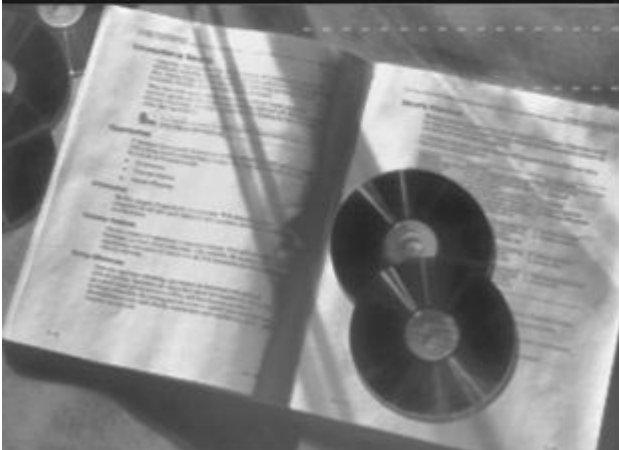


OpenVMS System and Network Node Management I

Lab Guide

Training



COMPAQ

OpenVMS System and Network Node Management I

Lab Guide

Training



COMPAQ

Notice

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

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

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

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

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

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

Aero, ALPHA, ALPHA AXP, AlphaServer, AlphaStation, Armada, BackPaq, COMPAQ, Compaq Insight Manager, CompaqCare logo, Counselor, DECterm, Deskpro, DIGITAL, DIGITAL logo, DIGITAL Alpha Systems, Digital Equipment Corporation, DIGITAL UNIX, DirectPlus, FASTART, Himalaya, InfoPaq, Integrity, LicensePaq, Ministation, NetFlex, NonStop, OpenVMS, PaqFax, Presario, ProLiant, ProLinea, ProSignia, QuickBack, QuickFind, Qvision, RDF, RemotePaq, RomPaq, ServerNet, SERVICenter, SmartQ, SmartStart, SmartStation, SolutionPaq, SpeedPaq, StorageWorks, Systempro/LT, Tandem, TechPaq, TruCluster, and Tru64 UNIX are registered in United States Patent and Trademark Office.

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

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

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

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

OpenVMS System and Network Node Management I

Lab Guide

December 1999

Module 1 — Understanding the System Management Environment

System Configuration	L1-1
Written Exercise 1-1	L1-1
Written Solution 1-1.....	L1-2
Terminology	L1-3
Written Exercise 1-2	L1-3
Written Solution 1-2.....	L1-4
Using DCL to Monitor the System.....	L1-5
Lab Exercise 1-3	L1-5
Lab Solution 1-3	L1-6

Module 2 — Developing Command Procedures

Writing Command Procedures That Perform Terminal I/O	L2-1
Lab Exercise 2-1: Displaying Text	L2-1
Lab Solution 2-1: Displaying Text	L2-2
Lab Exercise 2-2: Accepting Input	L2-3
Lab Solution 2-2: Accepting Input	L2-4
Using Symbols as Variables and Parameters	L2-5
Lab Exercise 2-3: Using a Symbol as a Variable	L2-5
Lab Solution 2-3: Using a Symbol as a Variable.....	L2-6
Controlling Flow of Execution.....	L2-7
Lab Exercise 2-4: Testing for Conditions	L2-7
Lab Solution 2-4: Testing for Conditions.....	L2-8
Lab Exercise 2-5: Transferring Control Within a Command Procedure	L2-9
Lab Solution 2-5: Transferring Control Within a Command Procedure	L2-10
Lab Exercise 2-6: Transferring Control Outside of a Command Procedure	L2-13
Lab Solution 2-6: Transferring Control Outside of a Command Procedure	L2-14
Lab Exercise 2-7: Redefining SYSS\$INPUT	L2-15
Lab Solution 2-7: Redefining SYSS\$INPUT	L2-16
Using Lexical Functions	L2-17
Lab Exercise 2-8: Lexical Functions	L2-17
Lab Solution 2-8: Lexical Functions.....	L2-18

Writing Command Procedures That Perform File I/O.....	L2-20
Lab Exercise 2-9: Writing to a File.....	L2-20
Lab Solution 2-9: Writing to a File.....	L2-22
Lab Exercise 2-10: Writing to and Reading from a File.....	L2-23
Lab Solution 2-10: Writing to and Reading from a File.....	L2-24
Lab Exercise 2-11: Using File I/O to Get System Information.....	L2-26
Lab Solution 2-11: Using File I/O to Get System Information.....	L2-28
Handling Error Conditions and Ctrl/Y.....	L2-30
Lab Exercise 2-12: Handling Errors in Command Procedures.....	L2-30
Lab Solution 2-12: Handling Errors in Command Procedures.....	L2-32
Lab Exercise 2-13: Handling Ctrl/Y Interrupts.....	L2-34
Lab Solution 2-13: Handling Ctrl/Y Interrupts.....	L2-36

Module 3 — Managing System Users

Using the Authorize Utility.....	L3-1
Lab Exercise 3-1: Displaying UAF Records.....	L3-1
Lab Solution 3-1: Displaying UAF Records.....	L3-2
Adding a User to the System.....	L3-3
Written Exercise 3-2: Adding a User.....	L3-3
Written Solution 3-2: Adding a User.....	L3-4
Lab Exercise 3-3: Using ADDUSER.COM.....	L3-5
Lab Solution 3-3: Using ADDUSER.COM.....	L3-6
Removing a User Account.....	L3-7
Written Exercise 3-4: Steps for Removing a User.....	L3-7
Written Solution 3-4: Steps for Removing a User.....	L3-8
Managing and Monitoring User Processes.....	L3-9
Lab Exercise 3-5: Controlling User Processes.....	L3-9
Lab Solution 3-5: Controlling User Processes.....	L3-10
Lab Exercise 3-6: Using the SYSMAN Utility.....	L3-11
Lab Solution 3-6: Using the SYSMAN Utility.....	L3-13
Monitoring the Use of Disk Space.....	L3-15
Lab Exercise 3-7: Monitoring Disk Space.....	L3-15
Lab Solution 3-7: Monitoring Disk Space.....	L3-16

Module 4 — Maintaining Queues

Monitoring Queues and Jobs.....	L4-1
Lab Exercise 4-1: Displaying Queue Information.....	L4-1
Lab Solution 4-1: Displaying Queue Information.....	L4-2
Lab Exercise 4-2: Monitoring Queues and Jobs.....	L4-3
Lab Solution 4-2: Monitoring Queues and Jobs.....	L4-4

Changing the Characteristics of a Queue.....	L4-5
Lab Exercise 4-3: Modifying Queue Characteristics.....	L4-5
Lab Solution 4-3: Modifying Queue Characteristics	L4-6
Managing Jobs in a Queue	L4-7
Lab Exercise 4-4: Managing Jobs.....	L4-7
Lab Solution 4-4: Managing Jobs.....	L4-8

Module 5 — Performing Backups and Restores

Introduction to the Exercises.....	L5-1
Lab Exercise 5-1: Backup to Tape.....	L5-2
Lab Solution 5-1: Backup to Tape.....	L5-4
Lab Exercise 5-2: Backup to Disk	L5-6
Lab Solution 5-2: Backup to Disk	L5-8

Module 6 — Shutting Down and Starting Up an OpenVMS System

Shutting Down an OpenVMS System	L6-1
Lab Exercise 6-1: Shutdown	L6-1
Lab Solution 6-1: Shutdown.....	L6-2
Starting Up an OpenVMS System.....	L6-3
Lab Exercise 6-2: Hardware.....	L6-3
Lab Solution 6-2: Hardware	L6-4
Lab Exercise 6-3: Console System	L6-5
Lab Solution 6-3: Console System	L6-6
Lab Exercise 6-4: Errors in Startup	L6-7
Lab Solution 6-4: Errors in Startup.....	L6-9

Understanding the System Management Environment

Module 1 Labs

System Configuration

Written Exercise 1–1

Match each description with the appropriate OpenVMS environment.
Environments can be used once, more than once, or not at all.

- | | | | |
|----|-------------------------|-------|---|
| a. | Single-processor system | | Appears to the world as a single system |
| b. | Multiprocessor | | May appear to a user as a single system |
| c. | VMScluster system | | Communicates with other nodes using DECnet for OpenVMS software |
| d. | DECnet network | | Disks on member nodes are accessible to other members |
| | | | Shares a single memory among multiple processors |
| | | | Requires point-to-point connectivity, on one interconnect, between members |
| | | | Considered to be a single system management domain |
| | | | Provides high-availability access to shared disk resources and queue facilities |
| | | | Allows booting from a remote system disk |
| | | | Requires routing knowledge |

Written Solution 1–1

In the exercise below, match each description with the appropriate OpenVMS environment. Environments can be used once, more than once, or not at all.

- | | | | |
|----|-------------------------|-------------------------|---|
| a. | Single-processor system | <u>a,b</u> | Appears to the world as a single system |
| b. | Multiprocessor | <u>a,b,</u>
<u>c</u> | May appear to a user as a single system |
| c. | VMSccluster system | <u>d</u> | Communicates with other nodes using DECnet for OpenVMS software |
| d. | DECnet network | <u>c</u> | Disks on member nodes are accessible to other members |
| | | <u>b</u> | Shares a single memory among multiple processors |
| | | <u>c</u> | Requires point-to-point connectivity, on one interconnect, between members |
| | | <u>a,b,</u>
<u>c</u> | Considered to be a single system management domain |
| | | <u>c</u> | Provides high-availability access to shared disk resources and queue facilities |
| | | <u>c</u> | Allows booting from a remote system disk |
| | | <u>d</u> | Requires routing knowledge |

Terminology

Written Exercise 1–2

Match each definition with the appropriate term. Terms can be used once, more than once, or not at all.

- | | | | |
|----|-------------------------------|-------|---|
| a. | Program | | A program included in the OpenVMS software |
| b. | Image | | Categories of users, for file protection purposes |
| c. | Object file | | Categories of access, for file protection purposes |
| d. | Utility | | A pointer to the system device |
| e. | Process | | A pointer to the node-specific portion of the system disk |
| f. | Batch process | | A pointer to that portion of the system disk shared by all nodes |
| g. | Detached process | | Type of process generated when a user logs in |
| h. | Network process | | Type of process created by another (parent) process |
| i. | Subprocess | | Type of process created on a remote node when that node's name is specified in a DCL command |
| j. | Working set | | Schedulable entity in an OpenVMS system |
| k. | Balance set | | Moving a process's working set to/from disk in response to the scheduler |
| l. | Process priority | | Moving a portion of a process's working set to/from disk in response to a process action |
| m. | Paging | | A logical sequence of instructions |
| n. | Swapping | | A sequence of machine code instructions with addressing information |
| o. | SYS\$SYSDEVICE | | A collection of machine code (binary) instructions |
| p. | SYS\$SPECIFIC | | Set of processes whose working sets are resident in physical memory |
| q. | SYS\$COMMON | | Pages of a process currently residing in physical memory (or most recently swapped out of memory) |
| r. | SYSTEM,OWNER,
GROUP,WORLD | | |
| s. | READ,WRITE,
EXECUTE,DELETE | | |

Written Solution 1–2

Match each definition with the appropriate term. Terms can be used once, more than once, or not at all.

a.	Program	<u>d</u>	A program included in the OpenVMS software
b.	Image	<u>r</u>	Categories of users, for file protection purposes
c.	Object file	<u>s</u>	Categories of access, for file protection purposes
d.	Utility	<u>o</u>	A pointer to the system device
e.	Process	<u>p</u>	A pointer to the node-specific portion of the system disk
f.	Batch process	<u>q</u>	A pointer to that portion of the system disk shared by all nodes
g.	Detached process	<u>g</u>	Type of process generated when a user logs in
h.	Network process	<u>i</u>	Type of process created by another (parent) process
i.	Subprocess	<u>h</u>	Type of process created on a remote node when that node's name is specified in a DCL command
j.	Working set	<u>e</u>	Schedulable entity in an OpenVMS system
k.	Balance set	<u>n</u>	Moving a process's working set to/from disk in response to the scheduler
l.	Process priority	<u>m</u>	Moving a portion of a process's working set to/from disk in response to a process action
m.	Paging	<u>a</u>	A logical sequence of instructions
n.	Swapping	<u>b</u>	A sequence of machine code instructions with addressing information
o.	SYS\$SYSDEVICE	<u>c</u>	A collection of machine code (binary) instructions
p.	SYS\$SPECIFIC	<u>k</u>	Set of processes whose working sets are resident in physical memory
q.	SYS\$COMMON	<u>j</u>	Pages of a process currently residing in physical memory (or most recently swapped out of memory)
r.	SYSTEM,OWNER, GROUP,WORLD		
s.	READ,WRITE, EXECUTE,DELETE		

Using DCL to Monitor the System

Lab Exercise 1–3

1. Try using DCL commands to look at some of the system characteristics and features discussed in Chapter 1. Some useful commands are shown below. Some of these commands will be discussed more in later chapters of this course. For information on any of the commands, use \$ HELP for SHOW, SPAWN, RUN PROCESS, and SUBMIT.

```
$ SHOW NETWORK$ SHOW PROCESS$ SHOW USERS
$ SHOW SYSTEM$ SHOW CLUSTER
```

2. Use the SHOW LOGICAL command to look at the definitions of the following logicals:

```
SYS$SYSDEVICESYS$SYSROOTSYS$SYSTEM
SYS$SPECIFICSYS$COMMONSYS$MANAGER
SYS$LOGIN
```

3. Use an editor to create a file called TEST.COM. It should contain some DCL commands that display information. For example, type the following:

```
$ EDIT/TPU TEST.COM

$ ! Sample command file
$ SHOW USERS
$ SHOW SYSTEM
$ SHOW PROCESS/SUBPROCESSES
$ EXIT
```

[Ctrl/Z]

4. Try executing the command file in various ways. The output should change based on how you execute it. Check the number and type of processes listed for you by SHOW USERS, the subprocesses listed by SHOW PROCESS, and the number of batch processes and subprocesses shown by SHOW SYSTEM.

In your current interactive process:

```
$ @TEST
```

In a subprocess:

```
$ SPAWN @TEST
```

In a batch process:

```
$ SUBMIT/NOTIFY/NOPRINT TEST
```

When you get the notification message saying that the batch job has completed, look at its output with the command:

```
$ TYPE TEST.LOG
```

Lab Solution 1–3

No solution needed.

Lab Solution 2–1: Displaying Text

Write a command procedure that will use two different methods to display three lines of text on a user's screen.

First line

Second line

Third line

Sample solution:

```
$                                ! OUTPUT.COM
$
$! Using the WRITE command:
$ WRITE SYS$OUTPUT "First line"
$ WRITE SYS$OUTPUT "Second line"
$ WRITE SYS$OUTPUT "Third line"
$
$! Using the TYPE command:
$ TYPE SYS$INPUT
First line
Second line
Third line
$ EXIT )
```

Lab Exercise 2-2: Accepting Input

Write a command procedure that will use two different methods to prompt for the user's name, and display the name after each time it is entered.

This image shows a full page of a document template designed for handwriting practice or general note-taking. It consists of approximately 28 evenly spaced horizontal dotted lines across the entire width of the page. The background is plain white, and there are no margins, headers, footers, or other markings present.

Lab Solution 2–2: Accepting Input

Write a command procedure that will use two different methods to prompt for the user's name, and display the name after each time it is entered.

Sample solution:

```
$                ! INPUT.COM
$
$! Using INQUIRE:
$ INQUIRE NAME1 "Please enter your name"
$ WRITE SYS$OUTPUT NAME1
$
$! Using READ:
$ READ/PROMPT="Please enter your name: " SYS$COMMAND NAME2
$ WRITE SYS$OUTPUT NAME2
$ EXIT
```

Sample run:

```
$ @INPUT
Please enter your name: Krystal
KRYSTAL
Please enter your name: Krystal
Krystal
$
```



Note

Although you have a choice of two ways to display text, you cannot use TYPE SYS\$INPUT to display the contents of a symbol.

Using Symbols as Variables and Parameters

Lab Exercise 2–3: Using a Symbol as a Variable

Write a simple command procedure that does the following:

1. Creates a local symbol COUNT
2. Executes a loop in which the value of COUNT is incremented
3. Displays the value of COUNT each time the loop is executed
4. Exits after COUNT reaches a value of 5

This image shows a full page of white paper with horizontal ruling lines. The lines are evenly spaced and extend across the width of the page, providing a template for handwriting practice or general writing. There are no margins, text, or other markings on the page.

Lab Solution 2–3: Using a Symbol as a Variable

Write a simple command procedure that does the following:

1. Creates a local symbol COUNT
2. Executes a loop in which the value of COUNT is incremented
3. Displays the value of COUNT each time the loop is executed
4. Exits after COUNT reaches a value of 5

Sample solution:

```
$! COUNT.COM
$
$ COUNT = 1           ! Create local symbol with
$                     ! initial value of 1.
$ LOOP:              ! Label for loop.
$  SHOW SYMBOL COUNT  ! Display value of counter.
$  COUNT = COUNT + 1   ! Increment counter.
$  IF COUNT .LE. 5 THEN GOTO LOOP
$                     ! Do it again if not more than 5.
$ EXIT               ! Terminate execution.
```

Sample run:

```
$ @COUNT
COUNT = 1  Hex = 00000001 Octal = 00000000001
COUNT = 2  Hex = 00000002 Octal = 00000000002
COUNT = 3  Hex = 00000003 Octal = 00000000003
COUNT = 4  Hex = 00000004 Octal = 00000000004
COUNT = 5  Hex = 00000005 Octal = 00000000005
$
```

Controlling Flow of Execution

Lab Exercise 2–4: Testing for Conditions

- Eliminate the GOTO commands
- Use the IF-THEN-ELSE command

```
$! GREETING.COM
$
$
$ DATE = F$TIME( )
$ TIME = F$EXTRACT(12,2,DATE)
$ IF TIME .GE. 12 THEN GOTO AFTERNOON
$ MORNING:
$ WRITE SYS$OUTPUT "Good morning!"
$ GOTO END_GREETING
$ AFTERNOON:
$ WRITE SYS$OUTPUT "Good afternoon!"
$ END_GREETING:
$ WRITE SYS$OUTPUT DATE
$ EXIT
```

Lab Solution 2–4: Testing for Conditions

Rewrite the command procedure shown below to:

- Eliminate the GOTO commands
- Use the IF-THEN-ELSE command

Sample solution:

```
$!GREETING2.COM
$
$ DATE = F$TIME( )
$ TIME = F$EXTRACT(12,2,DATE)
$ IF TIME .GE. 12
$ THEN WRITE SYS$OUTPUT "Good afternoon!"
$ ELSE WRITE SYS$OUTPUT "Good morning!"
$ ENDIF
$ WRITE SYS$OUTPUT DATE
$ EXIT
```

Sample run:

```
$ @GREETING2
Good morning!
14-APR-1999 13:01:06.45
$
```

Lab Exercise 2–5: Transferring Control Within a Command Procedure

1. Write a short command procedure that contains a GOTO command, followed by three WRITE statements that display some message at the terminal. Do not include the label referred to by the GOTO command.
2. Execute the command procedure. What happens?
3. Edit the command procedure, inserting the missing label before the second WRITE command. Execute the command procedure and explain what happens.
4. Edit the command procedure, inserting a duplicate label before the third WRITE command. Execute the command procedure and explain what happens.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Lab Solution 2–5: Transferring Control Within a Command Procedure

1. Write a short command procedure that contains a GOTO command, followed by three WRITE statements that display some message at the terminal. Do not include the label referred to by the GOTO command.

```
$ ! GOTO_EX1.COM
$ GOTO MISSING_LABEL
$ WRITE SYS$OUTPUT "Message 1"
$ WRITE SYS$OUTPUT "Message 2"
$ WRITE SYS$OUTPUT "Message 3"
$ EXIT
```

2. Execute the command procedure. What happens?

```
$ @GOTO_EX1
%DCL-W-USGOTO, target of GOTO not found - check spelling
and presence of label
$
```

An error occurs when the command procedure attempts to transfer control to the nonexistent label.

3. Edit the command procedure, inserting the missing label before the second WRITE command. Execute the command procedure and explain what happens.

Sample solution:

```
$                                ! GOTO_EX2.COM
$ GOTO MISSING_LABEL
$ WRITE SYS$OUTPUT "Message 1"
$ MISSING_LABEL: WRITE SYS$OUTPUT "Message 2"
$ WRITE SYS$OUTPUT "Message 3"
$ EXIT
```

Sample run:

```
$ @GOTO_EX2
Message 2
Message 3
$
```

The first WRITE command does not execute because control is transferred to the command that follows it.

4. Edit the command procedure, inserting a duplicate label before the third WRITE command. Execute the command procedure and explain what happens.

Sample solution:

```
$ ! GOTO_EX3.COM
$ GOTO MISSING_LABEL
$ WRITE SYS$OUTPUT "Message 1"
$ MISSING_LABEL: WRITE SYS$OUTPUT "Message 2"
$ MISSING_LABEL: WRITE SYS$OUTPUT "Message 3"
$ EXIT
```

Sample run:

```
$ @GOTO_EX3
Message 2
Message 3
$
```

Duplicate labels do not generate any errors. Control is transferred to the first line that has the specified label.

(You might be interested in further developing this command procedure to include a conditional GOTO command later and see which label becomes the target of the control transfer.)

Lab Exercise 2–6: Transferring Control Outside of a Command Procedure

1. Write a command procedure, DATE.COM, that displays the current date/time on the screen.
2. Write a second command procedure, AT.COM, that:
 - a. Displays the string "main procedure" on the screen
 - b. Executes the first command procedure
 - c. Displays the string "main procedure" again
3. Execute AT.COM and verify that DATE.COM also executes.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Lab Solution 2–6: Transferring Control Outside of a Command Procedure

1. Write a command procedure, DATE.COM, that displays the current date/time on the screen.

```
$      ! DATE.COM
$ SHOW TIME
$ EXIT
```

2. Write a second command procedure, AT.COM, that:
 - a. Displays the string "main procedure" on the screen
 - b. Executes the first command procedure
 - c. Displays the string "main procedure" again

```
$! AT.COM
$ WRITE SYS$OUTPUT "main procedure"
$ @DATE
$ WRITE SYS$OUTPUT "main procedure"
$ EXIT
```

3. Execute AT.COM and verify that DATE.COM also executes.

```
$ @AT
main procedure
 14-APR-1999 14:32:18
main procedure
$
```

DATE.COM displays the date/time string.

Lab Exercise 2-7: Redefining SYS\$INPUT

Write a command procedure that allows you to edit a text file. The command procedure should:

1. Prompt the user for the name of the file to be edited
2. Invoke the editor to edit the file
3. Display directory information about the file after you exit from the editor

[illegible]

Lab Solution 2–7: Redefining SYS\$INPUT

Write a command procedure that allows you to edit a text file. The command procedure should:

1. Prompt the user for the name of the file to be edited
2. Invoke the editor to edit the file
3. Display directory information about the file after you exit from the editor

Sample solution:

```
$                                     ! EDIT.COM
$
$ INQUIRE FILE "Name of file to edit"
$ ASSIGN/USER SYS$COMMAND SYS$INPUT      ! Just for
next image.
$ ! Could also use:
$ ! DEFINE/USER SYS$INPUT SYS$COMMAND
$ EDIT/TPU 'FILE'
$ DIRECTORY 'FILE'
$ EXIT
```

Sample run:

```
$ @EDIT
Name of file to edit: TEST.DATA

.
.
.
10 lines written to file WORK3:[ROUPER]TEST.DATA;5

Directory WORK3:[ROUPER]

TEST.DATA;5      TEST.DATA;4

Total of 2 files.
$
```

Using Lexical Functions

Lab Exercise 2–8: Lexical Functions

This exercise can be done interactively or in a command procedure.

Use the F\$TIME, F\$EXTRACT, and F\$EDIT lexical functions to do the following:

1. Create a local symbol containing the current date and time.

Example of symbol value:

14-NOV-1999 11:28:23.28

2. Redisplay the current date in the following format:

Nov. 14, 1999

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Lab Solution 2–8: Lexical Functions

This exercise can be done interactively or in a command procedure. The sample solution was done interactively.

Use the F\$TIME, F\$EXTRACT, and F\$EDIT lexical functions to do the following:

1. Create a local symbol containing the current date and time.

```
$ DATE = F$TIME()  
$ SHOW SYMBOL DATE  
DATE = "14-NOV-1999 11:28:23.28"  
$
```

2. Redisplay the current date in the following format:
Nov. 14, 1999

Sample solution:

```
❶ $ MONTH = F$EXTRACT(3,3,DATE)  
❷ $ NEW_MONTH = F$EXTRACT(0,1,MONTH) + -  
_ $ F$EDIT(F$EXTRACT(1,2,MONTH),"LOWERCASE")  
$ SHOW SYMBOL NEW_MONTH  
NEW_MONTH = "Nov"  
❸ $ DAY = F$EXTRACT(0,2,DATE)  
$ SHOW SYMBOL DAY  
DAY = "14"  
❹ $ YEAR = F$EXTRACT(7,4,DATE)  
$ SHOW SYMBOL YEAR  
YEAR = "1999"  
❺ $ NEW_DATE = NEW_MONTH + ". " + DAY + ", " + YEAR  
$ SHOW SYMBOL NEW_DATE  
NEW_DATE = "Nov 14, 1999"  
$
```

Notes on the sample solution:

- ❶ Extract the month field from the date: skip the first three characters of the date, and extract the next three characters.
- ❷ Change the format of the month field so it has an initial uppercase letter only: extract the first character without changing it, and extract the next two characters and convert them to lowercase.
- ❸ Extract the day field from the date: start with the first character (offset of 0), and extract the next two characters.
- ❹ Extract the year field from the date: skip the first seven characters, and extract the next four characters.
- ❺ Construct the reformatted date: put a period and space between the month and day, and put a comma and a space between the day and year.

Writing Command Procedures That Perform File I/O

Lab Exercise 2–9: Writing to a File

Write a command procedure, FILE1.COM, that does the following:

1. Creates a new file called TEST.DATA.
2. Prompts for the user's name and enters the name into TEST.DATA. (Set up a loop so this will be done more than once.)
3. Closes the file when the user presses Ctrl/Z.
4. Uses the TYPE command to display the contents of TEST.DATA.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Rev. 9.41 L2 – 21

Lab Solution 2–9: Writing to a File

Write a command procedure, FILE1.COM, that does the following:

1. Creates a new file called TEST.DATA.
2. Prompts for the user's name and enters the name into TEST.DATA. (Set up a loop so this will be done more than once.)
3. Closes the file when the user presses Ctrl/Z.
4. Uses the TYPE command to display the contents of TEST.DATA.

Sample solution:

```
$                                ! FILE1.COM
$
$ OPEN/WRITE OUTFILE TEST.DATA      ! Create new file.
$ INPUT_LOOP:
$  READ/END_OF_FILE=INPUT_DONE/PROMPT="What is your name? "
$  -
$      SYS$COMMAND NAME              ! Get name from user;
$                                ! Ctrl/Z = END_OF_FILE.
$  WRITE OUTFILE NAME                ! Write to the file.
$  GOTO INPUT_LOOP                  ! Go back for more data.
$
$ INPUT_DONE:                        ! User pressed Ctrl/Z.
$  CLOSE OUTFILE                    ! Close the file.
$
$ TYPE TEST.DATA                    ! Display file contents.
$ EXIT                              ! Terminate the procedure.
```

Sample run:

```
$ @FILE1.COM
What is your name? Krystal
What is your name? Oliver
What is your name? [ Exit ]
Krystal
Oliver
$
```

Lab Exercise 2–10: Writing to and Reading from a File

Copy FILE1.COM to a file called FILE2.COM, and edit FILE2.COM so that it:

1. Opens the existing file TEST.DAT so records can be added to it
2. Prompts the user for two pieces of information (perhaps name and badge number) and enters this information into TEST.DAT as one record
3. Displays the contents of TEST.DAT without using the TYPE command

[illegible]

Lab Solution 2–10: Writing to and Reading from a File

Copy FILE1.COM to a file called FILE2.COM, and edit FILE2.COM so that it:

1. Opens the existing file TEST.DATA so records can be added to it
2. Prompts the user for two pieces of information (perhaps name and badge number) and enters this information into TEST.DATA as one record
3. Displays the contents of TEST.DATA without using the TYPE command

Sample solution:

```
$                                     ! FILE2.COM
$
$ OPEN/APPEND OUTFILE TEST.DATA      ! Position the record
$                                     ! pointer at end of file.
$ INPUT_LOOP:
$ READ/END_OF_FILE=INPUT_DONE/PROMPT="What is your name? "
-
    SYS$COMMAND NAME                  ! Get name from user;
$                                     ! Ctrl/Z = END_OF_FILE.
$ INQUIRE BADGE "Badge number"      ! Get badge number.
$ WRITE OUTFILE NAME,BADGE           ! Write to the file.
$ GOTO INPUT_LOOP                    ! Go back for more data.
$
$ INPUT_DONE:                        ! User pressed Ctrl/Z.
$ CLOSE OUTFILE                      ! Close the file.
$
$ DISPLAY_LOOP:                      ! Display file contents.
$ OPEN/READ INFILE TEST.DATA         ! Reopen the file.
$ READ/END_OF_FILE=DISPLAY_DONE INFILE RECORD ! Get a
record.
$ WRITE SYS$OUTPUT RECORD             ! Display the record.
$ GOTO DISPLAY_LOOP                  ! Go back for another
$                                     ! record.
$ DISPLAY_DONE:                      ! End of file
encountered.
$ CLOSE INFILE                      ! Close the file.
$ EXIT                              ! Terminate the procedure.
```

Sample run:

```
$ @FILE2.COM
```

```
What is your name? Jessica
```

```
Badge number: 12
```

```
What is your name? Stephanie
```

```
Badge number: 13
```

```
What is your name? [ Exit ]
```

```
Krystal
```

```
Oliver
```

```
Jessica12
```

```
Stephanie13
```

```
$
```

Lab Exercise 2–11: Using File I/O to Get System Information

Write a command procedure that checks to see if the specified user is logged in to the system.

1. Direct a listing of system users to a file.
2. Read each record (line) of the file, using lexical functions to locate the specified user name in the record.
3. Display a message indicating whether or not the user name was found in the file.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

L2 - 27

Lab Solution 2–11: Using File I/O to Get System Information

Write a command procedure that checks to see if the specified user is logged in to the system.

1. Direct a listing of system users to a file.
2. Read each record (line) of the file, using lexical functions to locate the specified user name in the record.
3. Display a message indicating whether or not the user name was found in the file.

Sample solution:

```
$                                ! FIND_USER.COM
$ INQUIRE USER "Name of user"    ! Name of user to find.
$ SHOW USERS /OUTPUT=USER.OUT     ! Output goes to file.
$ OPEN FILE USER.OUT             ! Open the file.
$ LOOP:
$   READ/END=END_OF_FILE FILE RECORD
$                                   ! Read a record, go to
$                                   ! END_OF_FILE if done.
$                                   ! Get length of line.
$   SEARCH = F$LOCATE(USER,RECORD)
$                                   ! Check for user name.
$   IF   SEARCH .EQ. F$LENGTH(RECORD)
$   THEN GOTO LOOP                ! Name not found, try again.
$   ELSE CLOSE FILE              ! Name found, all done.
$       WRITE SYS$OUTPUT "LOGGED IN"
$       EXIT
$   ENDIF
$ END_OF_FILE:
$   CLOSE FILE                   ! Did not find user name.
$   WRITE SYS$OUTPUT "NOT LOGGED IN"
$   DELETE/NOLOG USER.OUT;      ! Delete SHOW USERS file.
$   EXIT
```

Sample run:

```
$ @FIND_USER.COM
```

```
Name of user: ABC
```

```
NOT LOGGED IN
```

```
$ @FIND_USER.COM
```

```
Name of user: USER1
```

```
LOGGED IN
```

```
$
```

Handling Error Conditions and Ctrl/Y

Lab Exercise 2–12: Handling Errors in Command Procedures

Write a command procedure that does the following:

1. Sets up an error handler to trap any error that occurs
2. Prompts the user to enter a DCL command, attempts to execute the DCL command, then exits
3. Displays the value of \$SEVERITY when an error occurs, then exits after displaying the severity message

Test your command procedure with a variety of correct and incorrect DCL commands.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

L2 - 31

Lab Solution 2–12: Handling Errors in Command Procedures

Write a command procedure that does the following:

1. Sets up an error handler to trap any error that occurs
2. Prompts the user to enter a DCL command, attempts to execute the DCL command, then exits
3. Displays the value of \$SEVERITY when an error occurs, then exits after displaying the severity message

Test your command procedure with a variety of correct and incorrect DCL commands.

Sample solution:

```
$                                ! ERROR_TRAP.COM
$ ON WARNING THEN GOTO ERROR_HANDLER      ! Set up error
handler.
$ INQUIRE COMMAND "Enter a DCL command"
$ 'COMMAND'                             ! Force symbol
substitution
$ EXIT                                  ! to execute the
command.
$!
$ ERROR_HANDLER:                    ! Handle the error.
$ SHOW SYMBOL $SEVERITY              ! Display value of
$SEVERITY.
$ EXIT
```

Sample run:

```
$ @ERROR_TRAP
Enter a DCL command: SHOW SMYBOL
%DCL-W-IVKEYW, unrecognized keyword - check validity and
spelling
```

```
$SEVERITY == "0"
$ @ERROR_TRAP
Enter a DCL command: DELETE *.*
%DELETE-E-DELVER, explicit version number or wild card
required
$SEVERITY == "2"
$ @ERROR_TRAP
```

```
Enter a DCL command: DELETE TEST.DIR;
%DELETE-W-FILNOTDEL, error deleting WORK3:[USER]TEST.DIR;1
-RMS-E-MKD, ACP could not mark file for deletion
-SYSTEM-F-DIRNOTEMPTY, directory file is not empty
$SEVERITY == "0"

$ @ERROR_TRAP
Enter a DCL command: SHOW DEFAULT
WORK3:[USER]
$ )
```


L2 - 35

Lab Solution 2–13: Handling Ctrl/Y Interrupts

1. Write a command procedure that does the following:
 - a. Prompts the user for a string and displays the string at the screen
 - b. In response to a Ctrl/Y sequence, displays the current date/time string and exits

```
$                                ! CTRL_Y_TRAP.COM
$ ON CONTROL_Y THEN GOTO CTRL_Y_HANDLER      ! Set up Ctrl/Y
    trap.
$ INQUIRE STRING "Enter any string"
$ WRITE SYS$OUTPUT STRING                    ! Display the
    string.
$ EXIT
$!
$ CTRL_Y_HANDLER:                        ! Handle the Ctrl/Y.
$  SHOW TIME                               ! Prove that we got
    here.
$  EXIT
```

2. Execute your command procedure, supplying any string as input.

```
$ @CTRL_Y_TRAP
Enter any string: abcdef
ABCDEF
$
```

3. Execute your command procedure again, pressing Ctrl/Y when the prompt appears.

```
$ @CTRL_Y_TRAP
Enter any string:
[ Interrupt ]

14-NOV-1999 16:18:44
$
```

Using the Authorize Utility

Lab Exercise 3–1: Displaying UAF Records

This exercise requires READ access to the User Authorization File.

Use the AUTHORIZE utility to display:

1. The default entry

.....

.....

.....

.....

.....

2. Your entry

.....

.....

.....

.....

.....

Lab Solution 3–1: Displaying UAF Records

This exercise requires READ access to the User Authorization File.

Use the AUTHORIZE utility to display:

1. The default entry

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

2. Your entry (where n is replaced by your assigned number)

```
UAF> SHOW OPERn
```

Adding a User to the System

Written Exercise 3–2: Adding a User

You have been told that Susan Jackson, who recently joined the finance group in your company, needs an account on your system. The account should have the same basic privileges as other accounts in the group. Members of the finance group have the group number 400 for their UIC. Each member has a default directory on the disk pointed to by the logical name FINANCE_DISK.

The password for Jackson should be her first name, and she should be required to change her password immediately upon logging in.

What steps should you follow?

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Written Solution 3–2: Adding a User

1. Create an account.

Use the Authorize utility to find an unused member number for UIC group 400.

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF> SHOW /BRIEF [400, *]
```

```
.
.
.
```

Add a UAF record for JACKSON.

```
UAF> ADD JACKSON/PASSWORD=SUSAN/PWDEXPIRED/UIC=[400,116] -
_UAF> /DIRECTORY=[JACKSON]/DEVICE=FINANCE_DISK:/FLAG=NODISUSER
```

The /PWDEXPIRED qualifier assures that user Jackson must change her password immediately upon logging in to her account.

2. If disk quotas are in effect, set up a disk quota for Jackson.

```
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> DISKQUOTA ADD [400,116] -
_SYSMAN> /PERMQUOTA=1000/OVER=200/DEVICE=FINANCE_DISK
```

3. Create a directory.
4. Use the following commands to create a directory for user Jackson:

```
$ SET DEFAULT FINANCE_DISK:[000000]
$ CREATE/DIRECTORY/OWNER=[JACKSON] [JACKSON]
```

5. Test the account by logging in to it before turning it over to the user. Because the password is preexpired, it must be reset.
6. Instruct the user about system procedures.

New users should receive information about logging in and appropriate use of the system. This can be done by you or another person in your organization.

In this example, Susan Jackson should be told that her user name is JACKSON, her password is SUSAN, and she must use a new password when she logs in. In addition, she should be made aware of any procedures specific to your site.

Lab Exercise 3–3: Using ADDUSER.COM

1. Use the procedure SYS\$EXAMPLES:ADDUSER.COM to add an account, STUDENT_nn, to the system. You will need a unique number to avoid conflict with other students.

.....

.....

.....

.....

.....

2. Read the procedure to see how it works (optional).

.....

.....

.....

.....

.....

Lab Solution 3–3: Using ADDUSER.COM

1. Use the procedure SYS\$EXAMPLES:ADDUSER.COM to add an account, STUDENT_nn, to the system. You will need a unique number to avoid conflict with other students.

```
$ @SYS$EXAMPLES:ADDUSER.COM
```

The procedure prompts for information. Use STUDENT_nn for the account name.

2. Read the procedure to see how it works (optional).

```
$ TYPE/PAGE SYS$EXAMPLES:ADDUSER.COM
```

Removing a User Account

Written Exercise 3–4: Steps for Removing a User

List the steps to follow when removing a user from the system.

This image shows a full page of white paper with horizontal dotted lines, typical of primary school writing paper. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Written Solution 3–4: Steps for Removing a User

List the steps to follow when removing a user from the system.

1. Prevent the user from logging in.
2. Remove disk files from the user directories.
3. Remove UIC entry from the disk quota file (if any).
4. Remove mail information for the user (if any).
5. Remove the user's UAF record.

Managing and Monitoring User Processes

Lab Exercise 3–5: Controlling User Processes

To practice controlling user processes, log in at two terminals or do the following with a partner:

1. Designate one terminal as the Manager terminal, and the other as the User terminal. Log in at both terminals. The process on the Manager terminal must have WORLD or GROUP privileges to affect the other processes.

.....
.....

2. The user should begin entering DCL commands that produce output, such as SHOW PROCESS and SHOW SYSTEM. (You can also create a command procedure to do this repeatedly.)

.....
.....

3. Find out the process ID number of the user process from the Manager terminal.

.....
.....

4. Suspend the user process from the Manager terminal (preferably while output is appearing on the User terminal).

.....
.....

5. Observe the User terminal. Note the lack of response from the keyboard (if you press any key or combination of keys, the state of the user process does not change).

.....
.....

6. Allow the user process to continue.

.....
.....

7. Stop the user process from the Manager terminal. Can the Manager allow the process to continue?

.....
.....

Lab Solution 3–5: Controlling User Processes

1. Designate one terminal as the Manager terminal, and the other as the User terminal. Log in at both terminals. The process on the Manager terminal must have WORLD or GROUP privileges to affect the other processes. No solution needed.
2. The user should begin entering DCL commands that produce output, such as SHOW PROCESS and SHOW SYSTEM. (You can also create a command procedure to do this repeatedly.)

The following command procedure produces an endless loop.

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

3. Find out the process ID number of the user process from the Manager terminal.
The commands SHOW SYSTEM and SHOW USERS/FULL display the PIDs of the users on the system. The PID is a hexadecimal number.
4. Suspend the user process from the Manager terminal (preferably while output is appearing on the User terminal).

```
$ SET PROCESS/SUSPEND/ID=21400FA1
```

Substitute the appropriate PID for 21400FA1.

5. Observe the User terminal. Note the lack of response from the keyboard (if you press any key or combination of keys, the state of the user process does not change).

No solution needed.

6. Allow the user process to continue.

```
$ SET PROCESS/RESUME/ID=21400FA1
```

Substitute the appropriate PID for 21400FA1.

7. Stop the user process from the Manager terminal. Can the Manager allow the process to continue?

```
$ STOP/ID=21400FA1
```

The process cannot be continued, since it has been deleted.

Lab Exercise 3–6: Using the SYSMAN Utility

You will need OPER privilege to run SYSMAN.

1. Invoke SYSMAN and see what information is returned by the SHOW PROFILE and SHOW ENVIRONMENT commands.

.....
.....
.....

2. Use SET PROFILE to change your default device and/or directory, and verify this with SHOW PROFILE.

.....
.....
.....

3. If your system is part of a cluster:

- a. SET ENVIRONMENT to another node in the cluster. SHOW ENVIRONMENT to verify this.

.....
.....

- b. Use DO SHOW SYSTEM to see what is happening on the other node.

.....
.....

- c. Look at some of the SYSGEN parameters on the other node.

.....
.....

- d. Try some of the CONFIGURATION SHOW commands.

.....
.....

- e. Now change your target environment so it includes the entire local cluster. Note that you do not need to include the /NODE qualifier unless you want your environment to be a remote cluster.

.....
.....

- f. Having set your environment to be the entire cluster, repeat the activities suggested above. How is the output different?

.....
.....

4. If your system is part of a network:

- a. SET ENVIRONMENT to a remote node. SHOW ENVIRONMENT to verify.

You will need OPER or SETPRV privilege on that node in order to execute commands there. If you lack the necessary privilege(s), you can still try the commands described below and see what error messages are returned.

.....
.....

- b. Use DO SHOW SYSTEM or DO SHOW USERS to see what is happening on the other node.

.....
.....

- c. Look at some of the SYSGEN parameters on the other node.

.....
.....

- d. Try some of the CONFIGURATION SHOW commands.

.....
.....

Lab Solution 3–6: Using the SYSMAN Utility

1. Invoke SYSMAN and see what information is returned by the SHOW PROFILE and SHOW ENVIRONMENT commands.

```
$ SET PROCESS/PRIVILEGE=OPER
$ RUN SYS$SYSTEM:SYSMAN
SYSMAN> SHOW PROFILE
SYSMAN> SHOW ENVIRONMENT
```

2. Use SET PROFILE to change your default device and/or directory, and verify this with SHOW PROFILE.

```
SYSMAN> SET PROFILE/DEFAULT=[directory]
SYSMAN> SHOW PROFILE
```

3. If your system is part of a cluster:

- a. SET ENVIRONMENT to another node in the cluster. SHOW ENVIRONMENT to verify this.

```
SYSMAN> SET ENVIRONMENT/NODE=other_node
SYSMAN> SHOW ENVIRONMENT
```

- b. Use DO SHOW SYSTEM to see what is happening on the other node.

```
SYSMAN> DO SHOW SYSTEM
```

- c. Look at some of the SYSGEN parameters on the other node.
For example:

```
SYSMAN> PARAMETERS SHOW /LGI
```

(Note that CMKRNL privilege is required.)

- d. Try some of the CONFIGURATION SHOW commands.
For example:

```
SYSMAN> CONFIGURATION SHOW TIME
SYSMAN> CONFIGURATION SHOW CLUSTER_AUTHORIZATION
```

- e. Now change your target environment so it includes the entire local cluster.

Note that you do not need to include the /NODE qualifier unless you want your environment to be a remote cluster.

```
SYSMAN> SET ENVIRONMENT/CLUSTER
```

- f. Having set your environment to be the entire cluster, repeat the activities suggested above. How is the output different?

The commands execute on each node in the cluster.

- 4. If your system is part of a network:

- a. SET ENVIRONMENT to a remote node. SHOW ENVIRONMENT to verify.

You will need OPER or SETPRV privilege on that node in order to execute commands there. If you lack the necessary privilege(s), you can still try the commands described below and see what error messages are returned.

```
SYSMAN> SET ENVIRONMENT/NODE=other_node
```

```
SYSMAN> SHOW ENVIRONMENT
```

- b. Use DO SHOW SYSTEM or DO SHOW USERS to see what is happening on the other node.

```
SYSMAN> DO SHOW SYSTEM
```

```
SYSMAN> DO SHOW USERS
```

- c. Look at some of the SYSGEN parameters on the other node.

For example:

```
SYSMAN> PARAMETERS SHOW /LGI
```

(Note that CMKRNL privilege is required.)

- d. Try some of the CONFIGURATION SHOW commands.

For example:

```
SYSMAN> CONFIGURATION SHOW TIME
```

```
SYSMAN> CONFIGURATION SHOW CLUSTER_AUTHORIZATION
```

Monitoring the Use of Disk Space

Lab Exercise 3–7: Monitoring Disk Space

1. Create a subdirectory with a version limit of 3, and set your default to that subdirectory.
 - a. Create a file called XYZ.TMP in the subdirectory.
.....
.....
 - b. Use an editor to change something in the file, then exit from the editor. Use the editor to change the file again in this way three more times.
.....
.....
 - c. Get a directory display to see how many copies of the file you ended up with.
.....
.....
 - d. Purge all but the most recent copy.
.....
.....
2. Find out how much space is left on the disk(s) on our system.
.....
.....
3. Use a DCL command to find out if disk quotas are enabled and, if so, what your disk quota is.
.....
.....
4. If disk quotas are enabled, use the SYSMAN utility to display everyone's quota.
.....
.....

Lab Solution 3–7: Monitoring Disk Space

1. Create a subdirectory with a version limit of 3, and set your default to that subdirectory.

```
$ CREATE/DIRECTORY/VERSION_LIMIT=3 [.MYSUB]
$ SET DEFAULT [.MYSUB]
```

- a. Create a file called XYZ.TMP in the subdirectory.

```
$ CREATE XYZ.TMP
```

Enter a line of text

[Ctrl/z]

- b. Use an editor to change something in the file, then exit from the editor. Use the editor to change the file again in this way three more times.

```
$ EDIT/TPU XYZ.TMP
```

Change something

[Ctrl/z] \$ EDIT/TPU XYZ.TMP

Change something else

[Ctrl/z]

```
$ EDIT/TPU XYZ.TMP
```

Change something else

[Ctrl/z]

```
$ EDIT/TPU XYZ.TMP
```

Change something else

[Ctrl/z]

- c. Get a directory display to see how many copies of the file you ended up with.

```
$ DIRECTORY XYZ.TMP;3 XYZ.TMP;4 XYZ.TMP;5
```

- d. Purge all but the most recent copy.

```
$ PURGE
```

2. Find out how much space is left on the disk(s) on our system.

```
$ SHOW DEVICE D
```

3. Use a DCL command to find out if disk quotas are enabled and, if so, what your disk quota is.

```
$ SHOW QUOTA
```

(This will either display your quota, or say that quotas are not enabled.)

4. If disk quotas are enabled, use the SYSMAN utility to display everyone's quota.

```
$ RUN SYS$SYSTEM:SYSMAN
  SYSMAN> DISKQUOTA SHOW *
  SYSMAN> [Ctrl/z]
```


Monitoring Queues and Jobs

Lab Exercise 4–1: Displaying Queue Information

1. Find out what batch queues are already on the system.

.....
.....

2. Find out what generic queues are already on the system.

.....
.....

Lab Solution 4–1: Displaying Queue Information

1. Find out what batch queues are already on the system.
\$ SHOW QUEUE/BATCH
2. Find out what generic queues are already on the system.
\$ SHOW QUEUE/GENERIC

Lab Exercise 4–2: Monitoring Queues and Jobs

Your instructor will have set up two queues, `STUDENT$BATCH` and `STUDENT$PRINT`, for this laboratory exercise. (If resources are available, the instructor may be able to set up queues for each student, `STUDENTnn$BATCH` and `STUDENTnn$PRINT`.) To avoid confusion, go through the steps of this exercise in coordination with the rest of the class.

1. Print a file to the output queue.

.....
.....

2. Send a print job to the output queue, including the `/HOLD` qualifier in the command.

.....
.....

3. Display the contents and characteristics of your output queue.

.....
.....

4. Delete your job, which is being held on the output queue.

.....
.....

Lab Solution 4–2: Monitoring Queues and Jobs

1. Print a file to the output queue.

```
$ PRINT/QUEUE=STUDENT$PRINT FILE.TXT
```
2. Send a print job to the output queue, including the /HOLD qualifier in the command.

```
$ PRINT/QUEUE=STUDENT$PRINT /HOLD FILE.TXT
```
3. Display the contents and characteristics of your output queue.

```
$ SHOW QUEUE/FULL/ALL STUDENT$PRINT
```
4. Delete your job, which is being held on the output queue.

```
$ DELETE/ENTRY=369
```

Substitute the entry number of your job for 369.

Changing the Characteristics of a Queue

Lab Exercise 4–3: Modifying Queue Characteristics

1. Modify the default characteristics of the STUDENT\$PRINT execution queue so that all jobs sent to the queue are printed with flag and trailer pages.

.....
.....

2. Send a short text file to STUDENT\$PRINT and observe the output.

.....
.....

3. Find out if a job limit has been set on STUDENT\$BATCH. Change the job limit to a different value, or set it to four if no limit is specified.

.....
.....

4. Display the characteristics of the queue after changing the job limit to check that the command was executed properly.

.....
.....

Lab Solution 4–3: Modifying Queue Characteristics

1. Modify the default characteristics of the STUDENT\$PRINT execution queue so that all jobs sent to the queue are printed with flag and trailer pages.

```
$ SET QUEUE/SEPARATE=(FLAG, TRAILER) STUDENT$PRINT
```

2. Send a short text file to STUDENT\$PRINT and observe the output.

```
$ PRINT/QUEUE=STUDENT$PRINT FILE.TXT
```

3. Find out if a job limit has been set on STUDENT\$BATCH. Change the job limit to a different value, or set it to four if no limit is specified.

```
$ SHOW QUEUE/FULL STUDENT$BATCH
```

If no JOB_LIMIT value is shown, no limit has been set

The command to use to initialize STUDENT\$BATCH with a job limit of two is

```
$ INITIALIZE/QUEUE/BATCH/START/JOB_LIMIT=2 STUDENT$BATCH
```

To change the JOB_LIMIT

```
$ SET QUEUE/JOB_LIMIT=4 STUDENT$BATCH
```

4. Display the characteristics of the queue after changing the job limit to check that the command was executed properly.

```
$ SHOW QUEUE/FULL STUDENT$BATCH
```

Managing Jobs in a Queue

Lab Exercise 4–4: Managing Jobs

1. Send two separate print jobs to your print queue, holding them until you release them.
.....
.....
2. Release the first job, then abort it.
.....
.....
3. Display the contents of the queue immediately after aborting your job.
.....
.....
4. Stop the queue using the /RESET qualifier. You now have complete control over the queue.
.....
.....
5. Release the next job being held on the queue.
.....
.....
6. Display the contents of the queue and note the status of the jobs.
.....
.....
7. Start the queue and observe the output. (This is particularly effective if you have individual terminal execution queues for the exercise.)
.....
.....
8. Stop and delete your output queue and your batch queue.
.....
.....
9. If necessary, have the instructor help you set the output terminal back to its original characteristics (if the output queues were terminal execution queues). You will need OPER and LOG_IO privileges to do this.
.....
.....

Lab Solution 4–4: Managing Jobs

Substitute your file names, entry numbers, and queue names in the commands listed below. If individual output and/or batch queues were created for this exercise, use the queue names STUDENTnn\$PRINT and STUDENTnn\$BATCH.

1. Send two separate print jobs to your print queue, holding them until you release them.

```
$ PRINT/QUEUE=STUDENT$PRINT/HOLD FILE1.TXT
```

```
$ PRINT/QUEUE=STUDENT$PRINT/HOLD FILE2.TXT
```

2. Release the first job, then abort it.

```
$ SET ENTRY 369/RELEASE
```

```
$ STOP/ABORT STUDENT$PRINT
```

3. Display the contents of the queue immediately after aborting your job.

```
$ SHOW QUEUE STUDENT$PRINT
```

Note the status of the jobs on the queue. Also note that trailer pages, if specified, will be printed for a job that was aborted.

4. Stop the queue using the /RESET qualifier. You now have complete control over the queue.

```
$ STOP/QUEUE/RESET STUDENT$PRINT
```

5. Release the next job being held on the queue.

```
$ SET ENTRY 369/RELEASE
```

Notice that the job did not print, as the queue is stopped.

6. Display the contents of the queue and note the status of the jobs.

```
$ SHOW QUEUE STUDENT$PRINT
```

7. Start the queue and observe the output. (This is particularly effective if you have individual terminal execution queues for the exercise.)

```
$ START/QUEUE STUDENT$PRINT
```

8. Stop and delete your output queue and your batch queue.

```
$ STOP/QUEUE/NEXT STUDENT$PRINT  
$ DELETE/QUEUE STUDENT$PRINT  
$ STOP/QUEUE/NEXT STUDENT$BATCH  
$ DELETE/QUEUE STUDENT$BATCH
```
9. If necessary, have the instructor help you set the output terminal back to its original characteristics (if the output queues were terminal execution queues). You will need OPER and LOG_IO privileges to do this.

If the output execution queue is on your terminal, the terminal characteristics will need to be reset. Use the correct device name for your output terminal.

```
$ SET DEVICE/NOSP00L TTC5:  
$ SET TERMINAL/PERMANENT/TYPE_AHEAD/BROADCAST TTC5:
```

Performing Backups and Restores

Module 5 Labs

Introduction to the Exercises

If a tape drive (either reel or cartridge) is available to you, perform the Backup to Tape exercise. If not, perform the Backup to Disk exercise.

Lab Exercise 5–1: Backup to Tape

Obtain a scratch tape and use it to perform this exercise. In this exercise, you will first use the copy command to save files and then the backup command.

1. Software allocate a tape drive, and physically load a tape on the drive.
.....
.....
2. Initialize and software mount (file structured) the magnetic tape. If the tape is not new and did not belong to you, you need VOLPRO privilege.
.....
.....
3. Using the COPY command, copy all of the .COM files from SYSS\$MANAGER and SYSS\$UPDATE to the tape.
.....
.....
4. Get a directory listing of the files on tape.
 - Did it keep the original directory structure?
.....
.....
5. Dismount the tape drive without unloading the tape.
.....
.....
6. Reinitialize the tape.
.....
.....
7. Using the BACKUP command, back up all of the .COM files from SYSS\$MANAGER and SYSS\$UPDATE to the tape creating a SAVE_SET called NAME.BCK.
.....
.....

8. Dismount the tape drive without unloading the tape.

.....
.....

9. Mount the tape (file structured) and take a directory listing.

- Did you see the files that you backed up?
- What is the name of the SAVE_SET?

.....
.....

10. Dismount the tape drive without unloading the tape.

.....
.....

11. Using BACKUP, get a listing of the SAVE_SET on tape.

- What directory were the files placed into?
- Did it keep the original directory structure?

.....
.....

12. Dismount and deallocate the tape.

.....
.....

Lab Solution 5–1: Backup to Tape

Substitute your tape device name for **MTA0:** and your label for **label**.

1. Software allocate a tape drive and physically load a tape on the drive.

```
$ ALLOCATE MTA0:
```

2. Initialize and software mount the magnetic tape.

```
$ INITIALIZE MTA0: label
```

Mount the tape using the following command:

```
$ MOUNT MTA0: label
```

3. Using the COPY command, copy all of the .COM files from SYS\$MANAGER and SYS\$UPDATE to the tape.

```
$ COPY SYS$MANAGER:*.COM,SYS$UPDATE:*.COM MTA0:
```

4. Get a directory listing of the files on tape.

```
$ DIRECTORY MTA0:
```

- Did it keep the original directory structure?

NO

5. Dismount the tape drive without unloading the tape.

```
$ DISMOUNT/NOUNLOAD MTA0:
```

6. Reinitialize the tape.

```
$ INITIALIZE MTA0: label
```

7. Using the BACKUP command, back up all of the .COM files from SYS\$MANAGER and SYS\$UPDATE to the tape creating a SAVE_SET called NAME.BCK.

```
$ BACKUP
```

```
_From: SYS$MANAGER:*.COM,SYS$UPDATE:*.COM
```

```
_To: MTA0:NAME.BCK/SAVE_SET/LABEL=label
```

8. Dismount the tape drive without unloading the tape.
\$ DISMOUNT/NOUNLOAD MTA0:
9. Mount the tape (file structured) and take a directory listing.
\$ MOUNT MTA0: label
\$ DIRECTORY MTA0:
 - Did you see the files that you backed up?
NO
 - What is the name of the SAVE_SET?
NAME.BCK
10. Dismount the tape drive without unloading the tape.
\$ DISMOUNT/NOUNLOAD MTA0:
11. Using BACKUP, get a listing of the SAVE_SET on tape.
\$ BACKUP/LIST MTA0:NAME.BCK
 - What directory were the files placed into?
SAME DIRECTORIES
 - Did it keep the original directory structure?
YES
12. Dismount and deallocate the tape.
\$ DISMOUNT MTA0:
\$ DEALLOCATE MTA0:

Lab Exercise 5–2: Backup to Disk

1. Create a new subdirectory one level below your main directory. Name the new subdirectory SUBA.
.....
2. Using BACKUP, back up all the .COM files from SYSS\$UPDATE to your new subdirectory SUBA.
.....
3. Get a directory listing of your new subdirectory.
.....
.....
 - Were the files copied?
4. Using BACKUP, back up all your files and directories into a SAVE_SET in your top level directory called NAME.BCK.
.....
.....
5. Using BACKUP, get a listing of your SAVE_SET.
.....
.....

6. Delete all of your files EXCEPT the SAVE_SET just created. Also delete the files in your subdirectories and the subdirectories themselves. You should have just one file left, the SAVE_SET.

.....
.....

7. Using BACKUP, restore the file AUTOGEN.COM from your SAVE_SET, and place it in your subdirectory SUBA. Backup will recreate the subdirectory.

.....
.....

8. Get a directory to see that you were successful.

.....
.....

9. Using BACKUP, restore all of your files from your SAVE_SET keeping them in their proper directories.

.....
.....

Lab Solution 5–2: Backup to Disk

Substitute your directory name for STUDENT in the following examples.

1. Create a new subdirectory one level below your main directory. Name the new subdirectory SUBA.

```
$ CREATE/DIRECTORY [ .SUBA]
```

2. Using BACKUP, back up all the .COM files from SYS\$UPDATE to your new subdirectory SUBA.

```
$ BACKUP _From: SYS$UPDATE:* .COM  
_To: [ .SUBA]
```

3. Get a directory listing of your new subdirectory.

```
$ DIRECTORY [ .SUBA]
```

- Were the files copied?
YES

4. Using BACKUP, back up all your files and directories into a SAVE_SET in your top level directory called NAME.BCK.

```
$ BACKUP  
_From: [...]*. *.*  
_To: NAME.BCK/SAVE_SET
```

5. Using BACKUP, get a listing of your SAVE_SET.

```
$ BACKUP/LIST NAME.BCK/SAVE_SET
```

6. Delete all of your files EXCEPT the SAVE_SET just created. Also delete the files in your subdirectories and the subdirectories themselves. You should have just one file left, the SAVE_SET.

```
$ SET PROTECTION=0:RWED [...]*. *.*
$ DELETE/EXCLUDE=NAME.BCK [...]*. *.*
```

DELETES ALL FILES

```
$ DELETE/EXCLUDE=NAME.BCK [...]*. *.*
```

DELETES SUBDIRECTORY

7. Using BACKUP, restore the file AUTOGEN.COM from your SAVE_SET and place it in your subdirectory SUBA. Backup will recreate the subdirectory.

```
$ BACKUP _From: NAME.BCK/SAVE_SET/SELECT=[STUDENT.SUBA]AUTOGEN.COM
    _To: [.SUBA]*.*
```

8. Get a directory to see that you were successful.

```
$ DIRECTORY [.SUBA]
```

9. Using BACKUP, restore all of your files from your SAVE_SET keeping them in their proper directories.

```
$ BACKUP _
From: NAME.BCK/SAVE_SET
    _To: [*]*.*.*
```

Shutting Down and Starting Up an OpenVMS System

Module 6 Labs

Shutting Down an OpenVMS System

Lab Exercise 6–1: Shutdown

You must perform the following exercises on a standalone system under the supervision of your instructor. You must start the system up after each shutdown.

1. Use the orderly shutdown method to shut down your system.

.....
.....

2. Use the emergency DCL shutdown method to shut down your system.

.....
.....

3. Force a shutdown using the CCL method.

.....
.....

Lab Solution 6–1: Shutdown

1. Use the orderly shutdown method to shut down your system.

Enter the command:

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

Respond to the prompts to shut down your system.

2. Use the emergency DCL shutdown method to shut down your system.

Follow these steps:

- a. Log in as SYSTEM on the console terminal.
- b. Issue the DCL command:

```
$ RUN SYS$SYSTEM:OPCCRASH
```

- c. When the SYSTEM SHUTDOWN COMPLETE message is displayed, halt the system.

3. Force a shutdown using the CCL method.

For the CCL method of forcing a shutdown, see the *Installation and Operation Guide* for your system.

Starting Up an OpenVMS System

Lab Exercise 6–2: Hardware

1. Under the supervision of your instructor, look at the front panels of as many Alpha and VAX processors as are available to you. Become familiar with the items on them.

.....
.....

2. Your instructor will shut the system down for you and will turn the power off. (If he or she cannot shut the power off, go to problem 3). It is not good practice to shut the power off and on very often, as it can lead to hardware problems. (However, if this is only done occasionally, it should not harm the system.)

.....
.....

3. Start the system from power on, following the instructions in the *VMS Installation and Operation Guide* for this type of processor.

.....
.....

4. If your instructor allows you to, and if your system has battery backup, set up your system for an automatic restart and turn the power off. Start the system from power off. (The system should have been running before you began this problem and you should have been logged in.)

.....
.....

Lab Solution 6–2: Hardware

1. No solution needed.
2. No solution needed.
3. No solution needed.
4. No solution needed. If you set up your system correctly, it should start, and you should be able to log in.

Lab Exercise 6–3: Console System

Look up the console system for the VAX system available to you in the hardware manual supplied with this course. Using that as reference, do each problem in this lab under the **direct** supervision of your instructor.

1. Your instructor will shut the system down for you. The console terminal is now in program mode. Log in at the console terminal.

.....
.....

2. While you are logged in at the console terminal, change to console mode. Do not enter any console commands.

.....
.....

3. Return to program mode and enter the DIRECTORY command.

.....
.....

4. Disable the use of console mode. Try to enter console mode at the console terminal.

.....
.....

5. If this system boots by means of boot files, examine the default boot file. You may have to copy it from the console volume to your directory to look at it. Notice that it boots the default system device on your system (or it should).

.....
.....

Lab Solution 6–3: Console System

1. No solution needed.
2. When you see the >>> prompt, you know you have been successful.
3. When you see the \$ prompt and can enter the DIRECTORY command, you know you are in program mode.
4. You should not be able to enter console mode if you set the switch correctly.
Enable console mode again when you are done with this problem.
5. On some VAX systems you can use CCL commands to examine the boot file. On others, you must be in program mode to copy a file from the console volume. Use the SYS\$UPDATE:DXCOPY command procedure. If your default boot file, DEFBOO.CMD, does not boot your default system device, inform your instructor.

Lab Exercise 6–4: Errors in Startup

Complete the following exercises only under the supervision of your instructor.

The purpose of this optional lab is to show you that different errors can occur at different points during the startup process. Some errors indicate a problem with the console volume. It could be corrupted, missing, or it could have been replaced by a scratch or data volume. Other errors indicate a problem with the system disk. It could be corrupted, off-line, write-locked, or certain files necessary for the startup process could be missing or corrupted. Finally, errors could indicate problems with data disks, buses, memory, or other parts of the system.

You can usually identify the problem by looking at the listing on the console terminal. Certain errors occur at certain points in the startup process. For example, if the console volume is missing, you will not even get to the part of the listing where the system volume is read. Or, if you cannot get to the part where the console volume is read, perhaps the problem lies in memory or some part of the CPU.

If you are having problems starting up your system, first check to be sure you are preparing your system properly for startup. If you still cannot start it, show the console terminal listings to a more experienced person or to a Digital Customer Services representative.

1. Shut the system down using the normal method.
2. Remove the console volume. (If this is not possible on your system, go to question 9. If you are working on a VAX-11/750 system, also set the BOOT DEVICE switch to A.)

**Caution**

If your console volume is a diskette, do not remove it if you hear a clicking noise from the drive. The noise indicates that the system is using the volume. If you remove the volume while the system is using it, you can damage the volume.

3. Attempt to boot the system by using the boot switch on the front panel or by entering a command at the console terminal.

What are the results?

4. Place a scratch volume in the console drive.
5. Attempt to boot the system by using the boot switch on the front panel or by entering a command at the console terminal.

What are the results?

6. Replace the console volume in the console drive.
7. Spin down the system disk.

8. Attempt to boot the system by using the boot switch on the front panel or by entering a command at the console terminal.

What are the results?

9. Spin up the system disk and write-protect it.
10. Attempt to boot the system by using the boot switch on the front panel or by entering a command at the console terminal.

What are the results?

11. Write-protect a data disk that your site-specific command procedure normally loads.
12. Attempt to boot the system by using the boot switch on the front panel or by entering a command at the console terminal.

What are the results?

Lab Solution 6–4: Errors in Startup

1. Use the SHUTDOWN.COM procedure.

2. No solution needed.

3. No solution needed.

You should receive an error message at the console terminal. The content of this message differs between processors, but its meaning is the same. The meaning is that the system cannot boot without the console volume.

4. No solution needed.

5. No solution needed.

You should receive a different error message at the console terminal than the one you received when no volume was loaded. The meaning of this second message is that the system cannot boot without a proper system console volume.

6. No solution needed.

7. No solution needed.

8. No solution needed.

You should see more messages on the console terminal this time before you see an error message. The meaning of this third error message is that the system cannot read the system volume when it is not spun up. It must be able to read the system volume to complete the startup process.

9. No solution needed.

10. No solution needed.

You should see an error message on the console terminal. The meaning of this message is that the startup process cannot complete unless it can both read from and write to the system disk.

11. No solution needed.

12. No solution needed.

The startup process will complete if a data disk is not loaded, but you will receive error messages at the console terminal.