Educational Services

digital™

VMS System Management II

EY-A769E-SG-0001

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

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

The following are trademarks of Digital Equipment Corporation:

BI
DEC
DEC/CMS
DEC/MMS
DECnet
DECsystem–10
DECSYSTEM–20
DECUS
DECwriter
DEUNA
DDCMP
DFA01
DFM
DF01
DF03
DF100
DF112
DF124
DF126
DF129
DF224

DF242
DHB32
DHF11
DHQ11
DHU11
DHV11
DIBOL
DMB32
DMF16
DMF32
DMR11
DMS
DMV11
DMZ32
DPV11
DSV11
DUP11
DZQ11
EduSystem
IAS
KCT32

KMS11
KMSIP
KMVIA
LA-12
LSI-11
MASSBUS
MicroVAX
PDP
PDT
Q-bus
RSTS
RSX
Scholar
Scholar Plus
UNIBUS
VAX
VAXcluster
VMS
VT
digital™

CONTENTS

EXAMPLES

FIGURES

TABLES

About This Course

COURSE DESCRIPTION

VMS System Management II is a lecture/lab course designed for students already familiar with the fundamentals of managing and maintaining a VMS system. This course teaches students the skills needed to perform long-term management of a VMS system. Each module builds on the topics covered in the *VMS System Management I* course and also on the individual student's system management experience.

The *VMS System Management II* course covers:

- Command procedures used to perform advanced system management tasks
- Customizing the DCL environment
- Advanced VMS concepts pertinent to system management
- Files-11 and RMS concepts
- Basic hardware and software troubleshooting
- Expanding an existing hardware configuration

RESOURCES

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

- *Guide to Using VMS Command Procedures*
- *VMS DCL Dictionary*
- *Guide to Setting up a VMS System*
- *Guide to Maintaining a VMS System*
- *VMS System Generation Utility Manual*
- *System Messages and Recovery Procedures Manual*
- *VMS System Dump Analyzer Manual*
- *VMS Monitor Utility Manual*

- *VMS Command Definition Utility Manual*
- *VMS Installation and Operations*
- *VMS General User's Manual*
- *Run-Time Library Manuals*
- *VMS SYSMAN Utility Manual*
- *VMS Mount Utility Manual*
- *VMS Backup Utility Manual*
- *VMS File Definition Language Facility Manual*
- *VMS Convert and Convert/Reclaim Utility Manual*
- *VMS Analyze/RMS File Utility Manual*
- *VMS Error Log Utility Manual*
- *VMS Analyze/Disk Structure Utility Manual*
- *VMS Authorize Utility Manual*
- *Guide to VMS Performance Management*
- *Guide to VMS File Applications*
- *Guide to VMS Files and Devices*
- *Guide to VMS System Security*
- *VAX/VMS Internals and Data Structures Manual*
- *VAX Systems and Options Catalog*

PREREQUISITES

Students must have experience in performing the fundamental system management functions listed under "Course Description." To perform these functions, students must know how to:

- Maintain the user authorization file (UAF).
- Create user file directories (UFDs).
- Control user processes.
- Create and manage batch queues.
- Manage devices.
- Install known images.
- Back up and restore files and volumes.
- Install maintenance updates and optional software.

Students can gain this information by taking the following course:

- *VMS System Management I*

Students should also have at least six months' experience as full-time system managers or one year's experience as part-time system managers.

COURSE ORGANIZATION

This course is divided into eight units, or modules. Each module covers a skill or related group of skills required to fulfill the course goals. Each module consists of:

- An introduction to the subject matter of the module.
- A list of the topics dicussed in the module.
- One or more objectives that describe what students can achieve by studying the module.
- A list of resources - the books or manuals needed to study the module. These resources should be available for reference in the classroom or lab.
- The text of each module, which includes outlines, tables, figures, and examples.

This student workbook is designed for use in a lecture classroom. Your instructor will use its topic outlines to present much of the subject matter of this course. As the instructor explains a particular subject, you can take notes in this textbook in the extra space surrounding the subject outline.

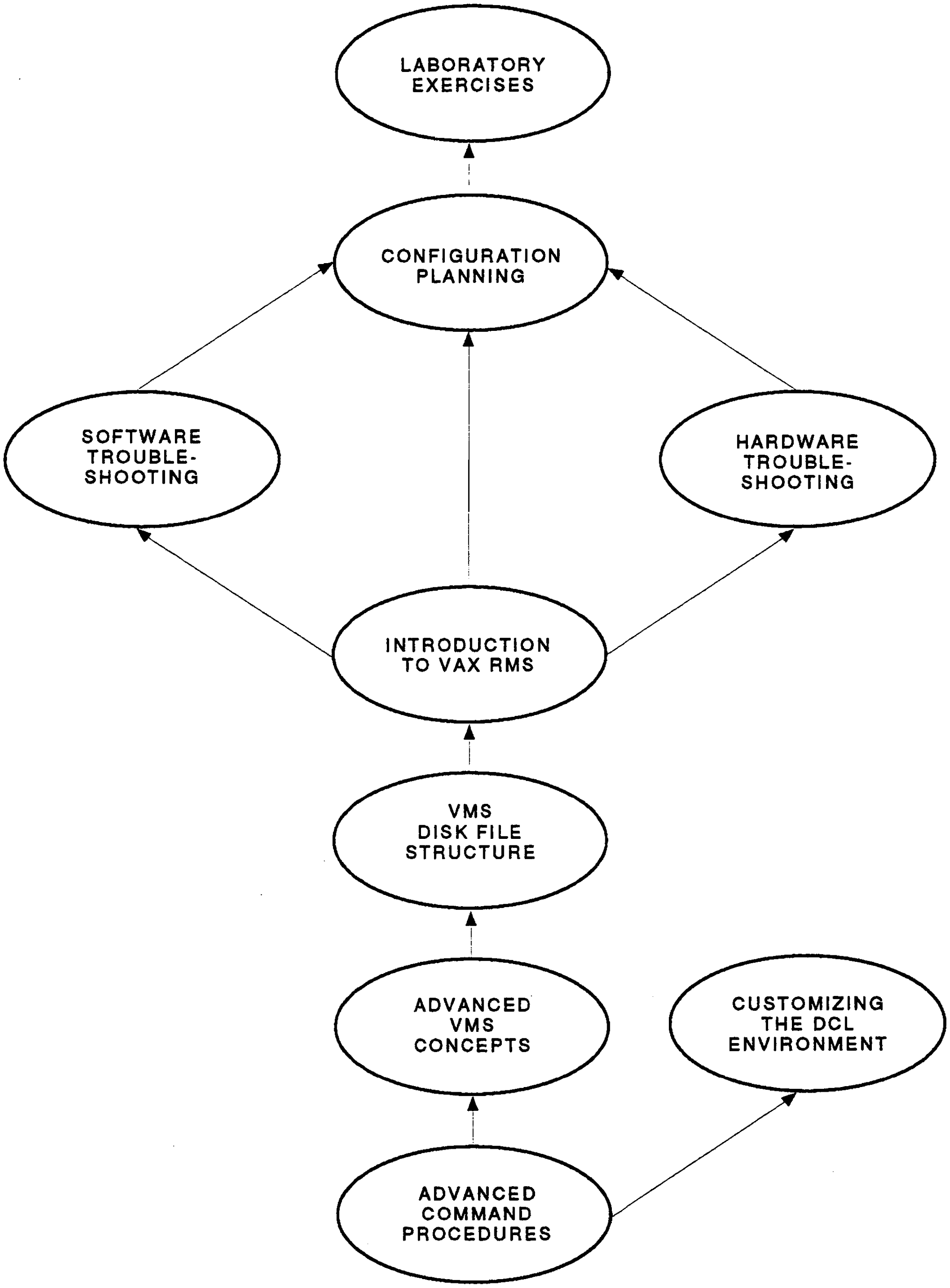
The outline will help you organize your notes. The tables, figures, and examples that accompany the outline should reduce the amount of writing required during lecture, enabling you to channel your energies into understanding the subject matter.

This course also includes a Laboratory Exercise module, which contains instructions for laboratory exercises. Use this module during your scheduled laboratory time.

The Course Map on the following page shows how each module is related to other modules and to the course as a whole. Before starting a new module, you should have a full understanding of all prerequisite modules. These prerequisites are located below the selected module in the Course Map and are connected to it by a chain of upward-pointing arrows.

This material can be effectively presented in many different ways. By providing prerequisite information as needed, your instructor may choose to use a module organization different from the one presented in the Course Map.

COURSE MAP



MKV_X2050_89

MODULE 1
ADVANCED 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. The system manager can create complex DCL procedures that compute an environment to maintain and check resources to ensure efficient use and provide adequate security.

This module discusses advanced command procedures used to create site-specific command procedures for system managers.

OBJECTIVES

To perform long-term system management functions, a system manager must be able to:

- List the DCL concepts and elements used in writing command procedures.
- Describe the three phases of symbol substitution.
- Write command procedures that:
  - Gather run-time information about the system.
  - Automatically mail information to the system manager.
  - Resubmit themselves for future execution.
  - Format and display run-time information on the system.
  - Perform tasks that require elevated privileges.

RESOURCES

- *Guide to Using VMS Command Procedures*
  - Developing command procedures
  - DCL concepts
  - Command procedure I/O
  - Using symbols and lexical functions
  - Design and logic
  - Controlling error conditions and CTRL/Y
- *VMS DCL Dictionary*
- *VMS General User's Manual*
  - Data representation
  - Command procedures
- *VMS DCL Concepts Manual*
  - Symbols and expressions
  - Symbol substitution

TOPICS

1. Review of basic concepts for developing command procedures
2. Symbols and expressions
3. Handling errors and debugging command procedures
4. Advanced DCL concepts and techniques
5. Symbol substitution
6. Repetitive and iterative substitution
7. Three-phase command processing
8. Passing and returning data to command procedures

REVIEW OF BASIC CONCEPTS FOR DEVELOPING COMMAND PROCEDURES

DCL Format Conventions

*better to put at bottom of command PROCEDURE! - GOOD but slow down execution! Keep two copies!

- Comments
  - Use comments to describe a procedure or block of command procedures.
  - Precede comments with an exclamation point (!).
  - Separate command sequences by inserting lines after logical sequences of command procedures with a combination of the ($!).
- Labels
  - Use labels to structure blocks of command procedures, especially in cases of recursion.
  - Labels are placed immediately following the dollar sign ($) and terminated by a colon (:).

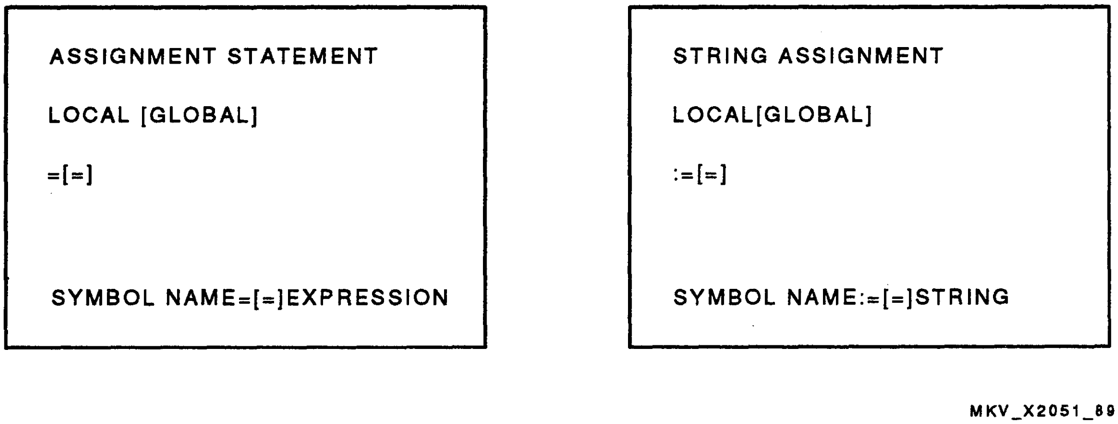
Example 1–1: A Formatted Command Procedure

```
$! 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 ERROR
$ SHOW MEMORY
$ SHOW MEMORY/POOL/FULL
$!
$! Run Accounting program for a daily summary of activity
$!
$ ACCOUNTING/SUMMARY=USER/REPORT=(RECORDS,PROCESSOR,-
  PAGES,PAGE_FILE,WORKING_SET)/SINCE=YESTERDAY
$!
$! Run Accounting program for a daily check of Log failures
$!
$ ACCOUNTING/FULL/TYPE=LOGFAIL/SINCE=YESTERDAY
$!
$! Find system files added and/or changed on the system disk.
$!
$ DIRECTORY/MODIFIED/SINCE=YESTERDAY SYS$SYSROOT:[000000...]
$!
$! Cleanup Section
$!
$CLEANUP:
$ EXIT
```

SYMBOLS AND EXPRESSIONS

- Symbols are names that represent character strings, integers, and logical values.
- Symbols are created when assigned a value. There are two types of symbols:

Figure 1–1: Symbol Types



ASSIGNMENT STATEMENT

LOCAL [GLOBAL]

=[=]

SYMBOL NAME=[=]EXPRESSION

STRING ASSIGNMENT

LOCAL[GLOBAL]

:=[=]

SYMBOL NAME:=[=]STRING

* Locals go away from local symbol table after command procedure exits! (login.com executes as long as you are logged in on your account).

can have " or '

Same { $ print == 'print /queue = Local'
$ print :== print /queue =Local

older syntax will probably not be supported forever.

Expressions are formed in two ways:

1. By combining literal character strings with operators

2. By combining literal numbers with operators

Table 1–1: Operators Used to Form Expressions

Character Operators	Name Operators	
* (asterisk)	.EQ.	equal
/ (slash)	.EQS.	strings equal
+ (plus sign)	.GE.	
- (minus sign)	.GES.	etc.
	.LE.	:
	.LES.	
	.LTS.	
	.NE.	
	.NES.	
	.NOT.	
	.AND.	
	.OR.	

= for assignment

but

.eq. for comparison!

Character String, Symbol, and Integer Expressions

For Assignment Statements:

- Character expressions contain literal character strings.
- A character string can represent a number.
- Symbols can be equated to character strings and numbers.
- The symbol's value is automatically substituted for the symbol.
- Integer expressions contain literal numbers.
- Specify integers as whole numbers.

For String Assignments:

- Use alphanumeric or special characters.
- Values are automatically converted to uppercase.
- Leading and trailing spaces and tabs are removed.
- Multiple spaces and tabs between characters are compressed to a single character.

Example 1–2: String Assignment

```
$ A = " Test    String A "
$ SHOW SYMBOL A
  A = " Test    String  A "

$ B :=  Test    String B
$ SHOW SYMBOL B
  B = "TEST STRING B"
```

Lexical Functions

- A lexical function performs operations on system and user data items.
- A lexical function substitutes the result of the operation for itself.
- The general format of a lexical function is:

```
F$function_name([args,...])
```

always need parentheses even if no args!

- Parentheses must always appear after the name of the lexical function, even if no arguments are specified. For example:

```
$ LENGTH = F$LENGTH("WOMBAT") + 1
$ SHOW SYMBOL LENGTH
  LENGTH = 7   Hex = 00000007  Octal = 00000000007

$ TIME = F$TIME()
$ SHOW SYMBOL TIME
  TIME = "20-MAR-1989 12:43:36.93"
```

* the only time # = length is retur
↓
(returns length when string not found)
(starts counting at 0!!! finds placement & returns #.

F$LOCATE - find substrings in strings
F$EXTRACT - extract substrings from strings

**can have lexical as an argument to another!!!
's and so on!

Example 1–3: Using Lexical Function F$LENGTH

Symbol:

```
$ MESSAGE   = "ERROR_MESSAGE"
$ MESSAGE_LEN = F$LENGTH(MESSAGE)
$ SHOW SYMBOL MESSAGE_LEN
  MESSAGE_LEN = 13   Hex = 0000000D  Octal = 00000000015
```

Symbol and Character String:

```
$ MESSAGE_LEN = F$LENGTH(MESSAGE + "S")
$ SHOW SYMBOL MESSAGE_LEN
  MESSAGE_LEN = 14   Hex = 0000000E  Octal = 000000000016
```

Lexical functions as arguments of Lexical Functions:

```
$ MESSAGE_LEN = F$LENGTH(F$DIRECTORY())
$ SHOW SYMBOL MESSAGE_LEN
  MESSAGE_LEN = 19   Hex = 00000013  Octal = 000000023
```

F$DIRECTORY gives the current value of the default directory
which happens to be [SYSTEM.COURSE_DEV]

F$GETQUI Lexical Function

- Format:

  F$GETQUI(Function, Item, Object-ID, Flags)

  Function is a $GETQUI system service function-code:

  DISPLAY_QUEUE
  DISPLAY_JOB
  DISPLAY_FILE
  DISPLAY_ENTRY
  DISPLAY_CHARACTERISTIC
  DISPLAY_FORM
  CANCEL_OPERATION
  TRANSLATE_QUEUE

  Item corresponds to a $GETQUI item-code.

  Object-ID specifies either the name or the number of an object (for example, a specific queue name or form number).

  Flags correspond to $GETQUI search-flags input item-code.

- Invokes the $GETQUI System Service to get information about queues and jobs currently in those queues.
- Allows access to characteristics, forms, queues, jobs in queues, and files contained in jobs in queues.
- Returns an integer, string, or Boolean value.
- Requires Read access to the job, or SYSPRV or OPER privilege.
- Returns one piece of information per call.

Example 1–4: Using the F$GETQUI Lexical Function

```
   $! Kill any previous $GETQUI activity
   $
1  $ TEMP = F$GETQUI( "CANCEL_OPERATION" )
   $
   $ Loop1:
   $
   $! Get names of batch queues. VMS will remember the queue name
   $! obtained for the next $GETQUI.  When all queues have been
   $! examined, the QNAME returned will be null.
   $
2  $ QNAME = F$GETQUI( "DISPLAY_JOB", "QUEUE_NAME", "*", "BATCH" )
   $ If QNAME .eqs. "" Then $ Exit
   $ Write SYS$OUTPUT "The working queue is ",QNAME
   $
   $ Loop2:
   $
3  $ JNAME = F$GETQUI( "DISPLAY_JOB", "JOB_NAME",, "ALL_JOBS" )
4  $ JPID  = F$GETQUI( "DISPLAY_JOB", "JOB_PID",,  "FREEZE_CONTEXT" )
   $ If JNAME .eqs. "" Then $ Goto Loop1
   $ Write SYS$OUTPUT "    Job name is: ",JNAME," and PID is: ",JPID
   $
   $ HEADER = "            Specified File(s): "
   $
   $ Loop3:
   $
5  $ FNAME = F$GETQUI( "DISPLAY_FILE", "FILE_SPECIFICATION" )
   $ Write SYS$OUTPUT HEADER,FNAME
   $ HEADER = "                               "
   $ If FNAME .nes. "" Then $ Goto Loop3
   $ Goto Loop2
```

Comments on Example 1-4

1. "Neutralize" any previous context.
2. Get the name of a BATCH queue.
3. Select a job from that queue.
4. Freeze the context on that job.
5. Get file names of all files in that job.

Control Statements

- The IF command tests the value of an expression and can execute one command, several commands, or take alternative action if the expression is false.
- The general format for executing a single command with no alternative action:

```
$ IF expression THEN command
```

- There are two ways to execute a block of commands.
  - Pass control to a label if the expression is true:

```
$ IF expression THEN  GOTO label
```

  - Use the block IF-THEN-ELSE statement:

```
$ IF expression
$ THEN
$   .
$   .   ! Commands to execute if true
$   .
$ ELSE
$   .
$   .   ! Commands to execute if false
$   .
$ ENDIF
  Note: The ELSE clause is optional.
```

- Use the GOTO command to pass control to a labeled line in a command procedure.
- Use loops to repeat statements until a condition is met.

Example 1-5: IF Commands Using Lexical Functions

```
$! Test whether the logical name TESTFILE has been assigned.
$! If there is no current assignment, the F$TRNLNM returns
$! a null string, and the command procedure continues executing
$! at ASSIGN.
$!
$ IF F$TRNLNM("TESTFILE") .EQS. "" THEN GOTO ASSIGN
  .
  .
  .
$ASSIGN:
  .
  .
  .
```

```
$! Try to give the process SYSPRV privilege.  Use F$PRIVILEGE to
$! test if the privilege was given, and if not, print an error
$! message.
$!
$ PRIVS = F$SETPRV("SYSPRV")
$ IF .NOT. F$PRIVILEGE("SYSPRV")
$  THEN
$    WRITE SYS$OUTPUT "You need SETPRV or SYSPRV privilege to run"
$    WRITE SYS$OUTPUT " this procedure"
$    GOTO CLEANUP
$  ENDIF
  .
  .
  .
$CLEANUP:
$ PRIVS = F$SETPRV(PRIVS)
$ EXIT
```

The GOSUB Statement

- Format:

  GOSUB label

  "Label" specifies an alpanumeric character label appearing as the first item on a command line; the label can precede or follow the GOSUB statement in the command procedure.

- Transfers control to command following the label; execution continues until a RETURN is encountered, at which point control returns to the command line following the GOSUB.
- If the label does not exist in the current command procedure, the procedure cannot continue and is forced to exit.
- GOSUB does not cause the creation of a new procedure level; all labels and local symbols are available to a subroutine invoked with a GOSUB.
- The GOSUB command can be nested up to a maximum of 16 levels per procedure level.

Example 1–6: Use of GOSUB in a Command Procedure

```
$ FOO = 99  ! Define a local variable
$ Gosub TEST1
$ Write SYS$OUTPUT "Successful completion !"
$ Write SYS$OUTPUT "The value of FOO is: ",FOO
$ Exit
$
$ TEST1:
$ Write SYS$OUTPUT "This is GOSUB Level 1"
$ Gosub TEST2
$ Return
$
$ TEST2:
$ Write SYS$OUTPUT "This is GOSUB Level 2"
$ Gosub TEST3
$ Return
$
$ TEST3:
$ Write SYS$OUTPUT "This is GOSUB Level 3"
$ Write SYS$OUTPUT "The value of FOO is: ",FOO
$
$ FOO = 100  ! Modify the local variable
$ Return
```

The CALL Statement

* JUST LIKE GOSUB BUT WITH ARGUMENTS!

- Format:

  CALL label [P1 [P2 [...P8]]] [/OUTPUT=filespec]

  "Label" is the name of a subroutine defined in the current command procedure.
  P1 - P8 represent parameters to be passed to the subroutine.
  The /OUTPUT qualifier can redirect the subroutine's SYS$OUTPUT to a specific file.

- Transfers control to a labeled subroutine and creates a new procedure level; execution continues in the subroutine until an EXIT is encountered, at which point control returns to the command line following the CALL.

- Similar to invoking a command file with the "@" command, except that it doesn't require DCL to open any additional files.

- Procedures can be nested up to 32 levels, including both subroutine CALLs and the use of the "@" command.

- Local symbols and labels within a nested subroutine are treated the same as if the routines had been invoked with the "@" command.

- The SUBROUTINE and ENDSUBROUTINE commands define the beginning and end of a subroutine.

- A subroutine can have only one entry point.

- The ENDSUBROUTINE command functions as an EXIT command if no EXIT is specified in the procedure.

Example 1–7: Use of CALL in a Command Procedure

```
$! Define subroutine SUB1
$
$ SUB1: Subroutine
$ Write SYS$OUTPUT "SUB1's P1 has the value: ",P1
$ Write SYS$OUTPUT "SUB1's P2 has the value: ",P2
 .
 .
$ Call SUB2  123
 .
 .
$ Exit
$ EndSubroutine
$
$
$! Define subroutine SUB2
$
$ SUB2: Subroutine
$ Write SYS$OUTPUT "SUB2's P1 has the value: ",P1
 .
 .
$ Exit
$ EndSubroutine
$
$
$! Start of main routine.  At this point, both SUB1 and SUB2
$! have been defined, but none of the previous commands have
$! been executed.
$
$ Call SUB1 "This is the P1 param"  "This is the P2 param"
 .
 .
 .
$ Call SUB2 999
 .
 .
 .
$ Exit
```

Example 1–8: Basic Concepts of a Command Procedure

WORTHLESS
COMMAND
PROCEDURE!!!

```
$! ADJTIME.COM is to be submitted as a batch job to adjust
$! the time for daylight savings time or standard time.
$! Submit it at some hour after midnight, since it does not
$! handle negative times or next day times.
~~$ PRIVS = ""~~ NOT NECESSARY
$ PRIVS = F$SETPRV("OPER,LOG_IO")
$ IF .NOT. F$PRIVILEGE("OPER,LOG_IO")
$  THEN
$    WRITE SYS$OUTPUT "You don't have the needed privileges"
$    GOTO DONE
$  ENDIF
$ TIME = F$TIME()
$ DAY = F$EXTRACT(0,2,TIME)
$ MONTH = F$EXTRACT(3,3,TIME)
$ IF MONTH .EQS. "APR"
$  THEN
$    IF F$CVTIME("TODAY",,"WEEKDAY") .NES. "Sunday" THEN GOTO DONE
$    IF DAY .GE. 08 THEN GOTO DONE
$    ADJUST = "+1"
$  ELSE
$    IF MONTH .EQS. "OCT"
$     THEN
$       IF F$CVTIME("TODAY",,"WEEKDAY") .NES. "Sunday" THEN GOTO DONE
$       IF DAY+7 .LE. 31 THEN GOTO DONE
$       ADJUST = "-1"
$     ELSE
$       GOTO DONE
$     ENDIF
$  ENDIF
$!
$! Adjust the time
$!
$ HOUR = F$EXTRACT(12,2,TIME)
$ HOUR = 'HOUR''ADJUST'
$ IF HOUR .LE. 9
$  THEN
$    HOUR[1,1] := 'HOUR'
$    HOUR[0,1] := 0
$  ENDIF
$ CUR_TIME = F$EXTRACT(12,8,TIME)
$ CUR_TIME[0,2] := 'HOUR'
$ SET TIME = 'CUR_TIME'
$!
$ MAIL NL: SYSTEM/SUBJECT="System time adjusted by ADJTIME.COM"
$!
$DONE:
$ PRIVS = F$SETPRV(PRIVS)
$ EXIT
```

Comments on Example 1-8

1. CUR_TIME := is a symbolic name for a global symbol substitution string assignment.
2. CHECK_APR is a DCL case statement.
3. F$CVTIME uses a character string in its symbol substitution.
4. CHECK_OCT is a DCL case statement.
5. ADJUST = is a literal number.
6. ADJUST_IT is a GOTO label.
7. HOUR = uses string concatenation.
8. MAIL mails correct time to the system.

HANDLING ERRORS AND DEBUGGING COMMAND PROCEDURES

→ can use to turn on verify in an if stmt.

- SET VERIFY or F$VERIFY are used to detect errors in a command procedure.
- Use SET NOON to prevent the command interpreter from checking the error status.
- Use SET ON to restore the default error checking by the command interpreter.

```
$ SET NOON
$ RUN TESTA
$ RUN TESTB
$ SET ON
```

This command procedure will run if either TESTA or TESTB returns an error condition.

- Use the ON command to perform an action if a severe error occurs.
  - — ON ERROR THEN GOTO EXIT
  - — ON CONTROL_Y THEN EXIT
- Override CTRL/Y interrupts by using the ON command.

Example 1–9: Handling Errors and Debugging (Sheet 1 of 2)

```
$! ANALYZE_DSK.COM analyzes the disks with /REPAIR and
$! resubmits itself to run on Tuesdays and Thursdays
$! at 3:00 am.
$!
$ SET NOON
$!
$! Give the needed privileges
$!
$ PRIVS = F$SETPRV("BYPASS,VOLPRO,WORLD")
$ IF .NOT. F$PRIVILEGE("BYPASS,VOLPRO,WORLD")
$  THEN
$    MAIL NL: SYSTEM/SUBJECT="ANALYZE_DSK not run - no privilege"
$    GOTO DONE
$  ENDIF
$!
$! Resubmit the command procedure for the next day
$!
$REQUEUE:
$ SUBMIT/QUEUE=SPECIAL_QUEUE/AFTER="TOMORROW+03:00"/RESTART -
   ANALYZE_DSK.COM
$!
$! Check that it is Tuesday or Thursday
$!
$ DAY = F$CVTIME(F$TIME(),,"WEEKDAY")
$ IF (DAY .EQS. "Tuesday" .OR. DAY .EQS. "Thursday")
$  THEN GOTO CHECK_THE_TIME
$  ELSE
$    MAIL NL: SYSTEM/SUBJECT="ANALYZE_DSK not run - wrong day"
$    GOTO DONE
$  ENDIF
$!
$! Make sure it isn't past 8:00
$!
$CHECK_THE_TIME:
$ SHOW TIME
$ IF F$EXTRACT(12,2,F$TIME()) .GE. 08
$  THEN
$    MAIL NL: SYSTEM/SUBJECT="ANALYZE_DSK not run - past 8:00"
$    GOTO DONE
$  ENDIF
$!
$! Analyze the disks and do any repairs
$!
$ ANALYZE/DISK_STRUCTURE/REPAIR DISK1:
$ ANALYZE/DISK_STRUCTURE/REPAIR DISK2:
```

Example 1-9: Handling Errors and Debugging (Sheet 2 of 2)

```
$! Check to see if the ERRFMT process is still running -
$! if not, restart it.
$!
$LOOP:
$ PID = F$PID(CONTEXT)
$ IF PID .EQS. ""
$  THEN
$    @SYS$SYSTEM:STARTUP ERRFMT
$    MAIL NL: SYSTEM/SUBJECT="ANALYZE_DSK done - ERRFMT restarted"
$  ELSE
$    IF F$GETJPI (PID,"PRCNAM") .EQS. "ERRFMT"
$     THEN MAIL NL: SYSTEM/SUBJECT="ANALYZE_DSK done"
$     ELSE GOTO LOOP
$     ENDIF
$  ENDIF
$!
$! Reset the original privileges
$!
$DONE:
$ PRIVS = F$SETPRV(PRIVS)
$ EXIT
```

Comments on Example 1-9

1. SET NOON prevents the command interpreter from checking the status returned from commands.

2. F$SETPRV sets the required privilege.

3. REQUEUE submits job for execution the next day.

4. ANALYZE/DISK/REPAIR is a DCL command.

5. MAIL mails results of the analyzed disk.

ADVANCED DCL CONCEPTS AND TECHNIQUES

Symbol Substitution

- Automatic symbol substitution -Does automatically replace symbols in Lexical functions!

  — ~~DCL automatically replaces symbol names or lexical functions in a command string with their current values.~~ ~~Not at the beginning of a line~~.

  — DCL automatically evaluates symbols and lexical functions when used in expressions.

  — Symbols are automatically evaluated at the beginning of a line when the symbol is not followed by an equal sign or a colon.

For more information on automatic symbol substitution, see Example 1-3.

- Nonautomatic symbol substitution
  - To request symbol substitution, use substitution operators.
  - DCL accepts two substitution operators:

    Apostrophe (')
    Ampersand (&)

  - Use apostrophes to enclose a string.

Example 1–10: Nonautomatic Symbol Substitution

```
$ ! This command procedure uses the apostrophes to enclose
$ ! a string.
$ !
$ ! The current default protection is saved before changing
$ ! the protection.
$ !
$ !
$ SAVE_PROT = F$ENVIRONMENT("PROTECTION")
$ SET PROTECTION = (SYSTEM:RWED, OWNER:RWED, GROUP, WORLD)/DEFAULT
    .
    .
    .
$ SET PROTECTION = ('SAVE_PROT')/DEFAULT
```

Use an ampersand to precede a string.

The following two symbol substitutions are functionally equivalent:

```
                    $ TYPE 'SYSTEM'
                    $ TYPE &SYSTEM
```

The result: String SYSTEM is equated to a character string value.

REPETITIVE AND ITERATIVE SUBSTITUTION

- ~~Repetitive substitution results when more than one type of substitution occurs in a single command string.~~

- ~~Iterative substitution occurs when symbols delimited by the apostrophe operator are translated during the first phase of command processing.~~

Example 1–11: Repetitive and Iterative Substitution

~~Repetitive Substitution:~~

Ex. Symbols and Lexical Functions

```
$ NEXT = F$CVTIME("TOMMOROW",,"WEEKDAY")
$ SHOW SYMBOL NEXT
  NEXT = "THURSDAY"
```

~~Iterative Substitution:~~

Ex. Concatenation of two Symbol Names

```
$ FILE = "FOO"
$ EXT = ".TXT"
$ PRINT 'FILE''EXT'
  FOO.TXT --->The file FOO.TXT is printed.
```

Symbols within Character String

```
$ PROMPT_STRING = "Creating file ''FILENAME' .TXT"
```

THREE-PHASE COMMAND PROCESSING

- Symbol substitution is performed by the command interpreter in ~~three~~ TWO phases of command processing.
  - — Command input ① scanning -- substitutes symbols 1st in a line or w/apostrophis
  - ~~— Command parsing~~ + ② EXECUTE - subs symbols in If expressions, lexical functions or w/ampersAND.
  - ~~— Expression evaluation~~
- The command interpreter does not perform substitution within data lines.
- Substitution does not occur on data read as input by commands or programs executed within a procedure.

Table 1–2: Three-Phase Command Processing

Phase	Processing
Command Input Scanning (Phase 1)	Reads command input. Replaces all tokens preceded with apostrophes.
Command Parsing (Phase 2)	Analyzes command string. Replace the symbol with the current value if the first value on the command line is a symbol used as a command synonym.
Expression Evaluation (Phase 3)	Replaces symbols during the actual execution of a command.

Table 1–3: Examples of Three-Phase Command Processing

Phase	Example
Command Input Scanning (Phase 1)	$ FILE = "MYFILE.TXT" $ TYPE 'FILE'
Command Parsing (Phase 2)	$ FILE = "MYFILE.TXT" $ TYPE &FILE
Expression Evaluation (Phase 3)	$ ANS = "YES" $ IF ANS THEN GOTO OK

Example 1–12: LOWPR.COM

```
$! LOWPR.COM allows the user to execute one DCL command at a
$! lowered priority.  The user's process name is modified.
$! However, once the DCL command completes executing, the
$! user's priority and process name are returned to "normal".
$!
$ ON SEVERE_ERROR THEN CONTINUE
$ ON CONTROL_Y THEN GOTO TERMINATION
$!
$ BASE_PRIORITY = F$GETJPI("","PRIB")
$ LOW_PRIORITY = BASE_PRIORITY - 2
$ PNAME = F$GETJPI("","PRCNAM")
$ UNAME := 'F$GETJPI("","USERNAME")'
$ SHORT_NAME = F$EXTRACT(0,9,UNAME)
$ NEW_NAME := "''SHORT_NAME'_PRI_''LOW_PRIORITY'"
$!
$ SET PROCESS/NAME='NEW_NAME'
$ SET PROCESS/PRIORITY='LOW_PRIORITY'
$!
$ ASSIGN/USER SYS$COMMAND SYS$INPUT
$ 'P1' 'P2' 'P3' 'P4' 'P5' 'P6' 'P7' 'P8'
$!
$ TERMINATION:
$ SET PROCESS/PRIORITY='BASE_PRIORITY'
$ SET PROCESS/NAME="''PNAME'"
$ EXIT
```

Comments on Example 1-12

1. base_priority :=... gets default priority
2. pname gets the process name
3. uname gets the username
4. shortuname extracts 9 characters from username
5. newuname :=... substitutes username at a lower priority
6. assign/user redirects sys$command.

PASSING DATA TO COMMAND PROCEDURES

- Use parameters to pass data to a command procedure.
- Use one or more parameter values to pass different data each time a command is executed.
- Specify a parameter as an integer, a string, or a symbol.
- Integers are converted to strings when assigned to one of the symbols from P1 through P8.
- Prompt the user at the terminal to enter data interactively.
- Use the INQUIRE and/or READ commands.
- Redefine SYS$INPUT to provide data from another file into commands and images.
- Redirect output of a command procedure to a file with the /OUTPUT qualifier.

RETURNING DATA FROM A COMMAND PROCEDURE

- To return a value to a command procedure, assign the value to be passed to a global symbol.
- Use logical names to return data from nested command procedures.

Example 1–13: Passing and Returning Data in a Command Procedure (Sheet 1 of 2)

```
$!
$!
$! ADD_ACNTS.COM adds an account to the system
$!
$! Includes info on
$! - changing the OWNER and ACCOUNT fields in the UAF.
$! - /PWDEXPIRED -- to make the PW pre-expired;
$! - /PASS=TEMP to /PASSWORD='username'
$! -  mailing of SYS$NOTES:NEWQUE.TXT.
$!
$!
$ VERIFICATION = 'F$VERIFY(0) ! 'F$VERIFICATION(DEBUG_COM)'
$!
$!
$ USERNAME := 'P1'
$ UIC := 'P2'
$ DISK := 'P3'
$ OWNER := 'P4'
$ ACCOUNT := 'P5'
$!
$GET_USER:
$ IF USERNAME .NES. "" THEN GOTO GET_UIC
$ INQUIRE USERNAME "Username"
$ GOTO GET_USER
$!
$GET_UIC:
$ IF UIC .NES. "" THEN GOTO GOT_UIC
$ INQUIRE UIC "UIC"
$ GOTO GET_UIC
$!
$GOT_UIC:
$ UIC = UIC - "[" - "]"
$ UIC = "[" + UIC + "]"
$!
$GET_DISK:
$ IF DISK .NES. "" THEN GOTO GOT_DISK
$ INQUIRE DISK "Disk"
$ GOTO GET_DISK
$!
$GOT_DISK:
$ DISK = DISK - ":" + ":"
$!
$GET_OWNER:
$ IF OWNER .NES. "" THEN GOTO GOT_OWNER
$ INQUIRE OWNER "Owner"
$ GOTO GET_OWNER
$!
$GOT_OWNER:
$!
```

Example 1-13: Passing and Returning Data in a Command Procedure (Sheet 2 of 2)

```
$GET_ACCOUNT:
$ IF ACCOUNT .NES. "" THEN GOTO GOT_ACCOUNT
$ INQUIRE ACCOUNT "Account"
$ GOTO GET_ACCOUNT
$GOT_ACCOUNT:
$!
$ ON ERROR THEN GOTO CLEAN_UP
$ PRIVS = ""
$ PRIVS = F$SETPRV("SYSPRV,BYPASS,CMKRNL")
$ IF  .NOT. F$PRIVILEGE("SYSPRV,BYPASS,CMKRNL")
$  THEN
$    WRITE SYS$OUTPUT "You don't have the needed privileges"
$    GOTO CLEAN_UP
$  ENDIF
$!
$ HOME = F$ENVIRONMENT("DEFAULT")
$ SET DEFAULT SYS$SYSTEM
$ UAF = "$SYS$SYSTEM:AUTHORIZE"
$ UAF ADD 'USERNAME'/UIC='UIC'/DEVICE='DISK'/DIR=['USERNAME'] -
  /OWNER="''OWNER'"/ACCOUNT='ACCOUNT'/PASSWORD='USERNAME' -
  /PWDEXPIRED
$ SET DEFAULT 'HOME'
$!
$ CREATE/DIR/OWNER='UIC' 'DISK'['USERNAME']
$ CREATE/OWNER='UIC' 'DISK'['USERNAME']LOGIN.COM
$DECK
$ IF F$MODE() .EQS. "INTERACTIVE"
$  THEN
$    TYPE SYS$SYSTEM:NOTICE.TXT
$    SET TERMINAL/INQUIRE
$  ENDIF
$EOD
$ CREATE/OWNER='UIC' 'DISK'['USERNAME']INIT.COM
$DECK
$ MAIL
SET MAIL_DIR [.MAIL]
EXIT
$ MAIL/SUBJECT="New User Info" SYS$COMMON:[NOTES]NEWUSER.TXT -
 'F$GETJPI(0,"USERNAME")'
$EOD
$!
$ SUBMIT/USER='USERNAME'/NOLOG 'DISK'['USERNAME']INIT.COM/DELETE
$!
$CLEAN_UP:
$ PRIVS = F$SETPRV(PRIVS)
$DONE:
$ IF VERIFICATION THEN SET VERIFY
$ EXIT
```

Comments on Example 1-13

1. VERIFICATION is locally substituted.
2. USERNAME, UIC, DISK, OWNER, and ACCOUNT are assigned to parameters.
3. PRIVS = - checks required privileges to enter UAF.
4. UAF ADD - builds an account with privileges for the new user.
5. CREATE/DIR/OWNER= - substitutes UIC, DISK, and USERNAME and creates the directory.
6. CREATE/OWNER= - provides the directory with login.com.
7. F$MODE allows new users to read system note.
8. SET MAIL_DIR - sets up the MAIL utility.
9. SUBMIT/USER='USERNAME' - submits username to the directory.

SUMMARY

This module provides the system manager with DCL commands necessary to build command procedures. Understanding these commands will help the system manager to maintain and protect the resources of the operating system in the most efficient manner.

Using symbol substitution as a primary structure in creating complex DCL procedures, the system manager can access the system interactively and noninteractively.

APPENDIX A

Summary of DCL Lexical Functions

Table 1–4: Lexical Functions That Obtain Information About a Process

Lexical Function	Description
F$ENVIRONMENT(item)	Returns information about the DCL command environment.
F$GETJPI(process-id,item)	Invokes the $GETJPI system service to return process information on the specific process.
F$MODE()	Returns a character string indicating the mode in which a process is running (INTERACTIVE, NETWORK, etc).
F$PID(context-symbol)	Returns the Process Identification (PID) number of a process.
F$PRIVILEGE(priv-list)	Returns the value of TRUE or FALSE depending on whether your process has the privileges in the specified list.
F$PROCESS()	Returns the current process name.
F$SETPRV(priv-list)	Invokes the $SETPRV system service to set the specified privileges for the process and returns the status of the privileges before the privileges were set.
F$USER()	Returns the current User Identification Code (UIC) in name format.
F$VERIFY([procedure-value],- [image-value])	Returns an integer value indicating whether procedure verification is enabled or disabled. When arguments are specified, F$VERIFY turns procedure and image verification on or off.

Table 1–5: Lexical Functions That Obtain Information About the System

Lexical Function	Description
F$CVTIME([input-time],[format],- [field])	Returns information about absolute combination or delta time strings.
F$GETDVI(device-name,item)	Invokes the $GETDVI system service to return information about a device.
F$GETQUI(function, [item],- [object-id],[flags])	Invokes the $GETQUI system service to return information about queues, jobs in queues, form definitions, and characteristic definitions.
F$GETSYI(item,[node])	Invokes the $GETSYI system service to return status and identification information about your system or a node in your cluster.
F$LOGICAL(logical-name)	Returns the equivalence string associated with the logical name. Does not perform iterative translation automatically.
F$MESSAGE(status-code)	Returns the message text associated with a specific system status code.
F$TIME()	Returns the current date and time string.
F$TRNLNM(logical-name,[table],- Returns the equivalence [index],[mode],[case],[item])	Returns the equivalence string or requested attributes associated with the logical name. Does not automatically perform iterative translation.

Table 1–6: Lexical Functions That Manipulate Character Strings

Lexical Function	Description
F$CVSI(bit-position,width,-string)	Extracts bit fields from character string data and converts the result, as a signed value, to an integer.
F$CVUI(bit-position,width,-string)	Extracts bit fields from character string data and converts the result, as an unsigned value, to an integer.
F$EDIT(string,edit-list)	Edits a string expression.
F$ELEMENT(element-number,-delimiter,string)	Extracts an element from a string. The elements must be separated by a delimiter. The first element is numbered zero.
F$EXTRACT(offset,number,-string)	Extracts a substring from a character string expression.
F$FAO(control-string,-[arguments])	Invokes the $FAO system service to convert the control string to a formatted ASCII output string.
F$INDENTIFIER(identifier,-conversion-type)	Converts an identifier from a character string to an integer, or vice versa.
F$INTEGER(string)	Returns an integer equivalent of the specified string.
F$LENGTH(string)	Returns the length of a string.
F$LOCATE(substring,string)	Locates the substring in the string and returns the offset position.
F$STRING(expression)	Returns the string equivalent of the result of the expression.
F$TYPE(symbol)	Returns the data type of the symbol.

Table 1–7: Lexical Functions That Obtain Information About Files and Directories

Lexical Function	Description
F$DIRECTORY()	Returns the current default directory as a character string. Does not return the associated device.
F$FILE_ATTRIBUTES(file-spec,- item) for a specific file.)	Returns attribute information item
F$PARSE(file-spec,[default-spec],- [related-spec],[field],[parse-type])	Invokes the $PARSE Record Management Service (RMS) to parse a file specification and return either the expanded file specification or the particular field you request.
F$SEARCH(file-spec,[stream-id])	Invokes the $SEARCH RMS to search a directory file and return the full file specifi- cation of a file you name.

MODULE 2
CUSTOMIZING THE DCL ENVIRONMENT

INTRODUCTION

The wide variety of standard VAX hardware and software configurations satisfy diverse application requirements. However, standard software and the default DCL environment may not entirely satisfy your specific application requirements. Most systems are customized to some degree, and the system manager usually performs the tasks related to system customization.

The VMS operating system provides many features that allow you to customize your system to meet your application and user requirements. This module discusses some methods for customizing your DCL environment and your boot procedures.

OBJECTIVES

To customize a VMS system to accommodate application-specific requirements, you should be able to:

- Customize the DCL environment by modifying the DCL tables.
- Create application-specific on-line HELP text and make it available to system users.
- Use device control libraries to customize your printing capabilities.
- Use application-specific logical names and tables to control access to application-specific resources.
- Use captive accounts for certain users to restrict their user environments.
- Customize system boot procedures to cause your system to boot from various devices.

RESOURCES

- *VMS DCL Dictionary*
  - Creating and Deleting Logical Names
  - Displaying Logical Names
  - Logical Name Tables
  - Command Descriptions
- *Guide to Setting Up a VMS System*
- *Guide to VMS System Security*
- *VMS Command Definition Utility Manual*
- *Run-Time Library Manuals*
- *Guide to VMS File Applications*
- *VMS Librarian Utility Manual*

TOPICS

1. Creating and deleting DCL commands
    a. Using the CDU to create a new command
    b. Using a user-defined command
    c. Removing a DCL command from the DCL tables
2. Creating and maintaining libraries
3. Adding HELP text to the system
    a. The format of a HELP text file
    b. Creating and using alternate HELP libraries
4. Device control libraries
    a. DCL commands related to device control libraries
    b. Steps in using a device control library
5. Creating application-specific logical names and tables
    a. Logical name tables - default and application-specific
    b. User categories and access modes
    c. Access to logical name tables
    d. Defining logical names for rooted directories
    e. Deleting logical name tables
6. Captive accounts
    a. Creating a UAF record for a captive account
    b. Creating a captive login command procedure
7. Customizing boot files
    a. Input parameters to the bootstrap program (VMB)
    b. Examples of boot command files
    c. Modifying boot files

just like adding the SAS invocation — wrong!

CREATING AND DELETING DCL COMMANDS

- SYS$LIBRARY:DCLTABLES.EXE is the master copy of the tables used to interpret DCL commands. By default they are copied into process P1 space at login.
- You can use the Command Definition Utility (CDU) to define command verbs that execute user-written programs.
- As you create new commands, they are added to your copy of the DCL command tables. No privilege is required to alter your own command tables.
- To make a new command available system-wide:
  - ~~Modify SYS$LIBRARY:DCLTABLES.EXE, thus creating a new version of the tables~~.
  - Run the INSTALL utility to use the new tables.
  - You must have SYSPRV and CMKRNL privileges.

NO! —CAN HAVE DANGEROUS SIDE EFFECTS!

PUT IN EVERYONE'S LOGIN.COM INSTEAD!!!

- You cannot alter DCL commands supplied with the system, but you can **remove** DCL commands, which allows you to limit users to a subset of DCL commands.

Using the CDU to Create a New Command

- When creating a new command, you must:
  - Supply an image to be executed by the command.
  - Use a text editor to create a command language definition file (extension .CLD).
  - Use SET COMMAND to add the command to your DCL tables.
- The .CLD file allows you to specify:
  - CDU statements that define the syntax of your command in DCL format.
  - Parameters or qualifiers (optional).
  - The image to be executed when someone enters the command at the DCL prompt.
- If your command definition allows parameters or qualifiers, the image must be prepared to interpret them. The program must call the appropriate run-time library routines (named CLI$xxx).

Table 2–1: Command Definition Utility Syntax Overview

CDU Statement	Parameters	Defines
DEFINE VERB	DISALLOW expression NODISALLOWS IMAGE image-string PARAMETER param-name [DEFAULT=string] [LABEL=label-name] [PROMPT=string] [VALUE=value-clause] NOPARAMETERS QUALIFIER qual-name NOQUALIFIERS ROUTINE routine-name SYNONYM synonym-name	A new verb and its characteristics
DEFINE TYPE	NAME DEFAULT LABEL=label-name [NON]NEGATABLE SYNTAX=syntax-name VALUE=value-clause [DEFAULT=string] [LIST of values] [REQUIRED] [TYPE=type-name]	Allowable syntax for values to be entered as parameters or qualifiers to a CDU statement
IDENT	none	Identifying information about an object module created by the SET COMMAND/OBJECT command
MODULE	none	A symbolic name for an object module

Example 2–1: The CLD File for the DCL Type Command

```
define verb type
   image type
   parameter p1,prompt="File",
       value(required,list,impcat,type=$infile),label=input
   qualifier backup
   qualifier before,  value(default=today,type=$datetime)
   qualifier confirm
   qualifier created
   qualifier exclude, value(required,list)
   qualifier expired
   qualifier modified
   qualifier output,  default,value(type=$file,default="SYS$OUTPUT")
   qualifier page,
   qualifier since,    value(default=today,type=$datetime)
   qualifier by_owner, value(type=$uic)

   disallow  output and page
```

Example 2–2: Adding a DCL Command to the Default Table

BAD - could lose old DCLTABLES.EXE

```
$ EDIT MY_COMMAND.CLD
  ! insert the following text

       DEFINE VERB MY_COMMAND
       IMAGE WORK1:[MY_DIR]MY_PROG.EXE
       NOPARAMETERS
       NOQUALIFIERS

   ! exit text editor

$ SET COMMAND /TABLE=SYS$LIBRARY:DCLTABLES.EXE -
_$ /OUTPUT=SYS$LIBRARY:DCLTABLES.EXE   MY_COMMAND.CLD

$ INSTALL
INSTALL> REPLACE SYS$LIBRARY:DCLTABLES.EXE
INSTALL> EXIT
```

Using a User-Defined Command

The following operations are performed to execute your command:

1. A user enters the command.
2. DCL parses the command and examines the command tables.
3. The image specified in your command definition is executed.
4. DCL passes control to the image.

Removing a DCL Command from the DCL Tables

- To remove a DCL command from your own process table:

```
$ SET COMMAND /DELETE=dcl_verb
```

- For example, to remove the MOUNT command:

```
$ SET COMMAND /DELETE=MOUNT
```

- When removing a DCL command from the system default table:
  - — Save the original DCL tables.
  - — Users currently logged in will continue to use the old version of the DCL table until they log out.
- To cause certain users to use a special DCL table:
  - — Use a different file name for the table.
  - — Alter CLITABLES field in UAF records for those users.

Example 2–3: Removing a DCL Command from the System Table

better * still vised!!!

```
$ COPY SYS$LIBRARY:DCLTABLES.EXE  SYS$LIBRARY:OLD_DCLTABLES.EXE
$ SET COMMAND /DELETE=MOUNT /TABLE=MY_DCLTABLES.EXE -
_$ /OUTPUT=SYS$LIBRARY:DCLTABLES.EXE

$ INSTALL
INSTALL> REPLACE SYS$LIBRARY:DCLTABLES
INSTALL> EXIT
```

must create first obviously!!!

Example 2–4: Removing a DCL Command and Creating a New Table

```
$ SET COMMAND /DELETE=MOUNT /TABLE=SYS$LIBRARY:DCLTABLES.EXE -
_$ /OUTPUT=SYS$LIBRARY:NEW_DCLTABLES.EXE
$SET PROTECTION = W:RE SYS$LIBRARY:NEW_DCLTABLES.EXE

$ INSTALL
INSTALL> ADD SYS$LIBRARY:NEW_DCLTABLES.EXE/OPEN/HEADER/SHARED
INSTALL> EXIT

$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF> MODIFY user /CLITABLES=NEW_DCLTABLES
UAF> EXIT
```

CREATING AND MAINTAINING LIBRARIES

- It is very important that libraries be maintained on a regular basis as they may become fragmented and take up increasing amounts of disk space.
- The table below lists some of the more frequently used qualifiers to the basic LIBRARY command.

Table 2–2: Qualifiers used in Maintaining Libraries

Qualifier	Function
/CREATE	Requests the creation of a new library.
/DELETE	Requests the deletion of one or more modules from library.
/INSERT	Requests the addition of one or more modules to the library.
/REPLACE	Requests the replacement of one or more existing modules with those specified.
/LIST [= output-file-spec]	Requests a listing of contents of a library.
/ONLY = module(s)	Specifies the individual modules for the LIBRARY command to operate.
/EXTRACT = module(s)	Copies one or more modules from a library into a file.
/OUTPUT = file-spec	Specifies the file where the output of Librarian commands will go.
/DATA = REDUCE	The library is stored in a data-reduced format.
/DATA = EXPAND	The library is changed to an expanded format.
/COMPRESS [=(option,[,...])]	Recovers space that was occupied by modules deleted from the library. It does this by creating a new library.

ADDING HELP TEXT TO THE SYSTEM

- You can create HELP text for:
  - User-defined DCL commands
  - Site-specific symbols and procedures
  - Local system information
- HELP text is found in HELP library files:
  - Default file type is .HLB
  - By default, HELP files reside in the SYS$HELP directory
  - The default VMS HELP library is SYS$HELP:HELPLIB.HLB
  - Other utilities have their own HELP libraries
- To add HELP text to a HELP library:
  - Place your HELP text in a specially formatted text file
  - Use default file type of .HLP
  - Use the LIBRARY command to insert your HELP text into the library
- To add or replace HELP text in a HELP library:

```
$ LIBRARY/HELP library_file.HLB  text_file.HLP
```

- To extract HELP text from a HELP library:

```
$ LIBRARY/HELP/EXTRACT=module /OUTPUT=module.HLP -
_$ library_file.HLB
```

- To remove HELP text from a HELP library:

```
$ LIBRARY /HELP /DELETE=module library_file.HLB
```

Format of a HELP Text File

Example 2–5: A Portion of HELP.HLP

```
1 HELP
 The  HELP  command  invokes  the  VAX/VMS  HELP  Facility
 to  display information about a VMS command or topic.
                   .
                   .
                   .
2 Parameters

 keyword ...
  Specifies one or more keywords that refer to the topic  or
  subtopic on which you want information from a HELP library.
                   .
                   .
                   .
2 Command_Qualifiers

/INSTRUCTIONS

 /INSTRUCTIONS (default)
 /NOINSTRUCTIONS

  Controls whether the HELP command displays information on
  how to use the  HELP facility.
                   .
                   .
                   .
2 Examples

  1.    $ HELP

        HELP
          .
          .  (HELP message text and list of topics)
          .
        Topic?

  Issuing the  HELP  command  without  any  qualifiers  or
  parameters produces a display of the HELP topics available
  from the root HELP library, SYS$HELP:HELPLIB.HLB.
                   .
                   .
                   .
```

Example 2–6: Output From the HELP HELP Command

```
$ HELP HELP

HELP

   The  HELP command  invokes  the  VAX/VMS  HELP  facility to
   display information about a VMS command or topic.
                                  .
                                  .
                                  .

  Additional information available:

  Parameters Command_Qualifiers
  /INSTRUCTIONS         /LIBLIST   /LIBRARY   /OUTPUT    /PAGE
  /PROMPT                /USERLIBRARY
  Examples

HELP Subtopic? PARAMETERS

HELP

  Parameters

     keyword ...
      Specifies one or more keywords that refer to the topic  or
      subtopic on which you want information from a HELP library.
                                  .
                                  .
                                  .
HELP Subtopic? /INSTRUCTIONS

HELP

  /INSTRUCTIONS

   /INSTRUCTIONS (default)
   /NOINSTRUCTIONS

    Controls whether the HELP command displays information on how
    to use the  HELP facility.
                                  .
                                  .
                                  .
HELP Subtopic?
Topic?
```

Creating and Using Alternate HELP Libraries

- Use a text editor to create a HELP file of the correct format.
- To create an alternate HELP library and insert a module:

```
$ LIBRARY/HELP/CREATE SYS$HELP:MY_HELP.HLB  MY_HELP.HLP
```

- To cause the HELP command to search your library:

```
$ HELP/LIBRARY=MY_HELP
```

- To cause the library to be searched automatically, use the following logical names:

HLP$LIBRARY

HLP$LIBRARY_1

HLP$LIBRARY_2

.

.

.

HLP$LIBRARY_n

Example 2–7: Site-Specific HELP File, FOOD.HLP

```
1 FOOD
         This help file contains information on the
         local restaurants. The system manager claims
         no responsibility for the quality of the food.

2 Command_qualifiers
  /Low_calorie
  /Vegetarian

2 Cheap

         The 'restaurants' listed here have one virtue -
         low price.

3 Burger_Biggie

         As its name implies, it serves big burgers.

2 Moderate

         Moderate restaurants serve decent food at
         moderate prices.  Of course, 'decent' and
         'moderate' are subject to interpretation.

3 Mels_Diner
          .
          .
2 Expensive
          .
3 Chez_Money
          .
          .
2 Outrageous
```

Example 2–8: Defining Additional HELP Libraries

```
$ ASSIGN /SYSTEM SYS$HELP:FOOD.HLB   HLP$LIBRARY
$ ASSIGN /SYSTEM SYS$HELP:TRAVEL.HLB HLP$LIBRARY_1
$ ASSIGN /SYSTEM SYS$HELP:MOVIES.HLB HLP$LIBRARY_2
```

There are two ways to access the Help libraries.

1. Go into the main Help library first:

```
$ HELP
    .
    .
    .
Additional Help libraries available (type @name for topics):

FOOD TRAVEL  MOVIES

Topic?  @FOOD

    .
    .
```

2. Go directly into the library:

```
$ HELP FOOD

FOOD
         This help file contains information on the
         local restaurants. The system manager claims
         no responsibility for the quality of the food.

  Additional information available:

  Cheap    Command_qualifiers    Moderate   Expensive
  Outrageous
```

DEVICE CONTROL LIBRARIES

- The system default device control library is:

  SYS$LIBRARY:SYSDEVCTL.TLB

- A device control library:
  - Is a text library
  - Contains user-written modules that cause printers to perform special functions
- Device control modules can contain:
  - Escape sequences and/or control characters that set up programmable printers for selected print options
  - Text to be inserted at specific points in the processing of a print job
- Device-specific print options include:
  - Point size
  - Character set
  - Bold or italic print
- Refer to your printer operation guide for the escape sequences and/or control characters to perform specific printer functions.

Table 2–3: DCL Commands Related to Device Control Libraries

Command	Function
DEFINE /FORM /SETUP=module	Specifies one or more modules that set up the device when the form is mounted before each file
DEFINE /FORM /PAGE_SETUP=module	Specifies one or more modules that set up the device before each page
INIT /QUEUE /LIBRARY=library_name START /QUEUE /LIBRARY=library_name	Specifies the file name of the device control library. The default is SYS$LIBRARY:SYSDEVCTL.TLB
INIT /QUEUE /SEPARATE=RESET=module START /QUEUE /SEPARATE=RESET=module SET QUEUE /SEPARATE=RESET=module	Specifies one or more modules to be extracted from the device control library and copied to the device at the end of each job. Default is NORESET.

Steps in Using a Device Control Library

- Use the following command to create a device control library:

```
$ LIBRARY /CREATE /TEXT libray-file
```

- Use a text editor to create a module that will cause your printer to perform the desired functions. You must enter the appropriate:
  - Escape sequences
  - Carriage control characters
  - Text
- Use the following command to insert the module into your device control library:

```
$ LIBRARY /INSERT /TEXT library-file module-file
```

- Use the following command to assign the device control library to a queue:

```
$ INITIALIZE /QUEUE /LIBRARY=library-file queue-name
```

(This step is not necessary if you use the default library SYSDEVCTL.TLB).

- When printing a file, request a device control library module by issuing one of the following commands:

```
$ PRINT /SETUP=module-file /QUEUE=queue file-name
$ PRINT /FORM=module-file /QUEUE=queue file-name
```

Example 2–9: Using Device Control Library Modules to Process a Print Job

```
$ LIBRARY /CREATE /TEXT SYS$LIBRARY:MYDEVCTL.TLB
$ EDIT BORDER.TXT

  ! these are the escape sequences that cause an
  ! LN01 laser printer to print a border around
  ! the perimeter of each output page.

  <ESC>[0;1;50;2400;10!|
  <ESC>[1;1;50;3150;10!|
  <ESC>[0;1;3200;2410;10!|
  <ESC>[1;2400;50;3150;10!|

  ! exit text editor

$ EDIT LANDSCAPE.TXT

  ! this is the escape sequence that causes
  ! the LN01 laser printer to print pages in
  ! 'landscape' mode, which means the text is
  ! printed laterally on the page.

  <ESC>[10m

  ! exit text editor

$ LIBRARY/INSERT SYS$LIBRARY:MYDEVCTL.TLB /TEXT BORDER.TXT
??? $ LIBRARY/INSERT SYS$LIBRARY:MYDEVCTL.TLB /TEXT LANDSCAPE.TXT
$ INIT /QUEUE /START /ON=TTA9 /LIBRARY=MYDEVCTL MY_QUE
$ SET QUEUE /SEPARATE=RESET=LANDSCAPE MY_QUE
$ SHOW QUEUE /FULL MY_QUE

Terminal queue MY_QUE, on TTA9
   /BASE_PRIORITY=4 /DEFAULT=(FEED) /FORM=DEFAULT
   /LIBRARY=MYDEVCTL /PROTECTION=(S:E,O:D,G:R,W;W)
   /OWNER=[1,4] /SEPARATE=RESET=(LANDSCAPE)

$ DEFINE /FORM /SETUP=BORDER FORM1 1
$ PRINT /FORM=FORM1 REPORT.TXT /QUEUE=MY_QUE
Job REPORT (Queue MY_QUE, entry 619) started on MY_QUE
```

CREATING APPLICATION-SPECIFIC LOGICAL NAMES AND TABLES

Logical Name Tables

- All logical names are cataloged in logical name tables.
- The system creates certain logical name tables by default.
  - Shareable tables are cataloged in the system directory table
  - Private tables are cataloged in process directory tables (one for each process)
- To display a list of the logical names in the system table:

```
$ SHOW LOGICAL /SYSTEM
```

- To display a list of the logical names in your process table:

```
$ SHOW LOGICAL /PROCESS
```

- To display the entire tree of logical name tables accessible to your process:

```
$ SHOW LOGICAL /STRUCTURE
```

Table 2–4: Default Logical Name Tables

Table	Name	Time of Creation	Logical Name
SYSTEM	LNM$SYSTEM_TABLE	At system initialization	LNM$SYSTEM
GROUP	LNM$GROUP_group_uic	At initialization of first process in group	LNM$GROUP
JOB	LNM$JOB_JIB-address	At initialization of first process in job	LNM$JOB
PROCESS	LNM$PROCESS_TABLE	At process initialization	LNM$PROCESS

Figure 2–1: The Structure of Logical Name Tables

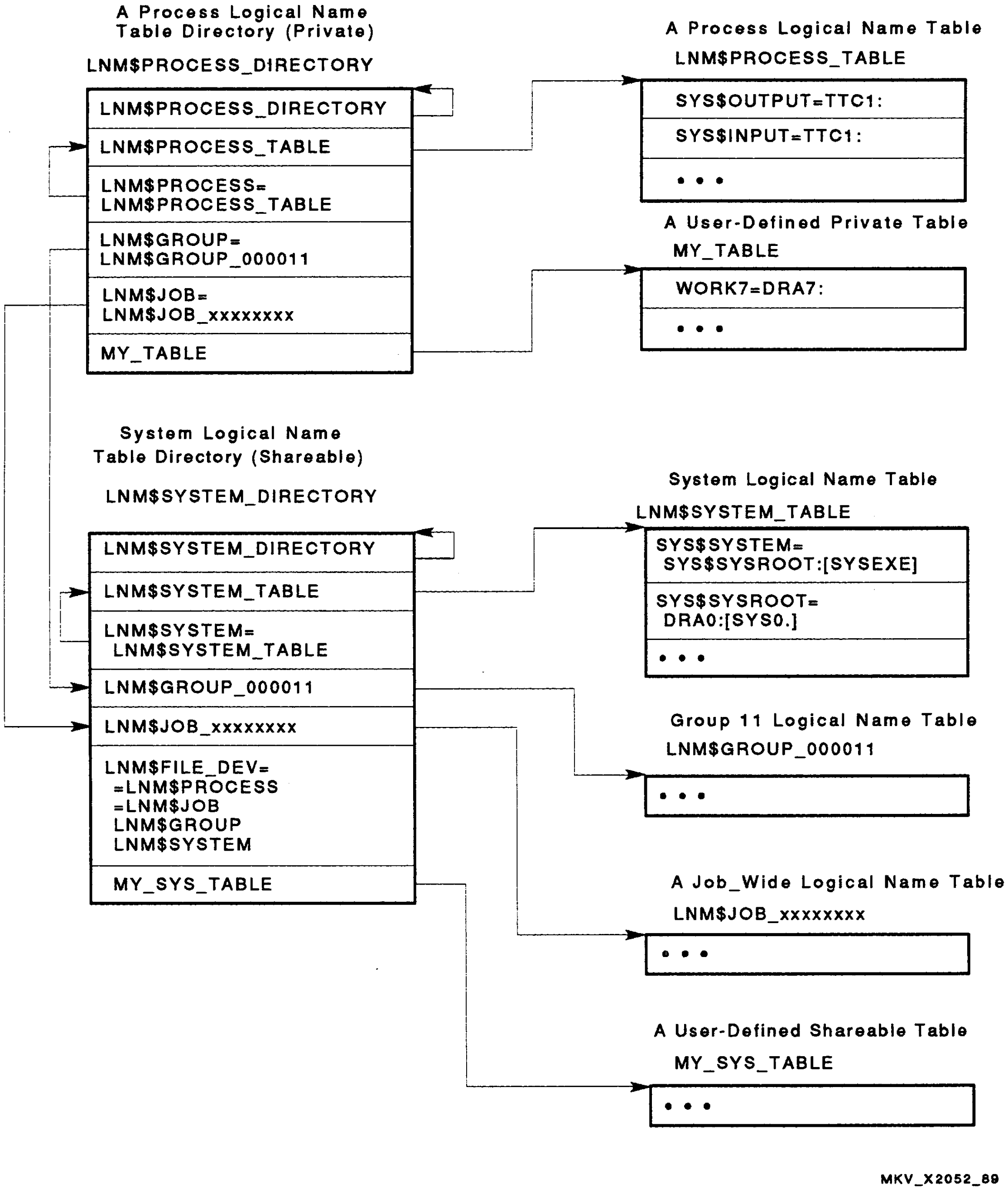


Table 2–5: Logical Name Table User Categories

Category	User Category
System	User has a system UIC
Owner	UIC of user matches UIC of table owner
Group	Group number of user matches group number of table owner
World	Neither part of user's UIC matches table owner's UIC

Table 2–6: Logical Name Table Access Codes

Code	Type of Access Allowed
R (Read)	Read and use logical names
W (Write)	Modify, add, and delete logical names
E (Enable)	Create a descendant table
D (Delete)	Delete the logical name table

Application-Specific Logical Name Tables

- You can create the following types of logical name tables:
  - Private tables, accessible only to your process
  - Shareable tables, accessible to multiple processes
- To create a process private logical name table:

```
$ CREATE /NAME_TABLE /PARENT_TABLE=LNM$PROCESS_DIRECTORY table_name
or simply (because LNM$PROCESS_DIRECTORY is assumed)
$ CREATE/NAME_TABLE table_name
```

- To create a shareable logical name table:

```
$ CREATE /NAME_TABLE /PARENT_TABLE=LNM$SYSTEM_DIRECTORY table-name
```

- To regulate access to shareable tables, you can use protection codes and ACLs. For example:

```
$ CREATE /NAME_TABLE /PARENT_TABLE=LNM$SYSTEM_DIRECTORY -
_$ /PROTECTION=(S:RWED,O:RWED,G:R,W) table_name
$ EDIT/ACL/OBJECT=LOGICAL_NAME_TABLE table_name
    use normal ACE syntax in editor
```

- To display the protection information:

```
$ SHOW LOGICAL/STRUCTURE/FULL (shows UIC protection)
$ SHOW ACL/OBJECT=LOGICAL_NAME_TABLE table_name
```

- To create a logical name in your user-defined table:

```
$ ASSIGN /TABLE=table_name equivalence_string  logical_name
```

- To display a list of logical names cataloged in your table:

```
$ SHOW LOGICAL /TABLE=table_name
```

Table 2–7: System-Defined and User-Defined Access to Logical Name Tables

Table	Accessible By	Type of Access Allowed
LNM$PROCESS_TABLE	The process only	Read, Write, Enable
LNM$JOB_JIB-address	Every process in the job	Read, Write
LNM$GROUP_group_uic	Every process in the same UIC Group	Read (all group users) Write (with GRPNAM)
LNM$SYSTEM_TABLE	All processes on the system	Read Write (with SYSNAM) Enable (with SYSPRV or a system UIC)
User-defined private table	Only the process that defines it	Read, Write, Enable Delete
User-defined shareable table (requires SYSNAM or system UIC to create)	Depends on the protection code	Depends on the protection code (Default is S:RWED,O:RWED,G,W)

Table 2–8: **Summary of DCL Commands Related to Logical Names and Tables**

Command	Function
ASSIGN	Creates a logical name and assigns equivalence string(s) to it.
CREATE	NAME_TABLE
DEASSIGN	Deletes a logical name or a logical name table.
DEFINE	Creates a logical name and assigns equivalence string(s) to it.
SHOW LOGICAL	Displays all logical names in one or more tables. Also displays the equivalence string(s) assigned to a specified logical name.

Deleting Logical Name Tables

- Certain tables are deleted by default.
- To explicitly delete a process-private table:

```
$ DEASSIGN /TABLE=LNM$PROCESS_DIRECTORY table_name
```

- To explicitly delete a shareable table:

```
$ DEASSIGN /TABLE=LNM$SYSTEM_DIRECTORY table_name
```

Table 2–9:, Default Logical Name Table Deletion

Table	Time of Deletion
System	At system shutdown
Group	At system shutdown
Job	At termination of the last process in the job
Process	At process termination

Rooted Directories

- Allows reference to directory trees as logical devices and top-level directories.
- Referencing this logical device may actually be accessing existing subdirectories.
- Reduces the number of top-level directories needed.
- Provides a way of making a directory or subdirectory appear as the master file directory (MFD) for the logical disk volume.
- To create a rooted directory, use a logical name assignment with the /TRANSLATION_ATTRIBUTES = CONCEALED qualifier.
    - The concealed device name must contain a trailing period (.)

      ```
      $ASSIGN/TRANSLATION_ATTRIBUTES=CONCEALED device_name:[directory.] logical_name
      ```

    - The directory specified during logical name definition serves as a base from which directories beneath it can be specified and is called the *root directory.*
    - The root directory must be specified using an alphanumeric specification rather than octal. For example, use [000000] rather than [0,0].

Example 2–10: Creating a Rooted Directory

```
$ASSIGN/TRANSLATION_ATTRIBUTES=CONCEALED DUA1:[DIRECTORY.] MYDIR

$SET DEFAULT MYDIR:[SUB]

$DIRECTORY *.DIR, [-]*.DIR
  MYDIR:[SUB]
   SUBSUB.DIR
  MYDIR:[000000]
   SUB.DIR
```

Using Rooted Directory Specifications

- To use a rooted directory, you can
  - Refer to the root directory itself, using the syntax [000000].
  - Refer to a specific subdirectory in the same way you refer to a top level directory using normal directory syntax.
  - Refer to subdirectories beneath the root directory using wildcards to traverse the directory tree.
- These commands will result in the translations below:

```
$ASSIGN ~~X~~ DUA0:[SMITH.]   X
$SET DEFAULT [JONES]
```

Table 2–10: Translations of Specifications

File Specification	Directories Accessed
X:	[SMITH.JONES]
X:[000000]	Root directory, [SMITH]
X:[]	[SMITH.JONES]
X:[-]	Root directory:[SMITH],listed as X:[000000]
X:[–]	Invalid (error)
X:[name]	[SMITH.name]
X:[.name]	[SMITH.JONES.name]
X:[name.*...]	All directories in all directory trees below [SMITH.name]
X:[*]	All directories one level below [SMITH.]
X:[*...]	All directories in all directory trees below [SMITH.]
X:[...]	All directories in all directory trees below [SMITH.JONES]

But can:

Username: name/Nocommand
also
" " /CLI=
" " /DISK =

CAPTIVE ACCOUNTS

- Permit very limited operations; for example, usually a specialized login command procedure controls which commands the user can access.
- To set up a captive account:
  - — Create an appropriate UAF record for the account
  - — Create a captive login command procedure

Creating a UAF Record for a Captive Account

- Authorize only privileges needed to do work
- Assign unique UIC group, restricting access to:
  - — Files owned by the captive account
  - — World accessible files
- * UIC group must be greater than 10 (octal) to distinguish it from the system group
  - — MAXSYSGROUP SYSGEN parameter determines maximum UIC group for system accounts

you can use this to break into system upon boot even if you have no privs!

- Set the following UAF flags:
  - — CAPTIVE
  - — DISCTLY
  - — LOCKPWD
  - — DISWELCOME
- Limit disk quota; allow only the amount needed
- Allow READ access to the captive login command procedure
- Set PRCLM to 0 to disable subprocess spawning

Captive Account UAF Record

Example 2–11: The UAF Record for a Captive Account

```
Username: SMITH            Owner:
Account:  SMITH            UIC:  [205,205] ([SMITH])
CLI:      DCL              Tables:
Default:  DEMONWORK1:[SMITH]
LGICMD:   SYS$MANAGER:SMITH_LOGIN.COM
Login Flags:  Disctly Lockpwd Captive Diswelcome
Primary days:   Mon Tue Wed Thu Fri Sat Sun
Secondary days:
No access restrictions
Expiration:            (none)    Pwdminimum:  6   Login Fails: 0
Pwdlifetime:        180 00:00    Pwdchange:   3-OCT-1988 10:06
Last Login: 7-OCT-1988 16:24 (interactive),(none) (noninteractive)
Maxjobs:         1  Fillm:       40   Bytlm:        8192
Maxacctjobs:     1  Shrfillm:     0   Pbytlm:          0
Maxdetach:       1  BIOlm:       18   JTquota:      1024
Prclm:           0  DIOlm:       18   WSdef:         150
Prio:            4  ASTlm:       24   WSquo:         200
Queprio:         0  TQElm:       10   WSextent:      300
CPU:        (none)  Enqlm:      200   Pgflquo:     10000
Authorized Privileges:
  TMPMBX NETMBX
Default Privileges:
  TMPMBX NETMBX
```

Creating a Captive Login Command Procedure

- Keep captive login procedures in SYS$MANAGER or where the user does not have write or delete access
- Use READ/PROMPT, not INQUIRE, to prompt for user input
- Validate all user input. Test for:
  - Correct syntax
  - Acceptable values
- Use error handling to handle all possible conditions
  - ON ERROR
  - SET NOON
- For highly restricted environments, the procedure should execute in a loop:
  - User can only exit by logging out
  - Procedure must handle all possible error conditions
- Reference: *Guide to VMS System Security*

Example 2–12: Captive Command Procedure (Sheet 1 of 2)

```
$! Comment Section
$!
$! This is the login command procedure for a captive account.
$! The characteristics of this account are defined in the UAF.
$!
$! Initialization Section
$!
$ ON CONTROL_Y THEN GOTO NEXT_CMD
$ SET CONTROL=(Y,T)
$ ESC[0,8] = %X1B
$NEXT_CMD:
$ ON ERROR THEN GOTO NEXT_CMD
$!
$! Main Section
$!
$WELCOME:
$ WRITE SYS$OUTPUT "''ESC'[2J"
$ WRITE SYS$OUTPUT "''ESC'[1;1H"
$ WRITE SYS$OUTPUT -
  "''ESC'[0;1m''ESC'#3 Welcome, captive user!''ESC'[0m''ESC'\"
$ WRITE SYS$OUTPUT -
  "''ESC'[0;1m''ESC'#4 Welcome, captive user!''ESC'\"
$ WRITE SYS$OUTPUT "''ESC'[0;m"
$ WRITE SYS$OUTPUT ""
$ TYPE SYS$INPUT
This captive account allows you to invoke the VAX MAIL utility
and to display information about system resource usage.

When you are ready to proceed, please choose one of the
following:

   0 - Logout
   1 - Invoke the VAX MAIL utility
   2 - Display general system statistics
   3 - Identify which processes are using the most CPU time
   4 - Display nonpaged pool statistics in graph form
$INQUIRE:
$ READ/PROMPT="Please enter your selection number " -
   SYS$COMMAND MENU_SEL
$ IF MENU_SEL .EQS. "4" THEN GOTO MONITOR_POOL
$ IF MENU_SEL .EQS. "3" THEN GOTO MONITOR_PROCESSES
$ IF MENU_SEL .EQS. "2" THEN GOTO MONITOR_SYSTEM
$ IF MENU_SEL .EQS. "1" THEN GOTO RUN_MAIL
$ IF MENU_SEL .EQS. "0" THEN GOTO LOGO
$ GOTO BAD_INPUT
```

Example 2-12: Captive Command Procedure (Sheet 2 of 2)

```
$MONITOR_SYSTEM:
$ MONITOR SYSTEM
$ GOTO WELCOME
$!
$MONITOR_PROCESSES:
$ MONITOR PROCESSES/TOPCPU
$ GOTO WELCOME
$!
$MONITOR_POOL:
$ MONITOR POOL/AVERAGE
$ GOTO WELCOME
$!
$RUN_MAIL:
$ DEFINE/USER SYS$INPUT SYS$COMMAND
$ MAIL
$ GOTO WELCOME
$!
$! Error Section
$! These commands are executed only if a problem or error
$!  occurs.
$!
$BAD_INPUT:
$ WRITE SYS$OUTPUT ""
$ WRITE SYS$OUTPUT -
  "Invalid selection please enter a single number from 0 to 4"
$ WRITE SYS$OUTPUT "''ESC'[5A"
$ WRITE SYS$OUTPUT "''ESC'[80D"
$ GOTO INQUIRE
$!
$! Cleanup Section
$! Deletes temporary symbols
$!
$LOGO:
$ DELETE/SYMBOL/LOCAL/ALL
$ DELETE/SYMBOL/GLOBAL/ALL
$ LOGOUT/BRIEF
$ EXIT
```

CUSTOMIZING BOOT FILES

- Boot files are processor-specific
- Several boot files reside on a processor's console device
  - Select one of them to be your default boot file
  - Name it DEFBOO.CMD or DEFBOO.COM, depending on your processor type
- Each boot file allows the processor to boot from a specific device
- You might want to modify a boot file to:
  - Boot from a disk drive other than the one specified in your boot file.
  - Set software boot control flags in R5 (see Appendix B).

Table 2–11: Device Codes Used in Boot File Names

Code	Boot Device
DB	RP05/06/07, RM03/05, RM80
DM	RK06/07
DQ	RL02
DJ	RA60
DU	RA80/81
CI	Any CI device, such as an HSC based disk

Table 2–12: Boot File Naming Conventions

File	Purpose
DEFBOO	Default boot procedure
dduGEN	Interactive boot from: dd (device code) u (unit number) For example: DU0GEN
dduBOO	Nonstop boot from: dd (device code) u (unit number) For example: DU1BOO
CIBOO	Nonstop boot from a CI device (HSC port number and unit from which to boot are specified in boot file)

Input Parameters to the Bootstrap Program (VMB)

- Register 0 (R0)
  - Bits 07:00 = Device type code
  - Bits 15:08 = Reserved for future expansion
  - Bits 31:16 = Device class dependent
- R1 - Boot device bus address
- R2 - Device dependent (refer to comments in boot files for more information)
- R3 - Unit number
- R4 - Logical block number from which to boot
- R5 - Software boot control flags that allow various booting functions, including:
  - Initiate a conversational boot
  - Initiate a diagnostic boot
  - Specify a top level directory number for a system disk containing multiple VMS systems; refer to Appendix B for details about boot control flags.

Example 2–13: VAX-11/780 Boot Command Procedure

```
!
! DB1 BOOT COMMAND FILE - DB1BOO.CMD
!
HALT   ! HALT PROCESSOR
UNJAM   ! UNJAM SBI
INIT   ! INIT PROCESSOR
DEPOSIT/I 11 20003800 ! SET UP SCBB
DEPOSIT R0 0  ! DISK PACK DEVICE TYPE
DEPOSIT R1 8  ! MBA TR=8
DEPOSIT R2 1  ! ADAPTER UNIT = 1
DEPOSIT R3 1  ! CONTROLLER UNIT = 1
DEPOSIT R4 0  ! BOOT BLOCK LBN (UNUSED)
DEPOSIT R5 4000  ! SOFTWARE BOOT FLAGS
DEPOSIT FP 0  ! SET NO MACHINE CHECK EXPECTED
START 20003000  ! START ROM PROGRAM
WAIT DONE  ! WAIT FOR COMPLETION
   !
EXAMINE SP  ! SHOW ADDRESS OF WORKING MEMORY+^X200
LOAD VMB.EXE/START:@ ! LOAD PRIMARY BOOTSTRAP
START @   ! AND START IT
```

Example 2–14: VAX 8800 Boot Command Procedure

```
!
! COMMAND PROCEDURE TO BOOT VMS FROM AN HSC OVER THE BCI.
!
! NEXT_PRIMARY is expected to point to the CPU that is to be used
! as the primary CPU.
!
! The following register deposits must be done before executing
! this command procedure or must be edited to correspond to the
! hardware configuration:
!
! R1 - Bus address information
! R2 - CI port # of HSC(s) to which drive is ported
! R3 - device unit number
! R5 - Software boot flags modify to boot from an alternate root of
!      a common system disk
!
SET VERIFY
SET CPU BOTH                  !SELECT BOTH CPUS
HALT                          !HALT BOTH CPUS
WAIT                          !WAIT FOR BOTH CPUS TO HALT
INIT                          !INIT BOTH CPUS,
                              ! COPY NEXT_PRIMARY TO CURRENT
SET CPU CURRENT_PRIMARY       !SELECT CURRENT_PRIMARY
UNJAM                         !CLEAR BI ERRORS,
                              ! NMI AND BI NODE STATE INFO
EXAMINE/L/P 3E000010          !READ MEM CSR4 (CLEARS INTERRUPTS)
EXAMINE/L/P 3E000014          !READ MEM CSR5 (CLEARS NMI FAULT)
DEPOSIT/L/P 3E000000 020F00F0 !CLEAR INTERLOCK TIMEOUT IN MEM CSR0
DEPOSIT/L/P 3E000008 F0000000 !CLEAR RDS ERROR BITS IN MEM CSR0
INIT                          !INIT PRIMARY
DEPOSIT R0 20                 !CI BOOT DEVICE TYPE CODE
DEPOSIT R1 00                 !BI BUS ADDRESS <5:4>=BI #
                              !  <3:0>=BI NODE #
DEPOSIT R2 00                 !CI PORT# OF HSC,<7:0>=CI PORT OF HSC
DEPOSIT R3 %D0                !UNIT # OF DRIVE, DECIMAL RADIX
DEPOSIT R4 0                  !NOT APPLICABLE
DEPOSIT R5 0                  !SOFTWARE BOOT CONTROL FLAGS
                              !ROOT DIRECTORY # IN <31:28>.DEFAULTS
                              !TO SYS0 OR UNROOTED DIRECTORY
DEPOSIT PSL 41F0000           !SET IPL=31, INTERRUPT STACK
FIND/MEM                      !FIND 64kb OF WORKING MEMORY,
                              !  SET COLD START BIT
IF NOT $STATUS THEN @EXIT     !IF UNSUCCESSFUL THEN DON'T BOOT
LOAD/MAIN/START=@ VMB.EXE     !LOAD VMB INTO GOOD MEM + 200
START @                       !START EXECUTING VMB
```

Modifying Boot Files

- You must modify boot files if:
  - — DEFBOO is not present on your console media
  - — Your DEFBOO specifies the wrong system device
- Always modify a copy of the distributed boot files, never the original version
- Example 2-15 shows how to modify CIBOO.COM for a VAX 8600 system

Example 2–15: Creating DEFBOO.COM for CIBOO.COM

```
(1)   $ RUN SYS$SYSTEM:SYSGEN
      SYSGEN> CONNECT CONSOLE
      SYSGEN> EXIT
(2)   $ EXCHANGE COPY CSA1:CIBOO.COM *
(3)   $ RENAME CIBOO.COM DEFBOO.COM
(4)   $ EDIT DEFBOO.COM
            .
            .
            .
(5)   $ EXCHANGE COPY DEFBOO.COM CSA1:
(6)   $ DISMOUNT CSA1
```

Comments on Example 2-15:

1. If the console device is not already connected, use the SYSGEN utility to connect it.
2. Use the Exchange utility to copy a boot command procedure from the console volume to your default directory:
   a. If the VAX 8600/8650 boots from an HSC50-based device, copy CIBOO.COM.
   b. If the VAX 8600/8650 boots from a UDA50-based device, copy DU0BOO.COM.
3. Rename the copy you just created to DEFBOO.COM.
4. Edit DEFBOO.COM as follows:
   a. If you copied CIBOO.COM (as assumed in this example), modify the DEPOSIT R2 statement giving the HSC50 CI node number(s).
   b. Modify the DEPOSIT R3 statement giving the system device unit number.
   c. Document your changes by modifying the comments.
5. Use Exchange to copy DEFBOO.COM to the console volume.
6. Dismount the console volume.

Example 2–16: CIBOO.COM from a VAX 8600 Console Volume

```
! Version: 001.000
!
! CI PORT BOOT COMMAND FILE - CIBOO.COM
!
! This CI port boot command file is set up to boot from a CI
! device; for example, an HSC-based disk.
!
! It assumes the CI780 is on SBIA #0, the TR level of the CI780
! is set to 14, the HSC node number is set to 15, and the disk's
! unit number is 0.
!
! If any of these assumptions are not true for your configuration,
! you may still use this command file by entering the BOOT/NOSTART
! console command and then altering the appropriate register values
! when the console command prompt reappears. Use the console
! command SHOW BOOT.HLP/ASCII to get more information on how to
! use the BOOT/NOSTART command and R5 boot options.
!
!
! Operating System Disk: CI DEVICE
!
!
SET SNAP ON         ! Enable ERROR_HALT snapshots
SET FBOX OFF        ! VMS will turn on Fbox
INIT                ! SRM processor unit
UNJAM               ! UNJAM SBIAs, Enable Master SBI interrupts
DEPOSIT CSWP 8      ! Turn off the cache (VMS turns the cache on)
!
DEPOSIT R0 20       ! Device Type is CI780
DEPOSIT R1 E        ! SBIA #0; TR number of the CI780 is 14
! DEPOSIT R2 F      ! HSC port number 15
! DEPOSIT R3 0      ! Unit number to boot from (in HEX)
DEPOSIT R2 0A09     ! HSC port numbers 10 and 9
DEPOSIT R3 1B       ! Unit number 27
DEPOSIT R4 0
! Logical block number to boot from if R5 bit 3 is set
                    ! Use R5 for optional boot control flags
FIND/MEMORY         ! Locate a 64KB chunk of good memory
EXAMINE SP          ! Display load address
LOAD/START:@ VMB
! Load VMB 200 bytes above the start of the good block
START @             ! Start VMB at the load address
```

SUMMARY

- You can customize the DCL tables by:
  - Using the CDU to create new DCL commands
  - Removing DCL commands from the DCL tables
- You can create your own on-line HELP by:
  - Writing your own HELP text
  - Placing your HELP text in the system HELP library
  - Creating alternate HELP libraries
- You can use device control libraries to customize your printing capabilities.
- You can use application-specific logical names and tables to control access to application-specific resources.
- You can create logical devices on a single volume by using rooted directories.
- You can enforce restricted user environments by using captive accounts for certain users.
- You can customize system boot procedures to cause your system to boot from various devices.
- You can customize system start-up and shutdown procedures to accommodate application requirements.

APPENDIX A

Examples of Logical Name Commands and Their Output

This appendix contains examples of DCL commands used to manipulate and display logical names and the contents of logical name tables.

Creating and Using Private Logical Name Tables

```
$ SHOW LOGICAL/STRUCTURE
(LNM$PROCESS_DIRECTORY)
    (LNM$PROCESS_TABLE)
(LNM$SYSTEM_DIRECTORY)
    (LNM$SYSTEM_TABLE)
    (LMF$LICENSE_TABLE)
    (LNM$GROUP_000100)
    (LNM$JOB_80776690)
    (LNM$JOB_80778E70)
    (LNM$JOB_8077CC50)
    (LNM$STARTUP_TABLE)

$ CREATE/NAME_TABLE MY_TABLE
$ SHOW LOGICAL/STRUCTURE
(LNM$PROCESS_DIRECTORY)
    (LNM$PROCESS_TABLE)
    (MY_TABLE)
(LNM$SYSTEM_DIRECTORY)
    (LNM$SYSTEM_TABLE)
    (LMF$LICENSE_TABLE)
    (LNM$GROUP_000100)
    (LNM$JOB_80776690)
    (LNM$JOB_80778E70)
    (LNM$JOB_8077CC50)
    (LNM$STARTUP_TABLE)

$ ASSIGN DRA7: WORK7
$ ASSIGN/TABLE=MY_TABLE FILE.DAT FILE
$ SHOW LOGICAL/PROCESS
(LNM$PROCESS_TABLE)
  "SYS$COMMAND" = "_SPLASH$RTA1:"
  "SYS$DISK" = "INS:"
  "SYS$ERROR" = "_SPLASH$RTA1:"
  "SYS$INPUT" = "_SPLASH$RTA1:"
  "SYS$OUTPUT" [super] = "_SPLASH$RTA1:"
  "SYS$OUTPUT" [exec] = "_SPLASH$RTA1:"
  "TT" = "RTA1:"
  "WORK7" = "DRA7:"
$ SHOW LOGICAL/TABLE=MY_TABLE FILE
   "FILE" = "FILE.DAT" (MY_TABLE)
```

```
$ SHOW LOGICAL/STRUCTURE
(LNM$PROCESS_DIRECTORY)
    (LNM$PROCESS_TABLE)
    (MY_TABLE)
(LNM$SYSTEM_DIRECTORY)
    (LNM$SYSTEM_TABLE)
    (LMF$LICENSE_TABLE)
    (LNM$GROUP_000100)
    (LNM$JOB_80776690)
    (LNM$JOB_80778E70)
    (LNM$JOB_8077CC50)
    (LNM$STARTUP_TABLE)

$ SET PROCESS/PRIVILEGE=SYSPRV
$ CREATE/NAME_TABLE/PARENT=LNM$SYSTEM_DIRECTORY MY_SYS_TABLE
$ CREATE/NAME_TABLE/PARENT=LNM$SYSTEM_DIRECTORY MY_SYS_TABLE1 -
_$ /PROTECTION=(S:RWED,O,G,W:RWED)
$ CREATE/NAME_TABLE/PARENT=LNM$SYSTEM_DIRECTORY MY_SYS_TABLE2 -
_$ /PROTECTION=(S:RWED,O,G,W)
$ SET PROCESS/PRIVILEGE=NOSYSPRV

$ SHOW LOGICAL/STRUCTURE
(LNM$PROCESS_DIRECTORY)
    (LNM$PROCESS_TABLE)
    (MY_TABLE)
(LNM$SYSTEM_DIRECTORY)
    (LNM$SYSTEM_TABLE)
    (LMF$LICENSE_TABLE)
    (LNM$GROUP_000100)
    (LNM$JOB_80776690)
    (LNM$JOB_80778E70)
    (LNM$JOB_8077CC50)
    (LNM$STARTUP_TABLE)
    (MY_SYS_TABLE)
    (MY_SYS_TABLE1)

$ ASSIGN/TABLE=MY_SYS_TABLE GRAPHICS.DAT GRAPHICS
$ ASSIGN/TABLE=MY_SYS_TABLE1 PAYROLL.DAT PAYROLL
$ ASSIGN/TABLE=MY_SYS_TABLE2 FORECAST.DAT FORECAST
%SYSTEM-F-NOPRIV, no privilege for attempted operation

$ SHOW LOGICAL/TABLE=MY_SYS_TABLE GRAPHICS
   "GRAPHICS" = "GRAPHICS.DAT" (MY_SYS_TABLE)
$ SHOW LOGICAL/TABLE=MY_SYS_TABLE1 PAYROLL
   "PAYROLL" = "PAYROLL.DAT" (MY_SYS_TABLE1)
$ SHOW LOGICAL/TABLE=MY_SYS_TABLE2 FORECAST
%SYSTEM-W-IVLOGTAB, invalid logical name table
%SHOW-S-NOTRAN, no translation for logical name FORECAST
```

Displaying the Contents of Directory Tables

```
$ SHOW LOGICAL/TABLE=LNM$DIRECTORIES

(LNM$PROCESS_DIRECTORY)
  "LNM$GROUP" = "LNM$GROUP_000100"
  "LNM$JOB" = "LNM$JOB_8077CC50"
  "LNM$PROCESS" = "LNM$PROCESS_TABLE"
  "LNM$PROCESS_DIRECTORY" [table] = ""
  "LNM$PROCESS_TABLE" [table] = ""
  "MY_TABLE" [table] = ""

(LNM$SYSTEM_DIRECTORY)
  "LMF$LICENSE_TABLE" [table] = ""
  "LNM$DCL_LOGICAL" = "LNM$FILE_DEV"
  "LNM$DIRECTORIES" = "LNM$PROCESS_DIRECTORY"
 = "LNM$SYSTEM_DIRECTORY"
  "LNM$FILE_DEV" [super] = "LNM$PROCESS"
 = "LNM$JOB"
 = "LNM$GROUP"
 = "LNM$SYSTEM"
  "LNM$FILE_DEV" [exec] = "LNM$SYSTEM"
  "LNM$GROUP_000001" [table] = ""
  "LNM$GROUP_000100" [table] = ""
  "LNM$JOB_80776690" [table] = ""
  .
  .
  "LNM$PERMANENT_MAILBOX" = "LNM$SYSTEM"
  "LNM$STARTUP_TABLE" [table] = ""
  "LNM$SYSTEM" = "LNM$SYSTEM_TABLE"
  "LNM$SYSTEM_DIRECTORY" [table] = ""
  "LNM$SYSTEM_TABLE" [table] = ""
  "LNM$TEMPORARY_MAILBOX" = "LNM$JOB"
  "LOG$GROUP" = "LNM$GROUP"
  "LOG$PROCESS" = "LNM$PROCESS"
 = "LNM$JOB"
  "LOG$SYSTEM" = "LNM$SYSTEM"
  "MY_SYS_TABLE" [table] = ""
  "MY_SYS_TABLE1" [table] = ""
  "MY_SYS_TABLE2" [table] = ""
  "TRNLOG$_GROUP_SYSTEM" = "LOG$GROUP"
 = "LOG$SYSTEM"
  "TRNLOG$_PROCESS_GROUP" = "LOG$PROCESS"
 = "LOG$GROUP"
  "TRNLOG$_PROCESS_GROUP_SYSTEM" = "LOG$PROCESS"
 = "LOG$GROUP"
 = "LOG$SYSTEM"
  "TRNLOG$_PROCESS_SYSTEM" = "LOG$PROCESS"
 = "LOG$SYSTEM"
```

```
$ CREATE/NAME_TABLE BUDGET
$ ASSIGN/TABLE=BUDGET CAR_EXPENSES.DAT CAR

$ SHOW LOGICAL/TABLE=LNM$DIRECTORIES LNM$FILE_DEV
   "LNM$FILE_DEV" = "LNM$PROCESS" (LNM$SYSTEM_DIRECTORY)
 = "LNM$JOB"
 = "LNM$GROUP"
 = "LNM$SYSTEM"
1  "LNM$PROCESS" = "LNM$PROCESS_TABLE" (LNM$PROCESS_DIRECTORY)
1  "LNM$JOB" = "LNM$JOB_8077CC50" (LNM$PROCESS_DIRECTORY)
1  "LNM$GROUP" = "LNM$GROUP_000100" (LNM$PROCESS_DIRECTORY)
1  "LNM$SYSTEM" = "LNM$SYSTEM_TABLE" (LNM$SYSTEM_DIRECTORY)

$ SHOW LOGICAL CAR
%SHOW-S-NOTRAN, no translation for logical name CAR

$ ASSIGN/TABLE=LNM$PROCESS_DIRECTORY -
_$ BUDGET,LNM$PROCESS,LNM$JOB,LNM$GROUP,LNM$SYSTEM  LNM$FILE_DEV

$ SHOW LOGICAL/TABLE=LNM$DIRECTORIES LNM$FILE_DEV
   "LNM$FILE_DEV" = "BUDGET" (LNM$PROCESS_DIRECTORY)
 = "LNM$PROCESS"
 = "LNM$JOB"
 = "LNM$GROUP"
 = "LNM$SYSTEM"
1  "BUDGET" [table] = "" (LNM$PROCESS_DIRECTORY)
1  "LNM$PROCESS" = "LNM$PROCESS_TABLE" (LNM$PROCESS_DIRECTORY)
1  "LNM$JOB" = "LNM$JOB_8077CC50" (LNM$PROCESS_DIRECTORY)
1  "LNM$GROUP" = "LNM$GROUP_000100" (LNM$PROCESS_DIRECTORY)
1  "LNM$SYSTEM" = "LNM$SYSTEM_TABLE" (LNM$SYSTEM_DIRECTORY)
   "LNM$FILE_DEV" = "LNM$PROCESS" (LNM$SYSTEM_DIRECTORY)
 = "LNM$JOB"
 = "LNM$GROUP"
 = "LNM$SYSTEM"
1  "LNM$PROCESS" = "LNM$PROCESS_TABLE" (LNM$PROCESS_DIRECTORY)
1  "LNM$JOB" = "LNM$JOB_8077CC50" (LNM$PROCESS_DIRECTORY)
1  "LNM$GROUP" = "LNM$GROUP_000100" (LNM$PROCESS_DIRECTORY)
1  "LNM$SYSTEM" = "LNM$SYSTEM_TABLE" (LNM$SYSTEM_DIRECTORY)

$ SHOW LOGICAL CAR
   "CAR" = "CAR_EXPENSES.DAT" (BUDGET)
```

Deleting a Logical Name Table Explicitly

```
$ SHOW LOGICAL/STRUCTURE
(LNM$PROCESS_DIRECTORY)
    (LNM$PROCESS_TABLE)
    (BUDGET)
(LNM$SYSTEM_DIRECTORY)
    (LNM$SYSTEM_TABLE)
    (LMF$LICENSE_TABLE)
    (LNM$GROUP_000100)
    (LNM$JOB_80776690)
    (LNM$JOB_80778E70)
    (LNM$JOB_8077CC50)
    (LNM$STARTUP_TABLE)

$ DEASSIGN/TABLE=LNM$PROCESS_DIRECTORY BUDGET

$ SHOW LOGICAL/STRUCTURE
(LNM$PROCESS_DIRECTORY)
    (LNM$PROCESS_TABLE)
(LNM$SYSTEM_DIRECTORY)
    (LNM$SYSTEM_TABLE)
    (LMF$LICENSE_TABLE)
    (LNM$GROUP_000100)
    (LNM$JOB_80776690)
    (LNM$JOB_80778E70)
    (LNM$JOB_8077CC50)
    (LNM$STARTUP_TABLE)
```

APPENDIX B

Software Boot Control Flags (Input to VMB in R5)

R5 Bit Meaning

0 RPB$V_CONV
Conversational boot. At various points in the system boot procedure, the bootstrap code solicits parameter and other input from the console terminal. If the DODGE is also on, the diagnostic supervisor should enter MENU mode and prompt user for the devices to test.

1 RPB$V_DEBUG
Debug. If this flag is set, VMS maps the code for the XDELTA Debugger into the system page tables of the running system.

2 RPB$V_INIBPT
Initial breakpoint. If RPB$V_DEBUG is set, VMS executes a BPT instruction immediately after enabling mapping.

3 RPB$V_BBLOCK
Secondary boot from the boot block. Secondary bootstrap is a single 512-byte block, whose LBN is specified in R4.

4 RPB$V_DIAG
Diagnostic boot. Secondary bootstrap is image called [SYSMAINT]DIAGBOOT.EXE.

5 RPB$V_BOOBPT
Bootstrap breakpoint. Stops the primary and secondary bootstraps with a breakpoint instruction before testing memory.

6 RPB$V_HEADER
Image header. Takes the transfer address of the secondary bootstrap image from that file's image header. If RPB$V_HEADER is not set, transfers control to the first byte of the secondary boot file.

7 RPB$V_NOTEST
Memory test inhibit. Sets a bit in the PFN bit map for each page of memory present. Does not test the memory.

8 RPB$V_SOLICT
File name. VMB prompts for the name of a secondary bootstrap file.

9 RPB$V_HALT
Halt before transfer. Executes a HALT instruction before transferring control to the secondary bootstrap.

10 RPB$V_NOPFND
No PFN deletion (not implemented; intended to tell VMB not to read a file from the boot device that identifies bad or reserved memory pages, so that VMB does not mark these pages as valid in the PFN bitmap).

11 RPB$V_MPM
Specifies that multiport memory is to be used for the total exec memory requirement. No local memory is to be used. This is for tightly coupled multiprocessing. If the DODGE is also on, the diagnostic supervisor enters AUTOTEST mode.

12 RPB$V_USEMPM
Specifies that multiport memory should be used in addition to local memory, as though both were one single pool of pages.

13 RPB$V_MEMTEST
Specifies that a more extensive algorithm be used when testing main memory for hardware uncorrectable (RDS) errors.

14 RPB$V_FINDMEM
Requests use of MA780 memory if MS780 is insufficient for booting. Used for 11/782 installations.

31:28 RPB$V_TOPSYS
Specifies the top level directory number for system disks with multiple systems.

APPENDIX C

11/780 Bootstrap HELP File

```
              Bootstrap Help File - BOOT.HLP

This file describes the input parameters to the bootstrap
program VMB.EXE .  Normally the bootstrap will look up the
file [SYSEXE]SYSBOOT.EXE on the specified device, load it
into memory and transfer control to it.

Two sets of command files are provided on the VMS
console floppy to perform the necessary bootstrap operations.
One set of these command files will boot, selecting an option
to stop in SYSBOOT to alter system parameters.  They are
invoked as console indirect command files.

        @DM0GEN                 ! Boot from RK07 unit 0
        @DM1GEN                 !                unit 1
        @DM2GEN                 !                unit 2
        @DM3GEN                 !                unit 3
        @DB0GEN                 ! Boot RM03/80 or RP05/6/7 unit 0
        @DB1GEN                 !      unit 1
        @DB2GEN                 !      unit 2
        @DB3GEN                 !      unit 3
        @DB4GEN                 !      unit 4
        @DB5GEN                 !      unit 5
        @DB6GEN                 !      unit 6
        @DB7GEN                 !      unit 7

The other set of these command files is normally invoked only
via the BOOT command but may be invoked explicitly as indirect
command files.  These command files perform a normal, non-
interactive boot without any stop in SYSBOOT to change
parameters.

BOOT DM0    or    @DM0BOO.CMD   ! Boot RK07 unit 0
BOOT DM1                        !           unit 1
BOOT DM2                        !           unit 2
BOOT DM3                        !           unit 3
BOOT DB0                        ! Boot RM03/80 or RP05/6/7 unit 0
BOOT DB1                        !      unit 1
BOOT DB2                        !      unit 2
BOOT DB3                        !      unit 3
BOOT DB4                        !      unit 4
BOOT DB5                        !      unit 5
BOOT DB6                        !      unit 6
BOOT DB7                        !      unit 7
```

The bootstrap is loaded into memory at least one page above the first available working memory to allow space for the Restart Parameter Block. The address of the base of the bootstrap is passed through SP, the stack pointer, where it also serves as a temporary stack pointer.

Input Parameters:

R0 - <07:00> = Device Type Code
 0 => MASSBUS Device (RM02/3,RP05/6/7,RM80)
 1 => Cartridge Disk (RK07)
 2 => Cartridge Disk (RL02)
 3 => IDC (almost an RM80) on 11/730
 11 => UDA50
 32 => HSC on CI
 64 => Console Block Storage Device

- <15:08> Reserved for future expansion

- <31:16> Device class dependent

UNIBUS - Optional vector address; 0 implies use the default vector

MASSBUS - not used

R1 - Boot device's bus address

11/780 &
11/730 - <31:04> MBZ
<03:00> TR number of Adapter

11/750 - <31:24> MBZ
<23:00> Address of the I/O page for the boot device's adapter

For most configurations the following convention has been used:

TR Number	Adapter / Controller
3	UNIBUS adapter number 0
4	UNIBUS adapter number 1
8	MASSBUS adapter number 0
9	MASSBUS adapter number 1
A	MASSBUS adapter number 2

```
R2  -  For UBA:
                 <31:18> = MBZ
                 <17:00> = UNIBUS Address of Control Register

    -  FOR MBA:
                 <31:04> = MBZ
                 <03:00> = Controller/Formatter Number
    -  FOR CI:
                 <31:08> = MBZ
                 <07:00> = HSC port number (station address)

R3  -  Unit Number

R4  -  Logical Block Number to boot from if bit 3 is set in R5
            (not supported on 11/750)

R5  -  Software Boot Control flags

        Bit     Meaning
        ---     -------

         0      RPB$V_CONV
                Conversational boot. At various points in the
                system boot procedure, the bootstrap code
                solicits parameter and other input from the
                console terminal.  If the DODGE is also on, then
                the diagnostic supervisor should enter "MENU"
                mode and prompt user for the devices to test.

         1      RPB$V_DEBUG
                Debug.  If this flag is set, VMS maps the code
                for the XDELTA Debugger into the system page
                tables of the running system.

         2      RPB$V_INIBPT
                Initial breakpoint. If RPB$V_DEBUG is set, VMS
                executes a BPT instruction immediately after
                enabling mapping.

         3      RPB$V_BBLOCK
                Secondary boot from the boot block.  Secondary
                bootstrap is a single 512-byte block, whose LBN
                is specified in R4.

         4      RPB$V_DIAG
                Diagnostic boot.  Secondary bootstrap is image
                called [SYSMAINT]DIAGBOOT.EXE.
```

5 RPB$V_BOOBPT
Bootstrap breakpoint. Stops the primary and secondary bootstraps with a breakpoint instruction before testing memory.

6 RPB$V_HEADER
Image header. Takes the transfer address of the secondary bootstrap image from that file's image header. If RPB$V_HEADER is not set, transfers control to the first byte of the secondary boot file.

7 RPB$V_NOTEST
Memory test inhibit. Sets a bit in the PFN bit map for each page of memory present. Does not test the memory.

8 RPB$V_SOLICT
File name. VMB prompts for the name of a secondary bootstrap file.

9 RPB$V_HALT
Halt before transfer. Executes a HALT instruction before transferring control to the secondary bootstrap.

10 RPB$V_NOPFND
No PFN deletion (not implemented; intended to tell VMB not to read a file from the boot device that identifies bad or reserved memory pages, so that VMB does not mark these pages as valid in the PFN bitmap).

11 RPB$V_MPM
Specifies that multi-port memory is to be used for the total exec memory requirement. No local memory is to be used. This is for tightly-coupled multi-processing. If the DODGE is also on, then the diagnostic supervisor enters "AUTOTEST" mode.

12 RPB$V_USEMPM
Specifies that multi-port memory should be used in addition to local memory, as though both were one single pool of pages.

13 RPB$V_MEMTEST
Specifies that a more extensive algorithm be used when testing main memory for hardware uncorrectable (RDS) errors.

```
        14      RPB$V_FINDMEM
                Requests use of MA780 memory if MS780 is insuffici-
                ent for booting. Used for 11/782 installations.

        <31:28> RPB$V_TOPSYS
                Specifies the top level directory number for
                system disks with multiple systems.

Output Parameters:

        R10 -      Base address of region containing secondary
                   bootstrap
        R11 -      Pointer to Restart Parameter Block (RPB)

        SP  -      ADDRESS+(^X200) of first working 64Kb memory
                   region usable as both stack pointer and pointer
                   to good memory.

        PR$_SCBB - System Control Block base register
```

```
       Memory Layout at Start of Secondary Bootstrap
       ---------------------------------------------

        +------------------------------------------+ :BASE
        !                                          !
        !       Restart Parameter Block (RPB)      !
        !                                          !
        +------------------------------------------+ :BASE+^X200
        !                                          !
        !       Primary Bootstrap Code             !
        !                                          !
        !                                          !
        +------------------------------------------+ :PR$_SCBB
        !                                          !
        !          System Control Block            !
        !                                          !
        +------------------------------------------+ :PFNMAP
        !                                          !
        !              PFN Bitmap                  !
        !                                          !
        +------------------------------------------+ :PFNMAP+^X800
        !                                          !
        !            Bootstrap Stack               !
        !                                          !
        +------------------------------------------+ :(SP)
        !                                          !
        !        Secondary Bootstrap Code          !
        !                                          !
        +------------------------------------------+
```

MODULE 3
ADVANCED VMS CONCEPTS

INTRODUCTION

The VMS operating system provides a variety of user facilities and resources that must be maintained and controlled by system management.

The system manager is responsible for:

- Making decisions that relate to optimizing the overall performance and efficiency of the system

- Performing tasks that relate to the overall management and control of the system

This module on advanced VMS concepts presents an overview of information aimed at helping the system manager diagnose and manage the VMS operating system. The materials cover information necessary to understand Module 6, "Software Troubleshooting."

OBJECTIVES

To successfully troubleshoot software problems, a system manager must be able to:

- Describe virtual memory and virtual address space.

- Describe the following system resources:

  - Dynamic memory (pool)

  - Memory other than pool

  - CPU

  - I/O

  - Pagefiles and Swapfiles

- Describe VMS process concepts.

- Describe the concepts and interrelationships of the following activities:
  - Paging
  - Swapping
  - Scheduling
- Describe the concepts and interrelationships of the following factors that affect system resource usage:
  - SYSGEN parameters
  - User quotas and privileges
- Describe the concept of the VMS lock manager.

RESOURCES

- *VAX Software Handbook*
- *VAX Architecture Handbook*
- *VAX/VMS Internals and Data Structures Manual*
- *VMS Authorize Utility Manual*
- *VMS System Generation Utility Manual*
- *Guide to Setting Up a VMS System*
- *VMS DCL Dictionary*

TOPICS

1. VMS process concepts
2. Virtual memory and virtual address space
3. Concepts of memory management
4. Concepts and interrelationships of:
    a. Paging
    b. Swapping
    c. Scheduling
5. System resource: dynamic memory (pool)
6. SYSGEN parameters and user quotas and privileges
7. VMS lock manager concepts

VMS PROCESS CONCEPTS

A process is the basic entity in VMS that can be scheduled. It is the consumer of various system resources (memory, I/O, and CPU time), and it provides the context within which an image executes. A process consists of:

1. Virtual address space
2. Hardware context
3. Software context

A process can be in one of three states:

1. Currently executing
2. Waiting for an event
3. Executable

Figure 3–1: Hardware PCB and Software PCB

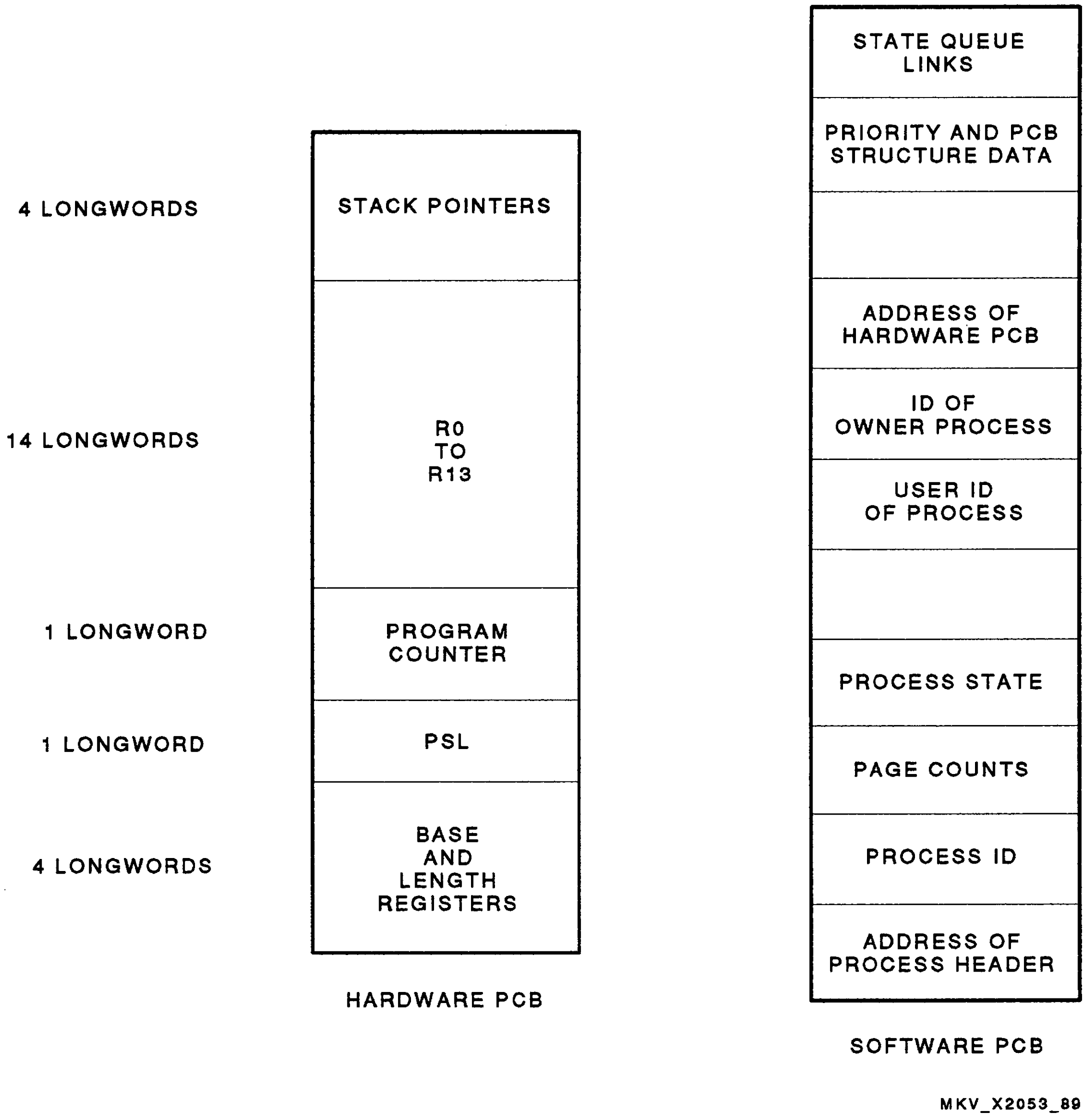
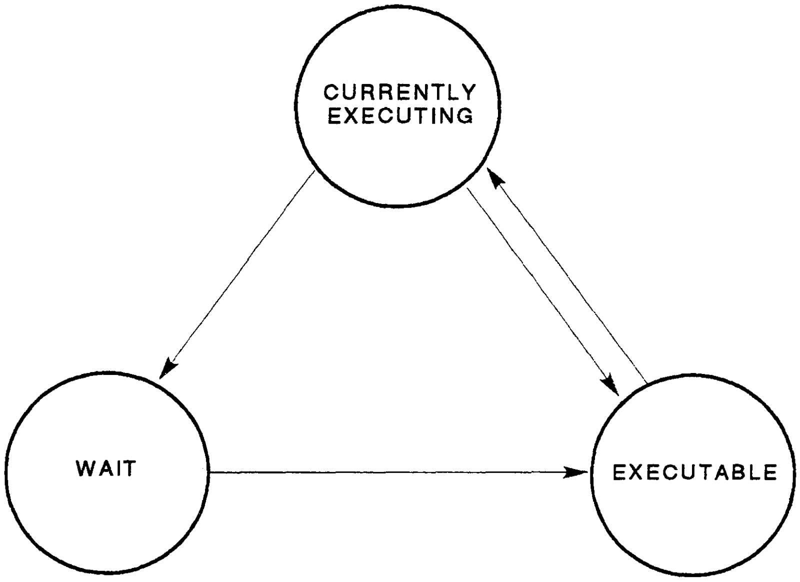


Figure 3–2: Process States



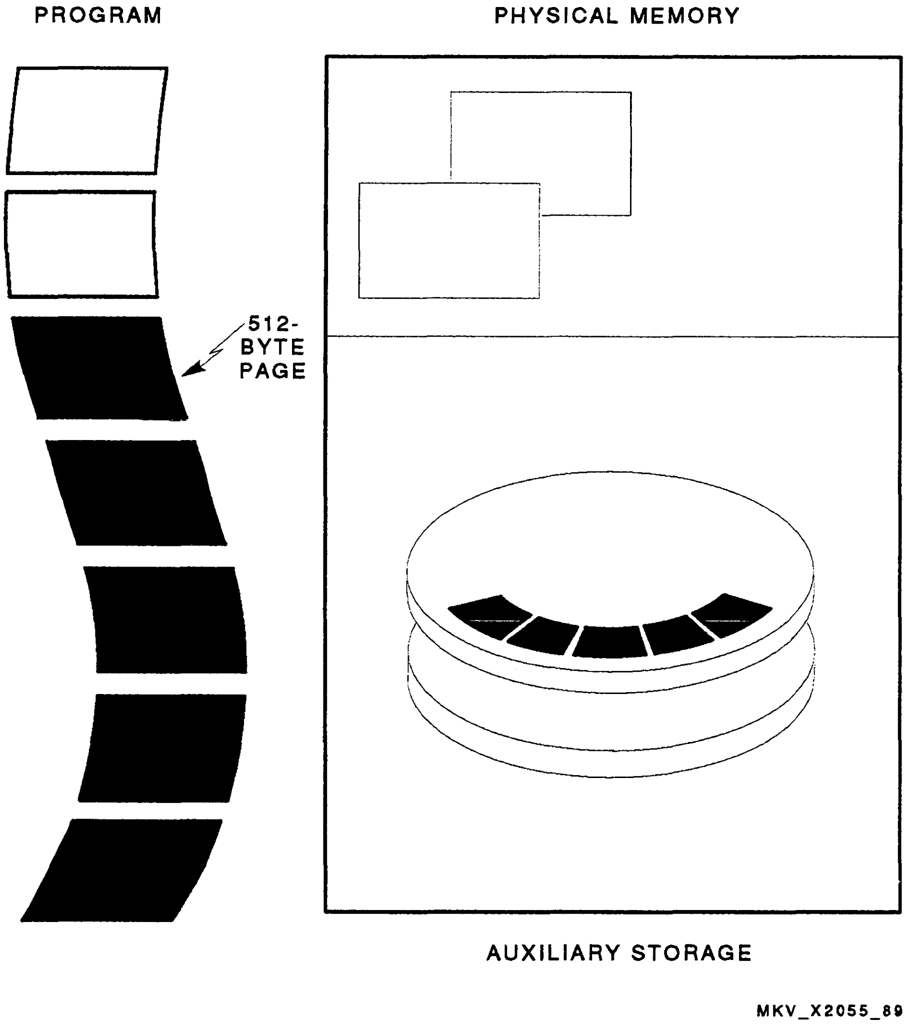
MKV_X2054_89

VIRTUAL MEMORY AND VIRTUAL ADDRESS SPACE

Virtual Memory

- The VMS memory management scheme uses two types of addresses: virtual addresses and physical addresses.
  - All addresses generated by a process are virtual, in that they are unique only to the process itself. They must be "translated" by the CPU into the addresses of specific storage locations.
  - Physical addresses, recognized by main memory hardware, refer to actual storage locations.
- At any point during program execution:
  - Part of the program resides in physical memory.
  - Part of the program can reside on disk.
- The current size of a process's virtual memory is the sum of its memory-resident parts, plus its disk-resident parts.
- Program size can exceed the amount of physical memory available.

Figure 3–3: VMS Virtual Memory System



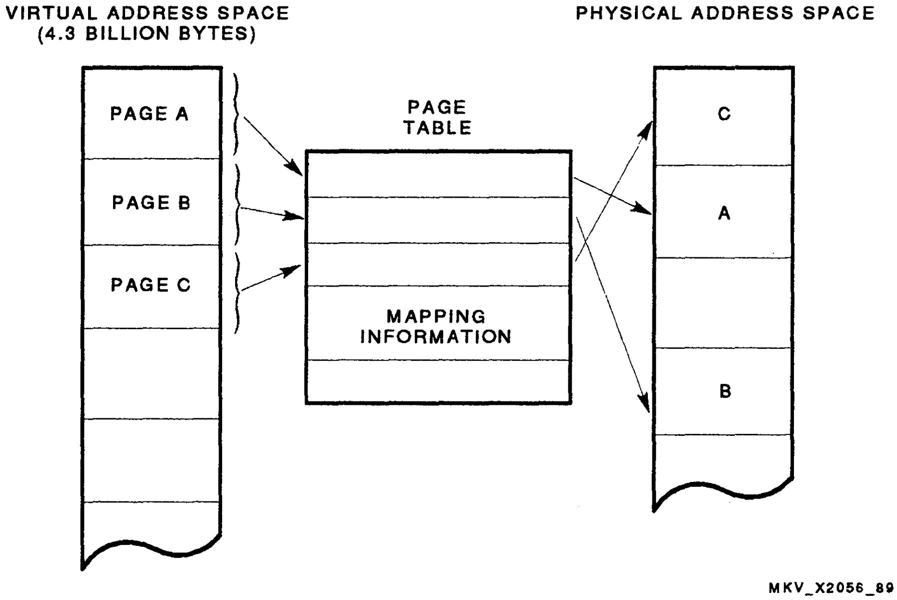
Virtual Address Space

- VAX architecture supports a 32-bit virtual address, allowing a maximum virtual address space of nearly 4.3 billion bytes. All 512-byte sections are called pages.
- A virtual page is the basic unit of relocation and protection in the virtual address space; one page consists of 512 consecutive bytes of virtual space.
- Virtual address space is divided into two equal parts: process space and system space.
  - Process space

    Consists of virtual addresses 0 to 7FFFFFFF (hexadecimal).
    Contains code and data specific to one process.
  - System space

    Consists of virtual addresses 80000000 to FFFFFFFF (hexadecimal).
    Contains the VMS executive itself and any data structures required by VMS to coordinate the operation of the system.

Figure 3–4: Mapping Virtual Pages into Physical Address Space



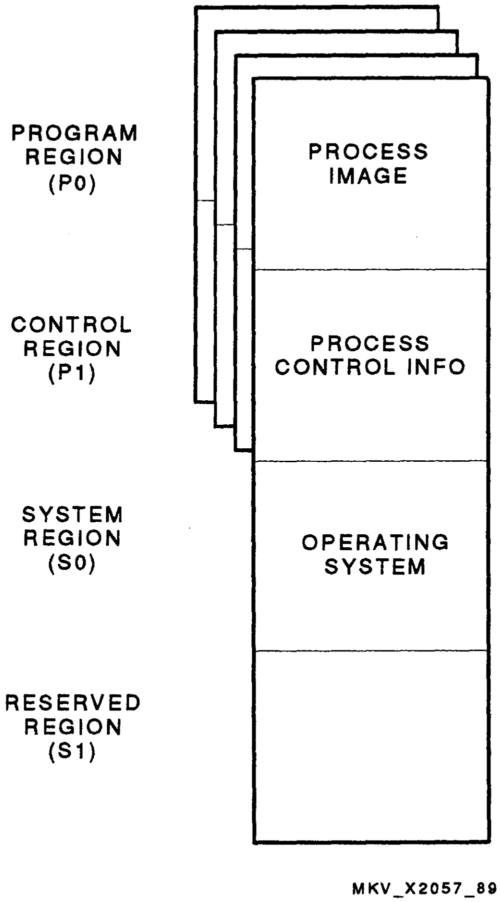
Process Space

- Program region P0 (addresses 0 to 3FFFFFFF) is reserved for the current image that the process is executing, plus any run-time libraries used by that image.
- Control region P1 (addresses 40000000 to 7FFFFFFF) is reserved for process control information, including the kernel, executive, supervisor, and user stacks, the DCL command interpreter, and the process I/O database.
- Every process has its own unique P0 and P1 space, and no process can access the P0 or P1 space of any other process.

System Space

- System region S0 (addresses 80000000 to BFFFFFFF) is reserved for the executable routines of the operating system, plus various data structures maintained by VMS. These data structures include:
  - Process control blocks
  - Process headers
  - System and group logical name tables
  - Mailboxes
  - Miscellaneous temporary structures necessary for operating system housekeeping
- All processes share the same S0 space.
- Reserved region S1 (addresses C0000000 to FFFFFFFF) is not used.

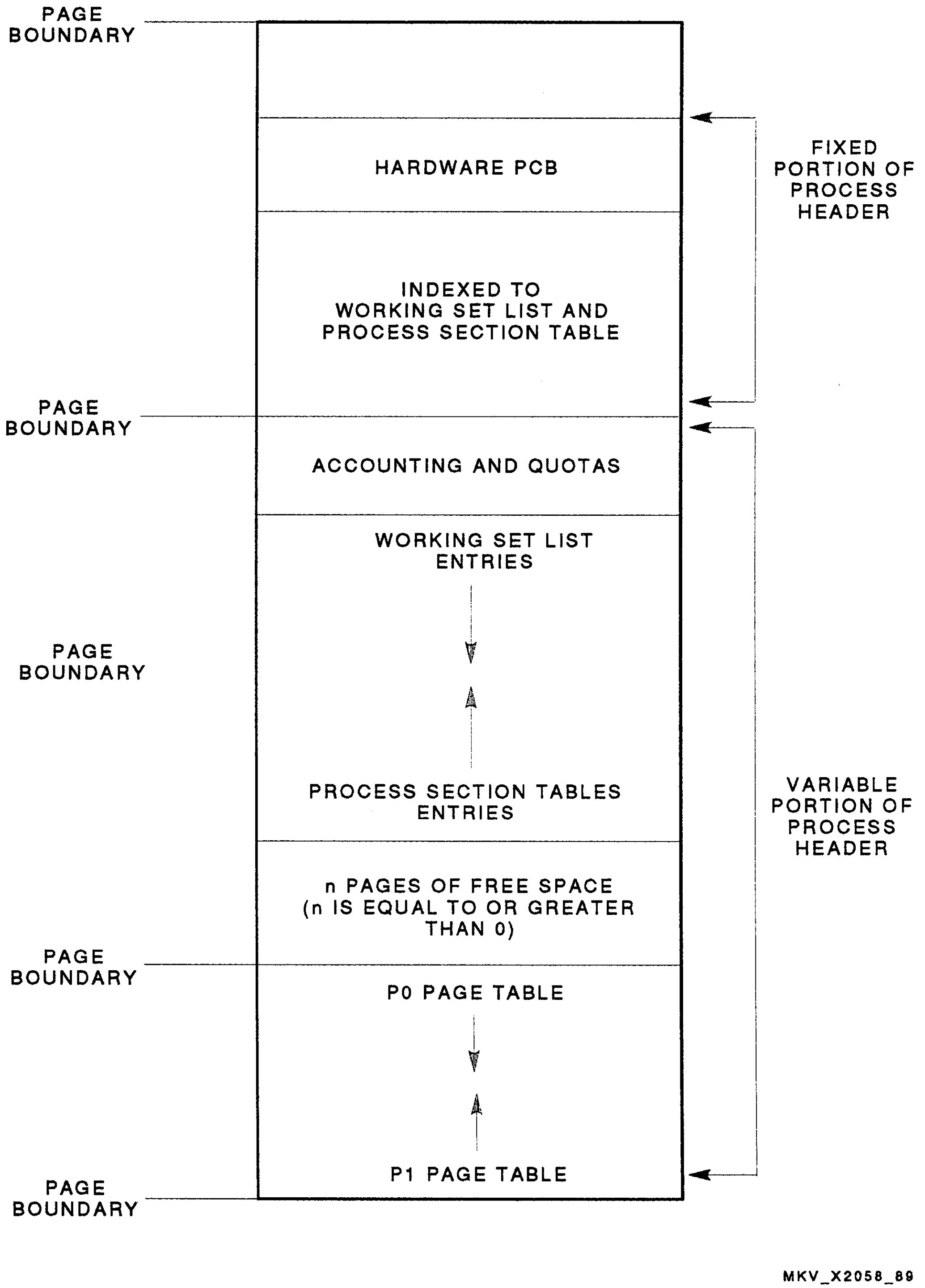
Figure 3–5: Virtual Address Space



CONCEPTS OF MEMORY MANAGEMENT

- Memory management hardware translates a virtual address into a physical address.
- Memory management software:
  - Maintains tables of mapping information commonly called page tables.
  - The CPU uses page tables to translate virtual addresses generated by a process into the physical addresses needed to refer to main memory.
  - Page tables keep track of where each 512-byte virtual page is located in physical memory.
  - One page table exists for each of the three regions - P0, P1, and S0.
- Process page tables (P0 and P1):
  - Reside in a pageable area of S0 space when the process is resident.
  - Can reside on disk and be paged and swapped as necessary.
  - Do not need to be mapped into contiguous physical pages.
- System page table
  - The only page table that must be resident in physical memory.
  - Must be mapped into contiguous physical pages.

Figure 3–6: Process Header

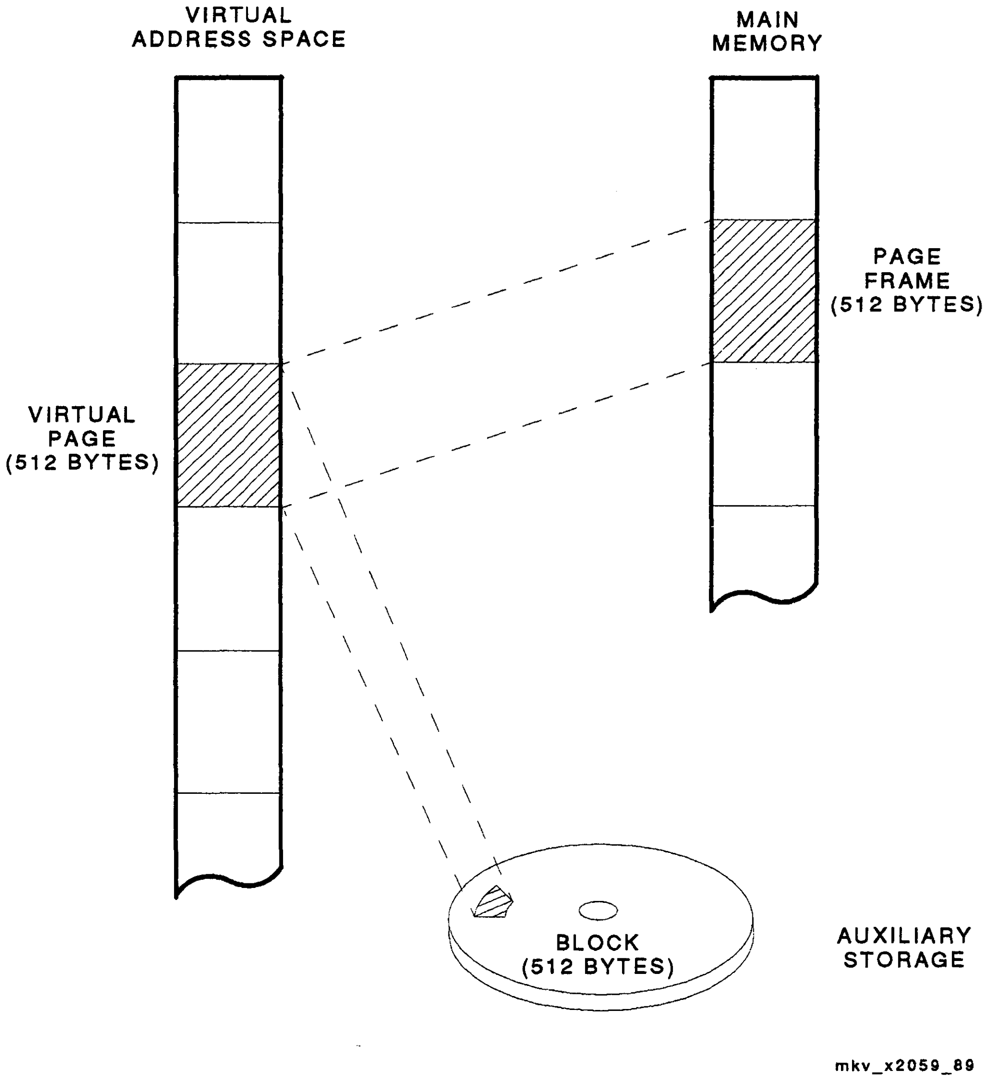

PAGE BOUNDARY
HARDWARE PCB
FIXED PORTION OF PROCESS HEADER
INDEXED TO WORKING SET LIST AND PROCESS SECTION TABLE
PAGE BOUNDARY
ACCOUNTING AND QUOTAS
WORKING SET LIST ENTRIES
PAGE BOUNDARY
PROCESS SECTION TABLES ENTRIES
VARIABLE PORTION OF PROCESS HEADER
n PAGES OF FREE SPACE (n IS EQUAL TO OR GREATER THAN 0)
PAGE BOUNDARY
P0 PAGE TABLE
P1 PAGE TABLE
PAGE BOUNDARY
MKV_X2058_89

PAGING, SWAPPING, AND SCHEDULING

Paging

- Paging is the action of making pages of an executing process available to the CPU's address-translation hardware; this is known as making the page "valid."
- The collection of currently valid pages for a particular process is known as that process' "working set."
- All pages of an executing process reside in virtual memory.
- Only the working set needs to reside in physical memory.
- Page faults (a hardware-detected condition) occur when a referenced page is not currently in the working set.
- In response to a page fault, the VMS pager brings pages into the working set from any of the following:
  - Image file on disk
  - Paging file on disk
  - Memory (that is, from the free or modified lists)
- The process can continue execution after the VMS Pager brings that page into the working set.
- The VMS pager maintains and manipulates the following databases:
  - Page frame number (PFN) database
  - Free page list
  - Modified page list
  - Working set lists
  - Page table entries

Figure 3–7: Organization of Virtual Address Space, Main Memory, and Auxiliary Storage



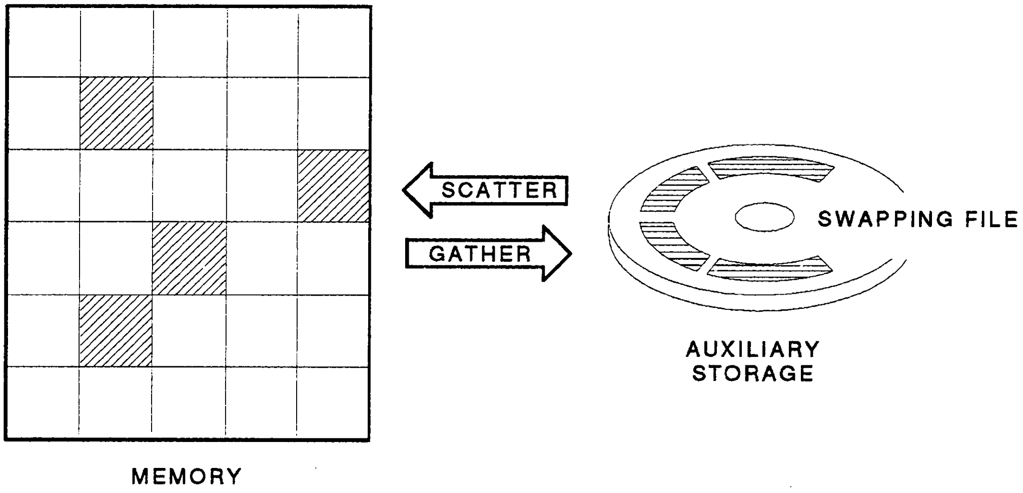
Swapping

- Memory use is controlled by swapping.
- Swapping occurs when a process is moved from memory into auxiliary storage while another process is brought into memory to be executed.
- Entire working sets are transferred between main memory and auxiliary storage to balance the available memory against the demands of user processes.
- Auxiliary storage contains swapping files that hold swapped-out processes.
- A process brought into memory is said to be swapped in.
- A process leaving memory is said to be swapped out.
- Swapping-out a process frees physical memory for another process.
- Swapping-in a process moves pages from auxiliary storage to physical memory.
- Swapping is accomplished by a swapper process.

- The swapper:
  - Is the process responsible for moving working sets between main memory and secondary storage.
  - Utilizes two conditions to determine which processes should be swapped in and which should be swapped out.

    Process priority
    Process status (which processes are executable and which are not)

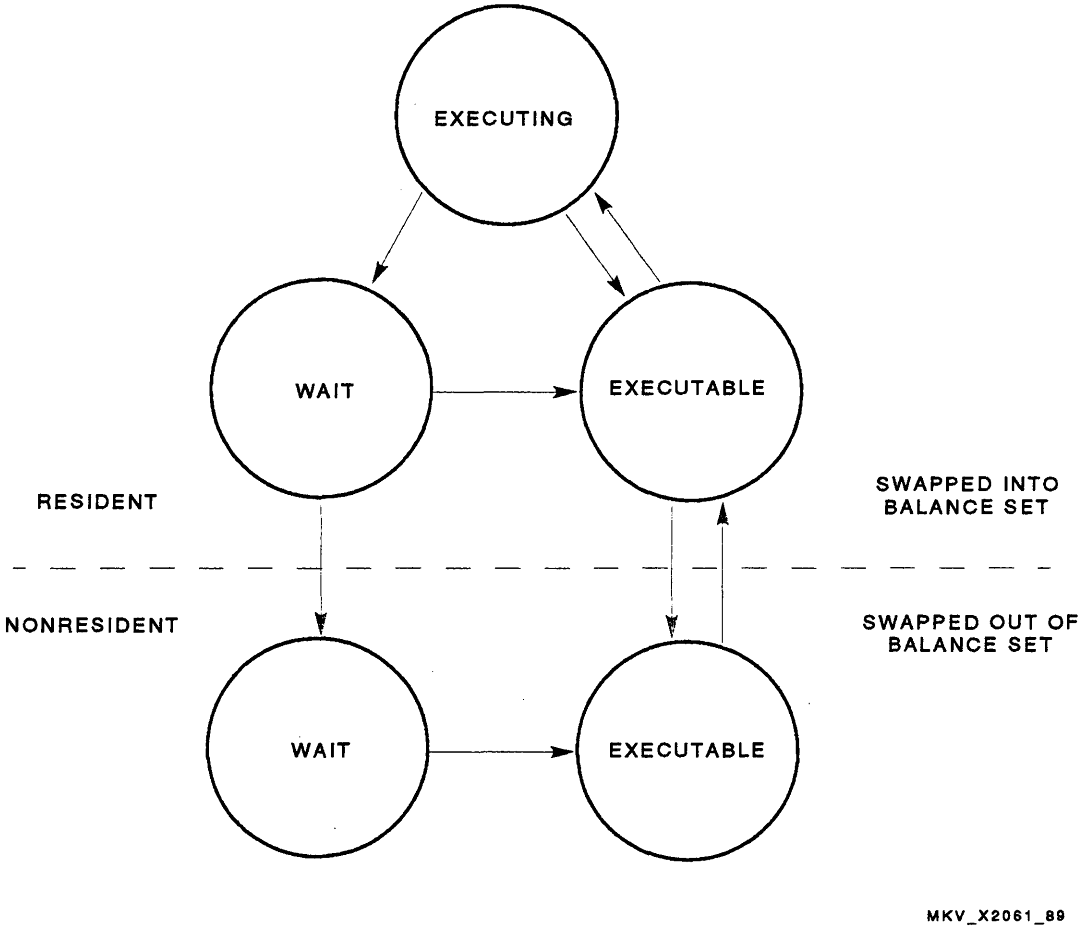
Figure 3–8: Scatter/Gather



MKV_X2060_89

- Swapping results in a 5-state model of a process that is either resident or nonresident (Figure 3-9).
- Swapping is necessary for two reasons:
  - — To replace lower priority or nonexecutable-resident processes with higher priority executable processes.
  - — To keep the scheduler well-supplied with executable processes on systems without enough main memory to contain all processes' working sets at once.
- On heavily loaded time-sharing systems, processes can be swapped out while waiting for input from a terminal user.

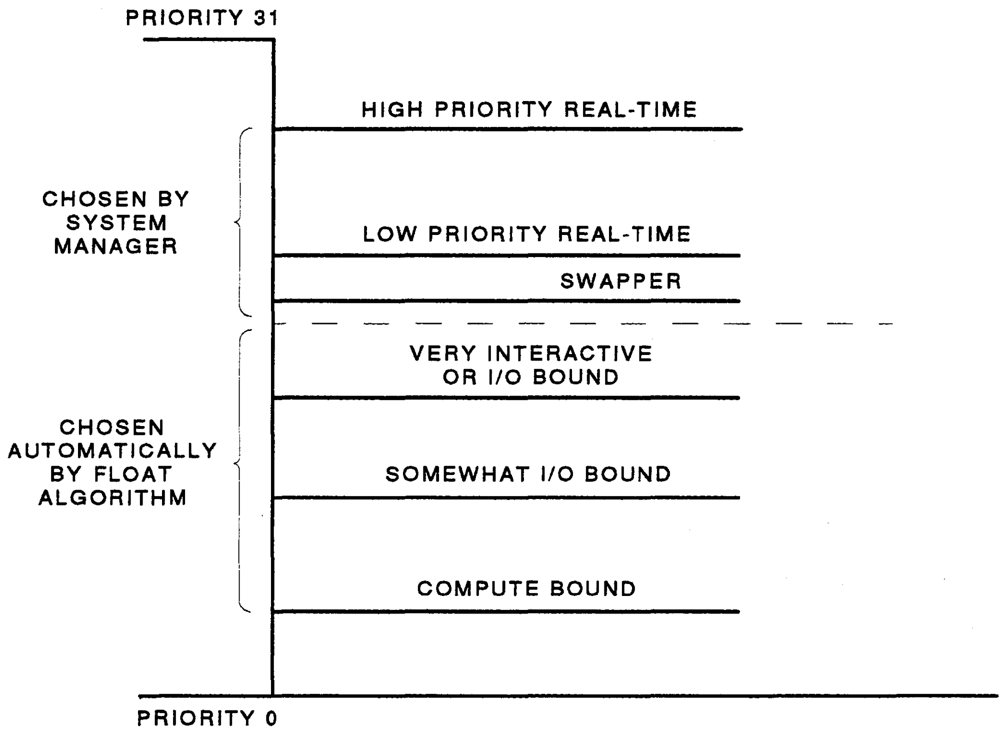
Figure 3–9: Swapping Between Process States



Scheduling

- Scheduling in VMS is priority-based.
- The highest priority executable process runs next, preempting the current process if the current process's priority is lower.
- Executable processes can be resident or nonresident.
- Executable processes exist at 1 of 32 priority levels. Half these priorities are for time-critical processes; the other half are for normal processes.
  - — Time-critical priority levels (16 - 31) are used for real-time applications that **cannot** be timed out.
  - — Normal priority levels (0 - 15) are used for interactive and batch applications that **can** be timed out.

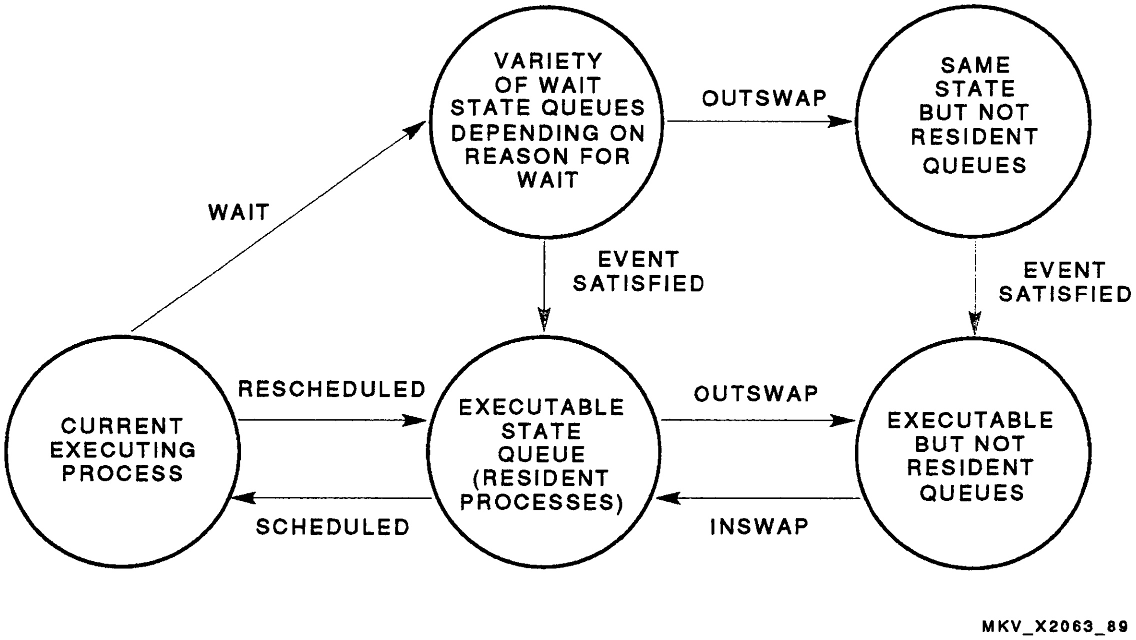
Figure 3–10: Process Scheduling



MKV_X2062_89

- Each process has a current priority and a base priority.
- The VMS executive dynamically changes the current priority in response to certain events within the process.
- Current priority levels are never less than the base priority levels.
- The highest priority process is selected to run.
- Pre-empted processes that are still executable are returned to the end of the queue.
- A process can be pre-empted many times before it has received its full quantum - the assigned execution time.
- Memory-resident processes are said to be in the balance set.
- A process remains in the balance set until one of the following occurs:
  - A nonresident, higher priority process requires service.
  - The process enters a wait state.
- High-priority, nonresident-executable processes can force a low-priority, resident-executable process out of memory.

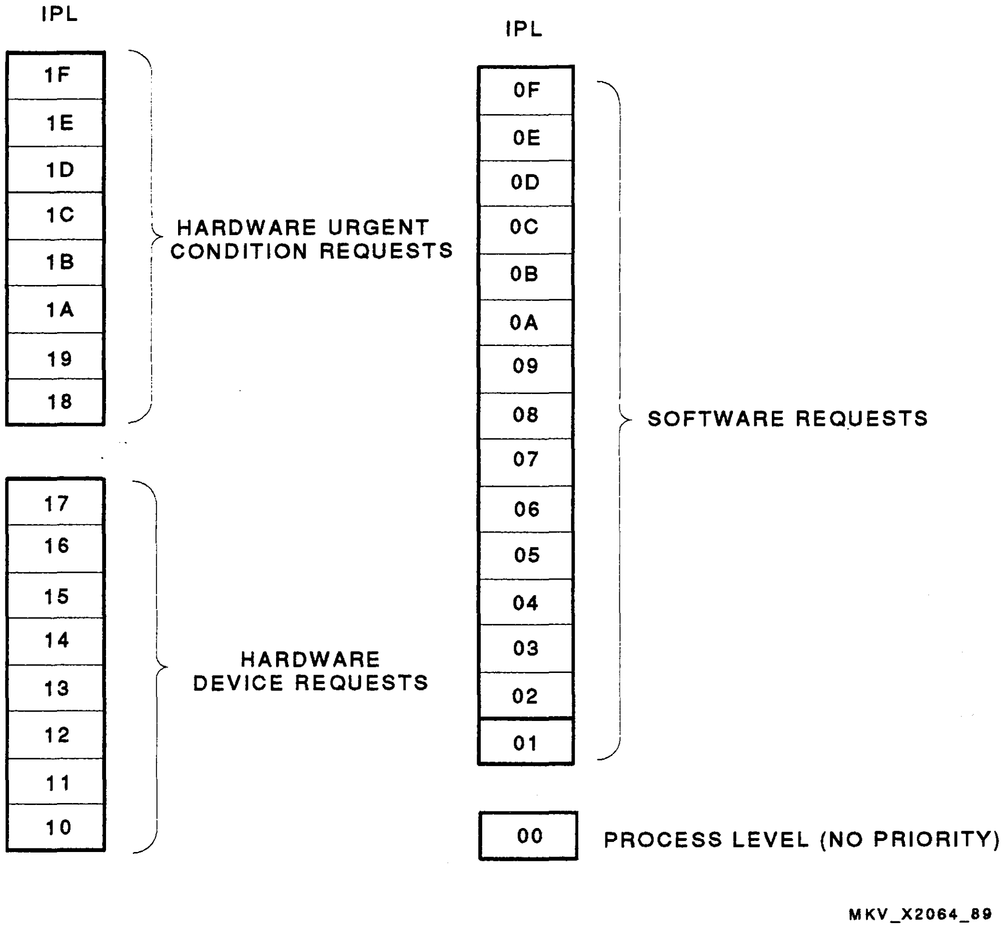
Figure 3–11: Process State Transition Cycle


VARIETY OF WAIT STATE QUEUES DEPENDING ON REASON FOR WAIT
OUTSWAP
SAME STATE BUT NOT RESIDENT QUEUES
WAIT
EVENT SATISFIED
EVENT SATISFIED
CURRENT EXECUTING PROCESS
RESCHEDULED
EXECUTABLE STATE QUEUE (RESIDENT PROCESSES)
OUTSWAP
EXECUTABLE BUT NOT RESIDENT QUEUES
SCHEDULED
INSWAP
MKV_X2063_89

Scheduling and Interrupt Priority Levels (IPL)

- Two types of events can influence the execution of a process:
  - — Synchronous events (exception)
  - — Asynchronous events (interrupt)
- The CPU is notified when an exception or interrupt event occurs.
- An exception is serviced in the context of the process that produced the exception condition.
- An interrupt is serviced independently from the currently running process.
- There are 31 Interrupt Priority Levels (IPLs) that assist in servicing multiple and simultaneous interrupts.
- IPLs are divided into two major categories:
  - — Software IPLs, for invoking various system routines in appropriate order.
  - — Hardware IPLs, for device and clock interrupts, and power failures.

Figure 3–12: Interrupt Priority Levels



DYNAMIC MEMORY (POOL)

- Pool areas are located in system space (S0).
- Pool areas provide blocks of memory that can be allocated when needed and deallocated when no longer needed.
- There are two types of pool:
  - Paged pool
  - Nonpaged pool
- Paged pool is used for data structures that are not required to be memory-resident. Therefore, they can be paged. For example:
  - Global section descriptors
  - Group and system logical name tables
- Nonpaged pool is used for data structures that are time-critical or for other reasons must remain memory-resident. For example:
  - Process control blocks
  - Device driver data structures
- Size of pool is controlled by SYSGEN parameters, which are set initially by AUTOGEN.
- Pool is a crucial system resource.
  - Too much pool can waste memory.
  - Too little pool can cause degradation of performance.
  - Exhaustion of pool causes the system to hang.

Non-paged dynamic pool (formal name)
area where VMS keeps critical data structure

like: PCB

TQE

I/O request packets etc.

SYSGEN PARAMETERS AND USER QUOTAS

- The System Generation utility (SYSGEN) allows the system manager to:
  - Create and modify system parameter files.
  - Dynamically modify some current system parameter settings.
  - Create swap, page, and dump files.
- Much of the memory management behavior of VMS can be modified by the settings of SYSGEN parameters (Table 3-1).
  - Size of the modified and free lists
  - Size of various pool areas
  - Maximum number of memory-resident processes
- Generally, it is best to let AUTOGEN with feedback choose the actual parameter settings.

Table 3–1: Various SYSGEN Parameters

Parameters	Definition
MAXPROCESSCNT	Maximum number of process
BALSETCNT	Maximum number of memory-resident processes
BORROWLIM	Free list size necessary to allow processes to grow past WSQUOTA at working-set adjustment time
GROWLIM	Free list size necessary to allow processes to grow past WSQUOTA at page-fault time
FREELIM	Lower limit of the free page list
FREEGOAL	Target free list length
MPW_WAITLIMIT	Modified page writer busy wait limit
MPW_HILIMIT	Modified page list maximum limit
MPW_LOWAITLIMIT	Modified page writer busy threshold to allow processes to continue
MPW_THRESH	Modified page writer threshold
MPW_LOLIMIT	Modified page list low limit
WSMAX	Maximum size of process working set
WSINC	Working set increment
WSDEC	Working set decrement
SYSMWCNT	System working set count
PAGFILCNT	Maximum number of paging files on system
SWPFILCNT	Maximum number of swapping files on system
PHYSICALPAGES	Maximum number of physical pages to be used
VIRTUALPAGECNT	Maximum virtual page count

VMS LOCK MANAGER CONCEPTS

- VMS lock manager can be used by cooperating processes to synchronize their access to shared resources.
- Processes lock a resource by specifying its name and the kind of access (locking mode) they wish to have to the resource (Table 3-2).
- If a lock cannot be granted (due to other locks already on the resource), the request can be queued and granted at a later time.
- The lock manager internal bookkeeping system uses four data structures:
  - Lock blocks (LKBs) that describe the locks requested by processes
  - Resource blocks (RSBs) that describe the various resource names known to the lock manager
  - Lock ID table that locates the lock blocks
  - Resource hash table that locates the resource blocks

Figure 3–13: Synchronizing Access to a Resource Using the Lock Manager

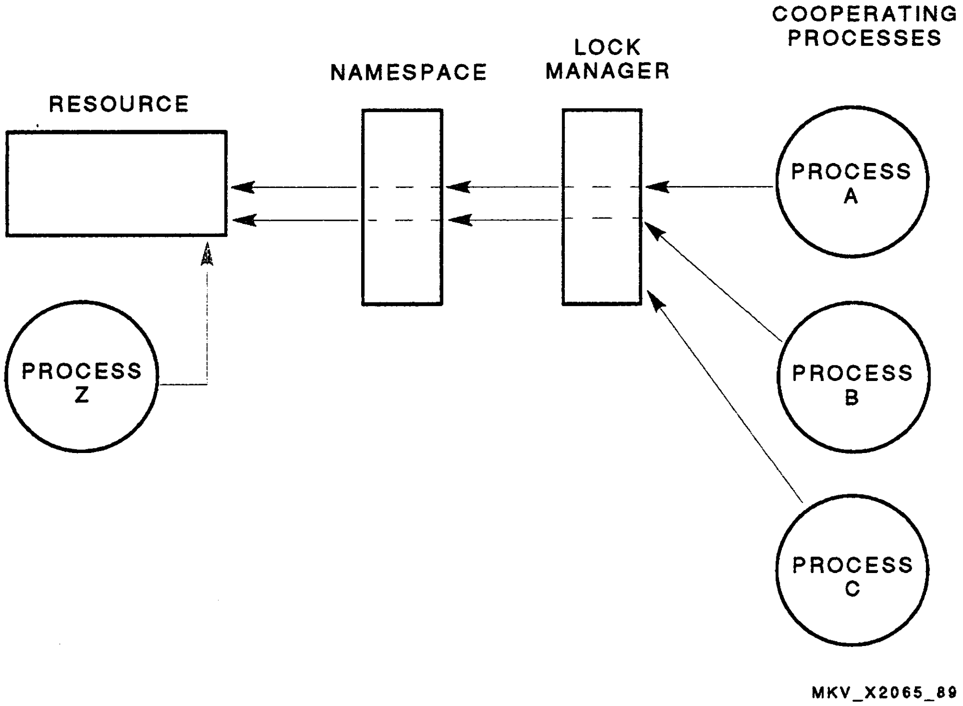


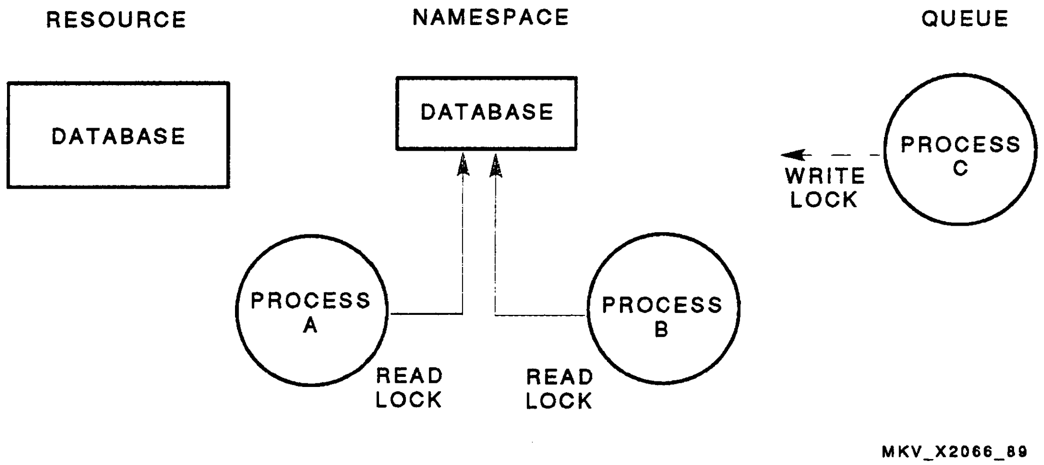
Table 3–2: Lock Modes

Mode	Meaning
NL	Null lock. Can neither read nor write.
CR	Concurrent read. Read access and sharing with other readers and writers.
CW	Concurrent write. Write access and sharing with other readers and writers.
PR	Protected read. Read access and sharing with other readers - no write allowed.
PW	Protected write. Write access and sharing with CR mode readers - no other writers.
EX	Exclusive access. Read or write access - denies access to any other readers or writers.

Lock Manager

- To use the lock manager, cooperating processes must:
  - Agree on a name for the shared resource.
  - Refer to that resource by its common name.
  - Always ask the lock manager's permission before accessing the resource.
- Many other parts of VMS use the lock manager to arbitrate access to shared resources:
  - Files-11 for controlling access to files and cache buffers
  - RMS for controlling access to records and data buckets

Figure 3–14: Creating Resource Locks



SUMMARY

This module has examined the following topics:

- Memory management
- Paging
- Swapping
- Scheduling
- Dynamic memory (pool)
- SYSGEN parameters
- Lock manager

MODULE 4
VMS DISK FILE STRUCTURE

INTRODUCTION

Most computer applications require storage, retrieval, and transfer of some type of data, while allowing users to share large amounts of information. Computer users are often so dependent on their data files that when those files become unavailable, human productivity either declines or comes to a halt.

The VMS system manager must protect the integrity of the data files maintained on the system. Furthermore, the system manager must effectively manage system resources that affect file-related activities. To perform these functions, you must understand disk structure and where it fits into the overall VMS disk I/O system.

This module describes Files-11 on-disk structure and provides information about managing disk file resources.

OBJECTIVES

To make effective use of disk resources and to troubleshoot problems related to disks and files, you should be able to:

- Recognize the components of the VMS disk I/O system.
- Describe both physical and software disk concepts.
- Describe disk software characteristics, and use the appropriate utilities and commands to create and display disk software characteristics.
- Describe the VMS ODS-2 disk file structure and recognize ODS-2 reserved files.
- Describe ODS-2 file characteristics, including file headers and directory structure.
- Use the appropriate utilities and commands to display disk file information and restore file contiguity.
- Describe the system disk directory structure.
- Describe the mount verification process, and how to respond to it.

- Use the appropriate utilities and commands to verify the validity of disk volumes and to recover files from corrupt directories.

- Describe the volume rebuild process and use the appropriate utilities and commands to enable, disable, or invoke volume rebuild.

- Describe the disk quota rebuild process and identify situations when you would invoke disk quota rebuild.

RESOURCES

- *Guide to VMS File Applications*
- *VMS Guide to Disk and Magnetic Tape Operations*
- *VMS DCL Dictionary*
- *VMS Backup Utility Manual*
- *Guide to Setting Up a VMS System*
- *Guide to Maintaining a VMS System*
- *VMS Analyze /Disk Structure Utility Manual*
- *VMS Mount Utility Manual*
- *VMS SYSMAN Utility Manual*
- *VMS System Messages and Recovery Procedures Reference*
- *VMS System Generation Utility Manual*

TOPICS

1. Overview of the VMS disk I/O system

   Components of the VMS disk I/O system

2. Disk concepts

   a. Physical disk characteristics

   b. Software blocks

   c. I/O concepts

3. Disk software characteristics

   a. INITIALIZE

   b. SHOW DEVICE /FULL

4. Disk file structure on VMS

   a. Files-11 on-disk structure level 2

   b. ODS-2 reserved files

   c. INDEXF.SYS file

5. ODS-2 file characteristics

   a. ODS-2 file headers

   b. ODS-2 directories

   c. Restoring file contiguity

6. Directory structure on a common system disk

7. Mount verification

8. Verifying validity of disk volumes

   a. ANALYZE/DISK_STRUCTURE utility

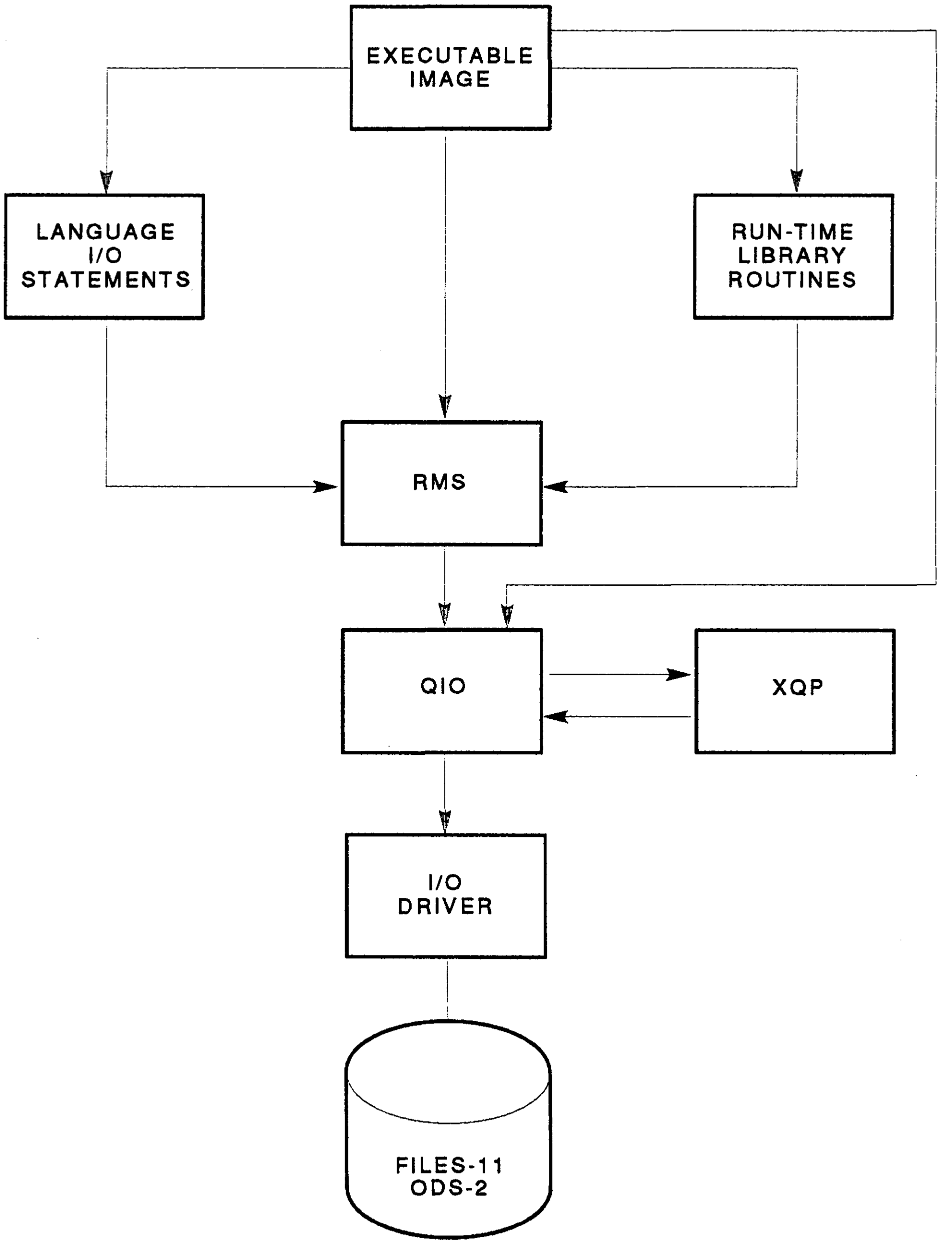
   b. Recovering files from a corrupt directory

9. Volume rebuild

10. Disk quota rebuild

OVERVIEW OF THE VMS DISK I/O SYSTEM

Figure 4–1: The VMS Disk I/O System



MKV_X2067_89

Components of the VMS Disk I/O System

- Executable images can be invoked by DCL commands, for example:
  - System-supplied programs, such as utilities
  - User-written or third-party programs
- Executable images perform virtual I/O
- I/O statements and run-time library routines
  - Are called by programs and DCL
  - Allow you to open, close, read from, and write to files
- Record management services
  - Are operating system procedures used to open, close, read from, and write to files
  - Can be called by:

    RMS utilities
    I/O statements and RTL routines
    Programs and DCL
  - Support various file organizations and access methods
- QIO (queue I/O request) system services
  - Are operating system procedures that prepare I/O requests for processing by the device driver or XQP
  - Can be called by:

    RMS services
    Programs

- XQP (extended QIO processor)
  - — An operating system procedure that performs I/O functions to Files-11 ODS-2 disks
  - — Supplements the driver's functions
  - — Performs functions such as:

    File creation
    File deletion
    File extension

- I/O drivers
  - — Operating system procedures and data structures
  - — Perform I/O to device controllers
- Files-11 ODS-2
  - — VMS default disk file structure
  - — Created when you initialize a disk

DISK CONCEPTS

Physical Disk Characteristics

- Tracks
- Cylinders
- ~~Disk blocks~~ SECTORS (which contain disk blocks)

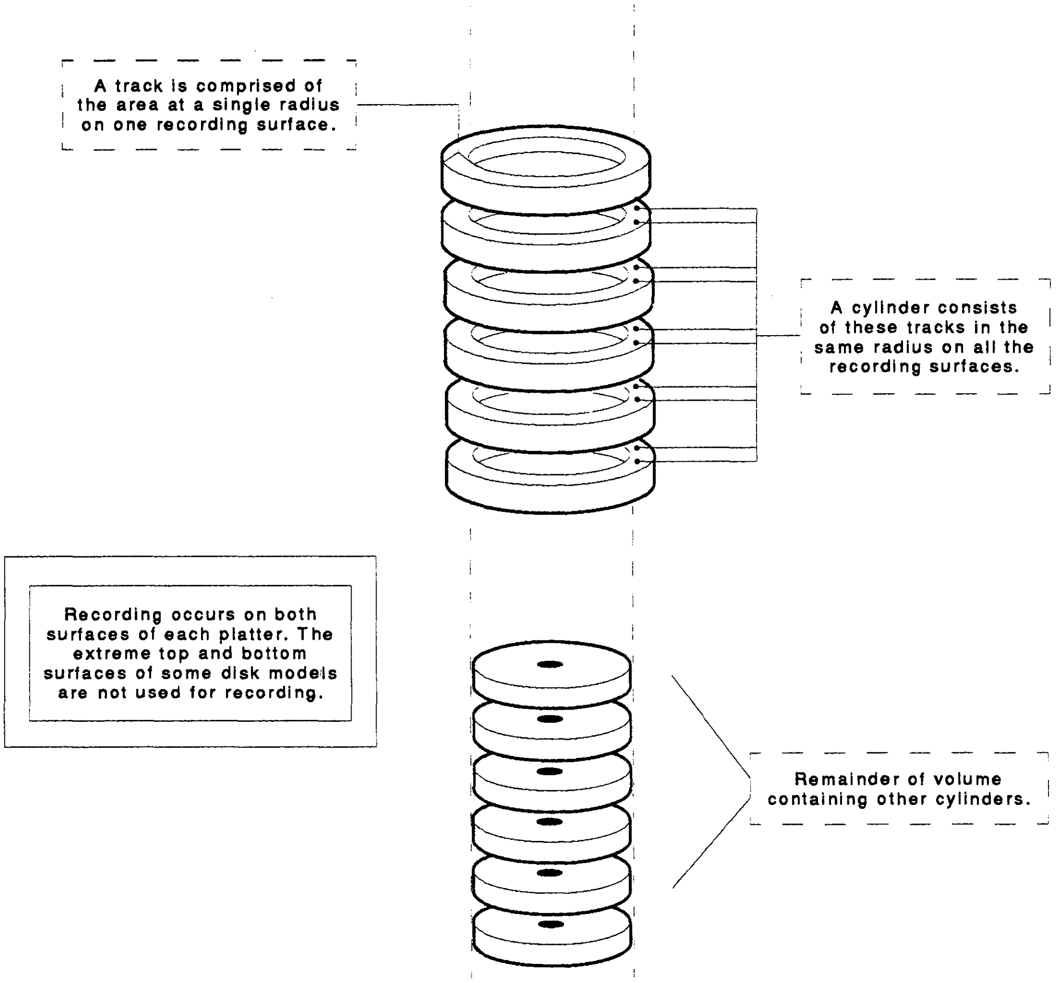
Software Blocks

- Logical blocks
- Virtual blocks

I/O Concepts

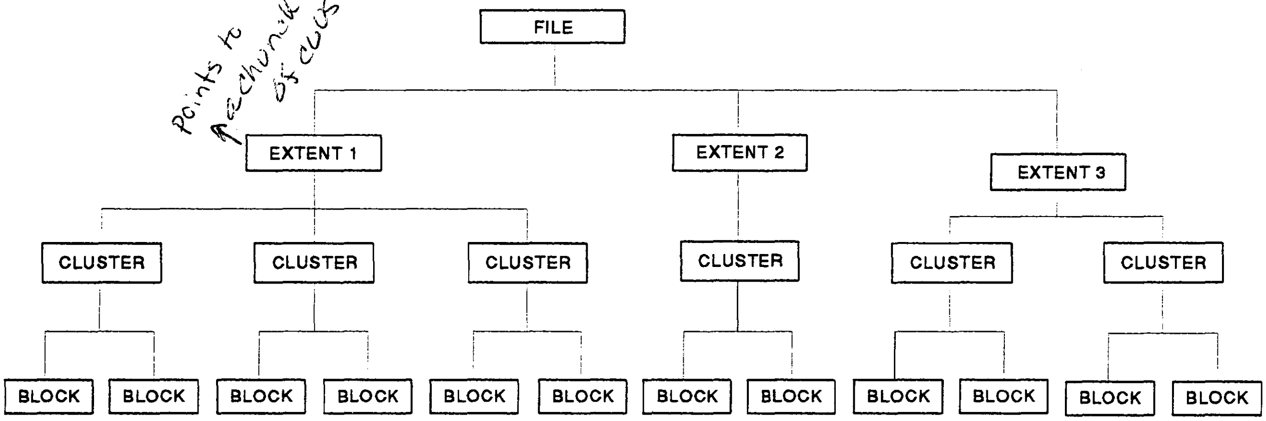
- Virtual I/O is performed by executable images and RMS.
- Logical I/O is performed by QIO.
- Physical I/O is performed by device driver.

Figure 4–2: Disk Physical Characteristics


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

MKV_X2068_89

Figure 4–3: Files-11 On-Disk Structure Hierarchy



Points to each chunk of CLUSTERS

MKV_X2069_89

- Disk structure is composed of logical disk **blocks** - each block is 512 bytes.
- Each logical disk block is a virtual block in the file.
- Blocks are grouped into **clusters.**
  - — Disk space is allocated in units of clusters.
  - — Cluster size can vary.

    Small clusters use disk space efficiently but create more overhead.
    Large clusters can waste disk space but cost less in system overhead.

  - — Cluster size should be a multiple/fraction of track size. - MAYBE NOT!
  - — Cluster size is defined at disk initialization:

    ```
    $ INITIALIZE /CLUSTER_SIZE
    ```

  - — Refer to Table 4-1 for INITIALIZE command qualifiers.
  - — To see disk cluster size, type:

    ```
    $ SHOW DEVICE /FULL
    ```

- Contiguous clusters allocated to a file are extent.
  - — A file can have one or more extents.
  - — Addition of extents over time can cause a file to become noncontiguous.

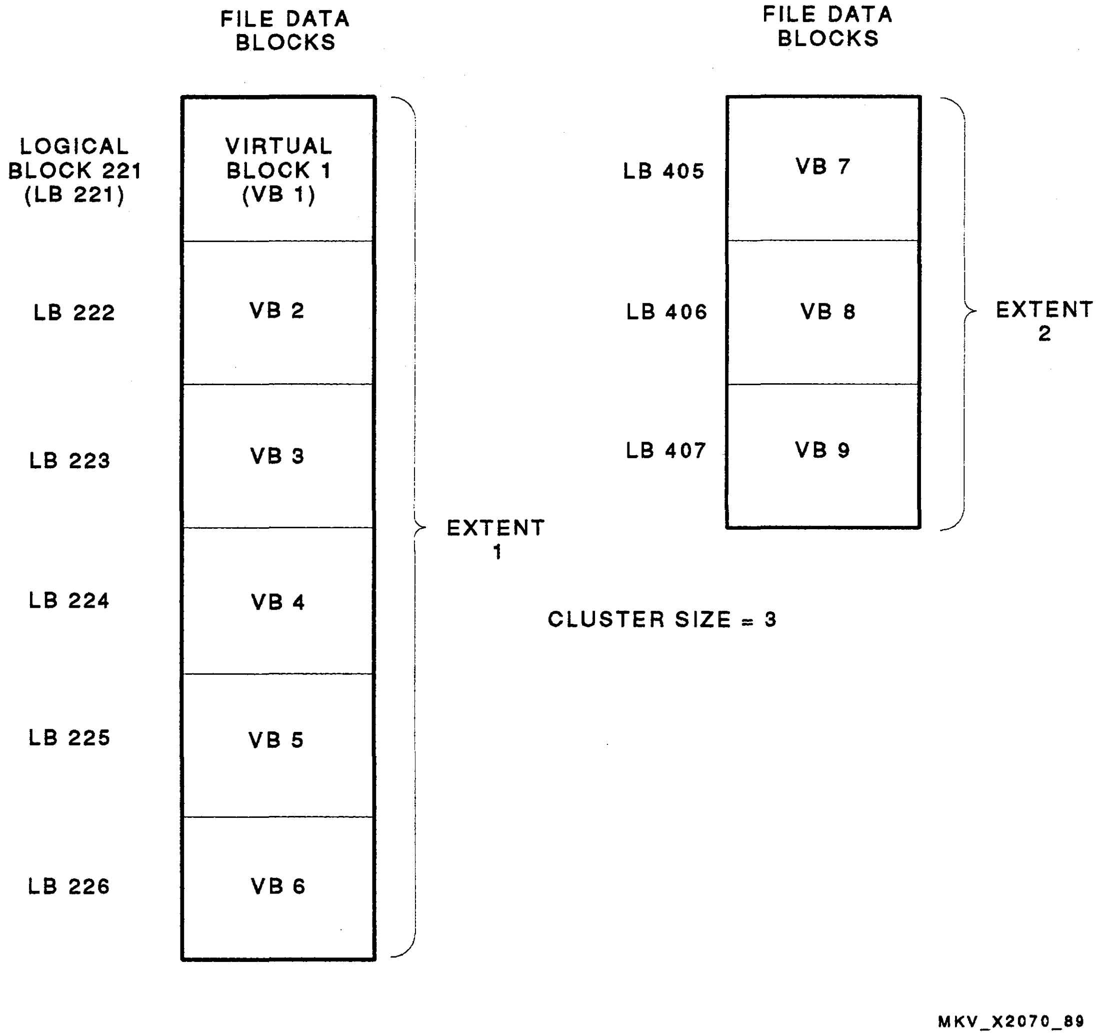
FOR EXAMPLE TRACKS ~~are~~ CAN BE 51 blocks!
only good cluster sizes
1,3,17,51 according to this
philosophy but the
hardware guys have
made it feasible to
span track boundries
which allows for different
cluster sizes!!!
with a cluster size of 5
you will only span
1 in 11 times
anyway!

$ Show device/full
↓
tells your
track size
in blocks.

Table 4–1: Important INITIALIZE Command Qualifiers

Qualifier	Function
/CLUSTER_SIZE	Defines minimum allocation size, in blocks
/DIRECTORIES	Specifies the number of entries to preallocate for user directories
/HEADERS	Specifies the number of file headers to be allocated initially for the index file
/INDEX	Requests that the index file for the volume's directory structure be placed in a specific disk location
/MAXIMUM_FILES	Restricts the maximum number of files the volume can contain
/WINDOWS	Specifies the number of mapping pointers to be allocated for file windows

Figure 4–4: Relationships Among Blocks, Clusters, and Extents



Example 4–1: The SHOW DEVICE/FULL Command

```
$ SHOW DEVICE/FULL WORK1

Disk DEMON$DUA1:, device type RA81, is online, mounted, file-oriented
    device, shareable, available to cluster, error logging is enabled.

Error count                    0  Operations completed              174075
Owner process                 ""  Owner UIC                          [1,1]
Owner process ID        00000000  Dev Prot     S:RWED,O:RWED,G:RWED,W:RWED
Reference count                8  Default buffer size                  512
Total blocks              891072  Sectors per track                     51
Total cylinders             1248  Tracks per cylinder                   14
Host name                "DEMON"  Host type, available           HS50, yes

Volume label              "FROG"  Relative volume number                 0
Cluster size                   3  Transaction count                      8
Free blocks               260142  Maximum files allowed             111384
Extend quantity                5  Mount count                            2
Mount status              System  Cache name         "_HARDY$DRA0:XQPCACHE"
Extent cache size             64  Maximum blocks in extent cache     26014
File ID cache size            64  Blocks currently in extent cache    2340
Quota cache size              90  Maximum buffers in FCP cache         348

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

DISK FILE STRUCTURE ON VMS

Files-11 On-Disk Structure Level 2

- Is the default disk file structure.
- Also called ODS-2.
- Used to maintain and control data on disk volumes.
- ODS-2 is created when you initialize a disk with the INITIALIZE command.

ODS-2 Reserved Files

- Define the Files-11 disk file structure.
- Are created when a volume is initialized.
- Are cataloged in the volume's master file directory (MFD).

  Master File Directory is [000000].

Table 4–2: Volume Information Contained in ODS-2 Reserved Files

Reserved File	Information
Index File (INDEXF.SYS)	Bootstrap block, home block, backup home block, backup index file header, index file bit map, file headers.
Storage Bit Map File (BITMAP.SYS)	Location of available clusters on a volume. The size of the bit map is determined by the disk cluster size.
Bad Block File (BADBLK.SYS)	Areas of volume not suitable for use (not used for DSA disks).
Pending Bad Block Log File (BADLOG.SYS)	Areas of volume suspected of being unusable (not used for DSA disks).
Master File Directory (000000.DIR)	Pointers to all User File Directories and reserved files.
Core Image File (CORIMG.SYS)	Not used by VMS. Provided to preserve structure of previous versions.
Volume Set List File (VOLSET.SYS)	Labels of other volumes in the volume set (if more than one).
Continuation File (CONTIN.SYS.)	The extension file identifier, if a file crosses from one volume to another in a loosely coupled volume set.
Backup Log File (BACKUP.SYS)	History of backups on the volume (not used).

Index File - INDEXF.SYS

- Bootstrap block
  - Physically, the first block on the volume.
  - Present on all Files-11 volumes.
  - Contains a bootstrap program on operating system volumes.
  - On other volumes, it contains a program that displays messages indicating that it is not a system volume.
- Home block
  - Essential to accessing any file on the volume.
  - Several backup copies of home block exist on all volumes.
  - Home block contains:

    Volume name
    Pointers to remainder of index file
    Maximum number of files allowed on this volume
    UIC of volume owner
    Volume protection code
- File headers
  - Describe part of one file on the volume.
  - Contain a list of extents that make up the file.
  - Have an identifying number.

ODS-2 FILE CHARACTERISTICS

Major Components

- An ODS-2 file header
- One or more data blocks

ODS-2 File Headers

- Part of the volume INDEXF.SYS
- Not part of the file it describes
- Locate file data blocks on the disk

File headers are divided into six areas:

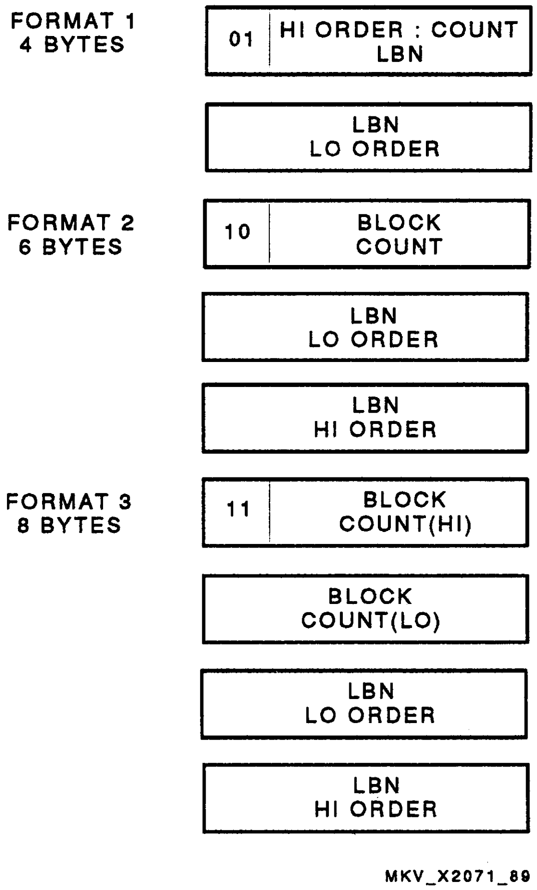
1. Header Area - contains basic information for checking access validity.
2. Ident Area - contains identification and accounting information.
3. Map Area - contains retrieval pointers to the blocks allocated to the file.
   a. Describes the physical location of the file.
   b. Files with many extents have a multiple block file header.
4. Access Area - contains access control list entries if any are defined.

   Files with many ACLs have a multiblock file header.
5. Reserved area - for use by customers and Digital Equipment Corporation.
6. Checksum area - validity check on the header's contents (the last word of the header).

To display contents of a file header:

```
$ DUMP /HEADER filename
```

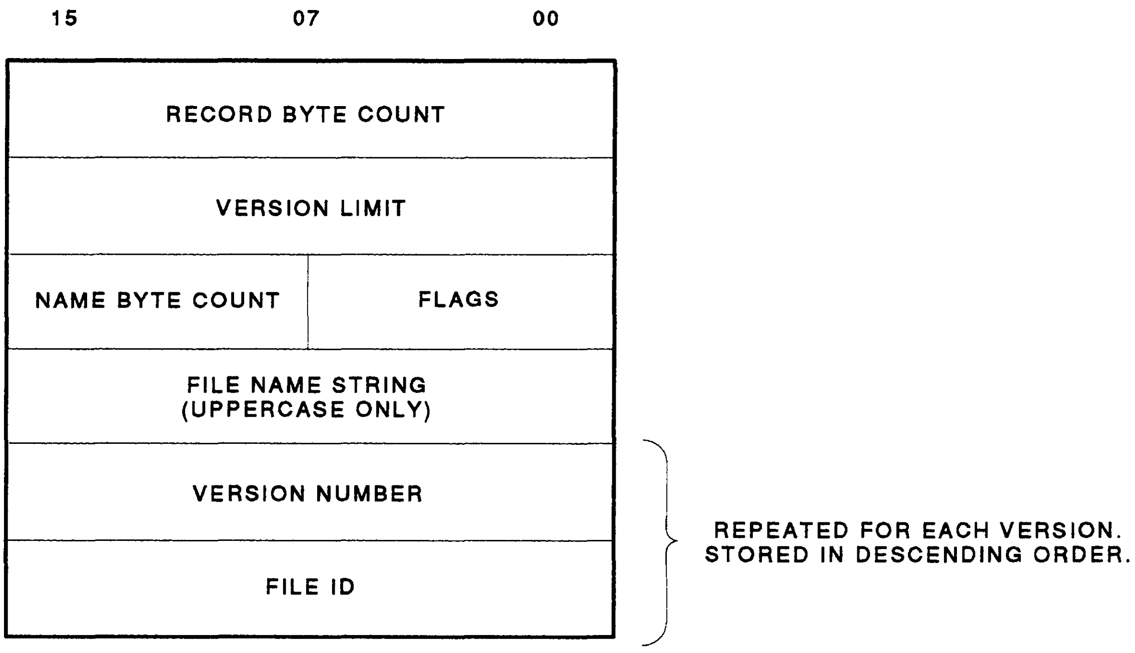
Figure 4–5: File Retrieval Pointers



ODS-2 Directories

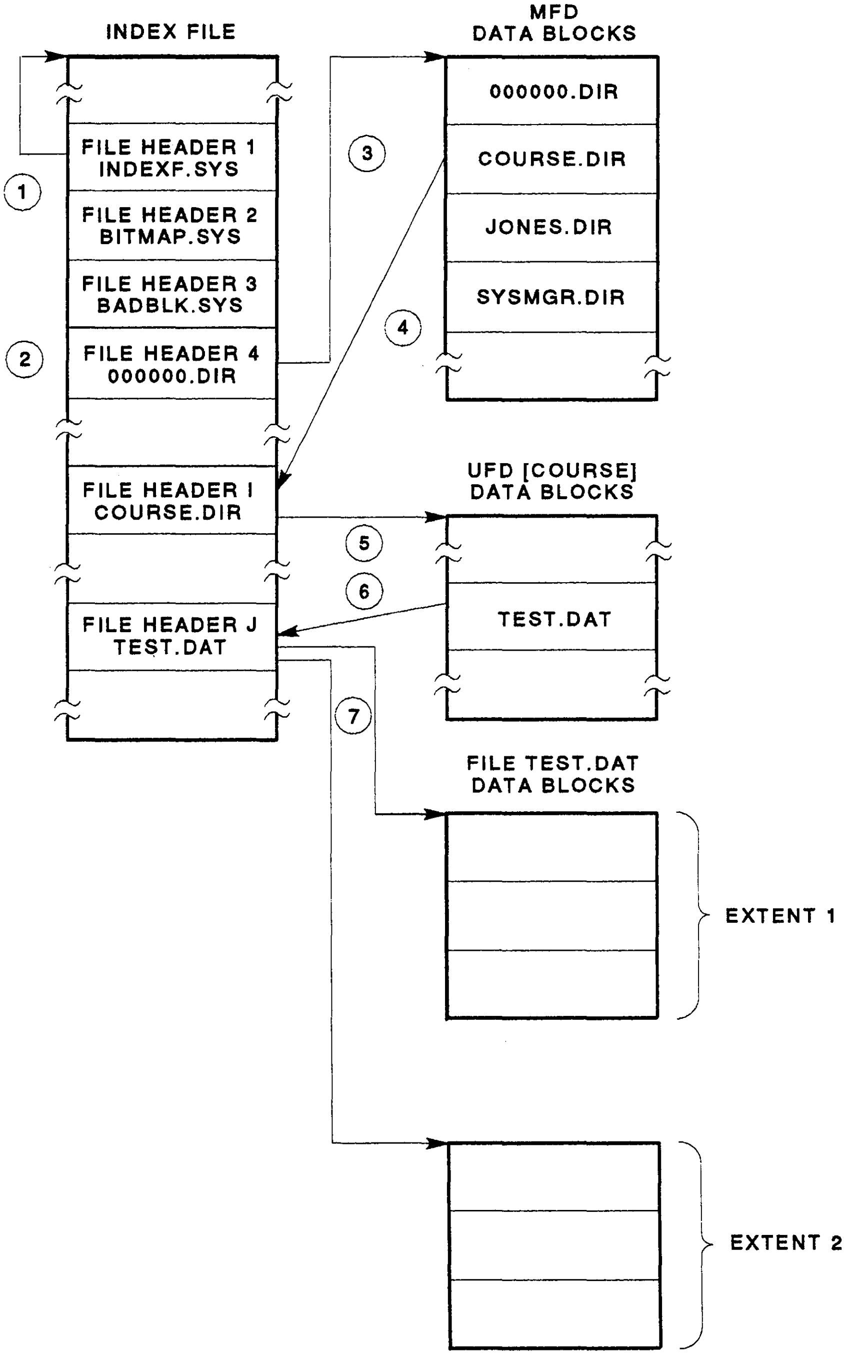
- Directories associate symbolic file name with file ID.
- User file directories (UFDs) are entered in the MFD.
- Subfile directories (SFDs) are entered in their parent directory file.
- Directory and subfile directory names are the same format as VMS file names.

Figure 4–6: Directory Record Structure



MKV_X2072_89

Figure 4–7: Using ODS-2 File Headers to Access a File



MKV_X2073_89

Example 4–2: The DUMP/HEADER Command

```
$ DUMP /HEADER /BLOCKS=COUNT=0  SPLASH_VPA.RPT
Dump of file INS:[SHERRY.SM2]SPLASH_VPA.RPT;7 on 30-DEC-1988 14:39:14.74
File ID (7836,13,0)   End of file block 89 / Allocated 90

                            File Header
Header area
    Identification area offset:           40
    Map area offset:                      100
    Access control area offset:           255
    Reserved area offset:                 255
    Extension segment number:             0
    Structure level and version:          2, 1
    File identification:                  (7836,13,0)
    Extension file identification:        (0,0,0)
    VAX-11 RMS attributes
        Record type:                      Variable
        File organization:                Sequential
        Record attributes:                Implied carriage control
        Record size:                      80
        Highest block:                    90
        End of file block:                89
        End of file byte:                 196
        Bucket size:                      0
        Fixed control area size:          0
        Maximum record size:              0
        Default extension size:           0
        Global buffer count:              0
        Directory version limit:          0
    File characteristics:                 <none specified>
    Map area words in use:                4
    Access mode:                          0
    File owner UIC:                       [INSTRUCT,SHERRY]
    File protection:                      S:RWED, O:RWED, G:RE, W:
    Back link file identification:        (3422,3,0)
    Journal control flags:                <none specified>
    Active recovery units:                None
    Highest block written:                90
Identification area
    File name:                            SPLASH_VPA.RPT;7
    Revision number:                      2
    Creation date:                        28-SEP-1988 12:47:48.85
    Revision date:                        30-DEC-1988 14:38:27.68
    Expiration date:                      <none specified>
    Backup date:                          <none specified>
Map area
    Retrieval pointers
        Count:         3        LBN:      590115 } file in 2 frags! one 3 blocks
        Count:        87        LBN:      590151 }                  one 87 blocks

Checksum:                                 34487
```

Example 4–3: The DUMP Command

```
$ DUMP SPLASH_VPA.RPT

Dump of file INS:[SHERRY.SM2]SPLASH_VPA.RPT;7 on 5-JAN-1989
File ID (7836,13,0)   End of file block 89 / Allocated 90

Virtual block number 1 (00000001), 512 (0200) bytes

796C616E 4120322E 31562041 5056004E N.VPA V1.2 Analy 000000
20202020 20202020 20202020 20736973 sis              000010
33382058 41562820 4B434155 51202020    QUACK (VAX 83 000020
20202020 20202020 20202020 20293030 00)              000030
31204547 41502020 20202020 20202020           PAGE 1 000040
726F7065 52202020 20202020 20200033 3.         Repor 000050
32202020 20202020 20202020 20202074 t              2 000060
206F7420 30303A30 30203838 50455337 7SEP88 00:00 to  000070
00000000 000D0001 00000039 353A3332 23:59........... 000080
20202020 20202020 2020002D 00000000 ....-.           000090
20202020 20202020 20202020 20202020                  0000A0
4D4D5553 20534953 594C414E 41202020    ANALYSIS SUMM 0000B0
20202020 20202020 002C0000 00595241 ARY...,.         0000C0
20202020 20202020 20202020 20202020                  0000D0
51206564 6F6E2072 6F662020 20202020       for node Q 0000E0
2020003F 00000000 00000000 4B434155 UACK........?.   0000F0
6F207265 626D754E 20202020 20202020         Number o 000100
7365636F 72502073 64726F63 65522066 f Records Proces 000110
2E2E2E2E 2E2E2E2E 2E2E2E2E 2E646573 sed............. 000120
003D0033 36352E2E 2E2E2E2E 2E2E2E2E ..........563.=. 000130
7265626D 754E2020 20202020 20202020           Number 000140
69746173 20736472 6F636552 20666F20  of Records sati 000150
646E6F63 20656C75 7220676E 69796673 sfying rule cond 000160
003F0036 2E2E2E2E 2E2E736E 6F697469 itions......6.?. 000170
7265626D 754E2020 20202020 20202020           Number 000180
20746F6E 20736472 6F636552 20666F20  of Records not  000190
20656C75 7220676E 69796673 69746173 satisfying rule  0001A0
00373535 2E2E736E 6F697469 646E6F63 conditions..557. 0001B0
626D754E 20202020 20202020 2020003D =.          Numb 0001C0
6E6F6973 756C636E 6F432066 6F207265 er of Conclusion 0001D0
2E2E2E2E 2E2E2E2E 2E2E2E2E 2E2E2E73 s............... 0001E0
00302E2E 2E2E2E2E 2E2E2E2E 2E2E2E2E ..............0. 0001F0

Dump of file INS:[SHERRY.SM2]SPLASH_VPA.RPT;7 on 5-JAN-1989
File ID (7836,13,0)   End of file block 89 / Allocated 90

Virtual block number 2 (00000002), 512 (0200) bytes

6E412032 2E315620 4150560C 004E0000 ..N..VPA V1.2 An 000000
20202020 20202020 20207369 73796C61 alysis           000010
   .
   .
   .
```

Example 4–4: The DIRECTORY/FULL Command

```
$ DIRECTORY /FULL  SPLASH_VPA.RPT

Directory INS:[SHERRY.SM2]

SPLASH_VPA.RPT;7              File ID:  (7836,13,0)
Size:           89/90         Owner:    [INSTRUCT,SHERRY]
Created:  28-SEP-1988 12:47:48.85
Revised:  30-DEC-1988 14:38:27.68 (2)
Expires:   <None specified>
Backup:    <No backup recorded>
File organization:  Sequential
File attributes:    Allocation: 90, Extend: 0,
      Global buffer count: 0, No version limit
Record format:      Variable length, maximum 80 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
```

Table 4–3: SET FILE Command Qualifiers for Changing File Attributes

SET FILE Command Qualifier	Function
/ACL	Modifies the access control list associated with a file.
/BACKUP	Determines whether the data in a file will be copied by the Backup utility.
/DATA_CHECK	Specifies reliability options (read check and/or write check).
/ENTER=newfilespec	Assigns an additional name to a file (alias). Use of this qualifier is discouraged.
/ERASE_ON_DELETE	Ensures erasure of confidential data when a file is deleted.
/EXPIRATION_DATE	Changes file expiration date (if any).
/GLOBAL_BUFFER	Specifies the number of global buffers that can be shared by processes accessing the file.
/NODIRECTORY	Removes the directory attributes of a file, allowing you to delete a directory.
/OWNER_UIC	Changes the file ownership.
/PROTECTION	Changes the protection code of a file.
/REMOVE	Removes one name from a file that has more than one name. Use of this qualifier is discouraged.
/TRUNCATE	Truncates a sequential file.
/VERSION_LIMIT	Changes the number of versions retained.

Restoring File Contiguity

- Many extents can make files noncontiguous.
- Use of BACKUP /IMAGE to restore contiguity:
  - — Processes the whole volume or volume set.
  - — Creates a functionally equivalent copy of the volume.
  - — Reorganizes file extents to make files contiguous.

GOOD Example!!

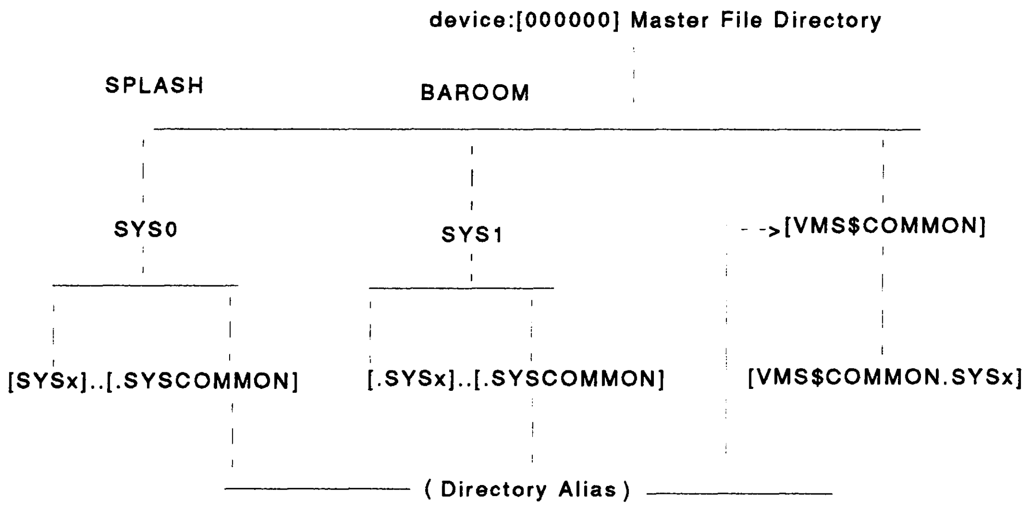
Example 4–5: Restoring File Contiguity

```
$ DUMP /HEADER /BLOCKS=END:0 DYA0:[SMITH]FILE.DAT
Dump of file DYA0:[SMITH]FILE.DAT;1 on 16-APR-1989 12:59:51.24
File ID (12,4,0)    End of file block 197 / Allocated 197

                    File Header
                         .
                         .
                         .
Map area
    Retrieval pointers
        Count:        12        LBN:        101
        Count:         3        LBN:        388
        Count:         1        LBN:        481
        Count:         1        LBN:          7
        Count:         4        LBN:         27
        Count:        30        LBN:         71
        Count:       146        LBN:        241

$ BACKUP /IMAGE DYA0: DYA1:
$ DUMP /HEADER /BLOCKS=END:0 DYA1:[SMITH]FILE.DAT
Dump of file DYA1:[SMITH]FILE.DAT;1 on 16-APR-1989 13:00:17.82
File ID (12,4,0)    End of file block 197 / Allocated 197

                    File Header
                         .
                         .
                         .
Map area
    Retrieval pointers
        Count:       197        LBN:        193
```

DIRECTORY STRUCTURE ON A COMMON SYSTEM DISK

- A common root directory that contains most of the operating system files and many files for optional products.
- Created automatically by the VMS installation or upgrade procedure.
- Each node has its own local root, known by the logical name SYS$SPECIFIC.
  - — In addition to specific system files, the root also contains a [SYSn.SYSCOMMON] directory.
  - — [SYSn.SYSCOMMON] is an *alias* for the disk's common directory, [VMS$COMMON].
  - — These system-specific root directories are generated by the command procedure, CLUSTER_CONFIG.COM
- Because the files listed in the SYS$SPECIFIC directory of each root are only *aliases*, there is only one actual file resident on disk.
  - — You can verify this by comparing the file identification numbers of the two files in question.
  - — Use the $SET FILE/REMOVE filename command to remove alias files without deleting the actual file.

Figure 4–8: Directory Structure on a Common System Disk



```
SYS$SPECIFIC  =  device:[SYSn.]
SYS$COMMON    =  device:[SYSn.SYSCOMMON]
SYS$SYSROOT   =  device:[SYSn.], device:[SYSn.SYSCOMMON]
```

KEY:

n = system root
x = system subdirectory

MKV_X3032_89

File Search Order on a Common System Disk

- The logical name SYS$SYSROOT is defined as a search list that points to a local root first (SYS$SPECIFIC) and then to the common root (SYS$COMMON).

- This means that when searching for a file, VMS always looks in the local root for each node first.

```
SYS$SYSTEM:--------> (1) SYS$SSPECIFIC:[SYSEXE]
           |
           |-------> (2) SYS$COMMON:[SYSEXE]
```

- When you manipulate files for a specific system on the system disk, this knowledge is important.
  - — For example, MODPARAMS.DAT contains parameters for a specific node.
  - — To create the file specific to SPLASH while logged into SPLASH:

```
$EDIT SYS$SPECIFIC:[SYSEXE]MODPARAMS.DAT
```

  - — To modify the file from SPLASH:

```
$EDIT SYS$SYSTEM:MODPARAMS.DAT
```

  - — As SYS$SYSTEM points first to [SYS0.SYSEXE], the root for SPLASH, it will find the correct file.
  - — To modify the file from BAROOM:

```
$EDIT [SYS1.SYSEXE]MODPARAMS.DAT
```

  - — Since the file is being accessed from another node on the cluster, the exact location must be specified.

MOUNT VERIFICATION - good x bad)

- Protects users when a mounted disk or tape becomes unreachable and then restored.
- Allows the system to recover, rather than wait indefinitely or crash.
- Is enabled when the device is mounted.

  MOUNT/MOUNT_VERIFICATION is the default.

- Mount verification is initiated when:
  - — Device is taken off-line and comes back on-line.
  - — Disk is write-locked and someone attempts a write operation.
  - — Disk drive contains the wrong volume.
  - — Hardware error is detected on the device.
- When mount verification is in progress:
  - — The device is marked invalid.
  - — I/O requests to that device are stalled.
  - — OPCOM displays messages indicating:

    Mount verification in progress
    Reason for mount verification

- Results of successful mount verification:
  - — Device is marked as valid.
  - — I/O operations resume.
  - — OPCOM displays message indicating successful completion.
  - — Generally leaves users unaware that the device has become unreachable and then restored.

MVTIMEOUT

problem: because VMS expects a shut off disk to be turned back on, it waits like (10 mins. - thr.) to notify users that the disk is off!

Resuming Operations During Mount Verification

- If the reason is "device off-line":
  - If the drive has spun down, spin it back up.
  - Mount verification completes.
  - All I/O operations to the disk resume.
- If the reason is "device write-locked":
  - Toggle the hardware WRITE LOCK switch on the drive.
  - I/O operations should resume.
- If the drive is faulty but a functioning drive is available on the same controller:
  - Move the disk to the functioning drive.
  - Swap the unit select plugs.
- If mount verification is unsuccessful, use one of three ways to cancel mount verification:
  - Allow it to time out.
  - Cancel it before it times out.
  - Dismount the volume.
- Any pending or future I/O operations to the disk will fail.

Allowing Mount Verification to Time Out

- Write lock mount verification will not time out.
- SYSGEN parameter MVTIMEOUT defines timeout period for disks and TAPE_MVTIMEOUT for tapes.
- Resetting MVTIMEOUT or TAPE_MVTIMEOUT will not affect a mount verification currently in progress.
- After timeout, all pending I/O requests to the volume will fail.
- Dismount and remount disk before it can be accessed again.

Canceling Mount Verification Before It Times Out

- Invoke the cancellation routine at your console terminal as follows:

```
<CTRL/P>
>>>HALT
>>>D/I 14 C
>>>CONT
IPC>
```

CAUTION

This procedure requires that you halt the processor.

- All system operation is suspended.
- Enter one of four commands:
  - IPC> C device-name - cancels pending mount verification.
  - IPC> X - transfers control to the XDELTA debugger (if it is loaded).
  - IPC> Q - recalculates quorum on a VAXcluster.
  - IPC> <CTRL/Z> - exit from the cancellation routine and resume system operation.
- After you successfully cancel a pending mount verification, you must dismount and remount the volume before you can access it.

Dismounting a Volume During Mount Verification

- Only possible if you can issue the DISMOUNT command for the volume.
- Use a terminal that has access to the volume.
- Enter the $ DISMOUNT/ABORT command for the volume.
- If you do not have access, you will get an error message.
- If dismount is successful, OPCOM displays a message indicating:

```
Mount Verification Aborted
```

- Remove the volume from the drive.

VERIFYING VALIDITY OF DISK VOLUMES

EXAMPLE p. 4-39

* Analyze /Disk_Structure Utility

#ANALYZE/DISK/REPAIR

-can use to recover files in dirs that have been deleted!!!

-can delete dirs w/files by:
Δ protection file.dir - default protect does not allow for delete!
then ~~[illegible]~~ set file/nodirectory which removes bit in file header to be not a dir but an ordinary file

-when you recover the files as above the "lost" files are put in DUA#:[SYSLOST]

- Checks Files-11 disk volumes for:
  - — Readability
  - — Consistency
  - — Validity
- Allows you to:
  - — Report errors and inconsistencies in disk structure.
  - — Repair disk structure.
- Allows you to reclaim disk space by:
  - — Identifying lost files (not cataloged in a directory).
  - — Identifying files marked for deletion.
  - — Rebuilding storage bit map file (BITMAP.SYS).
  - — Deleting unwanted files.
- You can invoke $ ANALYZE /DISK_STRUCTURE <device_name> in one of three modes:
  - — Error reporting with no repairs
  - — Error reporting with repairs
  - — User-controlled selective repairs
- ANALYZE /DISK_STRUCTURE displays various messages:
  - — Many messages are informational and do not indicate problems.
  - — Some indicate fatal problems with the disk volume.

Table 4–4: Common ANALYZE /DISK_STRUCTURE Messages

Ident	Message Text and Description
ALLOCCLR	Blocks incorrectly marked allocated. The specified blocks were marked allocated in the storage bit map but were not allocated to a file.
ALLOCSET	Blocks incorrectly marked free. The specified blocks were marked free in the storage bit map but were allocated to a file.
BACKLINK	Incorrect directory back link 'file-spec.' The back link of the file did not contain the file identification of the directory file in which the file is cataloged. This condition is normal if the file is cataloged in more than one directory.
ENTERLOST	File 'file-name' error entering file in directory [SYSLOST]. An error occurred during an attempt to enter the file in [SYSLOST]. The accompanying message gives additional information.
FINDHOME	No valid home block. Could not locate a valid home block on the volume.
LOSTHEADER	File 'file-name' not found in a directory. The file was not cataloged in a directory. This error is neither reported nor repaired if any errors occurred in processing the directories on the disk.
OPENQUOTA	Error opening QUOTA.SYS. An error occurred during an attempt to open the quota file on the volume. This is normal if quotas are not enforced on the volume.

Recovering Files from a Corrupt Directory

- Symptoms of a corrupt directory:
  - Files seem to have disappeared.
  - $ SHOW QUOTA and $ DIRECTORY indicate different amounts of disk space in use (for the same disk).
- What to do:
  - Use the following procedure to retrieve the lost files:

    ```
    $ SET FILE/NODIRECTORY <directory.dir>
    $ DELETE <directory.dir>
    $ ANALYZE /DISK_STRUCTURE /REPAIR
    ```

  - The lost files are moved from the corrupt directory and placed into [SYSLOST].

    Use DIR/OWNER to determine owner UIC.

  - You can then copy the lost files into a new directory.

Example 4–6: Recovering Files from a Corrupt Directory

```
$ SET FILE /NODIRECTORY DYA0:[000000]CLARK.DIR
$ DELETE DYA0:[000000]CLARK.DIR;*
$ ANALYZE /DISK_STRUCTURE /REPAR DYA0:
%VERIFY-I-OPENQUOTA, error opening QUOTA.SYS
-SYSTEM-W-NOSUCHFILE, no such file
%VERIFY-I-LOSTHEADER, file (11,1,1) A.DAT;1
      not found in a directory
%VERIFY-I-LOSTHEADER, file (12,1,1) A1.DAT;1
      not found in a directory
%VERIFY-I-LOSTHEADER, file (13,1,1) A2.DAT;1
      not found in a directory
%VERIFY-I-LOSTHEADER, file (14,1,1) AAA.DAT;1
      not found in a directory
%VERIFY-I-LOSTHEADER, file (15,1,1) ABC.DAT;1
      not found in a directory

$ DIR DYA0:[SYSLOST] /OWNER

Directory DYA0:[SYSLOST]

A.DAT;1                [GROUP11,CLARK]
A1.DAT;1               [GROUP11,CLARK]
A2.DAT;1               [GROUP11,CLARK]
AAA.DAT;1              [GROUP11,CLARK]
ABC.DAT;1              [GROUP11,CLARK]

Total of 5 files.

$ CREATE/DIR DYA0:[CLARK]
$ COPY DYA0:[SYSLOST]*.* DYA0:[CLARK]
$ DIR DYA0:[CLARK]

Directory DYA0:[CLARK]

A.DAT;1        A1.DAT;1       A2.DAT;1       AAA.DAT;1
ABC.DAT;1
```

*If you do this on master file directory you will be unable to access the disk at all!

* to fix ↑ – w/logical I/O priveledge – init another disk of same size find log. block of master file on O.K. disk of same size then reset bit on bad disk. w/log. block # write to disk use: Dump/header!

~~Dump/file-header~~

+ must also fix checksum bit!!! then dismount + remount!!!

* VOLUME REBUILD - automatic when volume comes up or mounts w/out proper dismount!

- Is necessary when a disk volume has been improperly dismounted.
- Recovers any caching limits that were enabled at time of dismount.
  - Preallocated free space (EXTENT cache)
  - Preallocated file numbers (FILE_ID cache)
  - Disk quota usage caching (QUOTA cache)
- The rebuild operation reads each file header and:
  - Updates index file bit map to show only valid files
  - Updates storage bit map to show allocated blocks
  - Updates quota file entries to show correct usage
  - Can take a considerable amount of time
- ACP_REBLDSYS enables automatic rebuild for **system** disk.
- Two ways to enable volume rebuild for a volume:

```
$   MOUNT /REBUILD

$   SET VOLUME /REBUILD
```

- If a volume was mounted /NOREBUILD:
  - It can be used immediately, but MOUNT displays the following message:

```
%MOUNT-I-REBLDREQD, rebuild not performed; some
free space unavailable; diskquota usage stale
```

  - You should perform the rebuild later with SET VOLUME /REBUILD.

DISK QUOTA REBUILD

- Reconstructs the usage counts for all entries on the volume.
- Use disk quota rebuild when you:
  - Create a quota file on a volume with existing files.
  - Re-enable disk quotas on a volume where they have been suppressed or suspended.
- To invoke disk quota rebuild:

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> DISKQUOTA REBUILD /DEVICE=<disk>
```

- During disk quota rebuild, file activity on the volume is suspended.

SUMMARY

- Files-11 ODS-2, the default VMS disk structure, is a component of the VMS disk I/O system.
- Disk's physical characteristics are predetermined, but you can define disk software characteristics.
- You can use the following commands to define and display disk software characteristics:

  ```
  $    INITIALIZE

  $    SHOW DEVICE /FULL
  ```
- ODS-2 reserved files, particularly INDEXF.SYS, are necessary for accessing any file on the volume.
- ODS-2 files consist of a file header and one or more data blocks and are cataloged in a directory.
- You can use the following commands to display information about disk files:

  ```
  $    DUMP /HEADER

  $    DUMP

  $    DIRECTORY /FULL
  ```
- You can use the BACKUP /IMAGE command to restore file contiguity to a noncontiguous file.
- Mount verification allows the system to recover when a mounted disk becomes unreachable and then restored.

- You can use ANALYZE /DISK_STRUCTURE to:
  - Verify the validity of disk volumes.
  - Recover files from a corrupt directory.
- Volume rebuild is necessary when a disk volume has been improperly dismounted. You can enable volume rebuild in two ways:

  $ MOUNT /REBUILD

  $ SET VOLUME /REBUILD
- Disk quota rebuild reconstructs the usage counts for all entries on a volume. You invoke disk quota rebuild by:

  ```
  $RUN SYS$SYSTEM:SYSMAN
  SYSMAN>DISKQUOTA REBUILD/DEVICE=<disk>
  ```

MODULE 5
INTRODUCTION TO VAX RMS

INTRODUCTION

VAX Record Management Services (RMS) provides generalized record access methods available to all VMS processes. RMS provides extensive capabilities for data storage, retrieval, and modification; and supports several file organizations and file access techniques. The VMS operating system uses RMS as an integral part of its operation and by most optional software products for file access. Many applications make extensive use of RMS files for storage and retrieval of data.

Data files are usually designed and created by programmers; however, the system manager must allocate and manage the system resources needed by file applications running on the system. This module provides an overview of RMS files, buffering concepts, and RMS utilities. It then discusses the management of system resources needed by file applications. Finally, it introduces RMS Journalling, an optional software product for improving data file integrity.

OBJECTIVES

To effectively manage system resources necessary for applications using VAX RMS files, you should be able to:

1. Describe and compare the various RMS file organizations, record formats, and access options.
2. Identify permanent and semipermanent RMS file attributes.
3. Describe RMS buffering, the advantages it offers, and how to control buffering from the DCL level at run time.
4. Describe the functions of the RMS utilities.
5. List the process and system resources used by file applications, and describe how to control and allocate them.
6. Describe the functions of the RMS journalling product.

RESOURCES

- *Guide to VMS File Applications*
- *VMS File Definition Language Facility Manual*
- *VMS Convert and Convert/Reclaim Utility Manual*
- *VMS Analyze/RMS_File Utility Manual*
- *Guide to Setting a VMS System*
- *Guide to Maintaining a VMS System*
- *VMS Analyze/Disk Structure Utility Manual*
- *System Messages and Recovery Procedures*

TOPICS

1. Introduction to VAX Record Management Services (RMS)
   a. RMS file organizations
   b. RMS record formats and access options
   c. RMS file attributes
   d. RMS I/O buffers
   e. Global buffers
2. Introduction to RMS utilities
   a. File definition language (FDL)
   b. Convert and Convert/Reclaim utilities
   c. Analyze/RMS_File utility
   d. Examples of using RMS utilities
3. Process and system resources for file applications
   a. Memory requirements
   b. Record locking
   c. Suggested values for resource limits
4. Introduction to RMS journalling
   a. After-image journalling
   b. Before-image journalling
   c. Recovery unit problem
   d. Recovery unit journalling

INTRODUCTION TO VAX RECORD MANAGEMENT SERVICES (RMS)

RMS File Organizations

- Sequential
- Relative
- Indexed sequential

RMS Record Formats

- Fixed length—supported for all file organizations
- Variable length—supported for all file organizations
- Variable with fixed control—supported for:
  - Sequential files
  - Relative files
- Stream—supported for sequential files

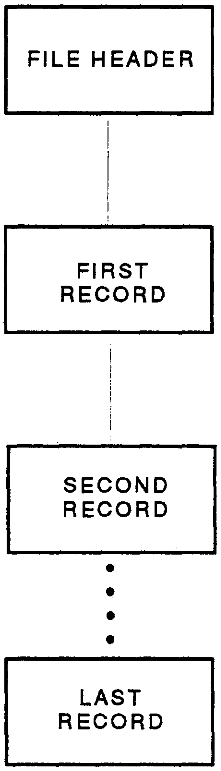
RMS Record Access Options

- Sequential—supported for all file organizations
- Random by key value—supported for indexed files
- Random by relative record number—supported for:
  - Sequential files
  - Relative files
- Random by the Record's File Address (RFA)—supported for all file organizations

Sequential Files

- Records in the file are arranged one after the other - records are stored in the order in which they are entered.
- Sequential files with fixed-length records have no overhead except the Files-11 file header.
- Records in sequential files are aligned on an even byte (except for stream format).
  - — If record size is an odd number, one byte is inserted on the disk before the next record.
  - — This process is transparent unless you examine a dump.

Figure 5–1: Sequential File Organization

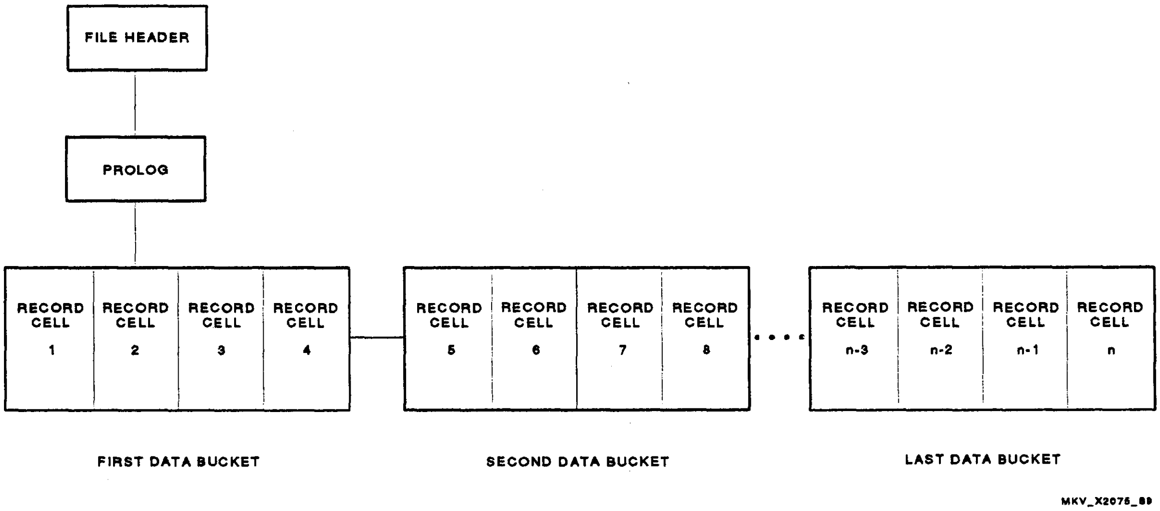


MKV_X2074_89

Relative Files

- Records are stored in a series of fixed-length positions called **cells** - each cell holds one record.
- Relative record number identifies the position of each record cell in relation to the beginning of the file.
- Relative organization allows random retrieval of records by means of the relative record number.
- In addition to the Files-11 header and data records, relative files also contain a **prolog,** which:
  - — Consists of one block
  - — Describes the file

Figure 5–2: Relative File Organization



Indexed Sequential Files

- Records are stored in the order defined by a key value (either ascending or descending order).
- Records can be retrieved sequentially or randomly.
- In addition to the Files-11 header and data records, indexed files include:
  - Prolog— contains information about the file
  - Primary key (key 0)
  - One or more alternate keys (optional)
  - Key descriptor for each key
  - Areas—sections of contiguous disk blocks that are independently allocated and managed
  - Area descriptors for each area
  - Primary index structure
  - Index buckets (contain index)
  - Data buckets (contain data records)
- Example use of alternate keys in an indexed file:
  - Primary key (employee badge number)
  - Alternate key (employee last name)

    An application could use either key to search the file for a given employee record.

Figure 5–3: Indexed File Organization

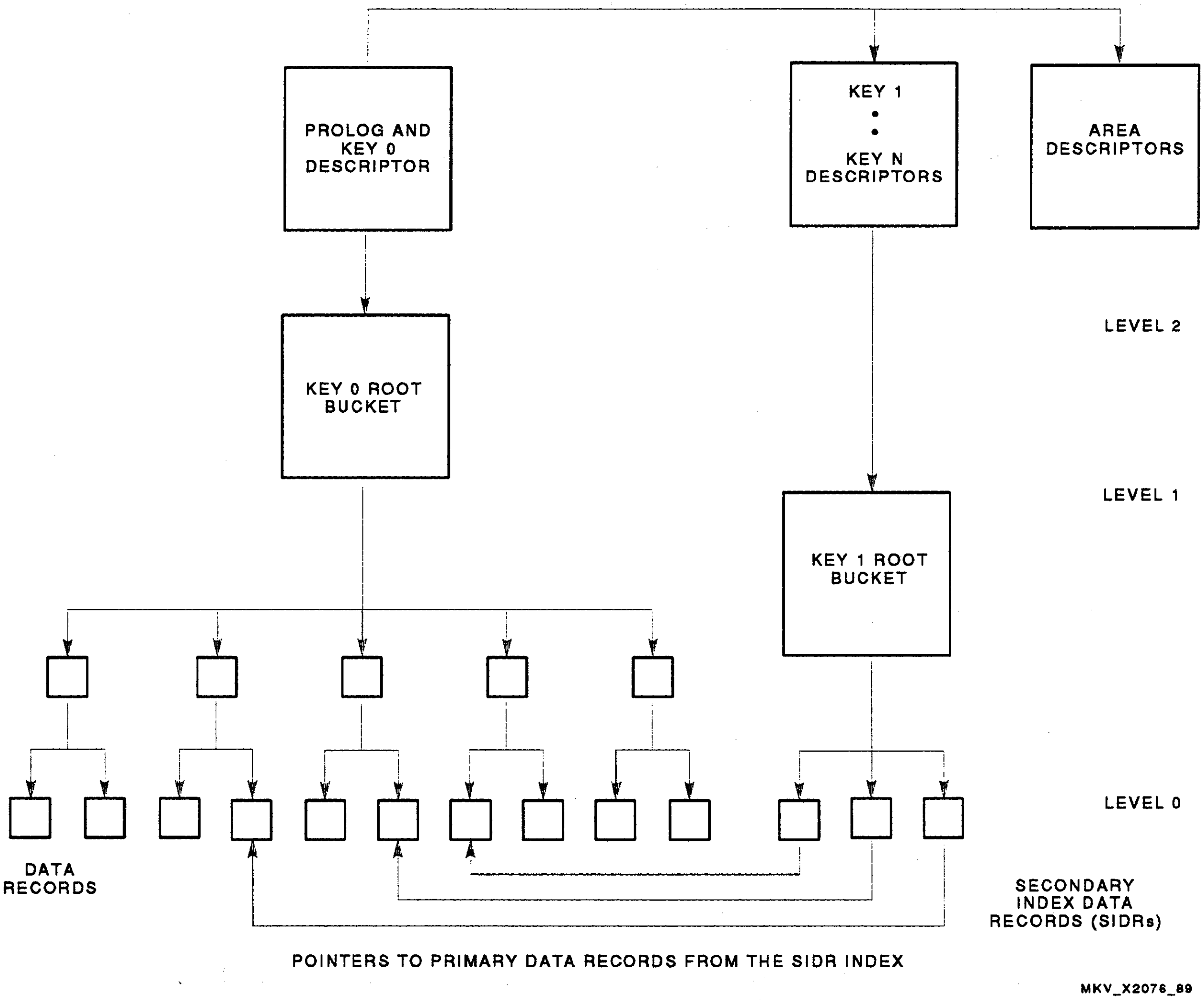


Table 5–1: Advantages and Disadvantages of RMS File Organizations

Organization	Advantages	Disadvantages
Sequential	Uses disk and memory efficiently	Has limited random access
	Provides optimal usage if the application accesses all records sequentially on each run	Cannot delete records
	Provides flexible record format	Can insert records only at the end of file
	Allows data to be stored on different types of media	
Relative	Allows sequential and random access for all languages	Allows data to be stored on disk only for access by relative cell number
	Allows random record deletion and insertion	Requires that files contain a record cell for each relative number allocated (files may not be densely populated)
		Requires that record cells be the same size
		Allows record insertion only to empty cells (or at the end of the file)
		Does not allow duplicate cell numbers
Indexed	Allows sequential and random access by key value for all languages and by RFA for some languages	Allows data to be stored on disk only
	Allows random record deletion and insertion	Usually requires more disk space
	Allows variable-length records to change length on update	Generally requires multiple disk accesses to process a record

RMS File Attributes

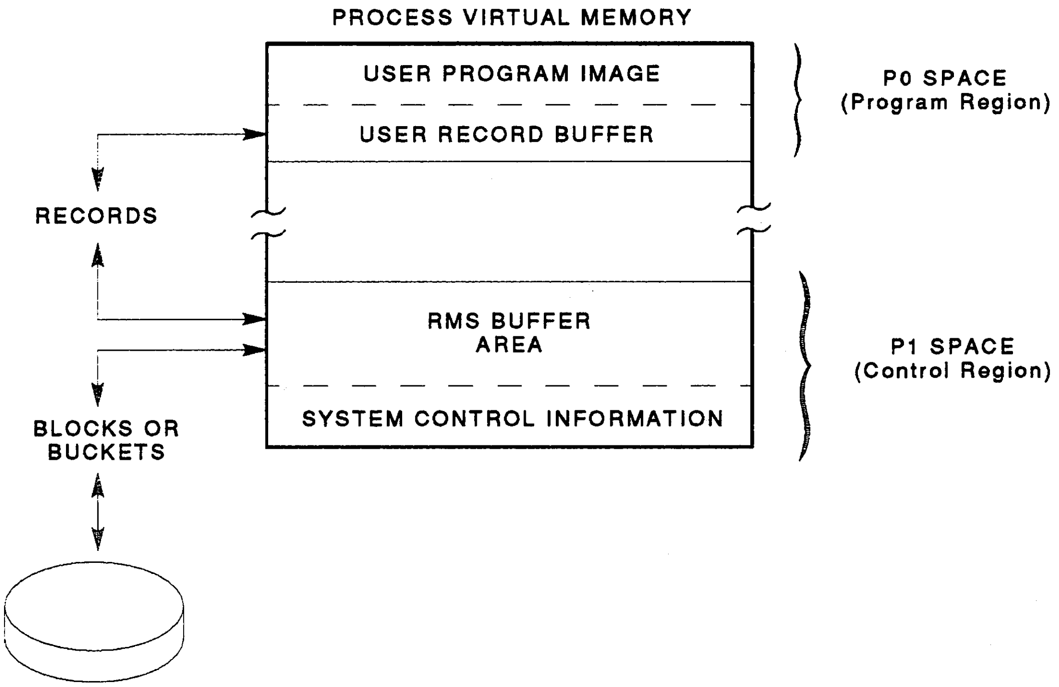
- Permanent file attributes
  - Cannot be changed without reorganizing or converting file
  - Examples:

    File organization
    Bucket size
    Record size
    Key

- Semipermanent file attributes
  - Can be changed with DCL SET FILE commands
  - Examples:

    Erase or delete
    Expiration date
    Number of global buffers
    Owner UIC
    Protection
    Version limit

Figure 5–4: VAX RMS Buffers and the User Program



MKV_X2077_89

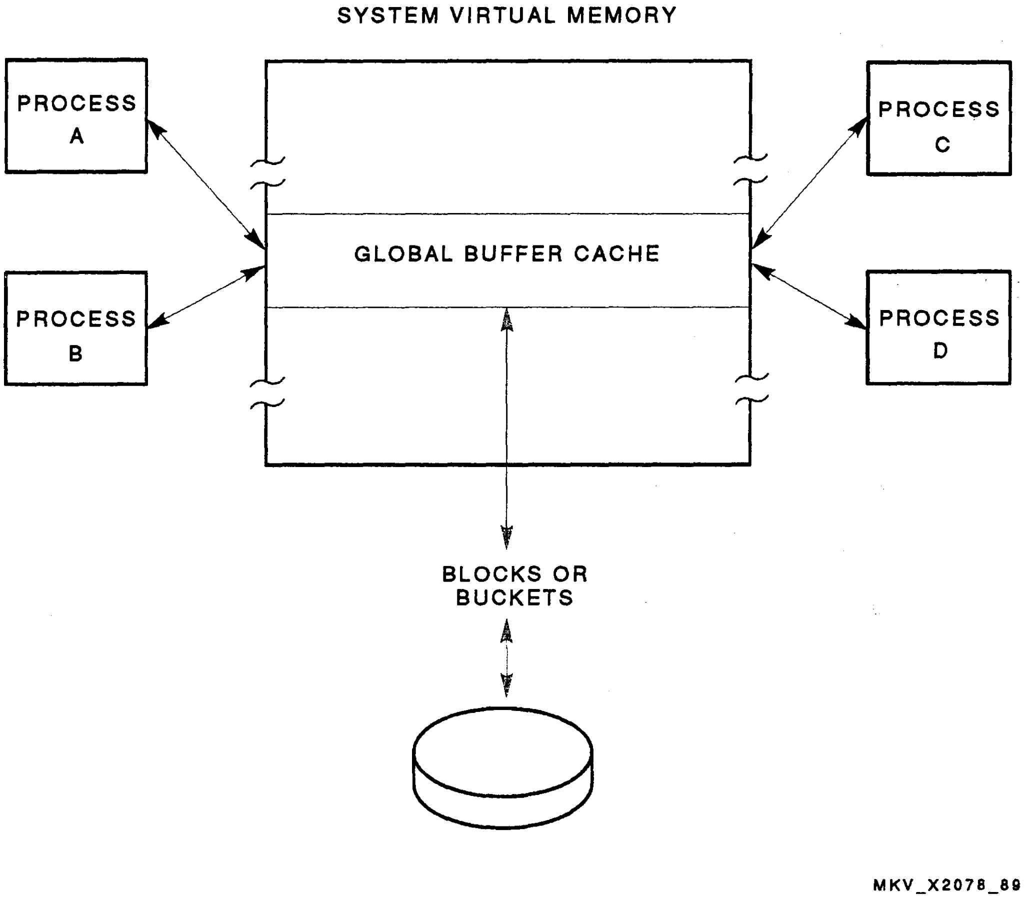
- RMS buffers reside in memory.
- Used by RMS to transfer blocks or buckets of data from disk to program.
- RMS buffering is transparent to a program.
  - Records appear to move directly between a file and the program.
  - When a program reads records, the records are:

    Read from file to RMS buffer area
    Transferred from RMS buffer area to program
  - When a program writes records, the records are:

    Transferred to the RMS buffer area
    Written to the file
- Proper use of RMS buffering can improve program performance. Multibuffering reduces the number of physical I/O operations required.
- To control and display the process or system default buffer count, use:
  - SET RMS_DEFAULT
  - SHOW RMS_DEFAULT

Global Buffers

Figure 5–5: Using Global Buffers for a Shared File



- For use by multiple processes that access the same files and records.
- Global buffers can improve performance of file-sharing:
  - Use global buffers in addition to local buffers.
  - Make global buffer large enough to contain entire index structure.
  - Use global buffers only when the file is opened for read- and write-sharing.
- Global buffers cannot be used for sharing process permanent files, for example:
  - SYS$COMMAND
  - SYS$ERROR
  - SYS$INPUT
  - SYS$OUTPUT
- To enable global buffers for a shared file:

  ```
  $ SET FILE /GLOBAL_BUFFERS=n
  ```
- Global buffers can also be enabled by programs at run time.

INTRODUCTION TO RMS UTILITIES

- File Definition facility
- Convert and Convert/Reclaim utilities
- Analyze/RMS_File utility

File Definition Language (FDL) Facility

1. FDL is used to write specifications for RMS data files.
2. The specifications are written to text files, called FDL files.
3. FDL files are used by other RMS utilities to create your data files.
4. FDL facility is comprised of:
   a. File Definition Language
   b. Edit/FDL utility
   c. Create/FDL utility
5. There are two steps used to create an RMS data file from the DCL level:
   a. $ EDIT/FDL filename.FDL
      1. Allows you to invoke a script that prompts you to define file specifications
      2. Creates an FDL file containing your file specifications
      3. EDIT/FDL also allows modification of existing FDL files

   b. $ CREATE/FDL=filename.FDL filename.dat
      1. Reads the specifications in the FDL file
      2. Creates a new, empty data file according to the specifications in the FDL file

The Convert Utility

- Copies records from source file(s) to an output file.
- Output file can differ in file organization and format from the source file.
- The Convert utility can be used to:
  - Convert a file from one organization to another.
  - Reformat a fragmented indexed file.
- To invoke Convert, type:

```
$   CONVERT [/qualifiers] in-file out-file
```

The Convert/Reclaim Utility

- Reclaims empty buckets in indexed files so that new records can be written to them.
- Helps to avoid extending the file when new records are added.
- To invoke this utility, type:

```
$   CONVERT/RECLAIM [/qualifier] filespec
```

Analyze/RMS_File Utility

The Analyze/RMS_File utility allows you to:

- Generate an FDL file from a data file.
- Examine the internal file structure of an RMS file.
- Explore the structure of a file interactively.
- Check the file structure for errors.
- Generate reports on a file's structure and use.

To invoke this utility, type:

```
$ ANALYZE/RMS_FILE filespec [qualifiers]
```

Examples of Using RMS Utilities

Using RMS Utilities to Change File Organization

- Create an FDL file to define the specifications for the new file.
- Using the specifications in the FDL file, create the new file and populate it with records from the existing input file.

```
$ EDIT/FDL new_indexed_file.FDL
$ CONVERT/FDL=new_indexed_file.FDL
_Input: old_sequential_file.DAT
_Output: new_indexed_file.DAT
```

- Alternatively, you can create a new, empty file and then populate the new file with records from an existing input file. For example:

```
$ EDIT/FDL new_indexed_file.FDL
$ CREATE/FDL=new_indexed_file.FDL
$ CONVERT
_Input: old_sequentail_file.DAT
_Output:  new_indexed_file.DAT
```

Using Convert to Reformat Fragmented Indexed Files

- Indexed files that have many records deleted or inserted become fragmented.
- Use Convert to eliminate fragmentation, as follows:

```
$ CONVERT
_Input: indexed_file.dat
_Output: new_indexed_file.dat
```

- The output file is created and then populated with the records from the existing input file.
- By default, file attributes are preserved.
- Fragmentation is eliminated.

Using RMS Utilities to Recover Data from Corrupt Files

- Indicators of a possible data corruption problem are:
  - Error message: bucket format check failed for VBN.
  - The file does not have as many records as it should have.
- Guidelines for recovering data from corrupt indexed files are:
  - Immediately make a backup copy of the file.
  - Use the backup copy when attempting to recover the data.
  - Be sure the file in question is the correct file (make sure there has not been a mix-up in files).
  - Use ANALYZE/RMS/INTERACTIVE or DUMP to check:

    The file header
    Prologue
    Key descriptors

Troubleshooting the Cause of Indexed File Corruption

- A logical name can cause programs to use the wrong file.
- Users sharing an account can inadvertently use the same file name for different data files.
- Incorrect use of DCL commands can make a copy of a file and give that file different attributes than the original file.
- Determine whether the file is really corrupt by using ANALYZE/RMS/CHECK.
- If ANALYZE/RMS/CHECK detects errors, the file has been corrupted.
- If a number of severe errors are indicated, restore the most recent uncorrupted backup copy.
- If a problem is identified in a particular virtual block:
  - Dump all blocks in the bucket starting with that virtual block.
  - The bucket could contain:

    Index records
    Data records

Using Convert to Restore Data from Corrupt Indexed Files

1. If none of the primary level 0 data buckets are corrupt, you can probably recover the data as follows:

   ```
   $ CONVERT
   _Input: corrupt_indexed_file.dat
   _Output: new_indexed_file.dat
   ```

2. If a primary index bucket is corrupt, Convert cannot recover data from the primary key. If you have at least one alternate key, follow these steps:

   a. If you do not have an FDL file for the corrupt file, create one.

      ```
      $ ANALYZE/RMS_FILE/FDL  corrupt_indexed_file
      ```

   b. You must edit the FDL file created by ANALYZE.

      ```
      $ EDIT good_indexed_file.FDL
      ```

      1. Delete the version number in the file name.

      2. Delete all the analysis sections at the end of the FDL file.

   c. Create an FDL file for a sequential file.

      ```
      $ EDIT/FDL seq_file.FDL
      ```

   d. Create a good sequential file from the data in the corrupt indexed file.

      ```
      $ CONVERT/KEY=1/FDL=seq_file.FDL
      _Input: corrupt_indexed_file.dat
      _Output: good_sequential_file.dat
      ```

   e. Create a good indexed file from the data in the sequential file by using the indexed file attributes specified in the FDL file.

      ```
      $ CONVERT/FDL=good_indexed_file.FDL
      _Input: good_sequential_file.dat
      _Output: good_indexed_file.dat
      ```

PROCESS AND SYSTEM RESOURCES FOR FILE APPLICATIONS

- A file application requires various resources.
  - System-wide
  - Per-process
- You may need to adjust specific resources and quotas for processes running a file application.
- You should be involved in the application design phase.
  - Discuss process and system requirements with your programming staff.
  - In some cases, you may want to obtain additional memory or disk drives.

Memory Requirements

- To improve application performance, you often need to:
  - Increase buffer sizes that can require larger working sets.
  - Allocate more buffers.
- Memory use generally increases with the number of files to be processed at the same time.
- The memory use (working set) of a process is governed by the following UAF fields:
  - WSDEFAULT
  - WSQUOTA
  - WSEXTENT
- These parameters can ensure that the process will have:
  - Sufficient memory to perform the application
  - A minimum amount of paging

Record Locking

- RMS uses record locking to control operations when two or more processes are accessing a file simultaneously. While one process is adding, deleting, or modifying a record, RMS automatically locks the record from access by another process.

- Process parameter ENQLM (enque quota) limits the number of locks a process can own concurrently.

- RMS takes one lock for every:
  - Shared file
  - Local buffer
  - Global buffer section
  - Outstanding record lock

- ENQLM should be increased for applications that:
  - Modify or add records to shared files.
  - Perform large amounts of file-sharing.

- Increase ENQLM by the number of records that can be locked at one time, multiplied by the number of open files.
  - Estimate the number of locks per process by:

```
  One lock per file
+ Multibuffer count for that file
+ Number of records locked (usually one)
__________________________________________
= Total locks per process
```

- To control process default multibuffer counts, type:
  - SHOW RMS_DEFAULT
  - SET RMS_DEFAULT

SYSGEN Parameters Related to Record Locking

- LOCKIDTBL limits the initial number of locks on the system.
- RESHASHTBL limits the number of entries in the lock manager resource name table.

  RESHASHTBL should be about one-quarter (1/4) the value of LOCKIDTBL.

Table 5–2: Error Messages Related to Record Locking

Message Code	Indicates	What to Do
SS$__EXENQLM	Insufficient enque quota for process	Increase ENQLM for the process
SS$__NOLOCKID	LOCKIDTBL system parameter too low	Increase LOCKIDTBL and RESHASHTBL system parameters

Suggested Values for Resource Limits

Table 5–3: Suggested Values for Process Resource Limits

Limit	Value	Type	Description
ASTLM	24	Nondeductible	AST queue limit
BIOLM	18	Nondeductible	Buffered I/O count limit
BYTLM	8192	Pooled	I/O byte count limit
CPU	0	Deductible	CPU time limit (0 = no limit)
DIOLM	18	Nondeductible	Direct I/O count limit
ENQLM	30	Pooled	Lock limit
FILLM	20	Pooled	Open file limit
JTQUOTA	1024	Pooled	Initial byte quota for job-wide logical name table
MAXACCTJOBS	0	System-wide	Maximum active processes for the same account (0 = no limit)
MAXDETACH	0	System-wide	Maximum detached processes for a single username (0 = no limit)
MAXJOBS	0	System-wide	Maximum active processes for a single username (0 = no limit)
PGFLQUO	12800	Pooled	Paging file limit
PRCLM	2	Pooled	Subprocess creation limit
SHRFILLM	0	Pooled	Maximum number of open shared files (0 = no limit)
TQELM	10	Pooled	Timer queue entry limit
WSDEFAULT	300	Nondeductible	Default working set size
WSEXTENT	700	Nondeductible	Maximum working set size
WSQUOTA	350	Nondeductible	Working set quota

Memory Allocation for Global Buffers

The memory use of global buffers is controlled by SYSGEN parameters.

Table 5–4: SYSGEN Parameters Related to Global Buffers

Parameter	Function
RMS_GBLBUFQUO	Specifies maximum number of RMS global buffers in use on a system simultaneously, regardless of the number of users or files
GBLSECTIONS	Specifies maximum number of global sections in use simultaneously on the system
GBLPAGES	Specifies the number of global page-table entries in use simultaneously on the system
GBLPAGFIL	Specifies the number of system-wide pages allowed for global page-file sections, or scratch global sections in use simultaneously on the system

INTRODUCTION TO RMS JOURNALLING

- Optional product designed to improve the data integrity of RMS files.
- Three types of RMS journalling:
  - After-image (AI) journalling—protects against hardware failure
  - Before-image (BI) journalling—protects against bad data being written to a file
  - Recovery unit (RU) journalling—protects against data inconsistency due to interruption of an application
- RMS journalling can be applied to any RMS file that is updated.
- RMS files can be marked for AI, BI, or RU journalling, or any combination.
- The following commands are used to mark files for journalling:
  - $ SET FILE/AI_JOURNAL
  - $ SET FILE/BI_JOURNAL
  - $ SET FILE/RU_JOURNAL
- The following types of RMS files can be journalled:
  - Sequential
  - Relative
  - Indexed-sequential

After-Image (AI) Journalling

1. Used to protect an important file from hardware failure.

2. To use AI journalling:

   a. Mark the file for AI journalling.

   ```
   $ SET FILE/AI_JOURNAL=(FILE=JNL_DISK:WEEKLY,CREATE) -
   _$ WORK_DISK:[PAYROLL]WEEKLY.DAT
   ```

   b. Back up the data file.

   ```
   $ BACKUP/RECORD WORK_DISK:[PAYROLL]WEEKLY.DAT -
   _$ JNL_DISK:[PAYROLL]WEEKLY.BCK
   ```

3. To recover from hardware failure, perform the following steps:

   a. Use the RMS Recovery utility to create a recovered file.

   ```
   $ RECOVER/RMS_FILE/FORWARD JNL_DISK:[PAYROLL]WEEKLY.BCK
   ```

   b. To replace the corrupt version, copy the recovered file to the old file specification.

   ```
   $ COPY  JNL_DISK:[PAYROLL]WEEKLY.BCK -
   _$ WORK_DISK:[PAYROLL]WEEKLY.DAT
   ```

   c. Mark the new file for AI journalling as before.

   d. Make a backup of the new data file.

Before-Image (BI) Journalling

- Protects an important file against data corruption.
- To use BI journalling:
  - Mark the file for BI journalling.

```
$ SET FILE/BI_JOURNAL=(FILE=WEEKLY,CREATE) -
_$ WORK_DISK:[PAYROLL]WEEKLY.DAT
```

- To recover from data corruption:
  - Use the RMS Recovery utility to roll back the file to a previous state.

```
$ RECOVER/RMS_FILE/BACKWARD/UNTIL=12-FEB-1989:8:20 -
_$  WORK_DISK:[PAYROLL]WEEKLY.DAT
```

  - If you are not sure when the bad data occurred, roll back a short time and then test. If needed, roll back some more.
- No additional steps are needed to use the file, and BI journalling is still enabled.

Recovery Unit Problem

- Protects against files being left in an inconsistent state due to failure of a process.
- In some instances, a series of file updates must occur for data to be consistent. For record-keeping data to be consistent, when a piece of company equipment is transferred from one department, an inventory record must be:
  - Removed from one department
  - Added to another department
- If the process doing the updates fails before the entire series completes, the data is inconsistent.

Recovery Unit Journalling

- Recovery unit journalling is used to ensure that either all or none of a series of transactions occur.
- To use recovery unit journalling:
  - Mark the file for recovery unit journalling:

    ```
    $ SET FILE/RU_JOURNAL  WORK_DISK:[SYSJNL]DEPT1.DAT
    $ SET FILE/RU_JOURNAL  WORK_DISK:[SYSJNL]DEPT2.DAT
    ```

  - Define recovery units in your application code by using recovery unit services.

    ```
    SYS$START_RU to mark the start of a recovery unit

    SYS$END_RU to mark the end of a recovery unit

    SYS$ABORT_RU to abort a recovery unit upon a
                 failure condition
    ```

- Once a file is marked for recovery unit journalling, all modifications to the file must be made within a recovery unit.
- Upon failure of a process within an incomplete recovery unit:
  - All files specified in the recovery unit that are marked for recovery unit journalling are locked by RMS journalling.
  - The first time a process tries to access any one of the files, RMS journalling attempts to return all files to their previous state.
  - If all files can be returned to their previous state, the files are unlocked and made available.
  - If any file cannot be returned to its previous state, all files remain unavailable to RMS access.
- The procedure is automatic and requires no action by the user.

SUMMARY

- VAX RMS supports a variety of:
  - File organizations
  - Record formats
  - Record access options
- RMS files can have various permanent and semipermanent attributes.
- RMS I/O buffers can improve performance of file applications.
- Global buffers are used by multiple processes accessing the same file.
- RMS supports the following utilities:
  - FDL facility
  - Convert and Convert/Reclaim
  - Analyze/RMS_file
- RMS utilities can be used to perform various tasks, including:
  - Define file characteristics.
  - Create files.
  - Change file organization.
  - Reformat fragmented indexed files.
  - Recover data from corrupt files.
- File applications require various system-wide and per-process resources, which the system manager must allocate and manage.
- RMS uses record locking when two or more processes are accessing a file simultaneously.
- RMS Journalling is an optional product designed to improve data integrity.

MODULE 6
SOFTWARE TROUBLESHOOTING

INTRODUCTION

The VMS operating system is a complex system that performs a wide range of functions. Most application systems require one or more layered software products in addition to the operating system. Like many layered software products, the VMS operating system can be customized in various ways. Therefore, in addition to the different VAX hardware configurations, there are many possible combinations of standard and customized software.

Troubleshooting software problems on a complex system is not a simple task. This module cannot cover all the problems that can arise in a VMS system running various layered software products and application programs.

This module discusses some common software problems that can occur in a VMS system and suggests some ways to troubleshoot these problems. However, this module does not attempt to present a guaranteed method for solving all possible software problems. The purpose of this module is to increase your awareness of some common software problems and troubleshooting methods. It provides a starting point from which you can begin your own troubleshooting activities.

OBJECTIVES

To troubleshoot software problems, a VMS system manager must be able to:

- List and describe the VMS tools that can be used to troubleshoot software problems.
- Use the following tools to gather information for performing preventive maintenance and diagnosing problems:
  - System dump analyzer (SDA)
  - Monitor utility
  - Error logger
  - DCL SET and SHOW commands
  - System Generation utility (SYSGEN)
  - Analyze/Disk_Structure utility
- Describe symptoms and possible causes of the following types of software problems:
  - System crashes
  - System hangs
  - Process hangs
  - Corrupt disk structure
- Use the appropriate methods and tools to troubleshoot these software problems.

RESOURCES

- *VMS System Dump Analyzer Manual*
- *VMS Monitor Utility Manual*
- *VMS Error Log Utility Manual*
- *VMS Analyze/Disk Structure Utility Manual*
- *VMS Authorize Utility Manual*
- *VMS System Generation Utility Manual*
- *Guide to Setting Up a VMS System*
- *Guide to Maintaining a VMS System*
- *VMS DCL Dictionary*
- *System Messages and Recovery Procedures Manual*
- *Guide to VMS Performance Management*
- *Guide to VMS File Applications*
- *VAX/VMS Internals and Data Structures Manual*

TOPICS

1. Review of VMS tools used in troubleshooting software problems

   a. System dump analyzer (SDA)

   b. Monitor utility

   c. Error logger

   d. DCL SET and SHOW commands

   e. The System Generation utility (SYSGEN)

2. Preventive troubleshooting

3. System crashes

   a. Bugcheck mechanism

   b. Exceptions

   c. Machine checks

   d. System hangs

   e. Pagefile full

   f. System dynamic memory (pool) exhausted

   g. Compute-bound processes

   h. System hangs caused by hardware problems

   I. Booting problems

4. Process Hangs

   a. Process awaiting resources

   b. Processes hung during login

   c. Process waiting for terminal I/O

   d. Troubleshooting a process awaiting terminal I/O

   e. Outswapped processes unable to be inswapped

   f. Processes with priority too low

   g. Process executing an infinite loop

VMS TOOLS USED TO TROUBLESHOOT SOFTWARE PROBLEMS

System Dump Analyzer (SDA)

- Helps to find the cause of system failure
- Allows you to analyze a running system
- To analyze a crash dump, type:

```
$   ANALYZE /CRASH_DUMP SYSDUMP.DMP
```

- To analyze a running system, type:

```
$   ANALYZE /SYSTEM
```

Monitor Utility

- Allows you to monitor system performance data, including:
- Process and system-wide statistics
- I/O statistics
- Memory management statistics
- Time spent in each processor mode
- Produces a variety of outputs - useful in preventive troubleshooting
- To invoke Monitor, type:

```
$   MONITOR classname
```

Error Logger

- Processes error log entries
- Produces reports of hardware errors and events
- To produce error log reports, type:

```
$   ANALYZE /ERROR SYS$ERRORLOG:ERRLOG.SYS
```

DCL SET and SHOW Commands

- SHOW displays information about characteristics of:
  - The system
  - The processes
  - The devices
- SET commands define these characteristics.

The System Generation Utility (SYSGEN)

- Modifies system parameters
- Creates, modifies and installs paging and swapping files
- AUTOGEN.COM works with SYSGEN
  - Analyzes any changes you make to system parameters
  - Automatically adjusts any related parameters
- SYS$UPDATE:SWAPFILES.COM can be used to create or extend primary page, swap, and dump files - it prompts you to enter file sizes

PREVENTIVE TROUBLESHOOTING

- Run ANALYZE/DISK_STRUCTURE periodically to check disk structure. ANALYZE/DISK_STRUCTURE is discussed in Module 4, "VMS Disk File Structure."
- Examine the operator log file daily.
- Examine the system console listing daily.
- Run SDA at system startup.
  - Place SDA commands in SYS$MANAGER:SYSTARTUP_V5.COM.
  - SDA will be run only if the system has just failed.
  - SDA ensures that crash dump information is displayed after each failure.

Example 6–1: Running SDA at System Startup

```
$ !    Print dump listing if system just failed
$ !
$    ANALYZE/CRASH_DUMP SYS$SYSTEM:SYSDUMP.DMP
     COPY SYS$SYSTEM:SAVEDUMP.DMP       ! Save dump file
     SET OUTPUT SYSDUMP.LIS             ! Create listing file
     SHOW CRASH                         ! Display crash information
     SHOW STACK                         ! Show current stack
     SHOW SUMMARY                       ! List all active processes
     SHOW PROCESS/PCB/PHD/REG           ! Display current process
     EXIT
```

- SHOW ERROR each time you log in - if errors are indicated, examine the error log.
- SHOW MEMORY each time you log in.
  - Identifies any bad memory
  - Checks for pagefile and swapfile usage
  - Shows any outswapped processes
- SHOW SYSTEM/FULL periodically to keep aware of:
  - What processes are running on your system
  - Owner UICs of all processes
  - Signs of potential problems such as many outswapped processes
- Use Monitor to continuously monitor system activity.
  - Table 6-1 summarizes helpful command procedures that produce Monitor reports.
  - Appendix A shows sample Monitor output.

Table 6–1: MONITOR Procedures

Command Procedure Name	Description
SYS$EXAMPLES:SUBMON.COM	Invokes MONITOR.COM. Invoke this procedure from your SYS$MANAGER:SYSTARTUP_V5.COM.
SYS$EXAMPLES:MONITOR.COM	Invoked by SUBMON.COM. Updates Monitor data files containing information about all classes every ten minutes.
SYS$EXAMPLES:MONSUM.COM	Generates Monitor summary reports and mails them to the system manager.

Table 6–2: **Summary of Important Monitor Data Items**

Item	Class	Comment
Compute Queue (COM, COMO)	STATES	Usually indicates CPU responsiveness. Typically, larger compute queues mean longer response time.
Idle Time	MODES	Good measure of available CPU cycles, but only when memory is sufficient and disk I/O subsystem is not overloaded.
Inswap Rate	IO	Can indicate memory management problems. Should be low.
Interrupt Stack Time and Kernel Mode Time	MODES	Time representing service performed by the system. Usually, should not exceed 40%.
Executive Mode Time	MODES	Time spent executing RMS and some database products.
Page Fault Rate	PAGE	Acceptable rates vary among different VAX processors.
Page Read I/O Rate	PAGE	The hard fault rate. Should be kept below 10% of overall rate.
System Fault Rate	PAGE	Should be minimized, no more than one fault per second.
Response Time	DISK	Expected values vary among different models of disks.
I/O Operation Rate	DISK	Following are normal load ranges for RA-series and MASSBUS disks: 1 - 8: lightly loaded 9 - 15: light to moderate 16 - 25: moderate to heavy More than 25: heavily loaded
Page Read I/O Rate + Page Write Rate + Inswap Rate (times 2) + Disk Read Rate + Disk Write Rate	PAGE PAGE IO FCP FCP	System I/O operation rate. The sum of these items shows the part of the overall rate initiated directly by the system.
Cache Hit Percentages	FILE_ SYSTEM_ CACHE	Should be kept as high as possible, no lower than 75% for active caches.

SYSTEM CRASHES

Bugcheck Mechanism

- Internal diagnostic check - routine name: EXE$BUG_CHECK
- Bugcheck processing can include:
  - Error log entry
  - System crash with console output and dump
- Bugcheck processing is determined by:
  - Access mode at which the exception occurred
  - SYSGEN parameters
  - Severity of bugcheck
- For fatal bugchecks, the bugcheck mechanism:
  - Writes error log entry
  - Displays bugcheck information on system console
  - Writes dump information to crash dump file if system parameter DUMPBUG=1
  - Reboots the system if parameter BUGREBOOT=1
  - Displays the following message if BUGREBOOT=0

    SYSTEM SHUTDOWN COMPLETE- USE CONSOLE TO HALT SYSTEM
- Do not stop system console when bugcheck information is being written, or crash dump file will not be written!

Exceptions

An exception is an event that alters normal instruction execution in system or user code.

- Exceptions are classified as:
  - Faults
  - Traps
  - Aborts
- Faults
  - Faults occur during execution of an instruction
  - Registers and memory are left in a consistent state
  - Instruction can be restarted, allowing the code to continue execution
- Traps
  - Traps occur after the execution of an instruction
  - Address of next instruction is placed in the PC, and execution continues
  - Some traps can be disabled
- Aborts
  - Aborts occur during the execution of an instruction
  - Register contents and memory are indeterminate
  - Instruction is usually not restartable
  - Execution cannot continue

- Unexpected, inner access mode exceptions cause bugchecks.
  - Some bugchecks are not fatal.
  - Fatal bugchecks cause a system crash.
- When the system reboots after a crash:
  - Use SDA to examine the crash dump.
  - See Example 6-2.
- Reference: *VMS System Dump Analyzer Reference Manual*

Table 6–3: Common Exceptions

Bugcheck	Description
FATALEXCPT	Fatal executive or kernel mode exception
INVEXCEPTN	Exception while above ASTDEL or on interrupt stack
SSRVEXCEPT	Unexpected system service exception
PGFIPLHI	Page fault with IPL too high

Example 6–2: Using SDA to Obtain Crash Information

```
1  $ ANALYZE/CRASH_DUMP SYSDUMP.DMP

   VAX/VMS System dump analyzer
   Dump taken on 24-AUG-1988 17:11:04.69
   SSRVEXCEPT, Unexpected system service exception

2  SDA> SHOW CRASH

   System crash information
   ------------------------
   Time of system crash: 24-AUG-1988 17:11:04.69
   Version of system: VAX/VMS VERSION V5.0
   System Version Major ID/Minor ID: 1/0
   System type: MicroVAX II
   Crash CPU ID/Primary CPU ID:  00/00
   Bitmask of CPUs active/available:  00000001/00000001
   CPU bugcheck codes:
    CPU 00 -- SSRVEXCEPT, Unexpected system service exception

   CPU 00 Processor crash information
   ----------------------------------
   CPU 00 reason for Bugcheck:
 SSRVEXCEPT, Unexpected system service exception

   Process currently executing on this CPU: SHERRY
   Current image file: $1$DJA0:[INSTRUCTORS.][SHERRY]CRASH1M.EXE;1
   Current IPL: 0  (decimal)
   CPU database address:  8053E000

   General registers:
     R0  = 00000000   R1  = 80002398   R2  = 00000004   R3  = 7FF93169
     R4  = 80273D80   R5  = 7FFE5EB4   R6  = 7FFED18A   R7  = 7FFED18A
     R8  = 7FFECA52   R9  = 7FFECC5A   R10 = 7FFED7D4   R11 = 7FFE2BDC
     AP  = 7FFE7794   FP  = 7FFE777C   SP  = 7FFE777C   PC  = 8000239E
     PSL = 00000000

3  SDA> SHOW SUMMARY/IMAGE
   Current process summary
   -----------------------
   Extended Indx Process name  Username State Pri   PCB      PHD    Wkset
   -- PID -- ---- ------------- -------- ----- --- -------- -------- -----
   00000021 0001 SWAPPER                HIB   16 801F2598 801F2400    0
   00000062 0002 SHERRY        SHERRY   CUR    6 80273D80 8054FE00  254
            $1$DJA0:[INSTRUCTORS.][SHERRY]CRASH1M.EXE;1
   00000024 0004 ERRFMT        SYSTEM   HIB    8 802798A0 80541200   96
     $1$DJA0:[SYS0.SYSCOMMON.][SYSEXE]ERRFMT.EXE;1
   00000025 0005 OPCOM         SYSTEM   HIB    8 80272EC0 8055EA00  121
     -- image name not available --
   00000026 0006 JOB_CONTROL   SYSTEM   HIB    8 8027A230 8056D600  429
     $1$DJA0:[SYS0.SYSCOMMON.][SYSEXE]JOBCTL.EXE;1
   00000027 0007 SYMBIONT_0001 SYSTEM   HIB    6 8028B250 8057C200   65
     -- image name not available --
   00000033 0013 NETACP        DECNET   HIB   10 8028D0A0 8058AE00  369
     $1$DJA0:[SYS0.][SYSEXE]NETACP.EXE;1
   00000034 0014 EVL           DECNET   HIB    6 8028DC30 80599A00   41
     -- image name not available --
```

Comments on Example 6-2

1. When you invoke SDA, it displays:

   a. An identification banner

   b. The date and time of the crash

   c. The bugcheck message

2. Enter the SHOW CRASH command to display crash information.

   a. Time of crash and VMS version

   b. Reason for bugcheck exception

   c. Process currently executing

   d. Current image file (the program that was executing at the time of the crash)

      Although a user program (CRASH1M.EXE) was executing, this program might not have been the actual cause of the crash. For example, CRASH1M.EXE could have been interrupted by system code (see comment about IPL below).

   e. Current IPL (interrupt priority level)

      Kernel mode software can run at elevated IPL, which allows it to interrupt programs running at lower IPL such as user programs.

   f. Contents of general registers at time of crash

3. The SDA SHOW SUMMARY/IMAGE command shows information similar to that shown by the DCL SHOW SYSTEM command.

Machine Checks

- CPU-specific exceptions, including:
  - Cache or parity error
  - CPU timeout
- Usually indicate hardware problems
- Logged in error log
- Symptoms indicating machine checks:
  - The error message SYSTEM-F-MCHECK
  - The fatal bugcheck MACHINECHK
  - A console halt code of 5
  - Machine check entries in the error log
  - DCL SHOW ERROR command indicates nonzero CPU errors
- What to do about machine checks:
  - `$ ANALYZE/ERROR/INCLUDE=MACHINE_CHECK ERRLOG.SYS`
  - For hardware problems, call Digital Field Service.
  - For software problems, call your system programmer.
  - When reporting problems, include the information below.

    The crash dump and output from these SDA commands:

    ```
        SHOW CRASH
        SHOW STACK
        SHOW SUMMARY
        SHOW PROCESS/ALL
    ```

    Error log report
    Console output
  - Save the crash dump file for future reference.

System Hangs

Symptom: the system does not respond to any terminal

Possible Causes of System Hangs

- System crash - check console for message indicating halt or bugcheck
- Console problems
  - Out of paper
  - Bad fuse
- Pagefile full
- Dynamic memory (pool) exhausted
- Execution of a compute-bound process
- System file is locked
- Booting problems
  - Pagefile badly fragmented or too small
  - SYSTARTUP_V5.COM contains errors
  - Pool too small
  - User authorization file (UAF) corrupt
  - Corrupt DCL tables
  - Hardware ECO revision level incorrect
  - 64 Kbytes of contiguous physical memory not available
- Hardware problems
- Disk undergoing mount verification (described in Module 4)
- VAXcluster problems (beyond the scope of this course)

Pagefile Full

- Possible messages displayed on console or user terminal include:
  - SYSTEM-W-PAGEFRAG,
    Pagefile badly fragmented, system continuing
  - SYSTEM-W-PAGECRIT,
    Pagefile space critical, system trying to continue
- To prevent system hang, determine which files are becoming full:

  ```
  $ SHOW MEMORY /FILES /FULL
  ```

  - In general, pagefiles should not be more than 50% full.
  - Install another pagefile.
  - Possibly increase SYSGEN parameter MPW_WAITLIMIT

    Limits rate at which any process can modify pages.
    If too small, modified page list can get too large.
    Must be >= MPW_HILIMIT, or system might hang.
  - Restart user processes or reboot system.
  - See Example 6-3.
- Reference: *VMS System Generation Utility Manual*

Example 6–3: Pagefile Problems

```
1 $ SHOW MEMORY /FILES /FULL

  System Memory Resources on  3-JAN-1989 16:18:31.34

   DISK$VAXVMSV50:[SYS0.SYSEXE]SWAPFILE.SYS
  Free Blocks             13032    Reservable Blocks       13032
  Total Size (blocks)     13032    Paging File Number          1
  Swap Usage (processes)      0    Paging Usage (processes)    0
  This file is used exclusively for swapping.

   DISK$VAXVMSV50:[SYS0.SYSEXE]PAGEFILE.SYS
  Free Blocks               999    Reservable Blocks       -7467
  Total Size (blocks)     29992    Paging File Number          3
  Swap Usage (processes)      0    Paging Usage (processes)   22
  This file can be used for either paging or swapping.

  $ RUN SYS$SYSTEM:SYSGEN

2 SYSGEN> SHOW MPW_WAITLIMIT

  Parameter Name  Current  Default Minimum Maximum  Unit Dynamic
  --------------  -------  ------- ------- -------  ---- -------
  MPW_WAITLIMIT       596      596       0   16384             D

2 SYSGEN> SET MPW_WAITLIMIT 16384
  SYSGEN> WRITE ACTIVE

3 SYSGEN> SHOW PAGFILCNT
  Parameter Name  Current  Default Minimum Maximum  Unit Dynamic
  --------------  -------  ------- ------- -------  ---- -------
  PAGFILCNT       2        2       1       63  Files

4 SYSGEN> CREATE <filespec> /SIZE=<size>
  SYSGEN> INSTALL <filespec> /PAGEFILE
5 SYSGEN> SET MPW_WAITLIMIT 596
6 SYSGEN> WRITE ACTIVE
  SYSGEN> EXIT
```

Comments on Example 6-3

1. This DCL command displays information about the usage of the paging and swapping file(s). If the **Free Blocks** counts for the paging files are very low, you must install a new paging file.

   Generally, the page and swap files should not be more than 50% full.

2. If the current size of the modified page list is approaching the value of the SYSGEN parameter MPW_WAITLIMIT, temporarily raise the parameter's value so that your own process will not be placed into a resource wait.

3. If the setting of the parameter PAGFILCNT is too low to allow installation of another page file:

   a. Have users log out.

   b. Use SYSGEN to alter PAGFILCNT.

   c. Shut down the system and reboot.

4. Create and install another page file.

5. Reset MPW_WAITLIMIT if you raised it.

6. The WRITE ACTIVE command writes the parameter values from the work area to the active system in memory.

System Dynamic Memory (Pool) Exhausted

- Some possible messages:
  - %SYSTEM-W-POOLEXPF, Pool expansion failure
  - %SYSTEM-W-INSFMEM, Insufficient dynamic memory
- Possible reasons:
  - Pool-related SYSGEN parameters need adjustment
  - The following user process quotas are too large:

    BYTLM
    BIOLM
    PRCLM
  - Not enough free physical memory
- Consider tradeoffs in adjusting pool-related parameters.
  - If value is too small, the system could hang or performance could degrade.
  - If value is too large, physical memory is wasted.

Table 6–4: Pool-Related SYSGEN Parameters

Parameter	Defines
NPAGEVIR	Maximum size to which NPAGEDYN can be increased
NPAGEDYN	Maximum size of nonpaged pool in bytes
PAGEDYN	Maximum size of paged pool in bytes
LRPCOUNTV	Upper limit to which LRPCOUNT can be increased
LRPCOUNT	Number of preallocated large request packets
IRPCOUNTV	Upper limit to which IRPCOUNT can be increased
IRPCOUNT	Number of preallocated intermediate request packets
SRPCOUNTV	Upper limit to which SRPCOUNT can be increased
SRPCOUNT	Number of preallocated small request packets

Example 6–4: Checking for Exhausted Pool

```
$ SHOW MEMORY /POOL /FULL /PHYSICAL

             System Memory Resources on  3-JAN-1989 16:39:20.57
Physical Memory Usage (pages):   Total      Free    In Use  Modified
 Main Memory (32.00Mb)          65536     25301     40182        53

Small Packet (SRP) Lookaside List        Packets     Bytes       Pages
  Current Total Size                        2288    219648         429
  Initial Size (SRPCOUNT)                   1024     98304         192
  Maximum Size (SRPCOUNTV)                  4096    393216         768
  Free Space                                 638     61248
  Space in Use                              1650    158400
  Packet Size/Upper Bound (SRPSIZE)                     96
  Lower Bound on Allocation                             32

I/O Request Packet (IRP) Lookaside List  Packets     Bytes       Pages
  Current Total Size                        2000    352000         688
  Initial Size (IRPCOUNT)                   2000    352000         688
  Maximum Size (IRPCOUNTV)                  8000   1408000        2750
  Free Space                                1376    242176
  Space in Use                               624    109824
  Packet Size/Upper Bound (fixed)                      176
  Lower Bound on Allocation                             97

Large Packet (LRP) Lookaside List        Packets     Bytes       Pages
  Current Total Size                          60     98880         194
  Initial Size (LRPCOUNT)                     60     98880         194
  Maximum Size (LRPCOUNTV)                   240    395520         773
  Free Space                                  23     37904
  Space in Use                                37     60976
  Packet Size/Upper Bound (LRPSIZE + 80)              1648
  Lower Bound on Allocation                           1088

Nonpaged Dynamic Memory
  Current Size (bytes)     11999744  Current Total Size (pages)   23437
  Initial Size (NPAGEDYN)  11999744  Initial Size (pages)         23437
  Maximum Size (NPAGEVIR)  35999744  Maximum Size (pages)         70312
  Free Space (bytes)       11057392  Space in Use (bytes)        942352
  Size of Largest Block    10993104  Size of Smallest Block          16
  Number of Free Blocks         132  Free Blocks LEQU 32 Bytes       14

Paged Dynamic Memory
  Current Size (PAGEDYN)    3697664  Current Total Size (pages)    7222
  Free Space (bytes)        3071984  Space in Use (bytes)        625680
  Size of Largest Block     3066704  Size of Smallest Block          16
  Number of Free Blocks          42  Free Blocks LEQU 32 Bytes       24

Of the physical pages in use, 35589 pages are permanently allocated
to VMS.
```

HARDLY EXHAUSTED!!!

You're unless have expanded pool!

- these #s are a little ~~up~~ inflated!!!
- not too realistic.

Comments on Example 6-4

1. This SHOW command helps to determine whether any of the nonpaged pool lists are becoming depleted. If so, consider altering the corresponding parameters and rebooting.

2. This command also displays information about the usage of each pool area, including:

   a. Amount of free space

   b. Expansion into virtual space

   c. Actual space in use

   d. Size of the largest contiguous block in each area

Compute-Bound Processes

- Symptoms: neither of the following has occurred for ten seconds:
  - Page fault
  - Direct or buffered I/O
- Possible solutions
  - Suspend the process
  - Control growth of the process
  - Submit compute-bound jobs to BATCH, to be run during off-hours

Example 6–5: A Compute-Bound Process

```
$ SHOW PROCESS /CONTINUOUS /ID=20402A2A
                   Process CLARK                      16:09:40
1   State               COM          Working set           331
    Cur/base priority   5/4          Virtual pages        2114
    Current PC          7FFEDF8A     CPU time      00:05:54.45
    Current PSL         03C00000     Direct I/O           3280   2
    Current user SP     7FF33D90     Buffered I/O        20954   2
    PID                 20402A2A     Page faults         42411   2
    UIC         [GROUP11,CLARK]      Event flags      E03D0036
                                                      F8000000
3   DRA0:[CLARK]COMPUTE.EXE
```

Comments on Example 6-5

1. Process state is COM.
2. A compute-bound process is characterized by the absence of the following operations for ten seconds or more:
   a. Direct I/O
   b. Buffered I/O
   c. Page fault
3. The name of the currently executing image is displayed.

Suspending a Compute-Bound Process

- Be sure it does not have a shared file locked (if so, other processes can become stalled).

  $ SHOW DEVICE/FILES

- To suspend a compute-bound process:

  $ SET PROCESS /PID=<pid> /SUSPEND=SUPERVISOR or /SUSPEND=KERNEL.
  SUPERVISOR suspend allows delivery of ASTs in EXEC or KERNEL mode.
  KERNEL suspend does not allow any ASTs to be delivered.

- If other processes become stalled, resume the compute-bound process.

  $ SET PROCESS /PID=<pid> /RESUME

Example 6–6: Files Used by a Compute-Bound Process

```
     $ SHOW DEVICE/FILES
     Files accessed on device DEMON_DUA1: on  2-APR-1988 15:00:37.53
     Process name      PID     File name
                     00000000  [000000]INDEXF.SYS;1
                     00000000  [000000]QUOTA.SYS;1
     Monitor         20402BD0  [CLARK.SYSMGT_II]MY_MONITOR.LOG;1
     Monitor         20402BD0  [CLARK.SYSMGT_II]MY_MONITOR.COM;2
     Monitor         20402BD0  [CLARK.SYSMGT_II]DUPER_MON.DAT;2
1    CLARK           20402A2A  [SHARE_DIR]SHARED_FILE.DAT;2
2    SMITH           20402157  [SHARE_DIR]SHARED_FILE.DAT;2
```

Comments on Example 6-6

1. Process CLARK, which is known to be compute-bound, is sharing a file with process SMITH.
2. If you suspend CLARK, SMITH may become stalled.

Controlling the Growth of a Compute-Bound Process

- If you want the process to run to completion, giving it a larger working set may reduce the required CPU time.
  - Larger working set reduces paging.
  - Paging uses substantial CPU time.
- Reducing the working set size will increase the process' page faulting, so it may not be appropriate if the process is already using too many resources.
- Try DCL commands to decrease working set size.

  ```
  $   SET WORKING_SET /EXTENT

  $   SET WORKING_SET /QUOTA
  ```

  Instruct the user to enter these commands before running the compute-bound process.
- If this achieves the desired effect, or if you need to increase the working set size, use Authorize to modify the corresponding UAF values.
  - UAF> MODIFY <username> /WSQUOTA
  - UAF> MODIFY <username> /WSEXTENT
- If the working set values were set by a system service:
  - Stop the program.

    CTRL/Y
    DCL STOP command
  - A programmer must modify the code before the program can be run again.

System Hangs Caused by Hardware Problems

Symptom: only the console terminal responds

- Possible causes:
  - No power to interconnect or adapter
  - Terminal controller not working
- Troubleshooting technique - refer to Module 7, "Hardware Troubleshooting"

Symptom: you cannot log in to any terminal or the console

- Possible causes:
  - System looping at high IPL, preventing user activity
  - System rebooting without having been shut down
  - Aborted fatal bugcheck because of:

    Corruption of system dump file
    Failure to initialize the system disk
    Continuous interrupts from malfunctioning device
- Troubleshooting technique:
  - Listen for console medium and/or system disk activity
  - If the system does not reboot automatically:

    Reboot it
    Examine the crash dump
    Examine the error log

Symptom: console does not respond to CTRL/P

- Possible causes:
  - Panel switch not in LOCAL ENABLE position
  - Console malfunctioning
- Troubleshooting technique:
  - Make sure switch is in LOCAL ENABLE position.
  - If console is malfunctioning:

    Power down, then power up the CPU and console.
    If that has no effect, call DIGITAL Field Service.
  - If the system is hung, use the console crash procedure to force a crash.

Symptom: all system activity appears to stop

- Possible cause - the system disk might become write-locked because:
  - Someone inadvertently pressed the write-protect button.
  - The device has gone off-line.
  - Someone is performing either of the following operations:

    ```
    DISKQ> REBUILD
    ANALYZE /DISK /REPAIR
    ```
  - The port select switch is in the wrong position.
- Troubleshooting technique:
  - Check for any of the possible causes listed above.
  - Take appropriate action.

Booting Problems

Symptom: system hangs during boot

- Possible causes:
  - SYSTARTUP_V5.COM contains errors.
  - Pagefile badly fragmented or too small.
  - Pool too small.
  - 64 Kbytes of contiguous physical memory are not available.

    Try powering down and back up.
    Might have been left in an inconsistent state after a power failure.
- Troubleshooting steps:
  - Initiate a conversational boot.
  - If you get the SYSBOOT prompt, the console subsystem is working.
  - Use SYSBOOT to set STARTUP_P1 to MIN to initiate a minimum boot, which bypasses SYSTARTUP_V5.COM.

    Later, change it back to blank or you will always get a minimum boot.
  - Continue.
  - If a minimum boot correctly enables the console and system disk, it indicates a problem with SYSTARTUP_V5.COM.
  - Use SET VERIFY to debug SYSTARTUP_V5.COM.

Symptom: minimum boot did not work, but there is no apparent problem with SYSTARTUP_V5.COM

- Troubleshooting steps:
  - — Initiate a conversational boot.
  - — Use SYSBOOT to check the sizes of:

    MAXPROCESSCNT
    Pool-related parameters
    Pagefile
  - — If necessary, use SYSBOOT to alter the appropriate parameters or pagefile.
  - — Attempt to reboot.

Symptom: 'FPLA VERSION MISMATCH' error message

- Possible cause: hardware ECO revision level does not match software (for example, when you get a new version of VMS).
- What to do: call Digital Field Service.
- Temporary solution: use your old console medium instead of the new one.

Symptom: system hangs during boot

- Possible cause: DCL tables are corrupt, possibly because of an aborted SET COMMAND/TABLES operation.
- Possible solution:
  - Boot from a different system disk.
  - Mount corrupt disk in another drive.
  - Copy DCL tables from good disk to corrupt disk, or restore them from a backup copy.

Symptom: system boots, but no one can log in

- Possible cause: user authorization file (UAF) is corrupt.
- Possible solution:
  - An alternate UAF named SYS$SYSTEM:SYSUAFALT must already exist.
  - Initiate a conversational boot.
  - Use SYSBOOT to set UAFALTERNATE to 1.
  - After the system boots, restore the primary UAF from backup.

Example 6–7: Using an Alternate UAF

```
SYSBOOT> set UAFALTERNATE 1
SYSBOOT> continue
          .
          .
          .
 (VMS operating system bootstraps)
 (Log in under the SYSTEM account)
          .
          .
          .
$ Backup...                    ! Restore a good copy of SYSUAF.DAT
$
$ Deassign SYSUAF              ! Stop using the alternate UAF
$
$ Run SYS$SYSTEM:SYSGEN        ! Disable use of alternate on next boot.
SYSGEN> set UAFALTERNATE 0
SYSGEN> exit

   NOTE:  Setting UAFALTERNATE to 1 on a system that has no
   SYSUAFALT.DAT will force VMS to accept ANY username
   and ANY password as valid, but only from OPA0.  A
   process logged in this way will autmatically have a
          UIC of [1,4] and ALL privileges enabled.
```

PROCESS HANGS

- Does the process in question still exist?

  ```
  $ SHOW SYSTEM
  ```

- If the process no longer exists:
  - It might have been deleted as a result of:

    A nonfatal bugcheck
    A malicious or mistaken user with privileges
    A problem occurring in the process

  - If process accounting is enabled, determine process exit status as follows:

    ```
    $ ACCOUNTING /SINCE=<today> /FULL /USER=<username>
    ```

  - To determine if the process was deleted after a nonfatal bugcheck:

    ```
    $ ANALYZE /ERROR /INCLUDE=BUGCHECKS /SINCE=<today>
    ```

- If the process still exists, it might be:
  - Awaiting resources
  - Hung during login
  - Awaiting completion of terminal I/O
  - Outswapped, unable to be inswapped
  - Too low in priority
  - Executing an infinite loop

Example 6–8: Using Accounting to Display Reason for Process Termination

```
     $ ACCOUNTING /SINCE=TODAY /FULL /USER=CLARK

     DETACHED Process Termination
     ----------------------------

     Username:     CLARK       UIC:           [GROUP1,SYSTEM]
2    Account:      VMS         Finish time:   2-APR-1989 11:47:28.07
2    Process ID:   204028B6    Start time:    2-APR-1989 11:33:35.50
     Owner ID:                 Elapsed time:           0 00:14:08.57
     Terminal name:            Processor time:         0 00:05:03.44
     Remote node addr:         Priority:           4
     Remote node name:         Privilege <31-00>: DFFFFFFF
     Remote ID:                Privilege <63-32>: FFFFFFFF
     Queue entry:              Final status code: 000184C4
     Queue name:
     Job name:

1    Final status text: %RMS-F-DEV, error in device name or
        inappropriate device type

     Page faults:          95        Direct IO:             0
     Page fault reads:      2        Buffered IO:           5
     Peak working set:    113        Volumes mounted:       0
     Peak page file:      679        Images executed:       1
```

Comments on Example 6-8

1. The process exit status indicates success or failure.
2. Process start and finish times (right-hand column) might be helpful if you want to know how long the process ran, or approximately when it terminated.

Process Awaiting Resources

- Sometimes processes must wait for:
  - Various system resources
  - Mutexes, which are semaphores that control access to data structures or other shared resources
- $ SHOW SYSTEM indicates process wait state.
  - Resource wait: various states (see Table 6-5)
  - Mutex wait: MWAIT state
  - Local event flag wait: LEF or LEFO state
- Troubleshooting technique:
  - Use $ SHOW SYSTEM or SDA to determine the type of wait (see Example 6-9).
  - You may not be able to determine exactly what caused the wait. This often requires knowledge of VMS internals and data structures.
- It may be necessary to do one of the following:

  Abort the image.
  Delete the process.
  Shut down and reboot the system.
- A process in MWAIT state cannot be deleted.

Table 6–5: Some Resource Wait States

State	Waiting for	Comment
RWAST	Delivery of an AST	Very general, most common
RWMBX	Mailbox space	BYTLM might be too small
RWNPG	Nonpaged pool space	Check pool-related parameters
RWPAG	Paged pool space	Check pool-related parameters
RWSWP	Swapping file space	Check swapfile usage
RWMPB	Modified page writer busy	MPW_WAITLIMIT too small

Table 6–6: Some Mutex Wait States

State	Possibly Waiting for
MWAIT	Common event block list
MWAIT	I/O database
MWAIT	Paged dynamic memory
MWAIT	Global section descriptor list
MWAIT	Shared memory global section descriptor list
MWAIT	Shared memory mailboxes

Table 6–7: Reasons for LEF and LEFO Waits

Process Awaiting	Possible Reasons
I/O completion	Normal wait for I/O completion Device failure or saturation Disk undergoing mount verification Software error
A lock grant	Another process has an incompatible lock
Timer expiration	Incorrect expiration time specified
Asynchronous system service completion	Completion is signaled by the setting of an event flag but the flag has not been set yet

Example 6–9: Determining the Reason for a Resource Wait (Sheet 1 of 2)

```
1  $ SHOW SYSTEM
   VAX/VMS V5.0-2 on node BAROOM 3-JAN-1989 17:11:48.50 Uptime 5 09:38:19
    Pid     Process Name    State Pri    I/O       CPU       Page flts Ph.Mem
  20200101 SWAPPER          HIB   16       0   0 00:00:02.73        0      0
  20200106 ERRFMT           HIB    8    7898   0 00:00:13.05       77    114
  20200109 OPCOM            HIB    6    2024   0 00:00:05.78      532    220
  2020010A JOB_CONTROL      HIB    8   17197   0 00:00:54.49      213    445
  2020010C SMISERVER        HIB    9     464   0 00:00:00.82      329    928
  20200118 NETACP           HIB   10    3349   0 00:01:00.36      263    455
  20200119 EVL              HIB    6     182   0 00:00:00.73   121799     60 N
  2020011F REMACP           HIB    9      24   0 00:00:00.06       78     58
  2020019F Dr Science       HIB    8   18957   0 00:00:48.66    11022    355
  202001A0 GONZO            RWAST  6      28   0 00:00:00.07       62     95

2  $ SET PROCESS/PRIV=CMKRNL
   $ ANALYZE/SYSTEM
   VAX/VMS System analyzer

3  SDA> SHOW SUMMARY
   Current process summary
   -----------------------
   Extended Indx Process name Username State Pri   PCB      PHD     Wkset
   -- PID -- ---- ----------- -------- ----- --- -------- -------- -----
   20200101 0001 SWAPPER                 HIB   16 80215D98 80215C00     0
   20200106 0006 ERRFMT        SYSTEM    HIB    8 806E8C60 82CE6E00   114
   20200109 0009 OPCOM         SYSTEM    HIB    6 806E9C40 82EB8E00   220
   2020010A 000A JOB_CONTROL   SYSTEM    HIB    8 806EA1C0 82F2D600   445
   2020010C 000C SMISERVER     AUDREY    HIB    9 806F19B0 83016600   928
   20200118 0018 NETACP        DECNET    HIB   10 8070DF60 833BA600   455
   20200119 0019 EVL           DECNET    HIB    6 8070F920 8342EE00    60
   2020011F 001F REMACP        SYSTEM    HIB    9 8072B280 831E8600    58
   2020019F 009F Dr Science    ART       COM    9 8069C850 82D5B600   360
   202001A0 00A0 GONZO         SHERRY    RWAST  6 806974E0 8325CE00    95
```

Comments on Example 6-9

1. The output from the DCL SHOW SYSTEM command indicates that the process GONZO is in the RWAST state (refer to the column labeled **State**).

2. To analyze the processes on a running system, use the ANALYZE/SYSTEM command.

3. The SDA SHOW SUMMARY command shows some of the same information as the DCL SHOW SYSTEM command.

Example 6-9 : Determining the Reason for a Resource Wait (Sheet 2 of 2)

```
4    SDA> SET PROCESS GONZO
5    SDA> SHOW PROCESS /PCB

     Process index: 00A0   Name: GONZO   Extended PID: 202001A0
     ----------------------------------------------------------
     Process status:  00140001    RES,PHDRES,LOGIN

     PCB address              806974E0  JIB address              8293CB60
     PHD address              8325CE00  Swapfile disk address    00000000
     Master internal PID      000100A0  Subprocess count                0
     Internal PID             000100A0  Creator internal PID     00000000
     Extended PID             202001A0  Creator extended PID     00000000
     State                       RWAST  Termination mailbox          0000
     Current priority                9  AST's enabled                KESU
     Base priority                   4  AST's active                 NONE
     UIC                [00100,000043]  AST's remaining                24
6    Mutex count                     0  Buffered I/O count/limit     0/18
     Waiting EF cluster              0  Direct I/O count/limit      18/18
     Starting wait time       1B001B1B  BUFIO byte count/limit  896/6896
     Event flag wait mask     00000001  # open files allowed left      15
     Local EF cluster 0       E0000000  Timer entries allowed left      8
     Local EF cluster 1       00000000  Active page table count         0
     Global cluster 2 pointer 00000000  Process WS page count          95
     Global cluster 3 pointer 00000000  Global WS page count            0

     SDA> EXIT
```

Comments on Example 6-9 (Cont)

4. The SDA SET PROCESS command makes the specified process (GONZO) the current process.

5. The SDA SHOW PROCESS/PCB command lists the data contained in the software process control block (PCB) for the current process.

6. The Buffered I/O count/limit and the Direct I/O count/limit (in the column on the right) indicate whether the process has any outstanding I/O requests.

   Since the number of buffered I/Os available is zero (indicated by 0/18), the process has eighteen outstanding buffered I/O requests. The process can also be awaiting a nineteenth buffered I/O, thus causing the RWAST state.

   Since the two numbers for Direct I/O are the same (18/18), the process has no outstanding direct I/O requests.

Processes Hung During Login

Symptom: users attempting to log in get no response, while users already logged in have no apparent problems

- Possible cause: a process has the UAF locked.
- Troubleshooting technique:
  - Use SDA to determine whether the hung process is awaiting a locked resource.
  - Use SDA to determine to which process the lock was granted.
  - If possible, stop the process that holds the lock.

Example 6–10: Troubleshooting Processes Hung During Login (Sheet 1 of 5)

```
       $ ANALYZE /SYSTEM
       SDA> show summary
       Current process summary
       -----------------------
        Extended Indx Process name    Username    State Pri   PCB
        -- PID -- ---- --------------- ----------- ----- --- --------
        21200080 0000 NULL                        CUR     0 800024A8
        21200081 0001 SWAPPER                     HIB    16 80002748
        21200085 0005 ERRFMT          SYSTEM      HIB     8 8016F170
        21200086 0006 CACHE_SERVER    SYSTEM      HIB    16 8017EF30
        21200088 0008 OPCOM           SYSTEM      LEF     8 8017F5C0
        21200089 0009 JOB_CONTROL     SYSTEM      HIB     8 8017F6E0
        2120008E 000E DBMS_MONITOR    SYSTEM      LEF     6 801A2340
        2120008F 000F ACMS_SWL        SYSTEM      HIB     8 801A08F0
        21200090 0010 VAXsim_Monitor  SYSTEM      HIB     8 801A0290
        21200091 0011 NETACP          DECNET      HIB    10 801A6D80
        21200092 0012 EVL             DECNET      HIB     6 801A7420
        21200093 0013 REMACP          SYSTEM      HIB     8 801ADC20
        21200096 0016 MAIL_2050       DECNET      LEF     6 8017EE10
        212000A6 0026 ACMS01ACC001000 SYSTEM      LEF     6 801CFD40
2       212000A7 0027 _LTA10:         <login>     LEF     4 801D05B0
        212000A8 0028 RDMS_MONITOR    ARCHDEACON  LEF    15 801D1910
        212000A9 0029 MAIL_1043       DECNET      LEF     4 801D0E20
1       212000AA 002A _LTA11:         <login>     LEF     4 801CEE70
1       212000AB 002B _LTA12:         <login>     LEF     5 801D21D0
        212000AD 002D MAIL_4117       DECNET      LEF     4 801D3580
1       212000AE 002E _LTA13:         <login>     LEF     4 801D0A80
1       212000B0 0030 _LTA14:         <login>     LEF     4 801A82E0
```

Comments on Example 6-10

1. Several processes attempting to log in are hung. The users are not getting any response after entering USERNAME.

2. This example examines one of these hung processes, process number 212000A7; its process index is 0027.

Example 6-10 Troubleshooting Processes Hung During Login (Sheet 2 of 5)

```
1     SDA> show process/index=0027/locks
      Process index: 0027   Name: _LTA10:   Extended PID: 212000A7
      ------------------------------------------------------------
      Lock data:
1     Lock id:  0023000A   PID:     00010027 Flags: VALBLK SYNCSTS SYSTEM
2,3   Par. id:  00000000   Waiting for   PW
      Sublocks:        0
      LKB:      8032BBC0   BLKAST
      Resource:        00010B1A 24534D52    RMS$....  Status:  ASYNC
       Length   26     435F4548 54020000    ...THE_C
       Exec. mode      00202020 544E554F    OUNT   .
       System          00000000 00000000    ........
      Local copy
```

Comments on Example 6-10 (Cont)

1. The SHOW PROCESS/LOCKS command allows you to examine the locks associated with the hung process. Locks are used to synchronize the use of shared resources such as the system UAF.

2. One lock (lock ID 0023000A) is associated with process 0027. Process 0027 requested this lock in order to gain access to a shared resource.

3. Notice that process 0027 is **waiting** for this PW (protected-write) lock to be granted. This indicates that another process has already requested, and been granted, a lock on the same resource desired by process 0027.

   Therefore, process 0027 must wait until the other process releases its lock. On the following page (sheet 3 of this example), notice that this lock (ID 0023000A) is listed among the locks in the waiting queue.

Example 6-10 Troubleshooting Processes Hung During Login (Sheet 3 of 5)

```
1    SDA> show resource/lockid=0023000A
     Resource database
     -----------------
     Address of RSB:  802A3380  Group grant mode:       PW
     Parent RSB:      00000000  Conversion grant mode:  PW
     Sub-RSB count:          8  BLKAST count:            1
     Value block:    00000166 00000165 00000000 00000002  Seq. #: 0000005A
     Resource:       00010B1A 24534D52    RMS$....
      Length   26    435F4548 54020000    ...THE_C        CSID: 00000000
      Exec. mode     00202020 544E554F    OUNT   .        Directory entry
      System         00000000 00000000    ........
3    Granted queue  (Lock ID / Gr mode):
          01D5026C  PW       004201F7  NL
     Conversion queue  (Lock ID / Gr/Rq mode):
          008E0086  NL/PW    006C0268  NL/PW    009C026F  NL/PW
          00330234  NL/PW    003301DD  NL/PW    001E0287  NL/PW
2    Waiting queue  (Lock ID / Rq mode):
          00230249  PW       0023000A  PW       0011028B  PW
          00230299  PW       00140251  PW       002C01F0  PW
     Resource database
     -----------------
          0073007A  PW       0041027B  PW       005301E2  PW
          00170215  PW       000402A0  PW       0017027F  PW
```

Comments on Example 6-10 (Cont)

1. The SHOW RESOURCE/LOCKID command displays information about the lock (ID 0023000A) that process 0027 has requested.

2. Information about the lock queues is displayed. Notice that lock 0023000A is in the Waiting queue, waiting to be granted. However, it cannot be granted until the blocking lock is released.

3. To determine which process owns the lock that is blocking process 0027, look at the locks listed in the Granted queue. There is one NL (null) lock, which is not likely to be the culprit.

   There is only one PW lock granted (lock ID 01D5026C), and this is probably the lock that is blocking process 0027. The following page shows how to examine this lock. Note that if several locks were listed in the Granted queue, the troubleshooting task would be more complex because you would need to examine each lock.

Example 6-10: Troubleshooting Processes Hung During Login (Sheet 4 of 5)

```
1    SDA> show lock 01D5026C
     Lock database
     -------------

2    Lock id:  01D5026C   PID:    00010026  Flags: VALBLK SYNCSTS SYSTEM
     Par. id:  00000000   Granted at    PW
     Sublocks:        2
     LKB:      8032BB00   BLKAST
     Resource:      00010B1A 24534D52    RMS$....  Status:  ASYNC  BLASTQD
      Length   26   435F4548 54020000    ...THE_C
      Exec. mode    00202020 544E554F    OUNT   .
      System        00000000 00000000    ........
     Local copy
```

Comments on Example 6-10 (Cont)

1. The SHOW LOCK command is used to examine lock 01D5026C, which is blocking process 0027.

2. This lock has been granted to the process identified by PID 00010026.

Example 6-10: Troubleshooting Processes Hung During Login (Sheet 5 of 5)

```
1    SDA> show summary
     Current process summary
     -----------------------
      Extended Indx Process name    Username    State Pri  PCB
     -- PID -- ---- --------------- ----------- ----- --- --------
      21200080 0000 NULL                        CUR     0 800024A8
      21200081 0001 SWAPPER                     HIB    16 80002748
      21200085 0005 ERRFMT          SYSTEM      HIB     8 8016F170
      21200086 0006 CACHE_SERVER    SYSTEM      HIB    16 8017EF30
      21200088 0008 OPCOM           SYSTEM      LEF     8 8017F5C0
      21200089 0009 JOB_CONTROL     SYSTEM      HIB     8 8017F6E0
      2120008E 000E DBMS_MONITOR    SYSTEM      LEF     6 801A2340
      2120008F 000F ACMS_SWL        SYSTEM      HIB     8 801A08F0
      21200090 0010 VAXsim_Monitor  SYSTEM      HIB     8 801A0290
      21200091 0011 NETACP          DECNET      HIB    10 801A6D80
      21200092 0012 EVL             DECNET      HIB     6 801A7420
      21200093 0013 REMACP          SYSTEM      HIB     8 801ADC20
      21200096 0016 MAIL_2050       DECNET      LEF     6 8017EE10
2     212000A6 0026 ACMS01ACC001000 SYSTEM      LEF     6 801CFD40
      212000A7 0027 _LTA10:         <LOGIN>     LEF     4 801D05B0
      212000A8 0028 RDMS_MONITOR    ARCHDEACON  LEF    15 801D1910
      212000A9 0029 MAIL_1043       DECNET      LEF     4 801D0E20
      212000AA 002A _LTA11:         <LOGIN>     LEF     4 801CEE70
      212000AB 002B _LTA12:         <LOGIN>     LEF     5 801D21D0
      212000AD 002D MAIL_4117       DECNET      LEF     4 801D3580
      212000AE 002E _LTA13:         <LOGIN>     LEF     4 801D0A80
      212000B0 0030 _LTA14:         <LOGIN>     LEF     4 801A82E0
      SDA> exit

3     $ STOP/ID=212000A6
```

Comments on Example 6-10 (Cont)

1. The SHOW SUMMARY command is used to find the process with an index value of 0026 (the last four digits of the PID indicated by the SHOW LOCK command).

2. Process 0026 (PID 212000A6) is a system process.

3. After exiting SDA, the DCL STOP command is used to stop process 212000A6.

   After that process was stopped, all the users who were hung in login were able to log in successfully.

Process Waiting for Terminal I/O

Symptoms:

- SDA> SHOW PROCESS/CHANNEL indicates terminal busy.
- User has not disabled broadcasts to terminal.
- **Keyboard Locked** and **Hold Screen** indicators are not on.
- Terminal RESET function has no effect.
- You have checked for bad fuse and/or paper out.
- Terminal is on-line (not in local mode).
- You have used $ SHOW TERMINAL to determine that the hardware and software characteristics are consistent:
  - Baud rate.
  - Terminal type.
  - SET TERM/INQUIRE in LOGIN.COM can be helpful.

Troubleshooting a Process Awaiting Terminal I/O

Example 6–11: Troubleshooting a Process Awaiting Terminal I/O

```
1     SDA> SET PROCESS/INDEX=002A
      SDA> SHOW PROCESS/CHANNEL

      Process index: 002A   Name: CLARK   Extended PID: 20402A2A
      ----------------------------------------------------------
                              Process active channels
                              -----------------------

      Channel  Window    Status  Device/file accessed
      -------  ------    ------  --------------------
        0010 00000000           DEMON$DUA1:
        0020 8034DC60           DRA0:[SYS0.SYSEXE]RTPAD.EXE;1
        0030 8033F560           DRA0:[SYS0.SYSLIB]LIBRTL.EXE;4
        0060 00000000   Busy    MBA7448:
        0090 80363020           DEMON$DUA1:(12731,13,0)
2       00A0 00000000   Busy    LTA648:
        00D0 801C87D0   Busy    NET6442:
      SDA> EXIT
```

Comments on Example 6-11

1. Before entering these SDA commands, use SDA> SHOW SUMMARY to obtain the process index of the process in question.

2. This display indicates that the process is awaiting terminal I/O on terminal LTA648. Terminal names are usually prefaced by one of the following:

   a. LTA

   b. TTx

   c. RTA (remote terminal)

   d. VTA (virtual terminal)

The process can have other active channels accessing:

- Executable images
- Data files
- Mailboxes (prefaced with MBA)
- Network links (prefaced with NET)

- Use the REPLY command to check terminal connection:

```
$ REPLY /TERM=<terminal> <some message>
```

  Connection is good if the message appears on the terminal. If the message does not appear, try the following:

- Look for related hardware errors.

```
$ SHOW ERROR
```

- Put terminal in local mode and try typing (to see if it works).
- Check any connections through a switch or LAT.
- Switch the plug with a good terminal to test whether the problem is in the terminal or the cable.
- If there is a problem with the cable, replace the cable.

Outswapped Processes Unable To Be Inswapped

Symptoms:

- $ SHOW SYSTEM shows process scheduling state is COMO
- $ SHOW MEMORY shows:
  - No free balance set slots
  - Free page count < 500

Troubleshooting technique:

- If a reboot is not acceptable, try stopping unimportant processes to free more balance set slots and memory.
- Try increasing SYSGEN parameter BALSETCNT.
- BALSETCNT should never be greater than MAXPROCESSCNT - 2.
- Shut down system.
- Reboot.

You might need more memory:

Example 6–12: Troubleshooting Outswapped Processes (Sheet 1 of 2)

```
$ SHOW SYSTEM

VAX/VMS V5.0-2 on node BAROOM 3-JAN-1989 17:11:48.50 Uptime 5 09:38:19
  Pid    Process Name    State Pri     I/O      CPU       Page flts Ph.Mem
20200101 SWAPPER          HIB   16       0  0 00:00:02.73        0      0
20200106 ERRFMT           HIB    8    7898  0 00:00:13.05       77    114
20200109 OPCOM            HIB    6    2024  0 00:00:05.78      532    220
2020010A JOB_CONTROL      HIB    8   17197  0 00:00:54.49      213    445
2020010C SMISERVER        HIB    9     464  0 00:00:00.82      329    928
20200118 NETACP           HIB   10    3349  0 00:01:00.36      263    455
20200119 EVL              HIB    6     182  0 00:00:00.73   121799     60 N
2020011F REMACP           HIB    9      24  0 00:00:00.06       78     58
2020019F Roger Rabbit     CUR    8   18957  0 00:00:48.66    11022    355
202001C5 Spaceman Spiff   COMO   9    4965  0 00:01:09.41    13560    230
202001D3 BODINE           COMO   9    9608  0 00:04:23.49   158962    208
202001D5 GONZO            COMO   6      28  0 00:00:00.07       62     95
    .        .              .    .      .   .      .              .      .
    .        .              .    .      .   .      .              .      .
    .        .              .    .      .   .      .              .      .
```

Example 6-12: Troubleshooting Outswapped Processes (Sheet 2 of 2)

```
    $ SHOW MEMORY
                 System Memory Resources on  3-JAN-1989 18:34:29.46

    Physical Memory Usage (pages):    Total      Free    In Use  Modified
  1   Main Memory (32.00Mb)           65536        50     65402        84

    Slot Usage (slots):               Total      Free  Resident   Swapped
      Process Entry Slots               100         3        60        37
  2   Balance Set Slots                  60         0        60         0

    Fixed-Size Pool Areas (packets): Total      Free    In Use      Size
      Small Packet (SRP) List          6400      2439      3961        96
      I/O Request Packet (IRP) List    2000      1359       641       176
      Large Packet (LRP) List            60        22        38      1648

    Dynamic Memory Usage (bytes):     Total      Free    In Use   Largest
      Nonpaged Dynamic Memory      11999744  11056528    943216  10993104
      Paged Dynamic Memory          3697664   3071648    626016   3066704

    Paging File Usage (pages):                  Free  Reservable    Total
      DISK$VAXVMSV50:[SYS0.SYSEXE]SWAPFILE.SYS  13032       13032    13032
      DISK$VAXVMSV50:[SYS0.SYSEXE]PAGEFILE.SYS  27036       -7659    29992

    Of the physical pages in use, 35589 pages are permanently allocated
    to VMS.
```

Comments on Example 6-12

1. There are very few free pages of main memory.

2. There are no free balance set slots. When there are no free balance set slots and a very low free page count, outswapped processes cannot be inswapped.

   The total number of balance set slots is controlled by the SYSGEN parameter BALSETCNT.

Processes with Priority Too Low

Symptoms: $ SHOW SYSTEM indicates:

- Process scheduling state is COM or COMO.
- Process priority is very low compared to other processes.

Troubleshooting technique:

- Use an account with ALTPRI and WORLD privileges.
- $ SET PROCESS /ID=<pid> /PRIORITY=<new-priority>.

Process Executing an Infinite Loop

Symptoms:

- Scheduling state is COM.
- Process priority is not the problem.
- CPU time increases.

  ```
  $ SHOW PROCESS /CONTINUOUS /ID=<pid>
  ```

Troubleshooting techniques:

- If the process is looping in user mode, the user can invoke the debugger and try to trace the loop:
  - CTRL/C
  - DEBUG

  This will not work if the program was installed with privilege.

- You can also try the following:
  - `$ SHOW PROCESS /CONTINUOUS /ID=<pid>`
  - Write down PC and PSL values.
  - Suspend the process for later examination:

    ```
    $ SET PROCESS /SUSPEND /ID=<pid>
    ```
  - Attempt further investigation using SDA. This requires knowledge of:

    The program in question
    VAX/VMS internals and data structures

Example 6–13: Troubleshooting a Process in Infinite Loop

```
$ SHOW PROCESS /CONTINUOUS /ID=20402A2A
                  Process CLARK                    16:09:40

    State             COM          Working set          331

    Cur/base priority 5/4          Virtual pages       2114

1   Current PC        7FFEDF8A     CPU time     00:05:54.45    2

1   Current PSL       03C00000     Direct I/O          3280    3

1   Current user SP   7FF33D90     Buffered I/O       20954

    PID               20402A2A     Page faults        42411

    UIC        [GROUP11,CLARK]     Event flags     E03D0036
                                                   F8000000
```

Comments on Example 6-13

1. When executing an infinite loop, the values of the current PC, PSL, and user SP will seem to change. However, you notice that the same few values continue to rotate.

2. If CPU time does not increase, there may be a higher priority compute-bound process.

3. If the Direct I/O value increases rapidly, this can indicate that the program is writing large amounts of data to disk, potentially causing disk utilization problems.

SUMMARY

Table 6–8: Summary of System Crashes

Cause of System Crash	Troubleshooting Tools
Exception	SDA
Machine check	Error logger

Table 6–9: Summary of System Hangs

Cause of System Hang	Troubleshooting Tools
System crash	SDA
Console problem	Physically check the hardware
Pagefile full	SYSGEN SHOW MEMORY /FILES /FULL
Dynamic memory exhausted	SYSGEN SHOW MEMORY /POOL /PHYSICAL
Compute-bound process	SHOW PROCESS /CONTINUOUS SYSGEN SET PROCESS SET WORKING_SET AUTHORIZE
Disk mount verification	SYSGEN Cancellation routine DISMOUNT /ABORT
Hardware problems	SDA Error logger Check hardware settings Call Digital Field Service
Booting problems	Conversational boot Minimum boot SYSBOOT Alternate UAF Replace corrupt DCL tables
VAXcluster problems	Beyond the scope of this course

Table 6–10: **Summary of Process Hangs**

Cause of Process Hang	Troubleshooting Tools
Process deleted	SHOW SYSTEM ACCOUNTING
Awaiting resources	SHOW SYSTEM SDA
Necessary resource locked by another process	SDA STOP /ID=<pid>
Awaiting completion of terminal I/O	Physically check the terminal SDA SHOW TERMINAL REPLY SHOW ERROR
Process outswapped, unable to be inswapped	SHOW SYSTEM SHOW MEMORY /SLOTS SYSGEN
Priority too low	SHOW SYSTEM SET PROCESS /PRIORITY
Infinite loop	SHOW PROCESS SET PROCESS /SUSPEND SDA

APPENDIX A

VMS TOOLS USED IN TROUBLESHOOTING SOFTWARE PROBLEMS

System Dump Analyzer (SDA)

- Helps find cause of system failure by allowing you to:
  - Display the date, time, and reason for the crash
  - Identify the process executing at the time of the crash
  - Examine memory locations in any process
  - Display contents of system and process data structures
  - Display information about system resources
- Uses data in the system dump file

  Default: SYS$SYSTEM:SYSDUMP.DMP
- Can also be used to analyze a running system
- Invoke SDA:
  - To analyze a dump file: $ ANALYZE/CRASH_DUMP filespec
  - To analyze a running system: $ ANALYZE/SYSTEM
- Reference: *VMS System Dump Analyzer Manual*

Table 6–11: Commonly Used SDA Commands

Command	Function
SHOW CRASH	Displays general crash information
SHOW STACK	Displays current contents of stacks
SHOW PROCESS	Displays software and hardware context of any process in the balance set
SHOW SUMMARY	Lists all active processes
SHOW POOL/SUMMARY	Displays a summary of pool contents and usage
SET LOG	Records SDA output in a log file

REQUIREMENTS FOR RUNNING SDA

- Privileges:
  - CMKRNL required to analyze a running system.
  - SYSPRV usually required to analyze a dump file.
- You need read access to the dump file.
- For a full crash-dump (DUMPSTYLE=0), the file must be at least the size of physical memory plus the value of the SYSGEN parameter, ERRORLOGBUFFERS.
- For a partial crash-dump (DUMPSTYLE=1), AUTOGEN will calculate an appropriate file-size.
- System parameter DUMPBUG must be set at time of crash.
- SYSGEN parameter VIRTUALPAGECNT must be at least the size of the dump file plus approximately 3000 pages.
  - This value (3000) can change in future releases.
  - Symptom of VIRTUALPAGECNT too small: SDA cannot read in the symbol table.
  - Keep increasing it by 1000 until it is large enough.
- The process pagefile quota (PGFLQUOTA in the UAF) must be equal to the value of VIRTUALPAGECNT as calculated above.

ERROR LOGGER

- System automatically writes messages to the error log file:

  ```
  SYS$ERRORLOG:ERRLOG.SYS
  ```

- Recorded events include:
  - Device errors and timeouts
  - Machine checks
  - Bus errors
  - Memory errors
  - Bugchecks
  - Volume mounts and dismounts
  - System failure (crash)
- The Error Log utility processes error log entries
- This utility produces various types of output, including:
  - Full report of selected entries
  - Brief report of selected entries
  - Summary report
- Invoking: $ ANALYZE /ERRORLOG [/qualifiers]
- Reference: *VMS Error Log Utility Manual*

Table 6–12: Error Log Command Qualifiers

Qualifier	Function
/BRIEF	Produces a brief report
/EXCLUDE	Excludes errors generated by specified device and/or entry types from the report
/INCLUDE	Includes errors generated by specified device and/or entry types in the report
/OUTPUT	Specifies an output file for the report
/SINCE	Includes only the entries dated later than the specified date in the report

DCL SET AND SHOW COMMANDS

- SHOW displays information about the status of:
  - The system
  - The processes
  - The devices
- SET commands define process characteristics.
- Reference: *VMS DCL Dictionary*

Table 6–13: SET and SHOW Commands for Software Troubleshooting

Command	Function
SET PROCESS/PRIORITY	Sets process priority
SET PROCESS/SUSPEND	Suspends a process
SET PROCESS/RESUME	Resumes a suspended process
SET WORKING_SET	Sets process working set size
SHOW DEVICE	Displays the status of a device
SHOW DEVICE/FILES	Lists all files open on a volume and their associated process name and PID
SHOW ERROR	Displays error count for the CPU, memory, and physical devices
SHOW MEMORY	Displays the availability and usage of memory-related resources
SHOW MEMORY/FILES	Displays usage of each paging and swap file currently installed
SHOW MEMORY/POOL	Displays usage of dynamic memory (pool)
SHOW MEMORY/PHYSICAL	Displays amount of physical memory and free pages
SHOW MEMORY/SLOTS	Shows balance set slot availability
SHOW PROCESS/CONTINUOUS	Continuously updates process information
SHOW SYSTEM	Shows status of all processes

THE SYSTEM GENERATION UTILITY (SYSGEN)

- Troubleshooting functions
  - Modify system parameters
  - Create paging and swapping files
- SYSGEN parameters
  - Divided into 11 categories (see Table 6-14)
  - SYS parameters are most often used in troubleshooting
- SYSGEN references
  - *VMS System Generation Utility Manual*
  - *Guide to Setting Up a VMS System*
  - *Guide to Maintaining a VMS System*
- AUTOGEN command procedure
  - File name - SYS$UPDATE:AUTOGEN.COM
  - Analyzes modifications you make to SYSGEN parameters
  - Automatically adjusts any related parameters
  - Reference: *Guide to Setting Up a VMS System*

Table 6–14: SYSGEN Parameter Categories

Category	Parameters Associated With
ACP	File system caches and Files-11 ACPs
CLUSTER	VAXcluster operation
JOB	Job control
LGI	Login security
MULTIPROCESSING	Process creation limits and quotas
PQL	Process creation limits and quotas
RMS	VAX RMS
SCS	System communication services
SPECIAL	For use by Digital Equipment Corporation personnel only (not documented)
SYS	Overall system operation
TTY	Terminal behavior

Table 6–15: SYSGEN Commands Used in Software Troubleshooting

Command	Description
CREATE filespec/SIZE=n	Creates or extends a file that can be used as a paging, swapping, or dump file
INSTALL filespec/PAGEFILE /SWAPFILE	Activates a new paging or swapping file
SET parameter-name value	Assigns a value to a parameter in the SYSGEN work area (does not modify parameter files or the active system)
SHOW parameter-name	Displays the values of parameters in the SYSGEN work area, and the default, minimum, and maximum values of the parameters
WRITE CURRENT	Writes parameter values from the SYSGEN work areas to the current system parameter file on disk (SYS$SYSTEM:VAXVMSSYS.PAR)
WRITE ACTIVE	Writes parameter values from the SYSGEN work area to the active system in memory

Using AUTOGEN Feedback To Modify Performance

- AUTOGEN feedback is:
  - A mechanism that allows AUTOGEN to allocate the resources under its control, based on the actual workload of the system
  - A powerful tool that can help tune a VMS system to the unique requirements of a particular site's workload
- Using AUTOGEN feedback:
  - Normal workload should have been in effect for information to be useful.

    You may elect not to use feedback information if this is an initial system installation and you know that the system has not been running long enough to reflect accurately your typical workload.
    In this case, AUTOGEN bases its calculations primarily on your hardware and on typical system workloads.
  - A warning is issued if:

    Collection occurred on another system.
    Collection occurred before the system had been up for 24 hours.
    Collection occurred over 30 days ago.
  - Format:

    ```
    @SYS$UPDATE:AUTOGEN [start_phase] [end_phase] [execution_type]
    ```
  - The start phase must either precede or be identical to the end phase, according to Table 6-16.
  - SYS$SYSTEM:AGEN$FEEDBACK.REPORT is generated, showing parameters and system files affected by feedback.

Table 6–16: AUTOGEN Phases

Phase	Function
SAVPARAMS	Collect feedback information from the running system.
GETDATA	Collect all data that will be required by the GENPARAMS, GENFILES, and TESTFILES phases, including user-supplied items, and feedback information.
GENPARAMS	Generate new system parameters. Create the installed-image list.
TESTFILES	Display the page, swap, and dump file sizes calculated by AUTOGEN.
GENFILES	Generate new page, swap, and dump files, if appropriate. (Cannot be specified as the start phase.)
SETPARAMS	Run SYSGEN to set system parameters specified in SETPARAMS.DAT and generate a new AUTOGEN.PAR file. (Current parameters are retained in VAXVMSSYS.OLD.)
SHUTDOWN	Prepare system to await a manual reboot.
REBOOT	Automatically reboot the system.

Table 6–17: AUTOGEN Execution-types

Type	Meaning
<Blank>	Use feedback in calculations. Stop in TESTFILES if information is suspect.
FEEDBACK	Use feedback regardless of warnings, and continue through specified end phase.
NOFEEDBACK	Use standard calculations (synonymous with INITIAL).

Example 6–14: Sample AUTOGEN Feedback Report (Sheet 1 of 2)

```
 9-MAR-1989 10:38          AUTOGEN FEEDBACK REPORT
                               ON NODE SPLASH

Feedback information was collected on  9-MAR-1989 10:37:10.
Old values shown below are the parameter values at the time of collection.
Feedback information is based on 17 hours of up time.
   *WARNING*    The system was up for less than 24 hours
                when the feedback information was recorded.

Parameter: MAXPROCESSCNT       Old value: 130       New value: 110
        Relevant feedback information:
           Maximum observed processes: 70

Parameter: NPAGEDYN            Old value: 1391616   New value: 1440000
        Relevant feedback information:
           Maximum observed non-paged pool size: 1456128 bytes.
           Non-paged pool request rate: 26 requests per 10 sec.

Parameter: PAGEDYN             Old value: 1133056   New value: 1133056
        Relevant feedback information:
           Current paged pool usage: 761728 bytes.
           Paged pool request rate: 1 requests per 10 sec.

Parameter: SRPCOUNT            Old value: 6322      New value: 5690
        Relevant feedback information:
           Maximum observed SRP list size: 6322

Parameter: IRPCOUNT            Old value: 2300      New value: 2070
        Relevant feedback information:
           Maximum observed IRP list size: 2300

Parameter: MSCP_BUFFER         Old value: 128       New value: 128
        Relevant feedback information:
           MSCP server I/O rate: 2 I/Os per 10 sec.
           I/Os that waited for buffer space: 0
           I/Os that fragmented into multiple transfers: 0

Parameter: LOCKIDTBL           Old value: 2516      New value: 2165
        Relevant feedback information:
           Current number of locks: 2850

Parameter: RESHASHTBL          Old value: 2048      New value: 2048
        Relevant feedback information:
           Current number of resources: 1850
```

Example 6-14: Sample AUTOGEN Feedback Report (Sheet 2 of 2)

```
 9-MAR-1989 10:38          AUTOGEN FEEDBACK REPORT
                               ON NODE SPLASH

Parameter: GBLPAGES            Old value: 400000    New value: 400000
        Relevant feedback information:
           Current used GBLPAGES: 30652
           Global buffer requirements: 8800

        User or Digital-supplied parameter modifications / overrides:
         The calculation has been disabled by a hard-coded value of 400000.

Parameter: GBLSECTIONS         Old value: 750       New value: 750
        Relevant feedback information:
           Current used GBLSECTIONS: 321

        User or Digital-supplied parameter modifications / overrides:
           The calculation has been disabled by a hard-coded value of 750.

Parameter: ACP_DIRCACHE        Old value: 480       New value: 480
        Relevant feedback information:
           Hit percentage: 93%
           Attempt rate: 5 attempts per 10 sec.

Parameter: ACP_DINDXCACHE      Old value: 120       New value: 120
        Relevant feedback information:
           Hit percentage: 92%
           Attempt rate: 9 attempts per 10 sec.

Parameter: ACP_HDRCACHE        Old value: 480       New value: 540
        Relevant feedback information:
           Hit percentage: 73%
           Attempt rate: 10 attempts per 10 sec.

Parameter: ACP_MAPCACHE        Old value: 8         New value: 8
        Relevant feedback information:
           Hit percentage: 21%
           Attempt rate: 0 attempts per 10 sec.

File name: $255$DUA2:[SPLASH]PAGEFILE2.SYS
        Current size: 150000          New size: 82800
        Maximum observed usage: 7826
File name: $255$DUA2:[SPLASH]SWAPFILE2.SYS
        Current size: 40002           New size: 20000
        Maximum observed usage: 0
```

MONITOR UTILITY

- Helps troubleshoot problems caused by resource overload
- Allows you to monitor system performance data, including:
  - Process and system-wide statistics
  - I/O statistics
  - Memory management statistics
  - Time spent in each processor mode
- Produces a variety of outputs
- Monitor classes:
  - System: system-wide resource utilization
  - Component: contribution of individual components to the overall measure
- References:
  - *VMS Monitor Utility Manual*
  - *Guide to VMS Performance Management*

Table 6–18: MONITOR Procedures

Command Procedure Name	Description
SYS$EXAMPLES:SUBMON.COM	Invokes MONITOR.COM. You should invoke this procedure from your SYS$MANAGER:SYSTARTUP_V5.COM.
SYS$EXAMPLES:MONITOR.COM	Invoked by SUBMON.COM. Creates a Monitor Summary File.
SYS$EXAMPLES:MONSUM.COM	Generates Monitor Summary Reports and mails them to the system manager.

EXAMPLES OF OUTPUT FROM MONITOR SUMMARY

Example 6–15: Monitor STATES

```
                VAX/VMS Monitor Utility
                    PROCESS STATES
                    on node CLAIR      From: 21-SEP-1988 15:07:37
                       SUMMARY         To:   22-SEP-1988 16:00:54

                                CUR       AVE       MIN       MAX

Collided Page Wait             0.00      0.00      0.00      0.00
Mutex & Misc Resource Wait     0.00      0.70      0.00      2.00
Common Event Flag Wait         0.00      0.00      0.00      0.00
Page Fault Wait                0.00      0.66      0.00      5.00
Local Event Flag Wait          2.00      1.07      0.00      7.00
Local Evt Flg (Outswapped)     0.00      0.09      0.00      3.00

Hibernate                      7.00      7.12      1.00      8.00
Hibernate (Outswapped)         1.00      1.45      1.00      7.00
Suspended                      0.00      0.00      0.00      0.00
Suspended (Outswapped)         0.00      0.00      0.00      0.00
Free Page Wait                 0.00      0.00      0.00      0.00
Compute                        5.00      3.68      1.00      7.00
Compute (Outswapped)           0.00      0.14      0.00      4.00
Current Process                1.00      1.00      1.00      1.00
```

- Processes spending excessive time in COM and COMO states can indicate that the CPU is overloaded.
- Processes in Mutex & Misc Resource Wait state indicate that there may be a shortage of a system-wide resource such as page or swapfile capacity.

  Check MONITOR PROCESSES data to determine the type of resource wait.

Example 6–16: Monitor MODES

```
                VAX/VMS Monitor Utility
                TIME IN PROCESSOR MODES
                     on node CLAIR       From: 21-SEP-1988 15:07:37
                        SUMMARY          To:   22-SEP-1988 16:00:54

                                    CUR       AVE       MIN       MAX

Interrupt Stack                    1.44      2.87      0.35      5.96

MP Synchronization                 0.00      0.00      0.00      0.00

Kernel Mode                       36.26     67.12     16.20     83.27

Executive Mode                     7.18      2.94      1.13      7.36

Supervisor Mode                    1.98      0.76      0.07      9.74

User Mode                         53.12     25.98      7.79     79.63

Compatibility Mode                 0.00      0.00      0.00      0.00

Idle Time                          0.00      0.30      0.00      8.37
```

- Interrupt Stack - CPU time spent handling interrupts from peripheral devices.
- MP Synchronization - On a multiprocessor, CPU time spent coordinating access to VMS internal data structures.
- Kernel - system functions such as local lock requests, file system requests, and memory management are performed in kernel mode.
- Executive - the major consumer of executive mode time is VAX RMS.
- Supervisor - command language interpreters (DCL and MCR) execute in supervisor mode.
- User - most user-written programs execute in user mode.
- Compatibility - PDP-11 code executes in compatibility mode.
- Idle Time - time consumed by the scheduler when there is no other work to do. Check process wait states to determine reasons.

Example 6–17: Monitor PAGE

```
             VAX/VMS Monitor Utility
            PAGE MANAGEMENT STATISTICS
                  on node CLAIR        From: 21-SEP-1988 15:07:37
                     SUMMARY           To:   22-SEP-1988 16:00:54

                                CUR       AVE       MIN       MAX

Page Fault Rate              424.32   1022.97    168.55   1523.86
Page Read Rate                10.32     10.19      3.10     32.85
Page Read I/O Rate             6.58      4.37      1.63     10.29
Page Write Rate               21.23     70.86      4.95    169.50
Page Write I/O Rate            2.12     19.64      0.28     45.64

Free List Fault Rate         149.52     92.26     45.58    155.61
Modified List Fault Rate     223.19    867.44     10.83   1461.37
Demand Zero Fault Rate         9.74      4.27      1.77     11.92
Global Valid Fault Rate       25.72      3.73      0.06     29.08
Wrt In Progress Fault Rate     9.28     50.33      0.00    140.32
System Fault Rate            121.30     63.94     11.92    133.23

Free List Size             27284.00  26808.52  25455.00  27500.00
Modified List Size            50.00    348.08      0.00   1475.00
```

- Free List Fault Rate - excessive rates can occur when working set quotas are too small.
- Modified List Fault Rate - excessive rates can occur when working set quotas are too small.
- Demand Zero Fault Rate - typically seen during image activation and whenever virtual address space is expanded.
- Global Valid Fault Rate - elevated rate can be caused by swapping or image activation.
- Write In Progress Fault Rate - typically very low.
- System Fault Rate - should be kept low (< 2 per second).
- Page Read I/O Rate - rate of I/O operations necessary to satisfy page faults.
- Page Write I/O Rate - rate of disk I/O operations to write pages to page files.

Example 6–18: Monitor POOL

```
                      VAX/VMS Monitor Utility
                      NONPAGED POOL STATISTICS
                           on node CLAIR        From: 21-SEP-1988 15:07:37
                              SUMMARY           To:   22-SEP-1988 16:00:54

                                      CUR        AVE        MIN        MAX

SRPs Available                      94.00      80.46      43.00     109.00
SRPs In Use                        338.00     351.53     323.00     389.00

IRPs Available                     268.00     249.94     231.00     268.00
IRPs In Use                         82.00     100.05      82.00     119.00

LRPs Available                      11.00      10.53       0.00      11.00
LRPs In Use                         19.00      19.46      19.00      30.00

Dynamic Kbytes Available           198.00     197.97     181.00     202.00
Dynamic Kbytes In Use              290.00     290.02     286.00     307.00
Holes In Pool                       42.00      36.40      32.00      47.00
Largest Block (Kbytes)             157.00     160.26     145.00     161.00
Smallest Block (Bytes)              16.00      16.00      16.00      16.00
Blocks Less or Eq 32 Bytes           9.00       8.11       6.00      16.00
```

Example 6–19: Monitor FCP

```
                       VAX/VMS Monitor Utility
                      FILE PRIMITIVE STATISTICS
                           on node CLAIR        From: 21-SEP-1988 15:07:37
                              SUMMARY           To:   22-SEP-1988 16:00:54

                                      CUR        AVG        MIN        MAX

FCP Call Rate                        2.02       0.74       0.29       3.18
Allocation Rate                      0.08       0.02       0.00       0.09
Create Rate                          0.00       0.00       0.00       0.05

Disk Read Rate                       0.44       0.13       0.00       1.83
Disk Write Rate                      0.26       0.08       0.02       0.30
Volume Lock Wait Rate                0.00       0.00       0.00       0.00

CPU Tick Rate                        1.92       0.61       0.20       3.42
File Sys Page Fault Rate             0.39       0.13       0.05       0.39
Window Turn Rate                     0.00       0.00       0.00       0.01

File Lookup Rate                     0.54       0.09       0.00       1.44
File Open Rate                       0.92       0.32       0.13       1.03
Erase Rate                           0.00       0.00       0.00       0.00
```

Example 6–20: Monitor I/O

```
                 VAX/VMS Monitor Utility
                  I/O SYSTEM STATISTICS
                      on node CLAIR       From: 21-SEP-1988 15:07:37
                         SUMMARY          To:   22-SEP-1988 16:00:54

                                     CUR        AVE        MIN        MAX

Direct I/O Rate                     6.65       3.71       1.48       7.05
Buffered I/O Rate                   4.96       1.96       0.54      10.65
Mailbox Write Rate                  0.00       0.01       0.00       0.50
Split Transfer Rate                 0.05       0.05       0.00       0.97
Log Name Translation Rate          16.99       4.83       1.32      18.40
File Open Rate                      0.92       0.32       0.13       1.03

Page Fault Rate                   424.32    1022.97     168.54    1523.90
Page Read Rate                     10.32      10.19       3.10      32.85
Page Read I/O Rate                  6.58       4.37       1.63      10.29
Page Write Rate                    21.23      70.86       4.95     169.50
Page Write I/O Rate                 2.12      19.64       0.28      45.64
Inswap Rate                         0.00       0.04       0.00       0.94
Free List Size                  27284.00   26808.00   25455.00   27499.00
Modified List Size                 50.00     347.70       0.00    1475.00
```

- High Page Fault Rate and/or high inswap rate can indicate a memory limitation.
- High Direct I/O Rate indicates excessive disk I/O activity, and can also be expensive in CPU cycles.
- High Inswap Rate indicates high swapping activity, which could be caused by:
  - — Lack of available balance set slots
  - — Working set quotas too high

Example 6–21: Monitor DISK /ITEM=ALL

```
                      VAX/VMS Monitor Utility
                        DISK I/O STATISTICS
                           on node CLAIR       From: 21-SEP-1988 15:07:37
                              SUMMARY          To:   22-SEP-1988 16:00:54

 I/O Operation Rate                  CUR      AVE      MIN      MAX

 $1$DJA0:   (CLAIR)  VMSRL5        16.26    16.02     1.45    35.96
 $1$DUA2:   (CLAIR)  INSTRUCTOR     0.58     0.77     0.42     1.71


                      VAX/VMS Monitor Utility
                        DISK I/O STATISTICS
                           on node CLAIR       From: 21-SEP-1988 15:07:37
                              SUMMARY          To:   22-SEP-1988 16:00:54

 I/O Request Queue Length            CUR      AVE      MIN      MAX

 $1$DJA0:   (CLAIR)  VMSRL5         1.39     1.37     0.24     3.09
 $1$DUA2:   (CLAIR)  INSTRUCTOR     0.03     0.05     0.01     0.18
```

```
The Average Response Time =

Average Queue Length
-------------------   *  1000
Average I/O Rate
```

Example 6–22: Monitor FILE_SYSTEM_CACHE

```
              VAX/VMS Monitor Utility
           FILE SYSTEM CACHING STATISTICS
                   on node CLAIR        From: 21-SEP-1988 15:07:37
                      SUMMARY           To:   22-SEP-1988 16:00:54

                              CUR        AVE        MIN        MAX

Dir FCB   (Hit %)          100.00      80.89       0.00     100.00
          (Attempt Rate)     0.54       0.09       0.00       1.46
Dir Data  (Hit %)          100.00      81.62       0.00     100.00
          (Attempt Rate)     0.54       0.13       0.00       2.13
File Hdr  (Hit %)           65.00      76.36      38.00     100.00
          (Attempt Rate)     1.30       0.47       0.17       2.37
File ID   (Hit %)            0.00      89.34       0.00     100.00
          (Attempt Rate)     0.00       0.00       0.00       0.04

Extent    (Hit %)          100.00      99.95      94.00     100.00
          (Attempt Rate)     0.08       0.02       0.00       0.09
Quota     (Hit %)            0.00       0.00       0.00       0.00
          (Attempt Rate)     0.00       0.00       0.00       0.00
Bitmap    (Hit %)            0.00      25.00       0.00      25.00
          (Attempt Rate)     0.00       0.00       0.00       0.01
```

Example 6–23: Monitor DECnet

```
              VAX/VMS Monitor Utility
                 DECNET STATISTICS
                   on node CLAIR        From: 21-SEP-1988 15:07:37
                      SUMMARY           To:   22-SEP-1988 16:00:54
                                 CUR       AVE       MIN       MAX

Arriving Local Packet Rate      0.00      0.21      0.00      3.17

Departng Local Packet Rate      0.00      0.22      0.00      3.46


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                 11.00     10.54      0.00     11.00
```

Example 6–24: Monitor LOCK

```
                  VAX/VMS Monitor Utility
                 LOCK MANAGEMENT STATISTICS
                       on node CLAIR        From: 21-SEP-1988 15:07:37
                          SUMMARY           To:   22-SEP-1988 16:00:54

                                  CUR       AVE       MIN       MAX

New ENQ Rate                     4.24      1.37      0.49      5.42
Converted ENQ Rate               0.57      0.29      0.09      7.01

DEQ Rate                         4.24      1.37      0.48      5.43
Blocking AST Rate                0.17      0.01      0.00      0.17

ENQs Forced To Wait Rate         0.29      0.04      0.00      0.30
ENQs Not Queued Rate             0.00      0.00      0.00      0.00

Deadlock Search Rate             0.03      0.00      0.00      0.04
Deadlock Find Rate               0.03      0.00      0.00      0.03

Total Locks                     70.00     71.65     65.00     99.00
Total Resources                 68.00     70.61     64.00     78.00
```

Example 6–25: Monitor DLOCK

```
                  VAX/VMS Monitor Utility
           DISTRIBUTED LOCK MANAGEMENT STATISTICS
                       on node CLAIR        From: 21-SEP-1988 15:07:37
                          SUMMARY           To:   22-SEP-1988 16:00:54

                                  CUR       AVE       MIN       MAX
New ENQ Rate         (Local)     4.24      1.37      0.49      5.42
                  (Incoming)     0.00      0.00      0.00      0.00
                  (Outgoing)     0.00      0.00      0.00      0.00
Converted ENQ Rate (Local)       0.57      0.29      0.09      7.01
                  (Incoming)     0.00      0.00      0.00      0.00
                  (Outgoing)     0.00      0.00      0.00      0.00
DEQ Rate             (Local)     4.24      1.37      0.48      5.43
                  (Incoming)     0.00      0.00      0.00      0.00
                  (Outgoing)     0.00      0.00      0.00      0.00
Blocking AST Rate  (Local)       0.17      0.01      0.00      0.17
                  (Incoming)     0.00      0.00      0.00      0.00
                  (Outgoing)     0.00      0.00      0.00      0.00
Dir Functn Rate (Incoming)       0.00      0.00      0.00      0.00
                  (Outgoing)     0.00      0.00      0.00      0.00
Deadlock Message Rate            0.00      0.00      0.00      0.00
```

MODULE 7
HARDWARE TROUBLESHOOTING

INTRODUCTION

When troubleshooting a system, the system manager must deal with both hardware and software problems. Sometimes it is difficult to determine if the problem is software or hardware. In these cases, a careful study of the problem must be made. If the trouble is software, the system manager can discuss with the system programmer what should be done. If the problem is hardware, the system manager can use the methods and tools discussed in this module to learn more about the problem. Once the problem has been established as hardware (except for very minor problems), Digital Field Service should be called.

The methods and tools discussed are for system managers to use to:

- Distinguish hardware problems from software problems.
- Gather and sort error information from the system.

The hardware tools presented here are information-gathering tools, not error-analysis tools. This module does **not** present material that will result in the system manager replacing any Field Service function. The purpose of this module is to show how the system manager can preprocess some of the system information and assist Field Service in diagnosing hardware problems.

OBJECTIVES

To troubleshoot hardware problems, a system manager must be able to:

- Define the system manager's function in hardware troubleshooting.
- Identify important hardware resources and potential hardware problems.
- Obtain information about hardware resources and diagnose problems by using VMS utilities and commands, including:
  - Operator log file
  - User environment test package
  - SHOW commands
  - VAX system integrity monitor
  - Error logger
- Differentiate hardware problems from software problems using information from the error log.
- Determine what several malfunctioning devices have in common using information from the error log.
- Anticipate and prevent hardware problems using information from the operator log file.

RESOURCES

- *VMS Installation and Operations*
- *VAX System Integrity Monitor (VAXsim) User's Guide*
- *VMS DCL Dictionary*
- *Guide to Setting up a VMS System*
- *Guide to Maintaining a VMS System*
- *VMS Error Log Utility Manual*
- *VAX Hardware Handbook*

TOPICS

1. Overview of hardware troubleshooting
2. Operator log file
3. User environment test package (UETP)
4. SHOW commands
    a. SHOW MEMORY
    b. SHOW DEVICES
    c. SHOW ERROR
5. VAXsimPLUS utility
6. Analyze/Error report generator
7. Error log report

OVERVIEW OF HARDWARE TROUBLESHOOTING

- Troubleshooting at the system management level is:
  - Monitoring the system for trouble spots.
  - Keeping track of changes to the system:

    Hardware changes
    New software
    Software version changes

  - Keeping a log of past problems, what caused them, and how they were solved.
  - Coordinating with Field Service for regular preventive maintenance.
  - Coordinating with your system programmer on software problems.
  - Analyzing and diagnosing problems (if possible) based on your experience.
- Troubleshooting at the system management level is **not**:
  - Solving all software problems.

    For most VMS internals-level problems, someone knowledgeable at the system programming level is required.

  - Solving all hardware problems.

    A few problems can be solved by the system manager.
    Most hardware problems should be referred to Field Service.

- Be sure the problem is hardware.
  - If the same software causes the same error on similar hardware devices, the problem might not be hardware.
  - If different software causes the same problem on a device, this suggests a hardware problem.
  - Two indications of a hardware problem are:

    Machine check.
    Errors in error log.
- Use the process of elimination to find out which device(s) is causing problems.

  Determining what devices work can lead you to find the one(s) that do not work.
- If several devices are causing problems, what do they have in common?

  A broken UNIBUS adapter will cause your terminals, line printers, and DECnet lines to stop functioning.
- Can you swap in another device?

  If you think your terminal is broken, swap in a different terminal. Check the **broken** terminal on a different terminal line.
- If it is not broken, do not fix it.

  When trying to find a problem, do not introduce additional variables by changing something that currently works. Find and fix the problem first. After that, go back and upgrade other items.

Malfunctioning I/O Subsystems

- A malfunctioning bus will cause poor communication to mass storage devices and peripheral devices.
- A malfunctioning adapter will cause poor communication between the synchronous backplane interconnect (SBI), which is an interface to the CPU and main memory, and the mass storage and peripheral devices.
- A malfunctioning SBI will cause a break in communication between the CPU and main memory, and between main memory and the adapters in the I/O subsystem.

Table 7–1: Hardware Resources and Problems

Problem		Symptom	Suggested Action
CPU			
	Internal Error	Machine check Bugcheck	Inform Field Service. Save dump file.
Memory			
	Read/Write Errors	Memory error count	Use Analyze/Error, VAXsim.
Disk Drives			
	Read/Write Errors	Device error count	Use Analyze/Error, VAXsim.
	Head Crash	Device error count, disk or sections of disk unusable	Spin down drive. Call Field Service. Do not replace or move disk.
Tape Drives			
	Read/Write Errors	Device error count	Use Analyze/Error, VAXsim. Replace tape.
Terminals			
	Internal Error	Unresponsive, unreliable	Swap terminals. Test TT: line.
Console Subsystem			
	CSA1: Problem	Unable to boot	Replace console media.
	Microprocessor	No >>> prompt	Reboot microprocessor.

Before fixing problems, you should:

- Obtain and store user's guides for all system devices.
- Check the user's guide for additional information.
- Talk to Digital Field Service for any suggestions.

OPERATOR LOG FILE

- Used with the console printout to provide a history of what has happened on the system
- ALL OPCOM messages appear on the console (unless disabled).
- Keep the console printout for a specified period of time for reference.
- ALL OPCOM messages appear in the operator log file.
- Examine the OPCOM messages periodically (the console printout or the operator.log file).

  To examine the current operator.log file:

  — Close the current log file, open a new log file, and then examine the file just closed:

  ```
  $ REPLY/ENABLE
  $ REPLY/LOG
  $ TYPE SYS$MANAGER:OPERATOR.LOG;-1
  $ REPLY/DISABLE
  ```

  or

  — Obtain a copy without closing the file:

  ```
  $ BACKUP/IGNORE = INTERLOCK -
  SYS$MANAGER:OPERATOR.LOG CURRENT.LOG
  ```

- Security messages that also appear in the operator.log file can be identified and examined as follows:

  ```
  $TYPE CURRENT.LOG
  $SEARCH CURRENT.LOG "Warning"
  $SEARCH CURRENT.LOG "Error"
  ```

USER ENVIRONMENT TEST PACKAGE (UETP)

- User environment test package (UETP) tests:
  - All standard peripheral devices
  - Various commands and operating system functions
  - The system's multiuser capability
  - DECnet VAX
- UETP consists of an initialization phase and a set of four other phases.
  - DEVICE - tests disks, tapes, printers, and terminals.
  - LOAD - simulates a number of terminal users.
  - DECnet - tests DECnet hardware and software.
  - CLUSTER - tests interprocess cluster communication.
    and locking
- The user can choose to run all four phases, run one or more phase, or test individual devices only. To run all four tests:
  - Log in with the username SYSTEST.
  - Prepare the devices for testing.
  - Execute UETP.
  - If UETP completes successfully, run the errorlog report formatter (ERF).
  - If UETP does not complete successfully, see the chapter on UETP in the *VAX Installation and Operations Manual.*

Example 7–1: Sample UETP Run of all UETP Phases

```
 Welcome to VAX/VMS UETP Version V5.0

%UETP-I-ABORTC, UETINIT00 to abort this test, type ^C

You are running on an 11/780 CPU with 16384 pages of memory.
The system was booted from _DMA3:[SYS0.].

Run "ALL" UETP phases or a "SUBSET" [ALL]?<CR>
How many passes of UETP do you wish to run [1]?<CR>
How many simulated user loads do you want [3]?<CR>
Do you want Long or Short report format [Long]? SHORT

UETP starting at  2-SEP-1988 19:38:01.09 with parameters:
DEVICE LOAD DECNET CLUSTER phases, 1 pass, 3 loads, short report.
 .      .     .      .      .        .       .       .     .
 .      .     .      .      .        .       .       .     .
 .      .     .      .      .        .       .       .     .
```

When finished running UETP, run the Analyze/Error utility to see what errors were recorded.

Example 7–2: Sample UETP Run of DEVICE Phase

```
 Welcome to VAX/VMS UETP Version V5.0

%UETP-I-ABORTC, UETINIT00 to abort this test, type ^C

You are running on an 11/780 CPU with 16384 pages of memory.
The system was booted from _DMA3:[SYS0.].

Run "ALL" UETP phases or a "SUBSET" [ALL]? SUBSET

You can choose one or more of the following phases:

        DEVICE, LOAD, DECNET, CLUSTER

PHASE(S):  DEVICE
How many passes of UETP do you wish to run [1]?<CR>
Do you want Long or Short report format [Long]? SHORT

UETP starting at  1-SEP-1988 19:57:15.07 with parameters:
DEVICE phases, 1 pass, 3 loads, short report.
 .      .     .      .      .       .      .      .     .
 .      .     .      .      .       .      .      .     .
 .      .     .      .      .       .      .      .     .
```

When finished running UETP, run the Analyze/Error utility to see what errors were recorded.

UETP Individual Device Tests

Before running an individual device test, check the file UETINIDEV.DAT.

- It must contain information about the device to be tested.
- If the file does not exist, create it.

Example 7–3: Creating UETINIDEV.DAT for MTA0

```
$ CREATE UETINIDEV.DAT
DDB T MTA
UCB T 00000
END OF UETINIDEV.DAT
^Z
$
```

Select the correct program to use for the particular device to be tested.

Table 7–2: Test Image File Names

Test Image Name	Devices Tested
UETDISK00.EXE	Disks
UETTAPE00.EXE	Magnetic Tapes
UETTTYS00.EXE	Terminals and Printers
UETLPAK00.EXE	LPA11-K
UETCOMS00.EXE	DMC11,DMR11
UETDMPF00.EXE	DMF32, DMP11
UETDR1W00.EXE	DR11-W
UETDR7800.EXE	DR780, DR750
UETMA7800.EXE	MA780
UETUNAS00.EXE	DEUNA

Select the proper image, prepare the device, and run the program.

Example 7–4: UETP Run for Single Device

```
$ INITIALIZE  MTA0:  UETP
$
$ RUN UETTAPE00
Controller designation?: MTA
%UETP-S-BEGIN, UETTAPE00 beginning at  3-SEP-1988 19:51:29:98
%UETP-I-ABORT,  TAPE_MTA to abort  this test, type ^C
 .    .   .         .     .    .      .    .     .   .
 .    .   .         .     .    .      .    .     .   .
 .    .   .         .     .    .      .    .     .   .
```

When finished, run the Analyze/Error utility to see what errors were recorded.

SHOW COMMANDS

$ SHOW MEMORY Command

Example 7–5: SHOW MEMORY Output

```
 System Memory Resources on  3-JAN-1989 15:10:52.36

Physical Memory Usage (pages):     Total       Free      In Use  Modified
  Main Memory (32.00Mb)            65536      17151       46746      1639

Slot Usage (slots):                Total       Free    Resident   Swapped
  Process Entry Slots                240        209          31         0
  Balance Set Slots                  187        158          29         0

Fixed-Size Pool Areas (packets):   Total       Free      In Use      Size
  Small Packet (SRP) List           6400       1586        4814        96
  I/O Request Packet (IRP) List     2000       1286         714       176
  Large Packet (LRP) List             60         21          39      1648

Dynamic Memory Usage (bytes):      Total       Free      In Use   Largest
  Nonpaged Dynamic Memory       11999744   11036128      963616  10993104
  Paged Dynamic Memory           3697664    3069728      627936   3066704

Paging File Usage (pages):                      Free  Reservable     Total
  DISK$VAXVMSV50:[SYS0.SYSEXE]SWAPFILE.SYS     13032       13032     13032
  DISK$VAXVMSV50:[SYS0.SYSEXE]PAGEFILE.SYS     20325      -36500     29992

Of the physical pages in use, 35589 pages are permanently allocated
to VMS.
```

Example 7–6: SHOW MEMORY Output with Bad Pages

```
 System Memory Resources on  3-JAN-1989 15:10:52.36

Physical Memory Usage (pages):      Total       Free       In Use  Modified
  Main Memory (32.00Mb)             65536      17151        46746      1639

  Bad Pages                         Total     Dynamic  I/O Errors    Static
                                        9          0            0         9
  .    .                                .          .            .         .
  .    .                                .          .            .         .
  .    .                                .          .            .         .
```

$ SHOW DEVICES Command

Example 7–7: Partial SHOW DEVICES Output

```
 $ SHOW DEVICES

Device         Device    Error      Volume          Free    Trans     Mnt
 Name          Status    Count      Label         Blocks    Count     Cnt
NODE1$DRA0:    Mounted    12      NODE1_SYS_V4    109745     261       1
NODE1$DRA1:    Mounted     0      BIG_BIRD        251367      12       1
NODE1$DRA2:    Mounted    10      BERT              1839      84       1
NODE1$DRA3:    Mounted     0      ERNIE            65067     116       1
  .    .          .        .        .                .        .        .
  .    .          .        .        .                .        .        .
  .    .          .        .        .                .        .        .
```

$ SHOW ERROR Command

Example 7–8: SHOW ERROR Output

```
$ SHOW ERROR

Device                 Error Count
NODE1$DRA0:                 12
NODE1$DRA2:                 10
```

Only devices with a nonzero error count appear in the SHOW ERROR display.

VAXsimPLUS UTILITY - THE VAX SYSTEM INTEGRITY MONITOR

- A system management tool provided to Digital Field Service customers at no additional cost
- Allows the system manager to monitor VAX systems
  - Stand-alone systems
  - VAXclusters
  - Networks
- Provides easy on-line access to error log data

Special Features

- Hierarchical graphic displays that highlight rising error rates
- Cumulative on-line summary of error information for immediate access
- Information from VAXcluster nodes and network systems displayed simultaneously on a single screen
- Automatic notification by Mail and/or OPCOM
- User interface automatically corrects typing errors
- Histogram display shows error counts over time
- Summary display shows all devices in warning or alarm condition

Error Analysis

- Cursory analysis of VMS error log records
  - Places them into a few general categories
  - Analyzes each error record individually
- VAXsimPLUS is a trouble indicator, **not** a repair tool
- Other error log analyzers such as ANALYZE/ERROR provide detailed failure information

VAXsimPLUS Startup and Shutdown

- VAXsimPLUS_Monitor Startup

```
$ @SYS$MANAGER:VAXsim$STARTUP.COM
```

- VAXsimPLUS Shutdown - Include the following line in SYSHUTDWN.COM:

```
$ VAXSIM SHUTDOWN
```

Activating VAXsimPLUS Display

$ VAXSIM [command]

- Any interactive commands, which are discussed later in this module, can be optionally entered when activating VAXsimPLUS
- A few VAXsimPLUS commands can be entered only at the DCL level

Table 7–3: Commands Allowed Only at DCL Level

Command	Command Usage
REPORT	Displays synopsis of broadcast messages from NOTIFY Example: $ VAXSIM REPORT
SHUTDOWN	Shuts down VAXsimPLUS_Monitor Process Example: $ VAXSIM SHUTDOWN
EDITOR NOTIFY	Enables/disables operator class(es) and Mail notification Example: $ VAXSIM EDITOR NOTIFY/ENABLE=MAIL

VAXsimPLUS Tree Structure

VAXsimPLUS employs a tree structure design composed of five hierarchical levels.

1. System level

   An overview of all nodes currently in the display database

2. Subsystem level

   A block diagram of a node's subsystems

   a. Disk errors

   b. Tape errors

   c. Memory errors

   d. Events

   e. Bus errors

   f. Node_Info

3. Unit level

   Information on each device in a particular subsystem (for example, DRA0:, DRA1:)

4. Error classification level

   Lists the errors for a specific device

   a. Disk and tape errors (device errors) - hard, soft, media, information

   b. Memory errors - hard, soft

   c. Events - description, not necessarily errors

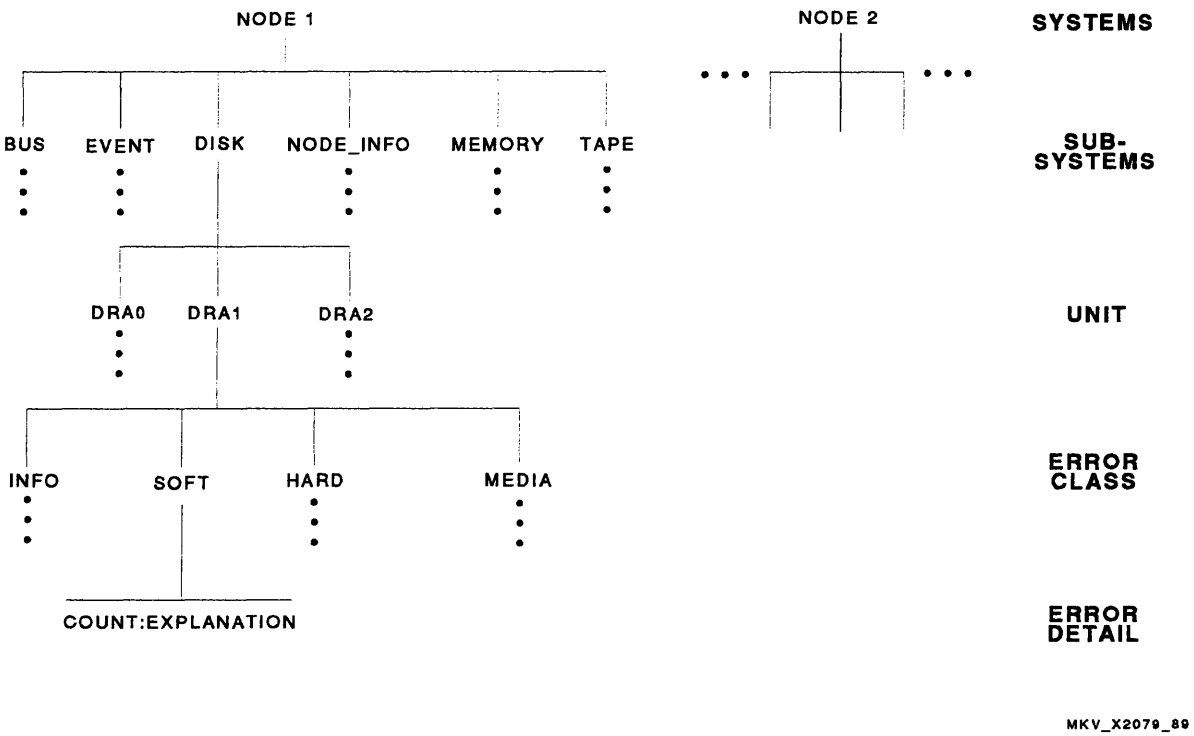
   d. Bus

   e. Info - provides node information (for example, VMS version number, CPU serial number)

5. Error detail level

   Gives the specific VMS error messages and the counts

Figure 7–1: VAXsimPLUS Tree Structure



VAXsimPLUS User Interface

- Ready to receive a command at the prompt

  VAXsimPLUS>

- Multiple commands are to be separated with a backslash (\) or a vertical bar (|)

VAXsimPLUS Interactive Command Groups

The VAXsimPLUS interactive commands are divided into the four groups described in Table 7-5.

Table 7–4: VAXsimPLUS Interactive Command Groups

Command Group	Command Usage
Display	Influences the content of your current VAXsimPLUS display
Directional	Changes your location within the VAXsimPLUS display tree
Special	Performs a variety of maintenance functions
Time	Adjusts the current evaluation period

Table 7–5: VAXsimPLUS Interactive Commands

Command	Command Action
Display Group	
ADD	Adds data from other node(s) to your display.
CYCLE	Enters dynamic display mode, 10 seconds per screen. Use <CTRL/C> to terminate.
DISPLAY	Sets the mode used to display the data (Box, Histogram, Summary, or Line).
REMOVE	Deletes node(s) data from display.
SHOW MESSAGE	Recalls last broadcast message.
SHOW MODES	Lists settings of various mode controls.
SHOW PARAMETERS	Lists current VAXsimPLUS parameters.
SHOW VERSION	Displays current VAXsimPLUS version.
WATCH	Monitors current single display. Reloads database every seven minutes.
Directional Group	
BACK	Moves in direction of descending order, same level.
DOWN	Moves down to the next (more detailed) display level.
LEFT	Scrolls display horizontally to reveal hidden items (note MORE flag).
NEXT	Moves in direction of ascending order, same level.
RIGHT	Scrolls display horizontally to reveal hidden items (note MORE flag).
TOP	Moves to highest (least detailed) display level.
UP	Moves up one (less detailed) display level.
ZOOM	Moves down along highlighted path. Highlighted paths indicate trouble areas.

Table 7–5: VAXsimPLUS Interactive Commands (Cont)

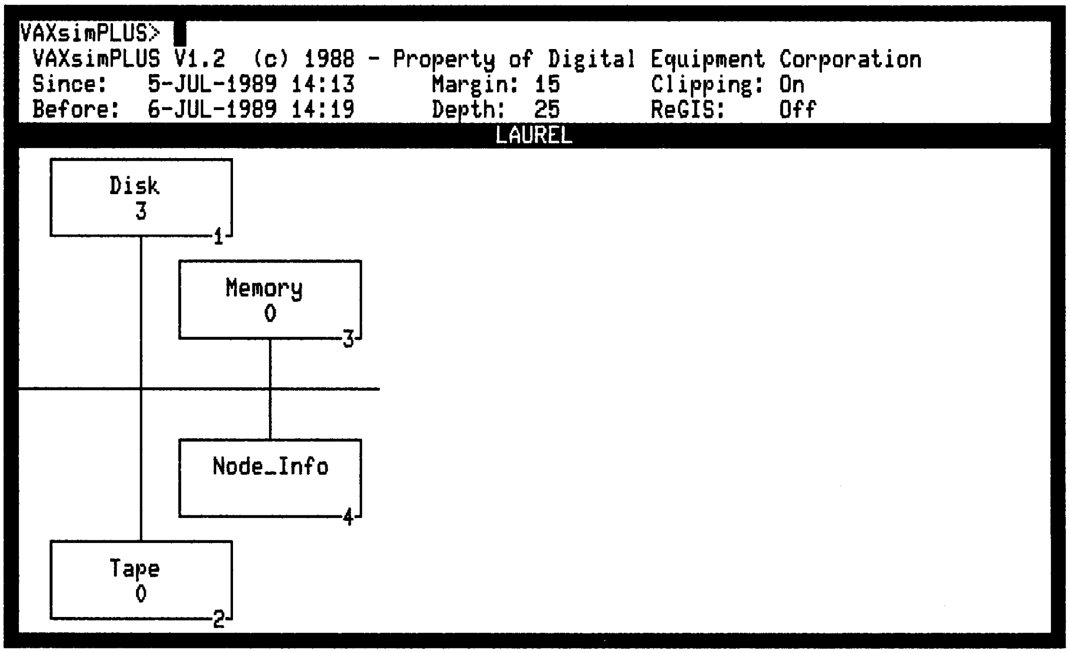
Command	Command Action
Special Group	
AUTOCORRECT	Enables/disables autocorrection facility.
CLIPPING	Enables/disables use of error rate clipping. Limits long-term effects of unusually high historical error rates.
EXIT	Leaves VAXsimPLUS. Returns control to DCL.
MARGIN	Adjusts sensitivity to error rate fluctuations.
PRINT	Produces a hard-copy dump of a ReGIS display screen onto attached graphic printer.
ReGIS	Enables/disables use of ReGIS graphic features.
UPDATE	Reloads data to bring display information up-to-date.
Time Group	
BEFORE	Sets ending time of evaluation period.
SINCE	Sets beginning time of evaluation period.
Other Commands	
DEPTH	Modifies the number of evaluation periods used in determining the error rate.
	Modifies the number of evaluation periods displayed on histograms.
HELP	Allows access to the HELP library.
SPAWN	Creates and executes a subprocess.

VAXsimPLUS Display Modes

VAXsimPLUS has four display modes:

- Box
- Line
- Histogram
- Summary

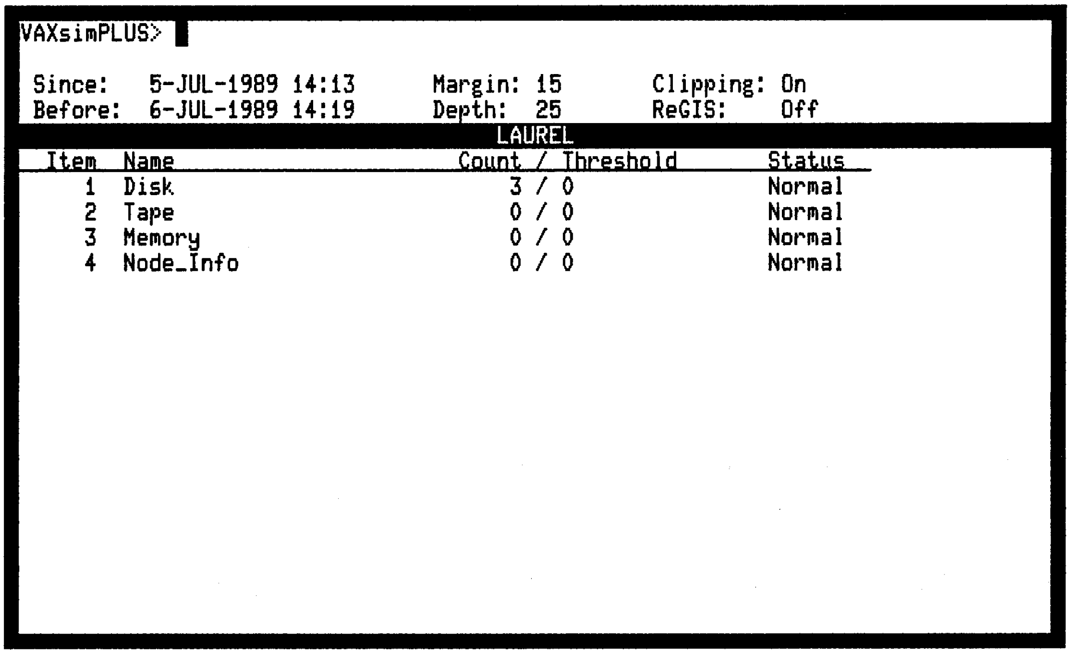
Figure 7–2: Box Display Mode - VAXsimPLUS Default



```
VAXsimPLUS>
 VAXsimPLUS V1.2  (c) 1988 - Property of Digital Equipment Corporation
 Since:   5-JUL-1989 14:13      Margin: 15       Clipping: On
 Before:  6-JUL-1989 14:19      Depth:  25       ReGIS:    Off
                                      LAUREL
```

- Since - Oldest data displayed
- Before - Newest data displayed
- Margin - Sensitivity to error rate fluctuations
- Depth - Number of evaluation periods
- Clipping - Effects calculation of error rate
- ReGIS - ReGIS graphics

Figure 7–3: Line Display Mode

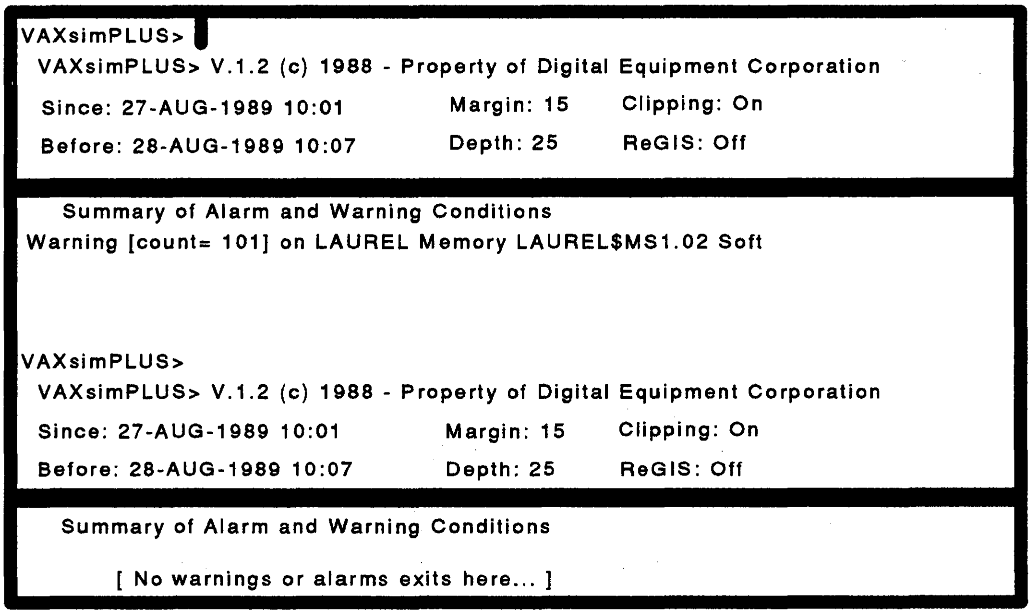


```
VAXsimPLUS> █

 Since:    5-JUL-1989 14:13      Margin: 15      Clipping: On
 Before:   6-JUL-1989 14:19      Depth:  25      ReGIS:    Off
                               LAUREL
  Item  Name                       Count / Threshold       Status
     1  Disk                           3 / 0               Normal
     2  Tape                           0 / 0               Normal
     3  Memory                         0 / 0               Normal
     4  Node_Info                      0 / 0               Normal
```

- Threshold - Lower limit of allowable errors
- Status - Current level of errors versus the threshold
  - — Normal - Error rates fall below thresholds
  - — Warning -

    Soft error rate exceeds threshold
    or
    Hard error rate is approaching threshold
  - — Alarm - Hard error rate exceeds threshold

Figure 7–4: Summary Display Mode



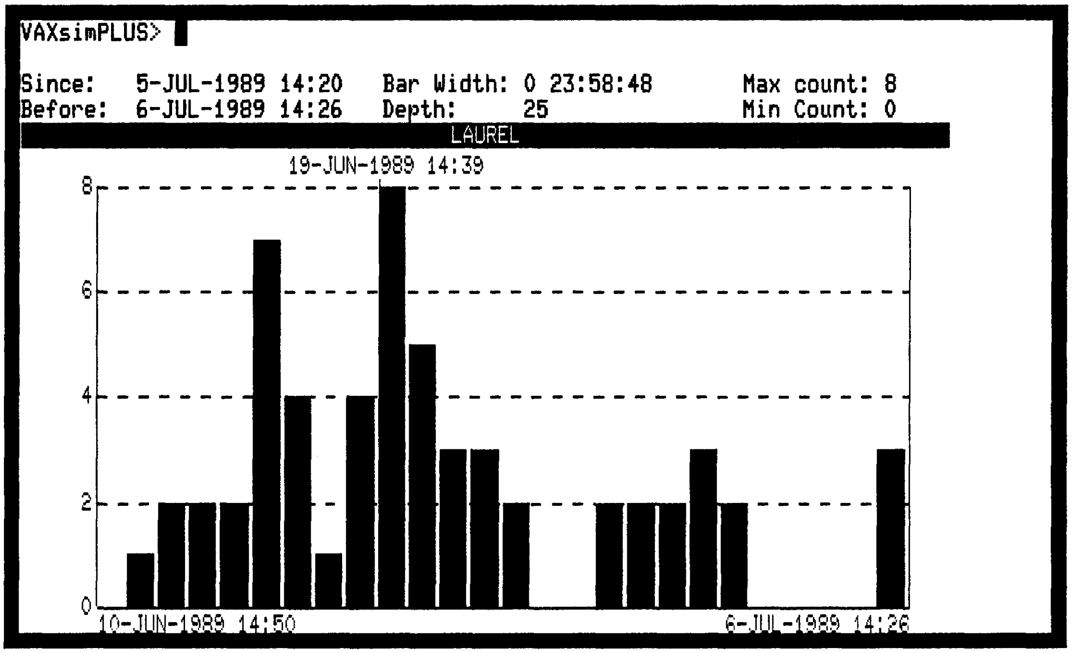
```
VAXsimPLUS>
  VAXsimPLUS> V.1.2 (c) 1988 - Property of Digital Equipment Corporation
   Since: 27-AUG-1989 10:01          Margin: 15     Clipping: On
   Before: 28-AUG-1989 10:07         Depth: 25      ReGIS: Off

     Summary of Alarm and Warning Conditions
 Warning [count= 101] on LAUREL Memory LAUREL$MS1.02 Soft



VAXsimPLUS>
  VAXsimPLUS> V.1.2 (c) 1988 - Property of Digital Equipment Corporation
   Since: 27-AUG-1989 10:01          Margin: 15     Clipping: On
   Before: 28-AUG-1989 10:07         Depth: 25      ReGIS: Off

     Summary of Alarm and Warning Conditions

          [ No warnings or alarms exits here... ]
```

MKV_X2080_89

For a summary of all errors for the selected time period, see BEFORE and SINCE dates. If there are no errors, the following line will be displayed:

[No warnings or alarms exits here...]

Figure 7–5: Histogram Display Mode



```
VAXsimPLUS>

Since:    5-JUL-1989 14:20   Bar Width: 0 23:58:48       Max count: 8
Before:   6-JUL-1989 14:26   Depth:      25              Min Count: 0
```

- Bar Width - Amount of time each histogram bar represents (period of time between SINCE and BEFORE)
- Depth - Span of time being plotted on graph (25 days)
- Max Count - Maximum error count plotted on graph (121)
- Min Count - Minimum error count plotted on graph (0)

ANALYZE/ERROR REPORT GENERATOR

- Used to generate summary reports of hardware errors logged.
- Used to generate detail reports of hardware errors logged.

Example 7–9: Analyze/Error Report

```
$ ANALYZE /ERROR /SUMMARY /NOFULL /INCLUDE=DUA2:

Error Log Report Generator    Version V05-000
 DEVICE SUMMARY LOGGED BY SID 068007FB
              ERRORS         TIMEOUTS    UCB ERROR  UCB OPERATION
         [HARD]  [SOFT]  [HARD]  [SOFT]   COUNT        COUNT
_SWISH$DUA2:
  "INSTRUCTOR"
                0.      6.      0.      0.          0.            0.
           -------- -------- -------- -------- ---------- ------------
   TOTALS       0.      6.      0.      0.          0.            0.
VOLUME LABEL(S) LOGGED BY SID 068007FB
                                QIO(S)    ERROR(S)  MOUNT(S)
      LABEL -- INSTRUCTOR
      _SWISH$DUA2:               55337.          0.        9.
SUMMARY OF ALL ENTRIES LOGGED BY SID 068007FB
      VOLUME MOUNT                          27.
      VOLUME DISMOUNT                        9.
      ERL$LOGMESSAGE                         6.
      DATE OF EARLIEST ENTRY           29-JUL-1988 09:39:25.24
      DATE OF LATEST ENTRY             29-DEC-1988 07:40:00.38
PROCESSED ENTRIES HOUR-OF-DAY HISTOGRAM LOGGED BY SID 068007FB
      00:00      0.
        .        .
        .        .
        .        .
      15:00      0.
      16:00      5. *****
      17:00      9. *********
      18:00      3. ***
      19:00      0.
      20:00      1. *
      21:00      0.
      22:00      0.
      23:00      0.
```

ERROR LOG REPORT

Identification Section and Device-Dependent Data

- Identification section consists of four lines:
  - First line is the error entry number.
  - Second line contains error sequence number and system identification value.
  - Third line specifies type of error.
  - Fourth line also specifies type of error.
- Device-dependent data section contains information on the selected error log entries:
  - First line can identify the device or subsystem where the error occurred.
  - Remaining space can contain the contents of the device, registers, and software information.

Example 7–10: Device Errors (Sheet 1 of 2)

```
V A X / V M S        SYSTEM ERROR REPORT      COMPILED 17-APR-1989 13:17
                                                               PAGE  1.
   ************************* ENTRY     274. *************************
1  ERROR SEQUENCE 376.                      LOGGED ON:      SID 02005F7C
   DATE/TIME 12-APR-1989 15:15:04.54                   SYS_TYPE 00000000

2  DEVICE ERROR  KA750    REV# 124.   UCODE REV# 95.
                SCS NODE: TEACH

3  MASSBUS SUB-SYSTEM, UNIT _TEACH$MTA0:

      "RH" CR         00000004
                                      INTERRUPT ENABLE
      "RH" SR         00003080
                                      "MASSBUS" EXCEPTION
                                      DATA TRANSFER ABORTED
                                      DATA TRANSFER COMPLETED
      "RH" VAR        0000024C
                                      76. BYTE, PAGE OFFSET
                                      MAPPING REGISTER #1. SELECTED
      "RH" BCR        00000000
      "RH" MPR #1.    8000444F
                                      VALID
                                      TRANSFER PAGE, 8743.5. K
      "RH" MPR #0.    80004828
                                      VALID
                                      TRANSFER PAGE, 9236.0. K

      TMCS1           00000838
                                      READ FORWARD
                                      DRIVE AVAILABLE
      TMDS            00005190
                                      SETTLE DOWN
                                      DRIVE READY
                                      DRIVE PRESENT
                                      MEDIUM ON-LINE
                                      COMPOSITE ERROR
4     TMER            00008080
                                      FORMAT ERROR
                                      CORRECTABLE DATA ERROR
.
.
.
```

Device Errors (Sheet 2 of 2)

```
V A X / V M S       SYSTEM ERROR REPORT      COMPILED 17-APR-1989 13:17
                                                             PAGE  2.

      UCB$B_ERTCNT            00
                                       0. RETRIES REMAINING
      UCB$B_ERTMAX            10
                                       16. RETRIES ALLOWABLE
      ORB$L_OWNER       00000000
                                       OWNER UIC [000,000]
      UCB$L_CHAR        0CC44021
                                       RECORD ORIENTED
                                       "SEQUENTIAL BLOCK" ORIENTED
                                       FILE ORIENTED
                                       AVAILABLE
                                       ERROR LOGGING
                                       ALLOCATED
                                       CAPABLE OF INPUT
                                       CAPABLE OF OUTPUT
      UCB$W_STS             0910
                                       ONLINE
                                       BUSY
                                       SOFTWARE VALID
      UCB$L_OPCNT       00000744
5                                      1860. QIO'S THIS UNIT
      UCB$W_ERRCNT          0001
                                       1. ERRORS THIS UNIT

      IRP$W_FUNC            000C
                                       READ PHYSICAL BLOCK
      IRP$W_BCNT            0050
                                       TRANSFER SIZE 80. BYTE(S)
      IRP$W_BOFF            01FC
                                       508. BYTE PAGE OFFSET
      IRP$L_PID         0003003F
                                       REQUESTOR "PID"
      IRP$Q_IOSB        046C01F4
                        000003C0       IOSB, 1132. BYTE(S) TRANSFERRED
```

Comments on Example 7-10

1. The time at which the error occurred.
2. The general type of error, the CPU type and revision-level.
3. The device having the error.
4. The specific nature of the error that occurred.
5. Total operations (QIOs) versus number of errors can indicate something about the reliability of the device.

Example 7–11: Control Entries

```
************************ ENTRY       1. ************************
ERROR SEQUENCE 0.                     LOGGED ON:     SID 068007FB
DATE/TIME 21-JUL-1988 12:37:33.80     SYS_TYPE 06000979

ERRLOG.SYS CREATED  KA855   REV# 0.
                    CPU # 1.
************************ ENTRY       2. ************************
ERROR SEQUENCE 0.                     LOGGED ON:     SID 068007FB
DATE/TIME 21-JUL-1988 12:37:33.80     SYS_TYPE 06000979

SYSTEM START-UP  KA855   REV# 0.
                 CPU # 1.

TIME OF DAY CLOCK     784C3764
************************ ENTRY       8. ************************
ERROR SEQUENCE 5.                     LOGGED ON:     SID 068007FB
DATE/TIME 21-JUL-1988 21:51:02.00     SYS_TYPE 06000979

TIME STAMP  KA855   REV# 0.
            SCS NODE: BAROOM, CPU # 1.
```

Example 7–12: Volume Changes

```
************************ ENTRY      63. ************************
ERROR SEQUENCE 79.                    LOGGED ON:     SID 068007FB
DATE/TIME 29-JUL-1988 09:39:25.24     SYS_TYPE 06000979

MOUNT VOLUME  KA855   REV# 0.
              SCS NODE: BAROOM, CPU # 1.

      UNIT _SWISH$DUA2:, VOLUME LABEL "INSTRUCTOR"

      9. QIO OPERATIONS THIS UNIT, 0. ERRORS THIS UNIT
************************ ENTRY      69. ************************
ERROR SEQUENCE 123.                   LOGGED ON:     SID 068007FB
DATE/TIME 29-JUL-1988 16:35:41.84     SYS_TYPE 06000979

DISMOUNT VOLUME  KA855   REV# 0.
                 SCS NODE: BAROOM, CPU # 1.

      UNIT _SWISH$DUA2:, VOLUME LABEL "INSTRUCTOR"

      29224. QIO OPERATIONS THIS UNIT, 0. ERRORS THIS UNIT
      29215. QIO OPERATIONS THIS VOLUME, 0. ERRORS THIS VOLUME
```

Example 7–13: Machine Check Error

```
V A X / V M S       SYSTEM ERROR REPORT      COMPILED 17-APR-1989 11:52
                                                             PAGE   1.

**************************** ENTRY       7. ****************************
ERROR SEQUENCE 5.                             LOGGED ON:      SID 02005F7C
DATE/TIME 10-APR-1989 15:00:11.74                        SYS_TYPE 00000000

MACHINE CHECK  KA750    REV# 124.   UCODE REV# 95.
               SCS NODE: TEACH

      EXCEPTION PC     000107CB

      ERROR PSL        03C00000
                                       INTERRUPT PRIORITY LEVEL = 00.
                                       PREVIOUS MODE = USER
                                       CURRENT MODE  = USER
      SUMMARY CODE     00000002
                                       TRANSLATION BUFFER OR BUS ERROR
      VA LAST REF      0002AC7C
      PC AT ERROR      000107CB
      MDR              00000038
      SMR              0000000B
                                       CPU MODE = USER
                                       VIRTUAL
                                       READ
      RLTO             00000000
      TBER             00000002
                                       TB G1 DATA ERROR
      CAER             00000000
                                       CACHE MISS
      BER              00000000
      MCESR            00000005
                                       XB REFERENCE
                                       TB PARITY ERROR
```

Example 7–14: Fatal Bugcheck

```
************************ ENTRY     423. ************************
ERROR SEQUENCE 292.                   LOGGED ON:    SID 068007FB
DATE/TIME 29-SEP-1988 14:44:00.       SYS_TYPE 06000979

FATAL BUGCHECK  KA855   REV# 0.
                SCS NODE: BAROOM, CPU # 1.
INVEXCEPTN, Exception while above ASTDEL or on interrupt stack

      PROCESS NAME    DTR_8385
      PROCESS ID      00020061
      ERROR PC        8017C1FE
      ERROR PSL       00080009
                                C-BIT
                                N-BIT
                                INTERRUPT PRIORITY LEVEL = 08.
                                PREVIOUS MODE = KERNEL
                                CURRENT MODE  = KERNEL
STACK POINTERS
KSP 7FFE76C0 ESP 7FFE9800 SSP 7FFECA4E USP 7FF0DD74 ISP 8074F200

GENERAL REGISTERS
R0  00000008  R1  00080000  R2 00001000  R3 00001408  R4 00000003
R5  804BD150  R6  00000000  R7 80610D80  R8 804BD1FC  R9 00000030
R10 00000031  R11 00000001  AP 00000388  FP 7FFE77E4  SP 7FFE7784

SYSTEM REGISTERS
      P0BR            8184CA00
                                  P0 PTE BASE (VIRT ADDRS)
      P0LR            000000CF
                                  TOTAL P0 PAGES
      P1BR            81094800
                                  P1 PTE BASE (VIRT ADDRS)
      P1LR            001FF85B
                                  TOTAL NON-EXISTENT P1 PAGES
      SBR             01F7EA00
                                  SYSTEM PTE BASE (PHYS ADDRS)
      SLR             00020580
                                  TOTAL PAGES "SYSTEM" VIRT MEM
      PCBB            004F2A20
                                  PCB BASE (PHYS ADDRS)
      SCBB            01F70A00
                                  SCB BASE (PHYS ADDRS)
      ASTLVL          00000000
                                  KERNAL MODE AST PENDING
      SISR            00000004
                                  PENDING INTERRUPT AT LEVEL 2.
                                  INTERRUPT REQUEST ACTIVE = 0.
```

Analyzing the Error Log Report

- Hardware problems can be differentiated from software problems on an Error Log report.
  - Hardware problems show the contents of the device registers.
  - Software problems show the contents of the I/O database at the time of the error.
- Common errors for most malfunctioning devices are:
  - Data transfer operations; specifically, translation of virtual addresses into physical
  - Read and write memory sequences

SUMMARY

This module has provided an overview of hardware troubleshooting from the system manager's perspective. Information describing the steps necessary to troubleshoot hardware problems and the utilities to be used were presented in detail. Examples were also included showing how to use the tools.

MODULE 8
CONFIGURATION PLANNING

INTRODUCTION

Computer systems often undergo changes that cause their workloads to gradually increase or become more diverse. Over time, new software products, application software, data files, and users may be added to a computer system. The increased workload creates the need for additional hardware.

During the life of your VAX system, you may want to buy additional hardware to accommodate new applications or other system growth. The process of adding new hardware to your VAX system is called **configuration planning.**

Carefully analyze your application requirements so that you can plan your configuration correctly. A detailed discussion of system analysis and design is beyond the scope of this module; entire seminars are devoted to the subject of system design. This module provides a brief overview of the system analysis process but concentrates on configuration planning.

The configuration planning described in this module is to be performed only after you have completed a careful analysis of your needs. After determining your needs, use the information in this module to select your new hardware.

OBJECTIVES

To plan for system expansion, a system manager must be able to:

- Identify the configuration planning parameters that need to be considered.
- Identify the criteria used in choosing between a large VAX system and many MicroVAX systems.
- Describe the differences between a VAXcluster and a network, and determine which type of configuration is the best solution for a given application.
- Given the resource requirements for an existing system, identify all hardware that must be added to meet the requirements.
- List the system management tasks that must be performed after new hardware is added to an existing system.

RESOURCE

VAX Systems and Options Catalog

TOPICS

1. General criteria for selecting additional hardware

2. Comparing hardware alternatives

    a. Current products compared to traditional products

    b. MicroVAX systems compared to large VAX systems

    c. Networks compared to VAXclusters

3. System expansion hardware

    a. Modules

        1) Bus loads
        2) BI nodes

    b. Backplanes

    c. Expansion boxes

    d. Expansion cabinets

4. Criteria for selecting additional mass storage devices

5. Expanding memory

6. The *VAX Systems and Options Catalog* (SOC)

    Using the SOC to select additional hardware

7. Example: Using the SOC to expand a VAX 6310 system

    a. Existing configuration

    b. Expansion requirements

    c. New hardware selection

    d. Summary of new hardware

8. Expanding your configuration

    a. Software requirements

    b. Integrating new equipment into the system

GENERAL CRITERIA FOR SELECTING ADDITIONAL HARDWARE

Defining Application Requirements

- Application software and data files
- Any new VAX programming languages or layered products
- Any increased user or application load
  - Growing business or department
  - More automation
- Total number of users
- Number of concurrent users
- Response time needed
  - Immediate
  - A few seconds
  - Many seconds
- Mass storage needed
  - Capacity
  - Speed
  - Removable or fixed media (disks)
- Computing and I/O capabilities
  - Compute-intensive application
  - I/O-intensive application

- Sharing data between systems
    - Quantity
    - Frequency
    - Speed required
- System availability
    - Acceptable amount of system downtime
    - Data availability
- Real-time capability

COMPARING HARDWARE ALTERNATIVES

Table 8–1: Current Products Compared to Traditional Products

Consideration	Current Product	Traditional Product
Purchase price	Higher	Lower
Service contract price	Lower	Higher
Power and floor space requirements	Less	More
Compatible with your older equipment	Maybe not	Probably
Requires new controller or other additional device	Maybe	Probably not
Can become obsolete	Later	Sooner

Table 8–2: Single-User MicroVAX Systems Compared to Large VAX Systems

Consideration	Single-User MicroVAX	Large VAX (Multiple Users)
Compute power per user	More	Less
I/O throughput	Slower	Faster
Amount of mass storage available	Less	More
Cost per user	Less	More
Who manages the system	User	System manager[1]
Technical expertise required of user	More	Less

[1] Except in the case of a local area VAXcluster (LAVc) running the local area VAXcluster optional software.

Table 8–3: Characteristics: Networks Compared to VAXclusters

Characteristic	Network	VAXcluster
CPU cabinet locations	Can be far apart	CI: Same room NI: About 2 miles
Number of systems	> 64,000	CI: 16 NI or MIVc: 42
System management	Separate	Integrated
File sharing between systems	Easy	Very easy
Hardware cost	Lower than CI	CI: Higher NI: Same
Software cost	Higher	Lower

Table 8–4: Functions: Networks Compared to VAXclusters

Function Required	Best Choice
Communication between systems in different departments	Network
Communication with remote systems	Network
Many file transfers and high-speed data sharing between systems	VAXcluster
High availability of system resources to all users	VAXcluster

SYSTEM EXPANSION HARDWARE

Four types of devices are used for expanding systems:

- Modules
- Backplanes
- Expansion boxes
- Expansion cabinets

Module

- A module is a printed circuit board.
- Provides electronics for a hardware component (for example, a disk or tape drive).
- Most hardware options consist of one or more modules.
- Module type is determined by the number of **connectors**.
  - — Dual-height module has two connectors.
  - — Quad-height module has four connectors.
  - — Hex-height module has six connectors.
- Connectors are inserted into a **backplane**.
- Each module requires **bus drive signals** from an interconnect.
  - — UNIBUS
  - — Q-Bus
  - — VAXBI

Bus Drive Signal

- Depending on the type of interconnect in use, bus drive signal level is measured in either:
  - **Bus loads**
  - **BI nodes**
- Most options consume one bus load or BI node.
- If you add enough options:
  - You will run out of bus loads or BI nodes.
  - Order the appropriate hardware for further expansion:

    UNIBUS repeater
    Another VAXBI channel

Backplane

- Provides space to mount modules.
- Space is divided into slots:
  - Module connectors are inserted into **slots**.
  - Each slot contains up to six rows.
  - Each row can accept one module connector tab.
  - Total number of slots depends on backplane type.
- Routes power and UNIBUS control signals to modules:
  - Certain rows of slots are dedicated to UNIBUS functions.
  - You cannot insert expansion modules in reserved rows.
- If you add enough options:
  - You will run out of slots.
  - Order another backplane for further expansion.

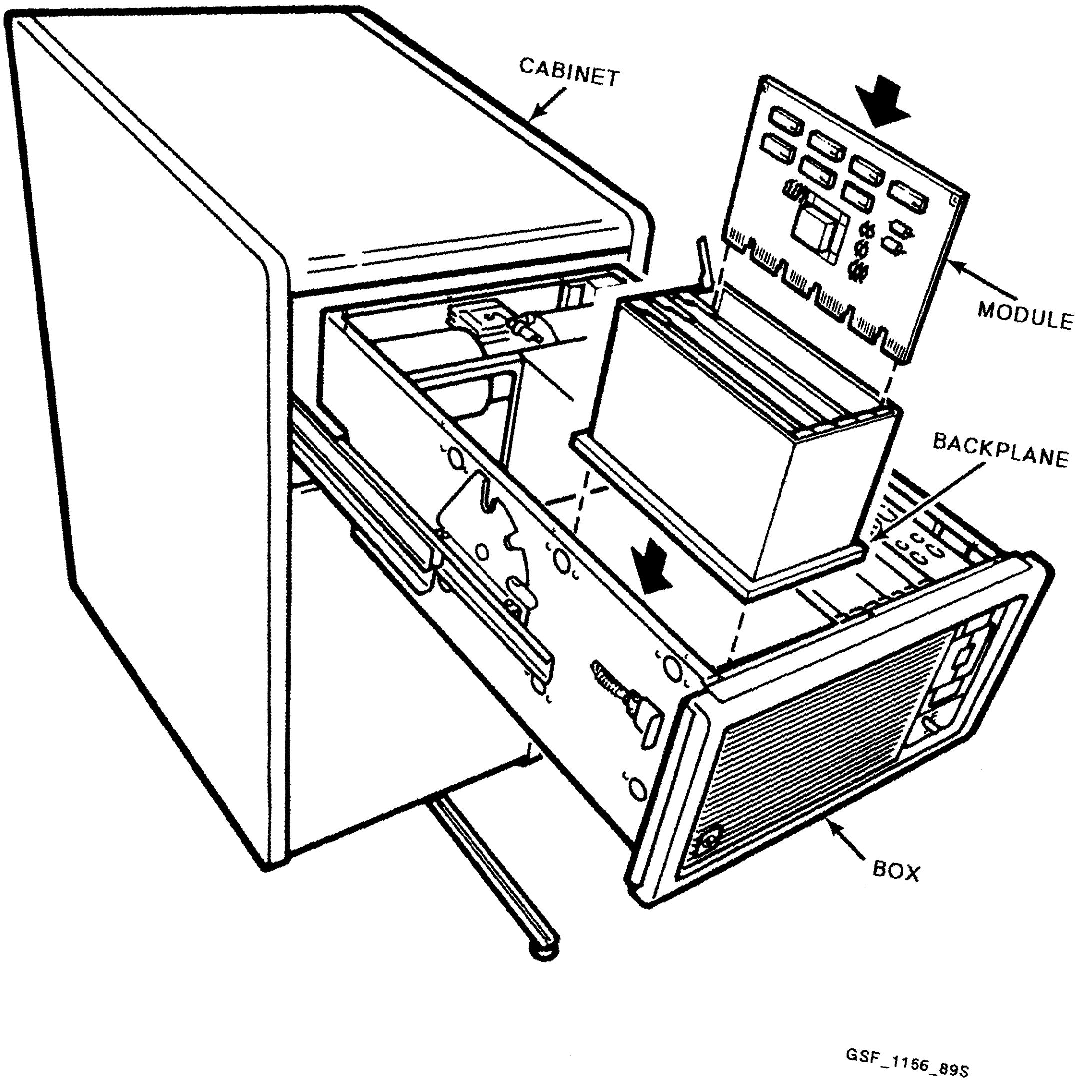
Expansion Box

- Provides mounting space for modules and backplanes.
  - Mounting space is measured in **system units (SU)**.
  - Different boxes have various SU of space.
  - Different backplanes require various SU of space.
- Provides dc power supply for modules - power is measured in amperes.
- Can also contain a UNIBUS repeater.

Expansion Cabinet

- Provides space for boxes - space is divided into **panel units (PU)** or **option panel spaces (OPS)**
- Protects the electronics contained within the box
- Shields against conducted and radiated emissions
- I/O connection panel
  - Section of the cabinet dedicated to I/O routing
  - Provides space to route I/O cables between peripherals and UNIBUS expansion cabinet
  - PU are used to connect peripherals to the cabinet
  - An optional hardware device may require a certain number of PU
- Different expansion cabinets support various numbers of:
  - Boxes
  - Panel units

Figure 8–1: System Expansion Hardware



CRITERIA FOR SELECTING ADDITIONAL MASS STORAGE DEVICES

- Type of controller required
  - Does your configuration already include one?
  - How many controllers does your system support?
- Number of devices connected to each controller
  - A controller supports a certain number of devices.
  - Connect fewer devices to the controller for better performance.
  - Buy more controllers for additional disk drives.
- Configuring requirements
  - Mounting - number of slots
  - Amperes drawn
  - I/O panel units
  - Bus load or BI nodes
- Footprint per Mbyte - amount of floor space required can be critical in large systems with many drives
- Expansion cabinet needed

EXPANDING MEMORY

- You must use the same generation of chips. For example, if your system has 256K chip memory boards:
  - You can add more 256K chip boards.
  - You cannot add 1M chip boards.
- Adding memory does not affect the amount of available:
  - Mounting space
  - Power
  - Panel units
- Additional memory can require:
  - Slots
  - Option panel spaces
- Different processors support various amounts of memory.

THE VAX SYSTEMS AND OPTIONS CATALOG (SOC)

- Lists current, actively marketed products (updated quarterly).
- Not all upgrade options are listed.
  - Mature products are not actively marketed.
  - DECdirect catalog lists additional upgrade options.
- Products not mentioned in the SOC:
  - Can still be supported
  - Are not preferred for new installations

    Check SPDs or Digital Field Service when in doubt.
- The SOC includes information about:
  - Preconfigured systems
  - System building blocks
  - System expansion
  - VAX options
  - Cables
  - Disks and tapes
  - Terminals and printers
  - Personal computers
  - Industrial systems
  - VAX software
  - VAX services

EXAMPLE: USING THE SOC TO EXPAND A VAX 6310 SYSTEM

Existing Configuration

- VAX 6310 CPU
- 32 Mbytes of 1M chip ECC MOS memory
- DEBNA 802.3 Ethernet communications interface
- LA100 console terminal and stand
- 1 TU81-Plus tape drive
- 1 TK70 tape drive and interface
- 1 SA600 storage array (2 RA90 drives)
- 1 KDB50 disk controller
- 2 VAXBI channels, which provide the following resources:
  - — 10 VAXBI slots
  - — 29 I/O panel units
- Table 8-5 lists components that consume expandable resources

Table 8–5: Expandable Resources Currently In Use

Component	VAXBI Slots	BI Nodes	I/O Panel Units
DEBNA/Ethernet port	1	1	1
1 KDB50 controller	2	1	2
1 TU81-Plus tape drive	1	1	1
1 TK70 tape drive	1	1	0
Total in use:	5	4	4
Remaining:	5	-	25

Expansion Requirements

- Approximately 4.0 GBytes of additional disk space - disks must be compatible with the existing SA600 configuration
- Another tape drive

New Hardware Selection

- Selecting a disk drive
  - The SA600 storage array building block is an array of four (4) RA90 drives supported by the VAX 6310.
  - SA600 prerequisites:

    A KDB50 disk controller
  - Select the appropriate combination of drives and cabinets.

    SA600 HA/HD containing four RA90 disk drives.
  - Select cables of the appropriate length.
  - Determine whether an additional controller is required:

    The KDB50 supports up to four disk drives.
    We already have two RA90 drives.
    To add four RA90 drives, we need another KDB50.
  - KDB50 configuring requirements:

    2 VAXBI slots
    1 BI node
    2 I/O panel units
    Cables (available in 12-foot, 25-foot, 50-foot, and 80-foot lengths)

- Select a tape drive.

  - The TU81-Plus tape drive is supported by the VAX 6310.

    TU81-Plus includes a controller.
    Does not require an external controller.

  - Configuring requirements for a TU81-Plus:

    1 VAXBI slot
    1 BI node
    1 I/O panel unit

  - TU81-Plus has cabinet space for one RA82 disk drive.

- Determine total configuring requirements for the additional TU81-Plus tape drive and KDB50 controller:

  - 3 VAXBI slots

  - 2 BI nodes

  - 3 I/O panel units

- Current configuration meets these requirements.

Summary of New Hardware

- 1 TU81-Plus tape drive
- 4 RA90 drives
  - 4 - RA90 - NA
  - Appropriate cables
- 1 KDB50 disk controller with appropriate cable

EXPANDING YOUR CONFIGURATION

Software Requirements

- STARTUP.COM automatically runs SYSGEN AUTOCONFIGURE ALL.
  - Connects all devices physically attached to the system
  - Loads the necessary I/O drivers
- You should run AUTOGEN after adding new devices, which modifies the appropriate SYSGEN parameters to reflect new hardware.
- Use SYS$MANAGER:SYCONFIG.COM for devices that are not automatically configured (for example, the virtual terminal driver).

Example 8–1: SYCONFIG.COM Procedure

```
$ RUN SYS$SYSTEM:SYSGEN
CONNECT VTA0 /NOADAPTER /DRIVER=TTDRIVER ! Enable virtual terminals
CONNECT LTA0 /NOADAPTER         ! Enable the LAT, loads LTDRIVER
CONNECT CONSOLE                 ! Load system console
EXIT
```

Integrating New Equipment into the System

- Before you move directories and files to a new disk:
  - Always use logical names, **not** physical device names, when referring to disks.
  - Instruct users to use appropriate logical names, **not** physical device names.
  - Update startup procedures to mount the new disk.
  - Ensure that the appropriate logical names are used in all:

    UAF records
    Login command procedures
    Startup and shutdown procedures
    Application programs and command procedures
    Site-specific messages or help text

- To move directories and files to a new disk:
  - Perform a complete backup of the old disk.
  - Initialize the new disk.
  - Use Backup to transfer directories and files to the new disk.
  - Assign the existing logical name to the new disk.
- When adding large amounts of memory, increase UAF working set sizes if you are not adding new users.
- When adding printers, use generic queues so users need not learn new print commands.

SUMMARY

- Before selecting additional hardware, you must:
  - Define application requirements.
  - Determine hardware alternatives.
  - Compare the alternatives.
- In comparing hardware alternatives, you should consider:
  - Current products versus traditional products
  - Single-user MicroVAX systems versus large VAX systems
  - Networks versus VAXclusters
- System expansion hardware includes:
  - Modules
  - Backplanes
  - Expansion boxes
  - Expansion cabinets
- In selecting additional mass storage devices, you must consider:
  - Type of controller required
  - How many devices are connected to each controller
  - Configuring requirements for the new device
  - Amount of floor space required
  - Whether an expansion cabinet is needed

- Expanding memory:
  - Requires the same generation of chips
  - Does not affect the amount of available mounting space, power, or panel units
  - May require slots or option panel spaces
  - Is based on the knowledge that different processors support various amounts of memory
- The *VAX Systems and Options Catalog:*
  - Describes current, actively marketed products
  - Does not list all possible upgrade options
  - Can be used to select additional hardware for your system
- After adding new hardware to your system:
  - STARTUP.COM will automatically invoke SYSGEN to connect the device and load the driver.
  - Use SYCONFIG.COM for devices that are not automatically configured.
  - Run AUTOGEN to adjust appropriate SYSGEN parameters.
  - Integrate the device into your system by modifying programs, command procedures, and logical names.

MODULE 9
LABORATORY EXERCISES

INTRODUCTION

This is the laboratory module for the *VMS System Management II* course. This module consists of topics corresponding to the course modules:

- Advanced Command Procedures
- Customizing the DCL Environment
- Advanced VMS Concepts
- VMS Disk File Structure
- Introduction to VAX RMS
- Software Troubleshooting
- Hardware Troubleshooting
- Configuration Planning

You will use this module during laboratory time. Each topics contains either practical or written exercises that allow you to practice the techniques discussed in the corresponding lecture.

This module contains many different lab exercises, which allow you to practice a wide variety of tasks related to system management. Because of the quantity of exercises provided, you might not have enough time to complete all of them. If this is the case, select only the exercises that interest to you.

COURSE FILES

Laboratory exercises for some modules instruct you to access on-line course files. These files reside in directories on the course disk, which should have the logical name COURSE$DISK.

Table 9-1 lists the names of the directories that contain the course files. For your convenience, these directories should have logical names assigned to them.

Table 9–1: Course Directories and Logical Names

Module Name	Directory (on COURSE$DISK)	Logical
Advanced Command Procedure	[COURSE.SYSMGT_II.COM]	ADV_COM
Customizing the DCL Environment	[COURSE.SYSMGT_II.DCL]	CUST_DCL
VMS Disk File Structure	[COURSE.SYSMGT_II.FILES_11]	FILES_11
Introduction to VAX RMS	[COURSE.SYSMGT_II.RMS]	INTRO_RMS
Software Troubleshooting	[COURSE.SYSMGT_II.SW_TROUB]	SW_TROUB
Hardware Troubleshooting	[COURSE.SYSMGT_II.HW_TROUB]	HW_TROUB

ADVANCED COMMAND PROCEDURES

Laboratory Exercise 1

Add a User

Write a command procedure to assist in adding a user account to the system. The requirements for the procedure are as follows:

- The procedure should set up the account, set up the disk quota entry, and create the directory.
- The procedure should prompt you for the proper information.

Hints

- The information you request for use with AUTHORIZE can be used for setting up the user's disk quota and creating the user's directory.
- AUTHORIZE can be used as a foreign command to execute a single AUTHORIZE command.

  Example:

  ```
  $ AUTHORIZE:=$AUTHORIZE
  $ AUTHORIZE SHOW username
  $
  ```

- Setting up disk quotas within the SYSMAN utility cannot be set up using a foreign command like AUTHORIZE. You can pass the information directly to the SYSMAN utility from within a command procedure.

  Example:

  ```
  $RUN SYS$SYSTEM:SYSMAN
  DISKQUOTA SHOW [*,*]
  EXIT
  ```

Solution

```
$ SET NOON         !Turns off automatic error checking
$!
$!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.
$!
$!Initialization Section
$!
$ AUTHORIZE :=$AUTHORIZE
$ USERDISK  :="DRA3:"        !The default disk for users
$
$ CURRENT_DISK = F$LOGICAL("SYS$DISK")
$ CURRENT_DIR  = F$DIRECTORY()
$ CURRENT_DEFAULT = CURRENT_DISK + CURRENT_DIR
$
$! SET DEFAULT SYS$SYSTEM
$
$ SET PROCESS/PRIV=SYSPRV
$ HAVE_SYSPRV = F$PRIVILEGE("SYSPRV")
$ IF HAVE_SYSPRV THEN $GOTO START
$!
$ WRITE SYS$OUTPUT "You do not have the privilege to add users"
$ WRITE SYS$OUTPUT "to the system"
$ EXIT
$!Main Section
$!
$ START:
$!
$! Request Account Information
$!
$ READ/PROMPT="Username: "  SYS$COMMAND  USERNAME
$ READ/PROMPT="Full Name: "  SYS$COMMAND  FULLNAME
$ READ/PROMPT="Password: "  SYS$COMMAND   PASSWORD
$ GET_GROUP_UIC:
$ READ/PROMPT="GROUP UIC (Give a number or an *) "-
   SYS$COMMAND   GROUP
$ WRITE SYS$OUTPUT " "
$ WRITE SYS$OUTPUT -
 "This is a listing of the group ''GROUP' UICs currently in use"
$ WRITE SYS$OUTPUT " "
$ AUTHORIZE SHOW ['GROUP',*]/BRIEF
$ IF GROUP .EQS. "*" THEN $GOTO GET_GROUP_UIC
$ READ/PROMPT="What member number do you want? " -
  SYS$COMMAND MEMBER
$ UIC = "[" + GROUP + "," + MEMBER + "]"
$ READ/PROMPT="Account Name: " SYS$COMMAND  ACCOUNT
$ READ/PROMPT="Privileges (If more than one, use commas: " -
  SYS$COMMAND  PRIVS
$ PRIVS = PRIVS - "(" - ")"
$ IF PRIVS .NES. "" THEN $PRIVS = "," + PRIVS
$ READ/PROMPT="Login Directory: " SYS$COMMAND  DIR
$ DIR = DIR - "[" - "]"
$ READ/PROMPT="Login Device: " SYS$COMMAND     DEVICE
$ DEVICE = DEVICE - ":"
$!
$AUTHORIZE ADD 'USERNAME'/OWNER="''FULLNAME'"-
 /ACCOUNT='ACCOUNT'/PASSWORD='PASSWORD'/DEV='DEVICE'-
 /DIRECTORY=['DIR']/UIC='UIC'/PRIV=(TMPMBX'PRIVS')
$!
$ AUTHORIZE SHOW 'USERNAME'
$!
$! Check for Disk quotas. If in use, request information
$!
$DQUOTA = 0
$ QUOTAS = F$SEARCH("''DEVICE':[0,0]QUOTA.SYS")
$ IF QUOTAS .EQS. "" THEN $GOTO ADD_DIRECTORY
$ READ/PROMPT="Disk Quota: " SYS$COMMAND DISK_QUOTA
$ READ/PROMPT="Disk Overdraft: " SYS$COMMAND OVERDRAFT
$ OPEN/WRITE FILE1 SYS$LOGIN:ADDQUOTA.TMP
$ WRITE FILE1 "$ SET DEFAULT ''DEVICE':[0,0]"
$ WRITE FILE1 "$ RUN SYS$SYSTEM:SYSMAN"
$ WRITE FILE1 -
  "  DISKQUOTA ADD ''UIC' /PERM=''DISK_QUOTA'/OVERDRAFT=''OVERDRAFT'"
$ CLOSE FILE1
$ @SYS$LOGIN:ADDQUOTA.TMP
$ DELETE SYS$LOGIN:ADDQUOTA.TMP;*
$ADD_DIRECTORY:
$ CREATE/DIR/OWNER='UIC' 'DEVICE':['DIR']/LOG
$  GOTO CLEANUP
$!Error Section
$!
$ CLEANUP:
$!Clean up Section
$!
$ SET DEFAULT  'CURRENT_DEFAULT'
$ EXIT
```

Laboratory Exercise 2

Backup Procedure

Write a command procedure to assist in backup operations on the system. The command procedure should have the following characteristics:

- The procedure should determine which disks are available on the system and prompt the user to indicate whether that disk should be backed up.
- The procedure should prompt the user for all the necessary information needed to back up the disk.
- The procedure can either ask if the user wants a **FULL** or **INCREMENTAL** backup, or it can make that decision based on the day of the week.

Hint

Before writing the command procedure, review the BACKUP command and determine:

- The exact form of the INCREMENTAL BACKUP command you want to use
- The exact form of the FULL BACKUP command you want to use

Solution

```
$ SET NOON         !Turns off automatic error checking
$!
$!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
$!
$!
$       DAY = F$CVTIME(,,"WEEKDAY")
$       BACKUP_TYPE = "INCREMENTAL"
$!
$!Main Section
$!
$       IF DAY .EQS. "Wednesday" THEN $ BACKUP_TYPE = "FULL"
$!
$ SHOW DEVICE D /MOUNTED /OUTPUT=SYS$LOGIN:MOUNTED.DAT
$
$ OPEN /READ IN_FILE SYS$LOGIN:MOUNTED.DAT
$ READ IN_FILE RECORD
$ READ IN_FILE RECORD
$ READ IN_FILE RECORD
$
$LOOP:
$ READ/END=END_LOOP IN_FILE RECORD
$
$ DISK = F$EXTRACT(0,14,RECORD)
$       DISK = F$EDIT(DISK,"COLLAPSE")
$       INQUIRE/NOPUN DOIT  -
        "Do you want to backup the disk ''DISK'? "
$       IF DOIT THEN $GOTO START_BACKUP
$ GOTO LOOP
$       START_BACKUP:
$       TAPE = ""
$       BPI = ""
$       VOL = ""
$       INQUIRE/NOPUN TAPE "BACKUP TO WHICH TAPE DRIVE? "
$       TAPE = TAPE - ":" + ":"
$       INQUIRE/NOPUN BPI  -
        "WHAT DENSITY SHOULD I USE (800/1600/6250)? "
$       INQUIRE/NOPUN VOL  -
        "WHAT VOLUME LABEL SHOULD BE PUT ON THE TAPE? "
$       IF BACKUP_TYPE .EQS. "FULL" THEN $GOTO FULL_BACKUP
$INCREMENTAL_BACKUP:
$ ALLOCATE 'TAPE'
$ INIT/DENSITY='BPI' 'TAPE' 'VOL'
$ MOUNT/FOREIGN 'TAPE'
$ BACKUP/REW/REC/FAST/IGNORE=INTER/DENS:'BPI'/BLOCK=32784-
 /BUFFER=5/MODIFIED/SINCE=BACKUP 'DISK'[*...]*.*;* 'vol'.bck
$ WRITE SYS$OUTPUT -
  "            THE INCREMENTAL BACKUP OF ''disk' IS COMPLETE"
$ DISMOUNT/NOUNLOAD 'TAPE'
$GOTO LOOP
$FULL_BACKUP:
$ ALLOCATE 'TAPE'
$ INIT/DENSITY='BPI' 'TAPE' 'VOL'
$ MOUNT 'TAPE'/FOR
$ BACKUP/REW/REC/DENS='BPI'/BLOCK=32784/BUFFER=5/IGNORE=INTER-
/IMAGE 'DISK' 'vol'.bck
$ WRITE SYS$OUTPUT -
  "           THE  FULL BACKUP OF ''DISK' IS COMPLETE"
$ DISMOUNT/NOUNLOAD 'TAPE'
$ GOTO LOOP
$END_LOOP:
$  GOTO CLEANUP
$!Error Section
$!
$ CLEANUP:
$!Clean up Section
$!
$ CLOSE IN_FILE /NOLOG
$ DELETE SYS$LOGIN:MOUNTED.DAT;* /NOLOG
$ EXIT
```

Laboratory Exercise 3

Disk Usage

The system manager uses the SYSMAN utility to enable disk quotas on a system to keep users from using too much disk space. However, it does not provide the following information:

- Total number of disk blocks that have been allocated on a particular disk
- List of users who have used more than 90%of their disk quota

Write a command procedure that performs the following functions for a specific disk that has disk quotas enabled:

- Calculate the percent used versus percent allocated for each user.
- Sum up the total blocks used.
- Sum up the total blocks allocated.
- Determine the total blocks available on the disk.
- Display the above items.
- Indicate which users have used more than 90%of their allocated disk quota.

Solution

```
$ SET NOON         !Turns off automatic error checking
$!
$!Comment Section
$!
$!      DISKUSE.COM
$!                  This procedure sums up the total disk
$!                  usage, disk allocation and calculates
$!                  the percent usage for each user on a
$!                  specified disk
$!
$!                  ASSUMPTION: This procedure is based on
$!                              output format of the DISKQUOTA
$!                              utility in SYSMAN. If that format
$!                              changes, this procedure will have
$!                              to be changed.
$!
$!Initialization Section
$!
$ON CONTROL_Y THEN GOTO CLEANUP
$USETOTAL=0
$PERMTOTAL=0
$DISK:='P1'
$ TAB := " "       !This symbol contains the TAB character
$!
$!Main Section
$!
$IF P1 .EQS. "" THEN $INQUIRE/NOPUN DISK  "Which disk? "
$ DISK = DISK - ":" + ":"
$!
$! Create a file with the DISKQUOTA information
$!
$ DEFINE WORK_DISK  'DISK'
$ DEFINE/USER  SYS$OUTPUT  SYS$LOGIN:DISK.TMP
$RUN SYS$SYSTEM:SYSMAN
 DISKQUOTA SHOW * /DEVICE=WORK_DISK
$!
$! If there was an error, (ie. quotas not enabled) print out
$! the file that has the message.
$!
$ IF .NOT. $SEVERITY
$ THEN
$   DEASSIGN WORK_DISK
$   TYPE SYS$LOGIN:DISK.TMP
$   GOTO CLEANUP
$   ENDIF
$!
$DEASSIGN WORK_DISK
```

```
$!
$! Read past the fist 3 header lines in the file
$!
$OPEN/READ  QUOTAFILE   SYS$LOGIN:DISK.TMP
$READ  QUOTAFILE   DUMMY
$READ  QUOTAFILE   DUMMY
$READ  QUOTAFILE   DUMMY
$!
$! Write the header for the output.
$!
$WRITE SYS$OUTPUT ""
$WRITE SYS$OUTPUT "   User ",tab,tab,tab," Number of ",tab,-
 "Total Blocks"
$WRITE SYS$OUTPUT "        ",tab,tab,tab,"Blocks Used",tab,-
 " Allocated",tab,"     Percent"
$WRITE SYS$OUTPUT ""
$!
$PERCENT = 0
$MARK = ""
$! Go through each record and get the user, blocks used, and permanent
$! quota.  Then calculate the percent of blocks used.
$!
$LOOP:
$READ/END=FINAL  QUOTAFILE  DISKRECORD
$!
$USER = F$EXTRACT(0,26,DISKRECORD)
$USENUMBER= F$INTEGER(F$EXTRACT(26,10,DISKRECORD))
$PERMNUMBER= F$INTEGER(F$EXTRACT(39,10,DISKRECORD))
$IF USENUMBER .EQ. 0 .AND. PERMNUMBER .EQ. 0 THEN GOTO LOOP
$USETOTAL=USETOTAL+USENUMBER
$PERMTOTAL=PERMTOTAL+PERMNUMBER
$PERCENT= (100 * USENUMBER)/PERMNUMBER
$IF PERCENT .GE. 90 THEN MARK = "***"
$!
$! Create and display the output record
$!
$OUT_REC[0,26] := 'USER'
$OUT_REC[34,10] := 'USENUMBER'
$OUT_REC[52,10] := "''PERMNUMBER'"
$OUT_REC[70,3] := 'PERCENT'
$OUT_REC[73,1] := "%"
$OUT_REC[74,3] := 'MARK'
$WRITE SYS$OUTPUT OUT_REC
$GOTO LOOP
$!
$! Display the final statistics
$!
$FINAL:
$CLOSE QUOTAFILE
$MAX_NUMBER = F$getdvi(DISK,"maxblock")
```

```
$WRITE SYS$OUTPUT ""
$WRITE SYS$OUTPUT  "The ''disk' disk blocks USED is   ",-
 USETOTAL
$WRITE SYS$OUTPUT  "            "
$WRITE SYS$OUTPUT  "The ''disk' disk blocks ALLOCATED is   ",-
 PERMTOTAL
$!
$WRITE SYS$OUTPUT "  "
$WRITE SYS$OUTPUT  "The ''disk' disk blocks AVAILABLE is   ",-
 MAX_NUMBER
$!
$!Clean up Section
$!
$CLEANUP:
$IF F$TRNLNM("WORK_DISK","LNM$PROCESS") .NES. "" THEN DEASSIGN WORK_DISK
$IF F$TRNLNM("QUOTAFILE","LNM$PROCESS") .NES. "" THEN CLOSE QUOTAFILE
$IF F$SEARCH("SYS$LOGIN:DISK.TMP") .NES. "" THEN -
  DELETE  SYS$LOGIN:DISK.TMP;*
$EXIT
```

Laboratory Exercise 4

Password Expiration

You have decided that you want a command procedure listing the password expiration dates for all your users. Since this information is contained in the SYSUAF.DAT file, you have decided to use a SYSUAF.LIS file as your input. The suggestion is that you write the procedure in two stages. Write your procedure using the criteria in stage one. Then, when that procedure works, modify it to include the items listed under stage two.

Stage One

The output list should contain the user's name and the date on which the password expires.

The solution can be found in the file ADV_CUST: PASSWORD_1.COM.

Stage Two

- The procedure will output a complete list of users and their password expiration dates (including those about to expire) by using the command:

  ```
  $ @filename FULL
  ```

- The output should include a series of apostrophes (') marking any user whose password expires within seven days or has already expired.

- The procedure will output a list of users and expiration dates for passwords about to expire in seven days, using the command:

  ```
  $ @filename
  ```

- The solution can be found in the file ADV_CUST: PASSWORD_2.COM.

Hints

- Your initial work file is SYSUAF.LIS. Can you think of any DCL command or VMS utility that filters the file leaving just the username and password lifetime information? Try to answer this question **without** looking at the solution.
- When comparing times in a command procedure, use the COMPARISON time format. When displaying times on the screen, use the ABSOLUTE time format.

Solution (Stage One)

```
$ SET NOON         !Turns off automatic error checking
$!
$!Comment Section
$!
$!        PASSWORD_1.COM
$!
$!        This procedure will output a complete list of users
$!        and their expiration dates.
$!
$!        Assumption:
$!                   This procedure is based on the current (V5.0)
$!                   SYSUAF.LIS file format. If that changes, the
$!                   procedure will have to be updated.
$!
$!Initialization Section
$!
$authorize:=$authorize
$!
$!Main Section
$!
$If f$logical("SYSUAF") .EQS. "" THEN -
  $DEFINE/USER SYSUAF  SYS$SYSTEM:SYSUAF
$authorize list */full
$search/output=sysuaf.tmp sysuaf.lis "USERNAME","PWDLIFETIME"
$OPEN/READ FILE1 SYSUAF.TMP
$LOOP:
$READ/END=CLEANUP FILE1 NAME_RECORD
$USER_NAME = F$EXTRACT(9,20,NAME_RECORD)
$READ FILE1 PWD_RECORD
$DAYS = F$EXTRACT(19,4,PWD_RECORD)
$DAYS = F$EDIT(DAYS,"COLLAPSE")
$IF DAYS .EQS. "" THEN $GOTO LOOP
$BASE_DATE = F$EXTRACT(45,11,PWD_RECORD)
$BASE_DATE = F$EDIT(BASE_DATE,"COLLAPSE")
$IF BASE_DATE .EQS. "(pre-ex" THEN $GOTO LOOP
$ D  = F$CVTIME("''BASE_DATE'+''DAYS'-","ABSOLUTE","DATE")
$WRITE SYS$OUTPUT " "
$WRITE SYS$OUTPUT "USERNAME= ",USER_NAME," ", -
 "Expiration date = ",D
$GOTO LOOP
$!Error Section
$ ERROR_SECTION:
$!
$!Clean up Section
$ CLEANUP:
$CLOSE FILE1
$DELETE SYSUAF.LIS;
$DELETE SYSUAF.TMP;
$ EXIT
```

Solution (Stage Two)

```
$ SET NOON         !Turns off automatic error checking
$!
$!Comment Section
$!
$!        PASSWORD_2.COM
$!
$!        This procedure will output a complete list of users
$!        and their expiration dates.
$!
$!        Procedure Format:
$!
$!        To get a complete list of users and their expiration
$!        dates including those about to expire, use the command
$!
$!               $@filename  FULL
$!
$!        The passwords that will expire within 7 days or already
$!        have expired are indicateD with a series of *'s.
$!
$!
$!        To get a list of PASSWORD expiration dates for users
$!        about to expire in 7 days, use the command
$!
$!              $@filename
$!
$!        Assumption:
$!                   This procedure is based on the current (V5.0)
$!                   SYSUAF.LIS file format. If that changes, the
$!                   procedure will have to be updated.
$!
$!
$!Initialization Section
$!
$P1 :="''P1'"
$NEXT_STEP = "LOOP"
$authorize:=$authorize
$!
$!Main Section
$!
$type sys$input
        This procedure gives a list of the PASSWORD expiration
        dates.
$
$IF P1 .EQS. "FULL" THEN $NEXT_STEP = "NOT_READY"
$IF F$LOGICAL("SYSUAF") .EQS. "" THEN -
  $DEFINE/USER SYSUAF  SYS$SYSTEM:SYSUAF
```

```
$AUTHORIZE LIST */FULL
$SEARCH/OUTPUT=SYSUAF.TMP SYSUAF.LIS "USERNAME","PWDLIFETIME"
$OPEN/READ FILE1 SYSUAF.TMP
$LOOP:
$READ/END=CLEANUP FILE1 NAME_RECORD
$USER_NAME = F$EXTRACT(9,20,NAME_RECORD)
$READ FILE1 PWD_RECORD
$DAYS = F$EXTRACT(19,4,PWD_RECORD)
$DAYS = F$EDIT(DAYS,"COLLAPSE")
$IF DAYS .EQS. "" THEN $GOTO LOOP
$BASE_DATE = F$EXTRACT(45,11,PWD_RECORD)
$BASE_DATE = F$EDIT(BASE_DATE,"COLLAPSE")
$IF BASE_DATE .EQS. "(pre-ex" THEN $GOTO LOOP
$ !
$ D  = F$CVTIME("''BASE_DATE'+''DAYS'-","COMPARISON","DATE")
$ CURRENT_DAY = F$TIME()
$ OFFSETDAYS = 7
$ THEN_DAY  = F$CVTIME("''CURRENT_DAY'+''OFFSETDAYS'-",-
  "COMPARISON","DATE")
$ IF D .GTS. THEN_DAY THEN $ GOTO 'Next_STEP'
$ D  = F$CVTIME("''BASE_DATE'+''DAYS'-","ABSOLUTE","DATE")
$WRITE SYS$OUTPUT " "
$WRITE SYS$OUTPUT "USERNAME= ",USER_NAME," ",-
  "Expiration date = ",D,"  ******"
$ GOTO LOOP
$ NOT_READY:
$ D  = F$CVTIME("''BASE_DATE'+''DAYS'-","ABSOLUTE","DATE")
$WRITE SYS$OUTPUT " "
$WRITE SYS$OUTPUT "USERNAME= ",USER_NAME," ", "Expiration date = ",D
$GOTO LOOP
$!Error Section
$ ERROR_SECTION:
$!
$!
$!Clean up Section
$ CLEANUP:
$!
$CLOSE FILE1
$DELETE SYSUAF.LIS;
$DELETE SYSUAF.TMP;
$ EXIT
```

<page>

Laboratory Exercise 5

Monitoring System Resources

Write a command procedure to be submitted as a batch job to gather system resource usage statistics. The command procedure should include the following:

- Use logical F$EXTRACT in an expression
- Use MONITOR STATE, PROC, PAGE, POOL, and IO/RECORD=
- Specify monitoring time
- Submit itself to SYS$BATCH to be executed at a designated time

Solution

```
$ ON CONTROL_Y then GOTO EXIT
$ !
$ SET NOVERIFY
$ !
$ ! MONITOR_RESOURCES.COM
$ !
$ DATE = F$EXTRACT(0,2,F$TIME()) + F$EXTRACT(3,3,F$TIME())
$ IF F$EXTRACT(0,1,DATE) .EQS. " " then DATE = F$EXTRACT(1,4,DATE)
$ !
$ MON_FILE := "MON"'DATE'.LOG
$ MONITOR STATE,PROC,PAGE,POOL,IO/RECORD=SYS$SYSROOT:[SYSMGR]'MON_FILE' -
         /INTER=60/NODISP/ALL/BEGIN=8:00/END=18:00
$ !
$ EXIT:
$ SET PROT=(SY:RWE,OW:RWE,WO,GR) SYS$SYSROOT:[SYSMGR]'MON_FILE';*
$ SUBMIT SYS$SYSROOT:[SYSMGR]MONITOR_RESOURCES.COM-
_/NOLOG/QUE=SYS$BATCH/AFTER=23:59:59
```

Laboratory Exercise 6

Submit a Job

Write a brief command procedure to assist you in submitting a job to a queue. The command procedure should include the following:

- Use logicals F$EDIT and F$GETSYI in an expression for the symbol name NODE
- Prompt for following information:
  - — The directory and name of the log file
  - — Queue name
  - — Job name
- Use parameters to pass information.

Solution

```
$ NODE = F$EDIT(F$GETSYI("NODENAME"),"TRIM")
$INQUIRE P1 "Enter directory and name for log file"
$INQUIRE P2 "Enter queue name"
$INQUIRE P3 "Enter job name"
$SUBMIT/NOTIFY/NOPRINT/LOG='P1'/QUEUE='P2' 'P3'
$SHOW QUEUE 'NODE'*
$EXIT
```

Laboratory Exercise 7

Get Queue Information

Write a command procedure using the F$GETQUI lexical function that will show information about print jobs having a block size larger than a certain value. The value for the block size limit can be:

- Passed as the first parameter
- Prompted for in the procedure

Display the following information about the print jobs:

- Queue name
- Entry number
- Job name
- Username
- Size

Solution

```
$! GETQUI.COM displays the print jobs that are larger than a specified
$! number of blocks.  This number can be passed as the first parameter
$! or will be asked for if it is not passed.  The information that is
$! shown includes the queue name, entry number, job name, username,
$! and size of each job.
$!
$ SET NOON
$!
$! If the size was not passed, ask for it. If a RETURN is input, assume
$! all print jobs will be displayed.
$!
$ IF P1 .EQS. "" THEN INQUIRE P1 "Enter the block limit to search "
$ IF P1 .EQS. "" THEN P1 = 0
$!
$! Cancel any previous F$GETQUI call
$!
$ TEMP = F$GETQUI("CANCEL_OPERATION")
$!
$! Print the header
$!
$ WRITE SYS$OUTPUT -
"         JOBS WITH MORE THAN ''P1' BLOCKS"
$ WRITE SYS$OUTPUT -
"         =================================
$ WRITE SYS$OUTPUT ""
$ WRITE SYS$OUTPUT ""
$ WRITE SYS$OUTPUT -
"    QUEUE       ENTRY     JOBNAME            USERNAME    SIZE"
$ WRITE SYS$OUTPUT -
"    -----       -----     _______            --------    ----"
$ WRITE SYS$OUTPUT ""
$!
$! Loop through each of the print queues (Printer, Server, and Terminal
$! queues) and get their names.  This will set up the queue context so we
$! can get information about the jobs in the queues.  When we have
$! examined all of them, the QNAME returned will be null.
$!
$QUEUE_LOOP:
$ QNAME = F$GETQUI("DISPLAY_QUEUE","QUEUE_NAME","*","SYMBIONT")
$ IF QNAME .EQS. "" THEN GOTO FINISH
$!
$! Loop through each job in the current queue.  When we have processed
$! all the jobs in the queue, go back to the queue loop to get the
$! next queue.
$!
```

```
$JOB_LOOP:
$ JNAME = F$GETQUI("DISPLAY_JOB","JOB_NAME",,"ALL_JOBS")
$ IF JNAME .EQS. "" THEN GOTO QUEUE_LOOP
$!
$! If the size of this job is less than or equal to what we are looking
$! for, don't process it anymore.. Go and get the next job.
$!
$ JSIZE = F$GETQUI("DISPLAY_JOB","JOB_SIZE",,"FREEZE_CONTEXT")
$ IF JSIZE .LE. P1 THEN GOTO JOB_LOOP
$!
$! The job is larger than the specified value, so get the username and
$! entry number.
$!
$ UNAME = F$GETQUI("DISPLAY_JOB","USERNAME",,"FREEZE_CONTEXT")
$ ENTRY = F$GETQUI("DISPLAY_JOB","ENTRY_NUMBER",,"FREEZE_CONTEXT")
$!
$! Set up an output record and print it.  Then get the next job.
$!
$ OUTPUT_RECORD[0,13] := 'QNAME'
$ OUTPUT_RECORD[16,5] := 'ENTRY'
$ OUTPUT_RECORD[23,18] := 'JNAME'
$ OUTPUT_RECORD[43,12] := 'UNAME'
$ OUTPUT_RECORD[57,5] := 'JSIZE'
$ WRITE SYS$OUTPUT OUTPUT_RECORD
$ GOTO JOB_LOOP
$!
$! All done.
$!
$FINISH:
$ EXIT
```

CUSTOMIZING THE DCL ENVIRONMENT

Laboratory Exercise 1

Perform the following exercises to practice managing logical names and logical name tables. Use the SYSPRV privilege to do some of this work.

1. Create a private logical name table named MY_TABLE.

2. Create a private logical name table named MY_CONF_TABLE with the CONFINE attribute.

3. Create a shareable logical name table named MY_SYS_TABLE. Specify the system directory table as its parent.

4. Display the names of all the tables to which you can gain access.

5. Create a command procedure containing the following commands:

```
$ SET VERIFY
$ SHOW LOGICAL/STRUCTURE
$ ASSIGN/TABLE=MY_TABLE DRA0: SYS_DISK
$ ASSIGN/TABLE=LNM$PROCESS DRA2: WORK2
$ ASSIGN/TABLE=MY_SYS_TABLE DRA3: WORK3
$ SHOW LOGICAL/TABLE=MY_TABLE SYS_DISK
$ SHOW LOGICAL/TABLE=LNM$PROCESS WORK2
$ SHOW LOGICAL/TABLE=MY_SYS_TABLE WORK3
```

6. Create a subprocess using the SPAWN command to execute your procedure. Send the output to a file by using the /OUTPUT qualifier with the SPAWN command.

7. After control returns to your process, execute the commands below. (They are the same as the last three commands in the command procedure executed by the subprocess.)

```
$ SHOW LOGICAL/TABLE=MY_TABLE SYS_DISK
$ SHOW LOGICAL/TABLE=LNM$PROCESS WORK2
$ SHOW LOGICAL/TABLE=MY_SYS_TABLE WORK3
```

8. Compare the results of these commands executed in your process with the results of the same ones executed in the subprocess (display the contents of the output file).

9. What are the differences between the results? Why are the results different?

Solution

Enter the following commands:

1. $ CREATE/NAME_TABLE MY_TABLE
2. $ CREATE/NAME_TABLE/ATTRIBUTE=CONFINE MY_CONF_TABLE
3. $ CREATE/NAME_TABLE/PARENT=LNM$SYSTEM_DIRECTORY MY_SYS_TABLE
4. $ SHOW LOG/STRUCTURE
5. No solution needed.
6. $ SPAWN/INPUT=CREATE_LOG.COM/OUTPUT=CREATE_LOG.LOG
7. $ SHOW LOGICAL/TABLE=MY_TABLE SYS_DISK
   $ SHOW LOGICAL/TABLE=LNM$PROCESS WORK2
   $ SHOW LOGICAL/TABLE=MY_SYS_TABLE WORK3
8. Contents of log file:

```
$ SHOW LOGICAL/STRUCTURE
(LNM$PROCESS_DIRECTORY)
    (LNM$PROCESS_TABLE)
    (MY_TABLE)
(LNM$SYSTEM_DIRECTORY)
    (LNM$GROUP_000013)
    (LNM$JOB_8016E180)
    (LNM$SYSTEM_TABLE)
    (MY_SYS_TABLE)
$ ASSIGN/TABLE=MY_TABLE DRA0: SYS_DISK
$ ASSIGN/TABLE=LNM$PROCESS DRA2: WORK2
$ ASSIGN/TABLE=MY_SYS_TABLE DRA3: WORK3
$ SHOW LOGICAL/TABLE=MY_TABLE SYS_DISK
  "SYS_DISK" = "DRA0:"   (MY_TABLE)
$ SHOW LOGICAL/TABLE=LNM$PROCESS WORK2
  "WORK2" = "DRA2"   (LNM$PROCESS_TABLE)
$ SHOW LOGICAL/TABLE=MY_SYS_TABLE WORK3
  "WORK3" = "DRA3"   (MY_SYS_TABLE)
```

9. Differences:

The SHOW LOGICAL command displays the equivalence strings of SYS_DISK and WORK2 while in the subprocess, but displays error messages in the parent process instead of translating the names.

Reason:

You created the logical names SYS_DISK and WORK2 in the subprocess' private logical name tables. The parent process cannot use those tables. (Actually, the system deletes the tables belonging to the subprocess when it deletes the subprocess.) Notice that the parent process does have access to the logical name in the shareable table that the subprocess created.

Laboratory Exercise 2

Find the following files on the course disk:

```
CUST_DCL:FORMAT.EXE - a user-written program
CUST_DCL:FORMAT.CLD - corresponding CLD file
CUST_DCL:FORMAT.HLP - corresponding HELP text
```

1. Create a DCL command that runs the FORMAT program.

   Hint: Since you are probably not the only person performing this exercise, create the command in your process DCL tables rather than in the system-wide tables.

2. Enter your command and observe the results.
3. Make the HELP text available to system users. Observe the HELP output.

Solution

1. Create a DCL command that runs the FORMAT program.

```
$ SET COMMAND CUST_DCL:FORMAT
```

2. Enter your command and observe the results.

```
$ FORMAT
```

3. Make the HELP text available to system users. Observe the HELP output.

```
$ LIBRARY/HELP SYS$HELP:HELPLIB.HLB CUST_DCL:FORMAT
$ HELP FORMAT
```

Laboratory Exercise 3

1. Create a text file named TEXT.HLP. The file should contain the following text:

   1 BAKER

   Baker is a fictitious character created by a user.

   2 SCHOOLS

   Baker, being a fictitious character, attended several nonexistent schools.

   2 JOBS

   As you may have guessed, Baker held several fictitious jobs.

2. Create an alternate HELP library, and insert TEXT.HLP into that HELP library.

3. Define a logical name that will cause your alternate HELP library to be searched automatically.

4. Enter the command HELP. Observe the output. Enter the command HELP BAKER. Request help for each of the items listed.

Solutions

1. Use the CREATE command or a text editor to create TEXT.HLP.

2. Create an alternate HELP library, and insert TEXT.HLP into that HELP library.

   ```
   $ LIBRARY/HELP/CREATE SYS$HELP:NANCY_HELP.HLB TEXT.HLP
   ```

3. Define a logical name that will cause your alternate HELP library to be searched automatically.

   ```
   $ DEFINE/SYSTEM HLP$LIBRARY SYS$HELP:NANCY_HELP.HLB
   ```

4. Enter the HELP command. Observe the output (the list of DCL commands now includes the command BAKER). Enter the command HELP BAKER. Request help for each of the items listed.

Laboratory Exercise 4

Find the following command procedure on the course disk:

CUST_DCL:CAPTIVE_LOGIN.COM

1. Create a captive account. Use CAPTIVE_LOGIN.COM to control the user environment when the captive user logs in to the system.
2. Ensure that the captive user cannot use CTRL/Y to abort the captive log-in procedure.
3. Log in to the captive account and observe the results.

Solution

1. The UAF record for your captive account should look similar to the following example. The LGICMD field must specify the captive log-in command procedure.

```
Username: SMITH            Owner:
Account:  SMITH            UIC:  [205,1] ([SMITH])
CLI:      DCL              Tables:
Default:  DISK$:[SMITH]
LGICMD:   CUST_DCL:CAPTIVE_LOGIN.COM
Login Flags:  Disctly Lockpwd Captive Diswelcome
Primary days:   Mon Tue Wed Thu Fri Sat Sun
Secondary days:
No access restrictions
Expiration:     (none)    Pwdminimum:  6   Login Fails: 0
Pwdlifetime: 180 00:00    Pwdchange:  19-MAY-1987 10:06
Last Login:  18-MAY-1987 16:24 (interactive), (none)
             (non-interactive)
Maxjobs:          1  Fillm:        40  Bytlm:         8192
Maxacctjobs:      1  Shrfillm:      0  Pbytlm:           0
Maxdetach:        1  BIOlm:        18  JTquota:       1024
Prclm:            0  DIOlm:        18  WSdef:          150
Prio:             4  ASTlm:        24  WSquo:          200
Queprio:          0  TQElm:        10  WSextent:       300
CPU:         (none)  Enqlm:       200  Pgflquo:      10000
Authorized Privileges:
  TMPMBX
Default Privileges:
  TMPMBX
```

2. The DISCTLY flag disables the captive user from using the CTRL/Y sequence to abort the captive log-in command procedure.

3. Log in to your captive account and observe the results.

Laboratory Exercise 5

To perform this exercise, you need a VMS system with a console storage device, and the console media must be on-line.

1. Display a directory listing of the files on the console device.
2. Select a boot file that is interesting to you, and copy it from the console device to your directory.
3. From what device and unit does this boot file cause the system to boot?
4. If there is a HELP file named BOOT.HLP on the console, copy it to your directory and read it. BOOT.HLP contains processor-specific information about booting.

Solution

1. Display a directory listing of the files on the console device.

   ```
   $ RUN SYS$SYSTEM:SYSGEN
   SYSGEN> CONNECT CONSOLE
   SYSGEN> EXIT
   $ EXCHANGE DIR CSA1:
   ```

2. Select a boot file that is interesting to you, and copy it from the console device to your directory.

   ```
   $ EXCHANGE COPY xxnBOO.COM *
   ```

   In place of xxnBOO.COM, you would specify the actual filename of the boot file of interest.

   Note that the file extension will be .CMD (instead of .COM) for some processors.

3. From what device and unit does this boot file cause the system to boot?

   You can usually determine the type of device and unit number from the name of the boot file. (Refer to the tables entitled Device Codes Used in Boot File Names and Boot File Naming Conventions in the Student Workbook).

   However, if the boot file name is DEFBOO.COM (or .CMD), you must read the comments in the boot file to determine the device and unit number.

   ```
   $ TYPE xxnBOO.COM
   ```

4. If there is a HELP file named BOOT.HLP on the console, copy it to your directory and read it. BOOT.HLP contains processor-specific information about booting.

   ```
   $ EXCHANGE COPY CSA1:BOOT.HLP *
   ```

   ```
   $ TYPE BOOT.HLP
   ```

VMS DISK FILE STRUCTURE

Written Exercises

The laboratory exercises for the VMS Disk File Structure module are divided into two sections, written exercises and on-line exercises. The written exercises allow you to review your knowledge of the topics covered in the corresponding lecture, and the on-line exercises allow you to practice using the VMS system management utilities and commands related to disk structure.

1. Describe briefly the uses of the following files:

   a. INDEXF.SYS

   b. BITMAP.SYS

   c. 000000.DIR

2. You have initialized a disk with a cluster size of 10. The average number of **bytes** of data your programs write to disk is 5630. How many blocks will be allocated to these files? Did you choose a good cluster size?

3. Where are file headers located and what purpose do they serve?

4. What happens if the disk write lock button is pushed while the disk is being accessed? Can it recover?

5. List at least two instances in which a disk will enter into mount verification.

6. What happens when a volume rebuild occurs?

7. Which of these commands would be dangerous to use on a disk that contains files you want to keep?

____$ANALYZE/ERROR
____$SET VOLUME/NOHIGHWATER_MARKING
____$ANALYZE/DISK
____$BACKUP
____$ANALYZE/MEDIA/EXERCISE
____$MOUNT
____$DISMOUNT
____$INITIALIZE
____$SET VOLUME/REBUILD

8. Which of these commands should not be used on a disk that users are currently accessing?

_____$ANALYZE/ERROR
_____$SET VOLUME/NOHIGHWATER_MARKING
_____$ANALYZE/DISK
_____$BACKUP
_____$ANALYZE/MEDIA/EXERCISE
_____$MOUNT
_____$DISMOUNT
_____$INITIALIZE
_____$SET VOLUME/REBUILD

On-Line Exercises

1. What is the cluster size on the system disk? Would it be possible to change the cluster factor while the system is running? (NOTE: DO NOT TRY THIS.)

2. Pick another disk on the system and find the cluster size. Is there a difference? If so, why?

3. Write a command procedure (to be submitted to batch) to perform the following DCL command:

   ```
   $DIRECTORY/SIZE=ALL [*...]
   ```

   Use the disk on which your directory resides. Determine how much disk space is wasted because of allocated space versus space in use.

   (Hint: Use $SUBMIT/NOPRINT so that you can look at the log file on-line.)

4. What is the maximum number of files that can be created on the system disk? Can you change it while the system is running?

5. Enter the following command to examine various sections of the cluster bitmap file BITMAP.SYS (substitute increasing values for n):

```
$DUMP/BLOCK=(START:n,COUNT:2) SYS$SYSDEVICE:[000000]BITMAP.SYS
```

See if there is a point where all the clusters are free. What would it mean if this continued for a large number of blocks?

NOTE

The following lab exercises allow you to examine file headers and contents of files using the DUMP command after trying several operations. Also, your instructor will submit a batch command procedure to run after class that will output the results of the following command:

$ANALYZE/DISK_STRUCTURE/REPAIR

The results of this command will be written to:

FILES_11:ANALYZE_DISK.TXT

This will enable you to see the results of the Analyze/Disk utility without locking the disk during lab time. Therefore, do not execute this command during class time.

6. Dump the file headers for the following files:

   SYS$MANAGER:ACCOUNTNG.DAT

   SYS$MANAGER:OPERATOR.LOG;-1

   SYS$ERRORLOG:ERRLOG.SYS

   Are any of these files fragmented? Why would they tend to be fragmented?

7. If you have fragmented files, you have two methods for making them contiguous. There is one that you can use while the system is running. Try that command on one of the fragmented files in the previous exercise and have the contiguous file go to your log-in directory.

8. Create a file in your directory. Looking at the file header for that file, use the back link file identification (in the header) to verify the directory in which it is cataloged.

9. Create a subdirectory with a version limit of 3. Can you use the DIRECTORY command to find the version limit established? Use the command that will give you this information.

10. Create a file in the subdirectory you created above. Use the command $SET FILE/ENTER to create another directory entry in your top-level directory for that file. Verify that both file records point to the same file header.

11. Delete the file in your subdirectory. Do a $DIR/FULL on the file name in your top-level directory and explain the results. Get rid of the directory record.

12. Create several files in your subdirectory. Rename the subdirectory to have a version number of 2 and try to get a listing of the files. Can you fix it?

13. Look at a full directory listing of the actual directory file (xxx.DIR). Now, issue the command $SET FILE/NODIRECTORY xxx.DIR and see what changes.

14. The .DIR file (from the previous exercise) is no longer a directory file because you have removed its directory attributes. Can you restore the directory attributes of the .DIR file to cause it to become a directory file again?

    The answer is **No,** you cannot restore the directory attributes of a file after you use the SET FILE/NODIRECTORY command. Therefore, you must delete the .DIR file, causing any files cataloged in that directory to become lost.

    After the instructor uses the $ANALYZE/DISK/REPAIR command to locate the lost files, check for the files in the appropriate location and examine their characteristics using the $DIRECTORY/FULL command. Where does it get all that information?

NOTE

The instructor will probably wait until after-hours to use ANALYZE/DISK/REPAIR. If so, you might have to wait until the next day to complete this exercise.

Solutions to Written Exercises

1. Describe briefly the uses of the following files:

   a. INDEXF.SYS

      1) Allows us to access any other file on the disk
      2) Contains the bootstrap program to enable a VAX-11/750 system to boot from this disk

   b. BITMAP.SYS - identifies which clusters are available when we need to create or expand a file.

   c. 000000.DIR - the master file directory, which contains the directory entry for all user file directories and reserved files on the disk.

2. You have initialized a disk with a cluster size of 10. The average number of **bytes** of data your programs write to disk is 5630. How many blocks will be allocated to these files? Did you choose a good cluster size?

   Twenty blocks will be allocated to these files, but only eleven will be used. The cluster size of 10 was **not** a good choice. Too many blocks are allocated but unused; therefore, they are not available to others.

3. Where are file headers located and what purpose do they serve? The file headers are located in INDEXF.SYS. They contain information to check for access validity, identification, accounting and RMS information, and the pointers to the actual data clusters.

4. What happens if the disk write lock button is pushed while the disk is being accessed? Can it recover?

   If someone is attempting to write to the disk, it will not allow the write to occur. It does recover by initiating a mount verification.

5. List at least two instances in which a disk will enter into mount verification.

   a. Disk is taken off-line and comes back on-line.

   b. Disk is write-locked and someone attempts a write operation.

   c. Device contains the wrong volume.

   d. Hardware error is detected on the disk.

6. What happens when a volume rebuild occurs?

   a. Index file bit map is updated.

   b. Storage bit map for allocated clusters is updated.

   c. Disk quota entries are updated.

7. Which commands should not be used on a disk containing files you want to keep?

____$ANALYZE/ERROR
____$SET VOLUME/NOHIGHWATER_MARKING
____$ANALYZE/DISK
____$BACKUP
_X__$ANALYZE/MEDIA/EXERCISE
____$MOUNT
____$DISMOUNT
_X__$INITIALIZE
____$SET VOLUME/REBUILD

8. Which commands should not be used on a disk that users are currently accessing?

____$ANALYZE/ERROR
____$SET VOLUME/NOHIGHWATER_MARKING
_X__$ANALYZE/DISK (it locks the disk while users are on it)
_X__$BACKUP (if backing up entire disk or files that someone is accessing)
_X__$ANALYZE/MEDIA/EXERCISE
_X__$MOUNT
_X__$DISMOUNT
_X__$INITIALIZE
_X__$SET VOLUME/REBUILD (it locks the disk)

Solutions to On-Line Exercises

1. What is the cluster size on the system disk? Would it be possible to change the cluster factor while the system is running? (NOTE: DO NOT TRY THIS.)

   $SHOW DEVICE/FULL SYS$SYSDEVICE - Look at the cluster size.

   You cannot change the cluster size without reinitializing the volume and then a $BACKUP/IMAGE/NOINITIALIZE back to it. (AGAIN - DO NOT ATTEMPT THIS!!!)

2. Pick another disk on the system and find the cluster size. Is there a difference? If so, why?

   $SHOW DEVICE/FULL D - Will show you all the disks. Look at the cluster size.

   If a disk has a cluster size = 1, it means many small files are probably on that disk, or it has less than 50,000 blocks. For disks larger than 50,000 blocks, the default cluster size is three.

3. Write a command procedure (to be submitted to batch) to perform the following DCL command:

   ```
   $DIRECTORY/SIZE=ALL [*...]
   ```

   Use the disk on which your directory resides. Determine how much disk space is wasted because of allocated space versus space in use.

   (Hint: Use $SUBMIT/NOPRINT so that you can look at the log file on-line.)

4. What is the maximum number of files that can be created on the system disk? Can you change it while the system is running?

   $SHOW DEVICE/FULL SYS$SYSDEVICE - Look at maximum files allowed.

   You cannot change the maximum number of files allowed without reinitializing the volume (NOTE: DO NOT TRY THIS.)

5. Enter the following command to look at various sections of the cluster bitmap file BITMAP.SYS (substitute increasing values for n):

```
$DUMP/BLOCK=(START:n,COUNT:2) SYS$SYSDEVICE:[000000]BITMAP.SYS
```

See if there is a point where all the clusters are free. What would it mean if this continued for a large number of blocks?

```
$DUMP/BLOCK=(START:500,COUNT:2) SYS$SYSDEVICE:[000000]BITMAP.SYS
```

If you see all FFFF in the dump, this indicates free blocks. Recall that a bit with the value of 0 means in use, 1 means free. You might notice a lot of free blocks near the end of a disk, especially after a recent image restore. This would make all files contiguous and leave all the free disk space at the end.

NOTE

The following lab exercises allow you to examine file headers and contents of files using the DUMP command after trying several operations. Also, your instructor will submit a batch command procedure to run after class that will output the results of the following command:

```
$ ANALYZE/DISK_STRUCTURE/REPAIR
```

The results of this command will be written to:

```
FILES_11:ANALYZE_DISK.TXT
```

This will enable you to see the results of the Analyze/Disk utility without locking the disk during lab time. Therefore, **do not** execute this command during class lab time.

6. Dump the file headers for the following files:

   ```
   SYS$MANAGER:ACCOUNTNG.DAT

   SYS$MANAGER:OPERATOR.LOG;-1

   SYS$ERRORLOG:ERRLOG.SYS.
   ```

   Are any of these files fragmented? Why would they tend to be fragmented?

   ```
   $DUMP/HEADER/BLOCKS=COUNT=0 filename
   ```

   If you see more than seven retrieval pointers for a file, it is fragmented. These files typically get fragmented over time because they are written to several different times. When an additional write takes place, there are probably not enough consecutive blocks, so a new mapping pointer needs to be used.

7. If you have fragmented files, you have two methods for making them contiguous. You can use one method while the system is running. Try that command on one of the fragmented files in the previous exercise and have the contiguous file go to your log-in directory.

   The two methods are:

   ```
   $BACKUP/IMAGE on the disk
   or
   $COPY/CONTIGUOUS  on the file (you can do this in lab)
   ```

   ```
   $COPY/CONTIGUOUS file_of_choice SYS$LOGIN:*.*
   ```

8. Create a file in your directory. Looking at the file header for that file, use the back link file identification (in the header) to verify the directory in which it is cataloged.

   ```
   $DUMP/HEADER/BLOCKS=COUNT=0  myfile.dat
        (find the back link file id)

   $DIRECTORY/FILE_ID [000000]mydir.dir
        (the file id should be the same)
   ```

9. Create a subdirectory with a version limit of 3. Can you use the DIRECTORY command to find the version limit established? Use the command that will give you this information.

   $ CREATE/DIRECTORY/VERSION_LIMIT=3 [mydir.sub]

   The DIRECTORY command only shows you the version limit on a particular file. You must use $DUMP/HEADER/BLOCKS=COUNT=0 on the sub.dir file and look at Directory Version Limit in the RMS attributes area.

10. Create a file in the subdirectory you created above. Use the command $SET FILE/ENTER to create another directory entry in your top-level directory for that file. Verify that both file records point to the same file header.

```
$ CREATE [mydir.sub]A.DAT
...text...
 <CTRL/Z>
 $ SET FILE/ENTER=[mydir]B.DAT [mydir.sub]A.DAT
 $ DIRECTORY/FILE_ID [mydir]B.DAT
 $ DIRECTORY/FILE_ID [mydir.sub]A.DAT
     (the file ids should be the same)
```

11. Delete the file in your subdirectory. Do a $ DIR/FULL on the file name in your top-level directory and explain the results. Get rid of the directory record.

```
 $ DELETE [mydir.sub]A.DAT;1
 $ DIRECTORY/FULL [mydir]B.DAT

    Indicates that there is no such file as B.DAT
    (the file no longer exists, but the directory
    entry for B.DAT still exists)

To delete the directory record:

 $ SET FILE/REMOVE [mydir]B.DAT;1
```

12. Create several files in your subdirectory. Rename the subdirectory to have a version number of 2 and try to get a listing of the files. Can you fix it?

```
$RENAME [mydir]sub.dir;1 [mydir]sub.dir;2
$DIRECTORY [.sub]

Indicates that the directory is not found.
You can recover by renaming SUB.DIR;2 back to SUB.DIR;1.
```

13. Look at a full directory listing of the actual directory file (xxx.dir). Now, issue the command $SET FILE/NODIRECTORY xxx.DIR and see what changes.

```
$DIRECTORY/FULL sub.dir
(note the file attributes section)

$SET FILE/NODIRECTORY
(the file attributes no longer show it as a directory file)
```

14. The .DIR file (from the previous exercise) is no longer a directory file because you have removed its directory attributes. Can you restore the directory attributes of the .DIR file to cause it to become a directory file again?

    The answer is **no**, you cannot restore the directory attributes of a file after you use the SET FILE/NODIRECTORY command. Therefore, you must delete the .DIR file, causing any files cataloged in that directory to become **lost**.

    After the instructor uses the $ANALYZE/DISK/REPAIR command to locate the lost files, check for the files in the appropriate location and examine their characteristics using the $DIRECTORY/FULL command. Where does it get all that information?

```
$SET PROTECTION=O:RWED [mydir]sub.dir
$DELETE [mydir]sub.dir;1
.
.
(wait until the instructor uses $ANALYZE/DISK/REPAIR)
.
.
.
$DIRECTORY/FULL SYS$SYSDEVICE:[SYSLOST]
```

The information comes from the file header, which was not deleted.

INTRODUCTION TO VAX RMS

Written Exercises

The laboratory exercises for the Introduction to VAX RMS module are divided into two sections, written exercises and on-line exercises. The written exercises allow you to review your knowledge of the topics covered in the corresponding lecture, and the on-line exercises allow you to practice using the VMS utilities and commands related to RMS files.

1. Which file organization has the most overhead (disk and memory) associated with it, and what would be your reason for using it?

2. Can you add records to the middle of a sequential file on disk?

3. What are RMS buffers used for, and why?

4. Can you change the size and number of buffers for a particular process/image?

5. Can you use the RMS utilities to create an empty file? Why would you want to do this?

6. If you were to run an application that opened 40 files at one time and set up a multibuffer count of 10, what value would you assign to the ENQLM quota?

On-Line Exercises

1. What kind of file organization is your MAIL.MAI file? If you do not have a MAIL.MAI file, send yourself a mail message and then examine MAIL.MAI. Check all the characteristics and see if they make sense.

2. Mail the following text file to yourself:

   ```
   INTRO_RMS:QUEUES.TXT
   ```

   It is a rather large file. Get out of MAIL and check to see how many .MAI files you have.

3. Look at the file characteristics for the two mail files. Are they the same file organization? Why do you think they are set up as they are?

4. Display the contents of the MAIL.MAI file to your terminal. What are the results? Try the same with the new mail file that was created. Which one makes more sense?

5. Get into the Mail utility and use the EXTRACT command for the message you just mailed yourself. Look at the contents and compare the extracted output with the output produced by typing the mail file. Which is the preferred method for accessing the data?

6. Copy the following files to your directory, perform an $ANALYZE/RMS/CHECK, and note the results:

   ```
   INTRO_RMS:CORRUPT1.DAT
   INTRO_RMS:CORRUPT2.DAT
   INTRO_RMS:CORRUPT3.DAT
   ```

   Now, use the following command on each of the files to try and reconstruct the files. Does it work? What would your options be if it does not work?

   ```
   $CONVERT CORRUPTn.DAT SYS$LOGIN:yourfile.dat
   ```

Solutions to Written Exercises

1. Which file organization has the most overhead (disk and memory) associated with it, and what would be your reason for using it?

   The indexed file organization uses extra disk space to store key value and pointer pairs, and uses extra CPU time to access the data records. However, it is the best choice when records need to be accessed by one or more key values other than a simple number such as the relative file organization uses.

2. Can you add records to the middle of a sequential file on disk?

   You can add records only to the end of a sequential file, even if it resides on disk. That is why many **update programs** are designed to read in all the records of a file, make changes, and create a new version of the file. For files that do not often change (and are accessed sequentially most of the time), the sequential file organization is still a good choice.

3. What are RMS buffers used for, and why?

   RMS buffers are the actual source/destination for the RMS records that are read by a program. This means that if you want a 50-byte record, a buffer is read in and the 50 bytes are pulled out. There may be less overhead in accessing the file, especially if future reads/writes resided in the same buffer.

4. Can you change the size and number of buffers for a particular process/image?

   You can change the size and number of buffers by using the command
   $SET RMS_DEFAULT/BLOCK=n/BUFFER=n.

5. Can you use the RMS utilities to create an empty file? Why would you want to do this?

   Using the RMS utilities, a file can be designed and created without any records in it. This is advantageous when there is a program that will be writing many records into a file because the optimal characteristics and size will already be defined. This could prevent constant extents from being added to the file, which can lead to fragmentation of the file.

6. If you were to run an application that opened 40 files at one time and set up a multibuffer count of 10, what value would you assign to the ENQLM quota?

   See the calculation in the lecture notes to get the number 51.

Solutions to On-Line Exercises

1. What kind of file organization is your MAIL.MAI file? If you do not have a MAIL.MAI file, send yourself a mail message and then examine MAIL.MAI. Check all the characteristics and see if they make sense.

   ```
   $DIRECTORY/FULL MAIL.MAI
   ```

   The MAIL.MAI file is an indexed file.

2. Mail the following text file to yourself:

   ```
   INTRO_RMS:QUEUES.TXT
   ```

   It is a rather large file. Get out of mail and look to see how many .MAI files you have.

   ```
   MAIL> SEND INTRO_RMS:QUEUES.TXT
   To: yourname
   Subject: What kind of files do I have?
   $DIRECTORY *.MAI
   ```

   There should be another file created with a long file name and extension of .MAI. The Mail utility will keep only files inside the MAIL.MAI file that are within certain size limitations.

3. Look at the file characteristics for the two mail files. Are they the same file organization? Why do you think they are set up as they are?

   ```
   $DIRECTORY/FULL *.MAI
   ```

   The original MAIL.MAI file is indexed, but the new file is sequential. The Mail utility keeps track of the new location of the new outside mail file, and will access it using the key/pointer pairs. This means that we need only sequential text for output.

4. Display the contents of the MAIL.MAI file on your terminal. What are the results? Try the same with the new mail file that was created. Which one makes more sense?

   ```
   $TYPE MAIL.MAI
   $TYPE MAILxxxxxxxxxxxxxx.MAI
   ```

   The MAIL.MAI file, being indexed, will not display correctly. The new mail file looks better, but still lacks the message heading that you see in the next exercise using the EXTRACT command within Mail.

5. Get into the MAIL utility and use the EXTRACT command for the message you just mailed yourself. Look at the contents and compare the extracted output with the output produced by typing the mail file. Which would be the preferred method for accessing the data?

   ```
       MAIL>EXTRACT FILE.TXT

       $TYPE FILE.TXT
       $TYPE MAILxxxxxxxxxxxxx.MAI
   ```

   The EXTRACT command gives the neatest and fullest information.

6. Copy the following files to your directory, perform an $ANALYZE/RMS/CHECK, and note the results:

   ```
       INTRO_RMS:CORRUPT1.DAT
       INTRO_RMS:CORRUPT2.DAT
       INTRO_RMS:CORRUPT3.DAT
   ```

   Now, use the following command on each of the files to try and reconstruct the files. Does it work? What are your options if it does not work?

   ```
       $CONVERT CORRUPTn.DAT SYS$LOGIN:yourfile.dat
   ```

   The CONVERT command should work on the first two files, but fail on the third. If the data is still in order, the index can be restored. However, there are still some errors that cannot be recovered. Other options to save the data would be to try and convert it into a sequential file, or write a program to pull out the blocks one at a time.

SOFTWARE TROUBLESHOOTING

Independent Exercises

The laboratory exercises for this module are divided into two sets. The exercises in the first set are **independent** exercises; they are to be performed by students working independently at their terminals.

The exercises in the second set are **group** exercises. The class performs these exercises as a group under the direction of the instructor. For these exercises, the instructor creates various software problems on the system. The group then troubleshoots the software problems.

1. Enter the SHOW SYSTEM command. You should see a process named PROCESS_1. Answer the following questions about PROCESS_1:

   a. What is the name of the program executing in PROCESS_1?

   b. What is PROCESS_1 doing?

   c. Who is the owner of PROCESS_1?

2. Enter the DIRECTORY command. You should have a subdirectory file named SUB1.DIR. Can you determine the names of the files in this subdirectory? (Hint: The DIRECTORY command should cause an error message to appear on your screen. Do you know what this message indicates?)

3. You should find a crash dump in:

   ```
   SW_TROUB:SYSDUMP.DMP
   ```

   Analyze this crash dump, and answer the following questions:

   a. What was the name of the image that was executing at the time of the crash?

   b. Who owned the process that was executing at the time of the crash?

4. Analyze the running system. Which of your fellow students are using the system dump analyzer? Which processes are awaiting terminal I/O?

Solutions

Independent Exercise 1

Comments on the solution to this exercise follow the solution.

```
1. $ SHOW SYSTEM
   VAX/VMS V5.0-2 on node CLAIR 5-JAN-1989 15:28:07.15 Uptime 0 19:50:11
    Pid    Process Name  State  Pri   I/O     CPU       Page flts Ph.Mem
   00000041 SWAPPER        HIB   16      0  0 00:00:03.97      0      0
   00000184 SHERRY         CUR    5     63  0 00:00:02.02    455    362
   00000045 OPCOM          HIB    8    125  0 00:00:01.58    209     88
   00000046 JOB_CONTROL    HIB    8  10081  0 00:01:34.00    233    466
   00000049 SYMBIONT_0001  HIB    4     21  0 00:00:00.97    268    108
   0000004B NETACP         HIB   10     92  0 00:00:15.82    193    377
   0000004C EVL            HIB    6     60  0 00:00:07.15  18723     39   N
   0000004D REMACP         HIB    9     17  0 00:00:00.23     79     50
   0000004E RDMS_MONITOR   LEF   15     18  0 00:00:01.33    830     44
   0000004F RPC$SWL        HIB    9     21  0 00:00:00.44    121    191
   00000053 SYMBIONT_0003  HIB    4     21  0 00:00:01.09    238    242
   00000054 TM_SERVER      HIB    6     27  0 00:00:01.52    289    340
   00000055 OA$FCV         SUSP   4      5  0 00:00:00.19     64     88
   00000171 PROCESS_1      COM    4      5  0 00:39:08.14     59     83

2. $ SHOW PROCESS/ID=171
   5-JAN-1989 15:28:18.18                      User: SHERRY
   Pid: 00000171   Proc. name: PROCESS_1        UIC: [INSTRUCT,SHERRY]
   Priority:   4   Default file spec: Not available

3. $ SHOW PROCESS/CONTINUOUS/ID=171

   Process PROCESS_1             15:29:21

   State              COM                Working set               83

   Cur/base priority  4/4                Virtual pages            862

   Current PC         00000205           CPU time     000:00:40:20.96

   Current PSL        03C00000           Direct I/O                 2

   Current user SP    7FF97380           Buffered I/O               3

   PID                00000171           Page faults               59

   UIC                [INSTRUCT,SHERRY]  Event flags         E0000000
                                                            00000000

 $1$DJA0:[INSTRUCTORS.][SHERRY]PROG_1.EXE;1
```

Comments on Solution to Independent Exercise 1

1. Enter the SHOW SYSTEM command, which lists all the processes currently running. It also lists the process identification number (PID) of each process.

2. Using the SHOW PROCESS command, you can identify the owner of PROCESS_1. The owner's username is SHERRY.

3. The SHOW PROCESS/CONTINUOUS command shows the complete file specification of the image currently executing, PROG_1.EXE.

   The SHOW PROCESS/CONTINUOUS command also indicates that PROCESS_1 is probably executing an endless loop. The CPU time continues to increase. When the current PC keeps changing, it changes in a definite pattern: it continuously changes from one address to another.

Solution

Independent Exercise 2

Comments on the solution to this exercise follow the solution.

```
1. $ DIR [CLARK.SUB1]
   %DIRECT-E-OPENIN, error opening DEMON$DUA1:[CLARK.SUB1]*.*;* as input
   -RMS-E-FND, ACP file or directory lookup failed
   -SYSTEM-W-BADIRECTORY, bad directory file format

2. $ DUMP [CLARK]SUB1.DIR

   Dump of file DEMON$DUA1:[CLARK]SUB1.DIR;1 on  7-AUG-1986 12:47:26.46
   File ID (10289,4,0)   End of file block 1 / Allocated 3

   Virtual block number 1 (00000001), 512 (0200) bytes

   442E4141 414B5241 4C430C00 7FFF0018 ......CLARKAAA.D 000000
   0E007FFF 001A0000 00022847 00015441 AT..G(.......... 000010
   00015441 442E4C49 46594D4B 52414C43 CLARKMYFIL.DAT.. 000020
   52414C43 0C007FFF 00180000 00022849 I(..........CLAR 000030
   00000002 28480001 5441442E 434D4E4B KNMC.DAT..H(.... 000040
   00000000 00000000 00000000 0000FFFF ................ 000050
                     .
                     .

   00000000 00000000 00000000 00000000 ................ 0001F0
```

Comments on Solution to Independent Exercise 2

1. The DIRECTORY command produces an error message indicating a corrupt directory file.

2. The output from the DUMP command indicates that there were three files in your subdirectory. The file names begin with your username and have the file name extension .DAT. To retrieve these lost files, you would:

   a. Invoke the Verify utility, using the command ANALYZE/DISK_STRUCTURE.

   b. Use the DIRECTORY/OWNER command to identify your files in the [SYSLOST] directory. (Your files will be owned by your UIC.)

   c. Copy the files from [SYSLOST] into another directory.

   NOTE

   Do not use the ANALYZE/DISK_STRUCTURE utility on a disk with concurrent activity. This would produce misleading messages.

Solution

Independent Exercise 3

Comments on the solution to this exercise follow the solution.

```
1. $ ANALYZE/CRASH_DUMP SW_TROUB:SYSDUMP.DMP
   VAX/VMS System dump analyzer
   Dump taken on 24-AUG-1988 17:11:04.69
   SSRVEXCEPT, Unexpected system service exception

2. SDA> SHOW CRASH
   System crash information
   ------------------------
   Time of system crash: 24-AUG-1988 17:11:04.69
   Version of system: VAX/VMS VERSION V5.0
   System Version Major ID/Minor ID: 1/0
   System type: MicroVAX II
   Crash CPU ID/Primary CPU ID:  00/00
   Bitmask of CPUs active/available:  00000001/00000001
   CPU bugcheck codes:
   CPU 00 -- SSRVEXCEPT, Unexpected system service exception

   CPU 00 Processor crash information
   ----------------------------------
   CPU 00 reason for Bugcheck:
    SSRVEXCEPT, Unexpected system service exception
   Process currently executing on this CPU: SHERRY
   Current image file: $1$DJA0:[INSTRUCTORS.][SHERRY]CRASH1M.EXE;1
   Current IPL: 0  (decimal)
   CPU database address:  8053E000

   General registers:
      R0  = 00000000   R1  = 80002398   R2  = 00000004   R3  = 7FF93169
      R4  = 80273D80   R5  = 7FFE5EB4   R6  = 7FFED18A   R7  = 7FFED18A
      R8  = 7FFECA52   R9  = 7FFECC5A   R10 = 7FFED7D4   R11 = 7FFE2BDC
      AP  = 7FFE7794   FP  = 7FFE777C   SP  = 7FFE777C   PC  = 8000239E
      PSL = 00000000

   Processor registers:
      POBR    = 80551C00     SBR     = 00272800     ASTLVL = 00000001
      POLR    = 00000003     SLR     = 00003600     SISR   = 00000000
      P1BR    = 7FD5EA00     PCBB    = 0013E820     ICCS   = 00000040
      P1LR    = 001FFA13     SCBB    = 0026FC00     SID    = 08000000
      TODR    = 89E7B231     SYSTYPE= 01010000
      ISP     = 8053F200
      KSP     = 7FFE777C
      ESP     = 7FFE9800
      SSP     = 7FFECA4E
      USP     = 7FF44D5C
```

```
3. SDA> SHOW SUMMARY
   Current process summary
   -----------------------
   Extended Indx Process name  Username State Pri   PCB      PHD    Wkset
  -- PID -- ---- ------------- -------- ----- --- -------- -------- -----
   00000021 0001 SWAPPER                HIB    16 801F2598 801F2400     0
   00000062 0002 SHERRY        SHERRY   CUR     6 80273D80 8054FE00   254
   00000024 0004 ERRFMT        SYSTEM   HIB     8 802798A0 80541200    96
   00000025 0005 OPCOM         SYSTEM   HIB     8 80272EC0 8055EA00   121
   00000026 0006 JOB_CONTROL   SYSTEM   HIB     8 8027A230 8056D600   429
   00000027 0007 SYMBIONT_0001 SYSTEM   HIB     6 8028B250 8057C200    65
   00000033 0013 NETACP        DECNET   HIB    10 8028D0A0 8058AE00   369
   00000034 0014 EVL           DECNET   HIB     6 8028DC30 80599A00    41
   00000035 0015 REMACP        SMITTY   HIB     9 802903F0 805A8600    49

  SDA> EXIT
```

Comments on Solution to Independent Exercise 3

1. Use the ANALYZE/CRASH_DUMP command to invoke SDA to analyze a crash dump.

2. The SHOW CRASH command shows:

   a. The name of the process that was executing at the time of the crash, SHERRY.

   b. The complete file specification of the image that was running, CRASH1M.EXE.

3. The SHOW SUMMARY command indicates the owner of the SHERRY process. The owner's username is SHERRY.

Solution

Independent Exercise 4

Comments on the solution to this exercise follow the solution.

```
1. $ ANALYZE/SYSTEM

   VAX/VMS System analyzer

2. SDA> SHOW SUMMARY

   Current process summary
   -----------------------
   Extended Indx Process name    Username State Pri   PCB      PHD     Wkset
  -- PID -- ---- --------------- -------- ----- --- -------- -------- -----
   20A00080 0000 NULL                      COM    0 800024A8 80002328     0
   20A00081 0001 SWAPPER                   HIB   16 80002748 800025C8     0
   20A00E82 0002 _RTA1:          CLARK     CUR    9 801CB3A0 80769000   500
   20A00E03 0003 ALBERT          ALBERT    LEF    4 801C7240 806DA800   300
   20A00085 0005 ERRFMT          SYSTEM    HIB    8 80163F70 8039B400    84
   20A00086 0006 CACHE_SERVER    SYSTEM    HIB   16 80164090 803B3000    72
   20A00087 0007 CLUSTER_SERVER  SYSTEM    HIB   10 801703E0 803E2800   240
   20A00088 0008 OPCOM           SYSTEM    LEF    8 80170500 803FA400   134
   20A00089 0009 JOB_CONTROL     SYSTEM    HIB    8 80170620 80412000   311
   20A0008A 000A CONFIGURE       SYSTEM    HIB    8 8017DCA0 80429C00   137
   20A0008E 000E NETACP          DECNET    HIB   10 801985C0 80488C00  1500
   20A0008F 000F EVL             DECNET    HIB    6 80198B50 804A0800   175
   20A00090 0010 VAXsim_Monitor  SYSTEM    HIB    8 801993F0 804B8400   200
   20A00091 0011 REMACP          SYSTEM    HIB    9 801A4AE0 804D0000    42
   20A00D96 0016 EDDINGS         EDDINGS   LEF    5 801A2800 804E7C00   465
   20A00C29 0029 MERCURI         MERCURI   LEF    6 801A50E0 805D5400   461
   20A00CAC 002C PIERCE          PIERCE    LEF    4 801A4C00 805A5C00   150
   20A00DBD 003D KENAH           KENAH     LEF    7 801A8270 80634400   500
   20A00F59 0059 PROCESS_1       CLARK     COM    4 801C7F00 807B0400    67
```

Comments on Solution to Independent Exercise 4

1. Use the ANALYZE/SYSTEM command to invoke SDA to analyze a running system.

2. Use the SDA> SHOW SUMMARY command to list the processes currently executing. This command also shows an index for each process, which SDA extracts from the PID.

```
3. SDA> SET PROCESS/INDEX=003D
4. SDA> SHOW PROCESS/CHANNEL
   Process index: 003D   Name: KENAH   Extended PID: 20A00DBD
   ------------------------------------------------------------
                    Process active channels
                    -----------------------
  Channel  Window   Status  Device/file accessed
  -------  ------   ------  --------------------
  0010   00000000          DEMON$DUA1:
  0020   8034FA00          DRA0:[SYS0.SYSEXE]NOTES$MAIN.EXE;3
  0030   8034BD40          DRA0:[SYS0.SYSLIB]NOTES$SHARE.EXE;3 (section file)
  0040   00000000          LTA264:
  0050   00000000          LTA264:
  0060   803439A0          DRA0:[SYS0.SYSLIB]LIBRTL.EXE;3 (section file)
  0070   8033D520          DRA0:[SYS0.SYSLIB]NPUSHR.EXE;3 (section file)
  0080   8034E4A0          DRA0:[SYS0.SYSLIB]NPU$CCTSHR.EXE;3 (section file)
  0090   80357CE0          DRA0:[SYS0.SYSMGR]APRILFOOL.EXE;1
  00A0   00000000 Busy     MBA6708:
  00B0   00000000 Busy     LTA264:
  00C0   80341780          DRA0:[SYS0.SYSLIB]NOTES$SECTION.TPU$SECTION;1
  00D0   8033F500          DEMON$DUA1:(5756,1,0)
  00E0   00000000          MBA6720:
  00F0   801CA880          NET8570:

  SDA> SET PROCESS/INDEX=29
  SDA> SHOW PROCESS/CHANNEL
  Process index: 0029   Name: MERCURI   Extended PID: 20A00C29
  ------------------------------------------------------------
                  Process active channels
                  -----------------------
  Channel  Window   Status   Device/file accessed
  -------  ------   ------   --------------------
  0010   00000000           DEMON$DUA1:
  0020   8034FEE0           DRA0:[SYS0.SYSEXE]MAIL.EXE;1 (section file)
  0030   803439A0           DRA0:[SYS0.SYSLIB]LIBRTL.EXE;3 (section file)
  0040   00000000           LTA254:
  0050   00000000           LTA254:
  0060   80344F60           DRA0:[SYS0.SYSLIB]SMGSHR.EXE;1 (section file)
  0070   8034A060           DEMON$DUA4:(2842,1,0)
  0080   80353C60           DEMON$DUA4:(2697,1,0)
  0090   00000000           LTA254:
  00A0   00000000           LTA254:
  00B0   00000000  Busy     LTA254:
  00C0   80346B80           DRA0:[SYS0.SYSMSG]CLIUTLMSG.EXE;1 (section file)
  00D0   803491C0           DRA0:[SYS0.SYSMSG]SHRIMGMSG.EXE;1 (section file)
  00E0   00000000           LTA254:
  00F0   8034ACC0           DEMON$DUA1:(7280,1,0)

SDA> EXIT
```

3. You can use the SDA> SET PROCESS/INDEX command to set the current process to any process you want to investigate.

4. You can use the SDA> SHOW PROCESS/CHANNEL command to show:

   a. Whether the process is awaiting terminal I/O

      1) Devices (in the column labeled **Device/file accessed**) that begin with **LTA** or **TT** indicate terminals.
      2) Devices that begin with **MBA** indicate mailboxes.

   b. What program(s) the process is executing

   c. What file(s) are being accessed

   If a process is in LEF state, the SHOW PROCESS/CHANNEL command can indicate the resource for which the process is waiting (for example, terminal I/O or mailbox I/O).

   If SHOW PROCESS/CHANNEL indicates that multiple processes are BUSY accessing the same device or file, it might indicate that too many users are trying to use that device or file.

Group Exercises

To prepare the system for group exercises, the instructor will produce problems that are detectable but not bad enough to cause a system hang.

The group exercises require cooperation between the instructor and the students. These exercises must be done one at a time. Follow this general procedure for each of these exercises.

- The instructor must prepare the system by creating a software problem.
- After the instructor prepares the system, the students will troubleshoot the problem. Use the VMS utilities and commands covered in this module to troubleshoot the problems.

  Use pencil and paper to write down your observations and proposed solutions to the problems. However, **do not** alter any parameters, files, or other system resources.
- After you have had time to troubleshoot the problem, the instructor may open the group to discussion. After the group discusses the symptoms, possible problems, and proposed solutions, make a decision about how to solve the problem.
- When a group decision has been made, one person (either the instructor or a student) can enter the recommended commands and report the results to the group. These exercises may require a reboot, so it is easier if the system console is nearby.
- After the proposed solution has been implemented, the students can use VMS utilities and commands to verify that the solution was effective. If the recommended solution did not actually solve the problem, repeat the troubleshooting cycle.

VAXsimPLUS

The VAXsimPLUS lab exercise is to be taken from the tutorial, "Getting Started with VAXsimPLUS."

CONFIGURATION PLANNING

Using the VAX Systems and Options Catalog to Expand a VAX-11/780

Existing Configuration

- VAX 6310 CPU
- 32 Mbytes of 1M-chip MOS memory
- DEBNA 802.3 Ethernet communications interface
- LA100 console terminal
- 1 TU81-Plus tape drive
- 1 TK70 tape drive and interface
- 1 SA600 storage array (2 RA90 drives)
- 1 KDB50 disk controller
- 2 VAXBI channels
  - — 10 VAXBI slots
  - — 29 I/O panel units

Table 9–2: Expandable Resources Currently In Use

Component	Slot	Node	Panel Unit
DEBNA/Ethernet port	1	1	1
1 KDB50 controller	2	1	2
1 TU81-Plus tape drive	1	1	1
1 TK70 tape drive	1	1	0
Total in Use:	5	4	4
Remaining:	5	-	25

Expansion Exercise

Use the *VAX Systems and Options Catalog* (SOC) to identify the hardware that you must order to expand the system as follows:

> Increase disk space by approximately 9 GBytes - media must be compatible with existing drives.

The solution to this exercise begins on the following page. Use the *VAX Systems and Options Catalog* to complete this exercise before you read the solution.

Solution to Expansion Exercise

Selecting New Hardware

1. Turn to the "Systems" chapter of the SOC.
2. Find the "VAX 6310 VMS System Building Blocks" section.
3. To select the new disk drives, find the "Mass Storage" subsection in the section entitled "Add-On Options."
4. The best choice is the SA600 storage array - compatible with existing configuration.
5. Find the description of the SA600 storage array in the SOC chapter entitled "Disks and Tapes."

   a. Prerequisites

      Either a KDA50 or a KDB50 controller

   b. Select the appropriate drives and cabinets

      SA600 JA/JD storage array containing 8 RA90 disks

   c. SA600 storage array plugs into the KDB50 controller

      See step 6 for KDB50 configuration requirements

   d. Cables come in various lengths

      Select the appropriate length - for example, order BC26V-25 if you need a cable 25 feet long

6. The KDB50 controller supports up to four RA90 drives.

   a. Two RA90s are already connected.

   b. To add the SA600 with 8 RA90s, we need 2 more KDB50s.

7. Find the description of the KDB50 controller in the SOC chapter, "Disks and Tapes."

   a. Configuring requirements for 2 KDB50s

      1) 4 VAXBI slots
      2) 2 BI node
      3) 4 I/O panel units

   b. The existing configuration meets these requirements: no expansion cabinets or boxes are needed.

Summary of New Hardware

- SA600 storage array
  - 2 SA600 HA/HD storage arrays containing 4 RA90 disk drives
  - Appropriate cables (2)
- 1 KDB50 disk controller

MODULE 10
POST-TEST

TEST

Circle the letter of the correct answer for each of the questions below.

1. How do you remove a DCL verb from the DCL tables?

   a. $ SET COMMAND/DELETE = dcl_verb

   b. $ EDIT SYS$LOGIN:DCLTABLES.EXE

   c. $ EDIT SYS$LIBRARY:DCLTABLES.EXE

   d. Use the Install utility to delete the DCL verb

2. Which of the following commands would you use to add information to an existing help file?

   a. $ HELP/LIBRARY= library_name

   b. $ EDIT help.HLB

   c. $ LIBRARY/HELP library_name text_file

   d. $ LIBRARY/HELP/CREATE help.HLB

3. 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

4. Which of the following commands would you use to make a change in your DEFBOO.CMD boot file?

   a. $ EXCHANGE DIR CSA1:

   b. $ EXCHANGE COPY CSA1::DEFBOO.CMD /LOG

   c. $ EDIT CSA1:DEFBOO.CMD

   d. $ BACKUP CSA1:DEFBOO.CMD /LOG

5. PROBLEM: You have encountered a process hang. A step likely to give you the most useful FIRST piece of information would be to:

a.	Determine the wait state of the process	SHOW SYSTEM
b.	Determine whether the process still exists	SHOW SYSTEM
c.	Check the priority of the process	SHOW SYSTEM
d.	Display the contents of a specific block of memory	SDA> SHOW PROCESS/PCB
e.	Obtain the PCB address for the process	SDA> SHOW PROCESS/PCB

6. PROBLEM: You are experiencing a system hang characterized by many processes awaiting unknown resources. One action you might have to take is to:

   a. Call Software Service

   b. Call Field Service

   c. Shut down and reboot

   d. Copy the lost file to a new directory

   e. Alter SYSGEN parameters

7. PROBLEM: A single user reports that the system is not responding. In trying to determine whether you have a system hang, the most useful FIRST step to take is to check:

   a. The console to see if the system has crashed

   b. Other users to see if the system responds to them

   c. Values of SYSGEN parameters

   d. Whether files are full

   e. The amount of free space in physical memory

8. You observe the following system message:

```
SYSTEM-W-PAGECRIT,
Pagefile space critical, system trying to continue
```

   At this point your appropriate next step is to:

   a. Install another pagefile

   b. Increase SYSGEN parameter MPW_WAITLIMIT

   c. Use SYSGEN to alter PAGEFILCNT

   d. Determine which files are becoming full

   e. Ask all users to log out

9. The appropriate command to issue following receipt of the message in the previous question is:

   a. $ SHOW MEMORY/FILES/ FULL

   b. $ SHOW SYSTEM

   c. $ SHOW MEMORY/POOL/ PHYSICAL

   d. SYSGEN> SHOW MPW_LIMIT

   e. SYSGEN> SHOW PAGFILCNT

10. PROBLEM: A file appears to be lost. You suspect a disk structure problem. The most helpful tool you can use to test this theory is the:

    a. Error logger

    b. DCL SHOW commands

    c. Monitor utility

    d. System dump analyzer

    e. Verify utility

11. Using an appropriate tool to follow up the problem, you observe a message containing the mnemonic code ENTERLOST. What does that indicate to you?

    a. Error entering file in directory [SYSLOST]

    b. Error creating directory [SYSLOST]

    c. Error opening directory

    d. Error deleting file

    e. No valid index file heading

12. For preventive maintenance purposes, you might routinely perform which of the following activities?

    a. Copy lost files into a new directory

    b. Move lost files into [SYSLOST]

    c. Check disk volumes for inconsistencies in disk structure

    d. Run SDA

13. Which of the following is not found in the OPERATOR.LOG file?

a. Security alarms

b. All reply messages

c. All OPCOM messages

d. All console messages

14. You are writing command procedures. For each of the functions below, select the Command or Qualifier that best implements that function. Each selection can be used once, more than once, or not at all.

Function

____ Putting the results of the SHOW QUOTA command in a file

____ Putting the results of a DIRECTORY command in a file

____ Displaying the content of a symbol at a terminal

____ Sending a copy of a file to the SYSTEM account

____ Displaying information that has been abstracted from a file

____ Putting the results of the ACCOUNT command into a file

____ Displaying results of the SHOW command at a terminal without unwanted information

Command/Qualifier

a. DEFINE

b. /OUTPUT= filename

c. TYPE symbol

d. MAIL

e. WRITE symbol

15. You are writing command procedures. For each of the below, select the Command or Lexical that best describes that function. Each selection can be used once, more than once, or not at all.

Function

____ Asking users if they want to see some information (Y/N)

____ Gaining access to information in a file

____ Informing VMS which file to read

____ Displaying information that has been obtained from a file

____ Displaying information in a different way than the system would display it

____ Restricting the information to be put into a symbol (for retrieval)

____ Displaying information obtained by means of a lexical and put into a symbol

____ Asking users to specify a particular device about which they want information

Command/Lexical

a. INQUIRE

b. WRITE

c. F$EXTRACT

d. OPEN/READ

16. There are a variety of ways to introduce communications between systems. For each situation below, select the most suitable solution for customer needs.

Situation

____ Occasional transfers of files between systems in different buildings

____ Need to integrate system management for systems in the same department

____ Frequent sharing of files by two or more systems in the same department

____ Need to minimize hardware costs when linking systems in the same building

Solution

a. VAXcluster

b. DECnet network

17. Identify the best solution for each of the situations below.

Situation

____ Users do not have very much technical (computer) expertise

____ Many users run a variety of compute-intensive software

____ System management expertise is not widespread

____ Large amounts of mass storage are needed for multiple users

Solution

a. Large VAX systems with many users

b. Single-user MicroVAX systems

18. Current and projected needs, and the kind of equipment currently in use, often determine the type of new equipment that will be purchased.

Select the Maximum Number that best describes each of the options below.

Option

____ Important to avoid early obsolescence

____ Essential that there be compatibility with existing equipment

____ Important to keep purchase cost as low as possible

____ There are plans for continuous updating and the use of whatever is purchased in new configurations

Maximum Number

a. Traditional equipment

b. Current equipment

19. Enter a letter from Column B that best corresponds to the description or definition in Column A. There may be items in Column A for which no term in Column B is appropriate. If so, enter f for blank.

Column A

____ Performing system-wide memory scheduling

____ Allowing a process to execute until it is preempted by a higher priority process

____ Rotating process execution based on process priority

____ Set of processes currently in memory

____ Transferring pages between physical memory and page files on disk

____ Containing current scheduling state

____ Moves entire process working sets between memory and disk

____ Contains data structures shared by multiple processes

____ A basic entity in VMS able to be scheduled

____ Contains system-wide data structures that do not have to be memory resident

Column B

a. Paged pool

b. Nonpaged pool

c. Paging

d. Scheduler

e. Swapper

f. Blank

20. Select the suggested immediate course of action for each of the following situations.

	Situation		Action
____	The computer repeatedly bug-checks with the Machine Check error	a.	ANALYZE/MEDIA
		b.	Interactive VAXsimPLUS
____	Produce a summary of all errors logged in the past 24 hours	c.	VAXsimPLUS monitor
____	Test one or more devices on the system to see if they generate hardware errors	d.	Call Field Service
		e.	Run all or part of UETP

21. For each of the following situations, select the problem that best describes the situation.

	Situation		Problem
____	Head crash	a.	Hardware problem
____	Terminal does not respond	b.	Software problem
____	Process(es) in RWAST wait state	c.	Unclear if it is hardware or software
____	Computer does not boot		

Circle the letter of the correct answer for each of the questions below.

22. Which is a component of the VMS disk I/O system?

a. Disk scheduler

b. Record management services

c. Pager

d. Disk working set

23. Which VMS disk I/O system component performs virtual I/O?

    a. Executable images

    b. Record management services

    c. QIO

    d. Both a and b

24. When is disk cluster size defined?

    a. System boot

    b. Disk initialization

    c. Programming

    d. Software installation

25. If you wanted to see disk cluster size, total blocks, total cylinders, or sectors per track, which command would you use?

    a. SHOW CLUSTER

    b. SHOW SIZE

    c. SHOW DEVICE/FULL

    d. SHOW ALL

26. What would you do to define cluster size?

    a. INITIALIZE/CLUSTERSIZE

    b. DEFINE CLUSTER SIZE

    c. SET CLUSTER SIZE

    d. SHOW CLUSTER

27. Which statement is true about ODS-2 disk file structure?

a. It is selected upon initialization

b. It is used to maintain and control data on disk volumes

c. It is created by the CREATE OSD-2 command

d. It has no effect on VMS disk volumes

28. Which statement is TRUE about ODS-2 file headers?

a. It is part of the file it describes

b. It is used to locate volumes on the disk

c. It is part of the volume INDEXF.SYS

d. There are ten different areas

29. If you wanted to display the contents of a file header, which command would you use?

a. $SHOW/HEADER filename

b. $DUMP/HEADER filename

c. $TYPE/HEADER filename

d. $PRINT/HEADER filename

30. As system manager you have done a DUMP/HEADER and found a file to be discontiguous. What would now be done to restore contiguity?

a. BACKUP/IMAGE

b. DUMP/HEADER/REORGANIZE

c. REORGANIZE/filename

d. RESTORE/filename

31. Which command is used to dismount a volume during mount verification?

   a. DISMOUNT/ENDPROCESS

   b. DISMOUNT/ABORT

   c. DISMOUNT/volumename

   d. DISMOUNT/VOLUME

32. As system manager you should check the validity of disk volumes. Which utility allows you to do this?

   a. ANALYZE/DISK_STRUCTURE

   b. SHOW DEVICE/VERIFY

   c. VERIFY/VALIDITY

   d. MOUNT/VERIFY

33. You have found a corrupt directory and want to recover the files in that directory. What should you do first?

   a. SET FILE/NODIRECTORY <directory.dir>

   b. DELETE <directory.dir>

   c. ANALYZE/DISK_STRUCTURE/REPAIR

   d. COPY FILES.*

34. Why would you want to rebuild a volume?

   a. The volume has failed because of a programming error

   b. The disk volume has been improperly dismounted

   c. None of the caching limits were in effect

   d. The volume is not write-locked

35. Which command would you use to enable volume rebuild?

a. SET/REBUILD

b. SET/VOLUME_REBUILD

c. MOUNT/REBUILD

d. MOUNT/VOLUME_REBUILD

36. As 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

37. What command is used to invoke disk quota rebuild?

a. DISK>REBUILD

b. DISKQ>REBUILD_VOLUME

c. DISKQ>REBUILD

d. DISK>REBUILD_VOLUME

ANSWERS

Circle the letter of the correct answer for each of the questions below.

1. How do you remove a DCL verb from the DCL tables?

   a. **$ SET COMMAND/DELETE = dcl_verb**

   b. $ EDIT SYS$LOGIN:DCLTABLES.EXE

   c. $ EDIT SYS$LIBRARY:DCLTABLES.EXE

   d. Use the Install utility to delete the DCL verb

2. Which of the following commands would you use to add information to an existing help file?

   a. $ HELP/LIBRARY= library_name

   b. $ EDIT help.HLB

   c. **$ LIBRARY/HELP library_name text_file**

   d. $ LIBRARY/HELP/CREATE help.HLB

3. 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

4. Which of the following commands would you use to make a change in your DEFBOO.CMD boot file?

   a. $ EXCHANGE DIR CSA1:

   b. $ EXCHANGE COPY CSA1::DEFBOO.CMD /LOG

   c. $ EDIT CSA1:DEFBOO.CMD

   d. $ BACKUP CSA1:DEFBOO.CMD /LOG

5. PROBLEM: You have encountered a process hang. A step likely to give you the most useful FIRST piece of information would be to:

a.	Determine the wait state of the process	SHOW SYSTEM
b.	**Determine whether the process still exists**	**SHOW SYSTEM**
c.	Check the priority of the process	SHOW SYSTEM
d.	Display the contents of a specific block of memory	SDA> SHOW PROCESS/PCB
e.	Obtain the PCB address for the process	SDA> SHOW PROCESS/PCB

6. PROBLEM: You are experiencing a system hang characterized by many processes awaiting unknown resources. One action you might have to take is to:

   a. Call Software Service

   b. Call Field Service

   c. Shut down and reboot

   d. Copy the lost file to a new directory

   e. Alter SYSGEN parameters

7. PROBLEM: A single user reports that the system is not responding. In trying to determine whether you have a system hang, the most useful FIRST step to take is to check:

   a. The console to see if the system has crashed

   b. **Other users to see if the system responds to them**

   c. Values of SYSGEN parameters

   d. Whether files are full

   e. The amount of free space in physical memory

8. You observe the following system message:

   ```
   SYSTEM-W-PAGECRIT,
   Pagefile space critical, system trying to continue
   ```

   At this point your appropriate next step is to:

   a. Install another pagefile

   b. Increase SYSGEN parameter MPW_WAITLIMIT

   c. Use SYSGEN to alter PAGEFILCNT

   d. **Determine which files are becoming full**

   e. Ask all users to log out

9. The appropriate command to issue following receipt of the message in the previous question is:

   a. **$ SHOW MEMORY/FILES/ FULL**

   b. $ SHOW SYSTEM

   c. $ SHOW MEMORY/POOL/ PHYSICAL

   d. SYSGEN> SHOW MPW_LIMIT

   e. SYSGEN> SHOW PAGFILCNT

10. PROBLEM: A file appears to be lost. You suspect a disk structure problem. The most helpful tool you can use to test this theory is the:

   a. Error logger

   b. DCL SHOW commands

   c. Monitor utility

   d. System dump analyzer

   e. **Verify utility**

11. Using an appropriate tool to follow up the problem, you observe a message containing the mnemonic code ENTERLOST. What does that indicate to you?

   a. **Error entering file in directory [SYSLOST]**

   b. Error creating directory [SYSLOST]

   c. Error opening directory

   d. Error deleting file

   e. No valid index file heading

12. For preventive maintenance purposes, you might routinely perform which of the following activities?

   a. Copy lost files into a new directory

   b. Move lost files into [SYSLOST]

   c. **Check disk volumes for inconsistencies in disk structure**

   d. Run SDA

13. Which of the following is **not** found in the OPERATOR.LOG file?

- **a.** Security alarms
- **b. All reply messages**
- **c.** All OPCOM messages
- **d.** All console messages

14. You are writing command procedures. For each of the functions below, select the Command or Qualifier that best implements that function. Each selection can be used once, more than once, or not at all.

Function

a Putting the results of the SHOW QUOTA command in a file

b Putting the results of a DIRECTORY command in a file

c Displaying the content of a symbol at a terminal

d Sending a copy of a file to the SYSTEM account

e Displaying information that has been abstracted from a file

b Putting the results of the ACCOUNT command into a file

e Displaying results of the SHOW command at a terminal without unwanted information

Command/Qualifier

a. DEFINE

b. /OUTPUT= filename

c. TYPE symbol

d. MAIL

e. WRITE symbol

15. You are writing command procedures. For each of the functions below, select the Command or Lexical that best describes that function. Each selection can be used once, more than once, or not at all.

Function

_a__ Asking users if they want to see some information (Y/N)

_d__ Gaining access to information in a file

_d__ Informing VMS which file to read

_b__ Displaying information that has been obtained from a file

_b__ Displaying information in a different way than the system would display it

_c__ Restricting the information to be put into a symbol (for retrieval)

_b__ Displaying information obtained by means of a lexical and put into a symbol

_a__ Asking users to specify a particular device about which they want information

Command/Lexical

a. INQUIRE

b. WRITE

c. F$EXTRACT

d. OPEN/READ

16. There are a variety of ways to introduce communications between systems. For each situation below, select the most suitable solution for customer needs.

Situation

b__ Occasional transfers of files between systems in different buildings

a__ Need to integrate system management for systems in the same department

a__ Frequent sharing of files by two or more systems in the same department

b__ Need to minimize hardware costs when linking systems in the same building

Solution

a. VAXcluster

b. DECnet network

17. Identify the best solution for each of the situations below.

	Situation		Solution
a	Users do not have very much technical (computer) expertise	a.	Large VAX systems with many users
b	Many users run a variety of compute-intensive software	b.	Single-user MicroVAX systems
a	System management expertise is not widespread		
a	Large amounts of mass storage are needed for multiple users		

18. Current and projected needs, and the kind of equipment currently in use, often determine the type of new equipment that will be purchased.

Select the Maximum Number that best describes each of the options below.

	Option		Maximum Number
b	Important to avoid early obsolescence	a.	Traditional equipment
a	Essential that there be compatibility with existing equipment	b.	Current equipment
a	Important to keep purchase cost as low as possible		
b	There are plans for continuous updating and the use of whatever is purchased in new configurations		

19. Enter a letter from Column B that best corresponds to the description or definition in Column A. There may be items in Column A for which no term in Column B is appropriate. If so, enter f for blank.

Column A

e Performing system-wide memory scheduling

d Allowing a process to execute until it is preempted by a higher priority process

d Rotating process execution based on process priority

f Set of processes currently in memory

c Transferring pages between physical memory and page files on disk

f Containing current scheduling state

e Moves entire process working sets between memory and disk

b Contains data structures shared by multiple processes

f A basic entity in VMS able to be scheduled

a Contains system-wide data structures that do not have to be memory resident

Column B

a. Paged pool

b. Nonpaged pool

c. Paging

d. Scheduler

e. Swapper

f. Blank

20. Select the suggested immediate course of action for each of the following situations.

	Situation		Action
d	The computer repeatedly bugchecks with the Machine Check error	a.	ANALYZE/MEDIA
b	Produce a summary of all errors logged in the past 24 hours	b.	Interactive VAXsimPLUS
e	Test one or more devices on the system to see if they generate hardware errors	c.	VAXsimPLUS monitor
		d.	Call Field Service
		e.	Run all or part of UETP

21. For each of the following situations, select the problem that best describes the situation.

	Situation		Problem
a	Head crash	a.	Hardware problem
c	Terminal does not respond	b.	Software problem
b	Process(es) in RWAST wait state	c.	Unclear if it is hardware or software
c	Computer does not boot		

Circle the letter of the correct answer for each of the questions below.

22. Which is a component of the VMS disk I/O system?

a. Disk scheduler

b. Record management services

c. Pager

d. Disk working set

23. Which VMS disk I/O system component performs virtual I/O?

- **a.** Executable images
- **b.** Record management services
- **c.** **QIO**
- **d.** Both a and b

24. When is disk cluster size defined?

- **a.** System boot
- **b.** **Disk Initialization**
- **c.** Programming
- **d.** Software installation

25. If you wanted to see disk cluster size, total blocks, total cylinders, or sectors per track, which command would you use?

- **a.** SHOW CLUSTER
- **b.** SHOW SIZE
- **c.** **SHOW DEVICE/FULL**
- **d.** SHOW ALL

26. What would you do to define cluster size?

- **a.** **INITIALIZE/CLUSTERSIZE**
- **b.** DEFINE CLUSTER SIZE
- **c.** SET CLUSTER SIZE
- **d.** SHOW CLUSTER

27. Which statement is true about ODS-2 disk file structure?

- **a.** It is selected upon initialization
- **b. It is used to maintain and control data on disk volumes**
- **c.** It is created by the CREATE OSD-2 command
- **d.** It has no effect on VMS disk volumes

28. Which statement is TRUE about ODS-2 file headers?

- **a.** It is part of the file it describes
- **b.** It is used to locate volumes on the disk
- **c. It is part of the volume INDEXF.SYS**
- **d.** There are ten different areas

29. If you wanted to display the contents of a file header, which command would you use?

- **a.** $SHOW/HEADER filename
- **b. $DUMP/HEADER filename**
- **c.** $TYPE/HEADER filename
- **d.** $PRINT/HEADER filename

30. As system manager you have done a DUMP/HEADER and found a file to be discontiguous. What would now be done to restore contiguity?

- **a. BACKUP/IMAGE**
- **b.** DUMP/HEADER/REORGANIZE
- **c.** REORGANIZE/filename
- **d.** RESTORE/filename

31. Which command is used to dismount a volume during mount verification?

    a. DISMOUNT/ENDPROCESS

    b. **DISMOUNT/ABORT**

    c. DISMOUNT/volumename

    d. DISMOUNT/VOLUME

32. As system manager you should check the validity of disk volumes. Which utility allows you to do this?

    a. **ANALYZE/DISK_STRUCTURE**

    b. SHOW DEVICE/VERIFY

    c. VERIFY/VALIDITY

    d. MOUNT/VERIFY

33. You have found a corrupt directory and want to recover the files in that directory. What should you do first?

    a. **SET FILE/NODIRECTORY <directory.dir>**

    b. DELETE <directory.dir>

    c. ANALYZE/DISK_STRUCTURE/REPAIR

    d. COPY FILES.*

34. Why would you want to rebuild a volume?

    a. The volume has failed because of a programming error

    b. **The disk volume has been improperly dismounted**

    c. None of the caching limits were in effect

    d. The volume is not write-locked

35. Which command would you use to enable volume rebuild?

- **a.** SET/REBUILD
- **b.** SET/VOLUME_REBUILD
- **c.** **MOUNT/REBUILD**
- **d.** MOUNT/VOLUME_REBUILD

36. As 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

37. What command is used to invoke disk quota rebuild?

- **a.** DISK>REBUILD
- **b.** DISKQ>REBUILD_VOLUME
- **c.** **DISKQ>REBUILD**
- **d.** DISK>REBUILD_VOLUME