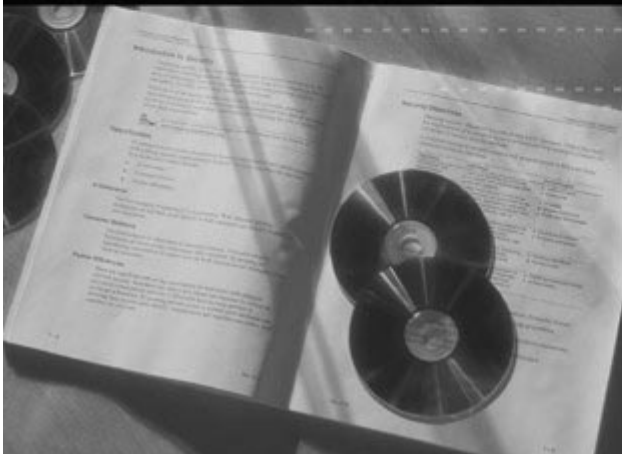


OpenVMS Skills for Users

Lab Guide

Training



COMPAQ

OpenVMS Skills for Users

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 Skills for Users

Lab Guide

December 1999

Module 1 — Logging In and Out of an OpenVMS System

Written Exercise: Terminology	2
Written Solution: Terminology	3
Lab Exercise: Terminal Server	4
Lab Solution: Terminal Server	5
Lab Exercise: Remote Session	7
Lab Solution: Remote Session	8
Written Exercise: Terminology	9
Written Solution: Terminology	10
Lab Exercise: Logging In, Changing Passwords, and Logging Out	11
Lab Solution: Logging In, Changing Passwords, and Logging Out	12

Module 2 — Issuing Commands

Constructing a DCL Command	2
Written Exercise: DCL Command Components	2
Written Solution: DCL Command Components	3
Using DCL to Interact with the System	4
Written Exercise: Understanding System Messages	4
Written Solution: Understanding System Messages	5
Lab Exercise: Error Messages	6
Lab Solution: Error Messages	7
Using DCL Features	8
Lab Exercise: Recalling and Editing Command Lines	8
Lab Solution: Recalling and Editing Command Lines	9

Module 3 — Naming and Storing Files

Constructing an OpenVMS File Specification	2
Written Exercise: Constructing OpenVMS File Specifications	2
Written Solution: Constructing OpenVMS File Specifications	3
Creating and Identifying Subdirectories	4
Written Exercise: User Subdirectories	4
Written Solution: User Subdirectories	5
Lab Exercise: User Subdirectories	6
Lab Solution: User Subdirectories	7
Displaying Information About Files	8
Written Exercise: Using Wildcards	8
Written Solution: Using Wildcards	10
Lab Exercise: DIRECTORY Command	11
Lab Solution: DIRECTORY Command	12

Module 4 — Creating Memos, Reports, and Data Files

Lab Exercise 1: Creating a File and Issuing Commands	2
--	---

Lab Solution 1: Creating a File and Issuing Commands	3
Lab Exercise 2: Editing a File	4
Lab Solution 2: Editing a File	5
Lab Exercise: Requesting Help	6
Lab Solution: Requesting Help	7
Lab Exercise 1: Including a File in a Document and Issuing DCL Commands	8
Lab Solution 1: Including a File in a Document and Issuing DCL Commands	9
Lab Exercise 2: Alternate Key Definitions	10
Lab Solution 2: Alternate Key Definitions	11
Lab Exercise 1: Saving an Edited File	12
Lab Solution 1: Saving an Edited File	13
Lab Exercise 2: Recovering an Editing Session.....	14
Lab Solution 2: Recovering an Editing Session.....	15

Module 5 — Manipulating Files

Making Copies of Files	2
Lab Exercise: Copying Files	2
Lab Solution: Copying Files	3
Displaying the Contents of Files	4
Lab Exercise: Typing Files	4
Lab Solution: Typing Files.....	5
Printing Files	6
Lab Exercise: Printing Files	6
Lab Solution: Printing Files	7
Renaming Files	9
Lab Exercise: Renaming Files.....	9
Lab Solution: Renaming Files.....	10
Removing Files from Disk	11
Lab Exercise: Deleting and Purging Files.....	11
Lab Solution: Deleting and Purging Files.....	12

Module 6 — Communicating with Other Users

Using the Mail Utility	2
Lab Exercise 1: Sending and Reading Mail Messages	2
Lab Solution 1: Sending and Reading Mail Messages	3
Lab Exercise 2: MAIL Commands.....	5
Lab Solution 2: MAIL Commands.....	6
Lab Exercise 3: Working with Folders	7
Lab Solution 3: Working with Folders	8
Lab Exercise 4: Working with Files	11
Lab Solution 4: Working with Files	12

Module 7 — Getting Help

Using Online Help	2
Lab Exercise 1: HELP Command	2
Lab Solution 1: HELP Command.....	3
Lab Exercise 2: More Complicated Investigation.....	4
Lab Solution 2: More Complicated Investigation	5
Lab Exercise: Locating Information.....	6

Module 8 — Protecting Your Data

Using File Protection	2
Written Exercise 1: File Protection	2
Written Solution 1: File and Directory Protection	3
Written Exercise 2: UIC Protection.....	4
Written Solution 2: UIC Protection.....	5
Lab Exercise 1: Protecting Files.....	6
Lab Solution 1: Protecting Files.....	7
Lab Exercise 2: Deleting a Subdirectory	9
Lab Solution 2: Deleting a Subdirectory	10
Saving Copies of Files	11
Lab Exercise: Backing Up and Restoring Your Files.....	11
Lab Solution 1: Backing Up and Restoring Your Files.....	12

Module 9 — Customizing Your Working Environment

Using Logical Names.....	2
Lab Exercise: Logical Names	2
Lab Solution: Logical Names.....	3
Using Symbols.....	5
Lab Exercise: Creating Symbols	5
Lab Solution: Creating Symbols	6
Using Keypad Definitions.....	8
Lab Exercise: Defining a Key to Recall Commands.....	8
Lab Solution: Defining a Key to Recall Commands <align_char>(#).....	9
Lab Exercise: Writing a LOGIN.COM.....	10
Lab Solution: Writing a LOGIN.COM.....	11

Logging In and Out of an OpenVMS System

Module 1

Written Exercise: Terminology

Match the definitions with the terms below. Terms may be used more than once, and some definitions may match up with more than one term.

- | | |
|--------------------|--|
| a. [Break] | A word or phrase, not echoed when typed, that is required for access to a system. |
| b. CONNECT | Command used when accessing a system through a terminal server. |
| c. DCL prompt | Command used to display the number of sessions at the terminal server prompt. |
| d. Password | Command used to initiate access to a system through a terminal server. |
| e. Session | \$ (dollar sign) |
| f. SET FORWARD | Required if you are to be able to have more than one terminal session in progress at a time. |
| g. SET PASSWORD | Required information that must be entered by a user during the login procedure. |
| h. SHOW SESSIONS | Used to return to the terminal server prompt without logging out. |
| i. Terminal server | |
| j. User name | |

Written Solution: Terminology

- | | |
|--------------------|---|
| a. [Break] | d. A word or phrase, not echoed when typed, that is required for access to a system. |
| b. CONNECT | b,f. Command used when accessing a system through a terminal server. |
| c. DCL prompt | h. Command used to display the number of sessions at the terminal server prompt. |
| d. Password | b. Command used to initiate access to a system through a terminal server. |
| e. Session | c. \$ (dollar sign) |
| f. SET FORWARD | i. Required if you are to be able to have more than one terminal session in progress at a time. |
| g. SET PASSWORD | d. Required information that must be entered by a user during the login procedure. |
| h. SHOW SESSIONS | a. Used to return to the terminal server prompt without logging out. |
| i. Terminal server | |
| j. User name | |

Lab Exercise: Terminal Server



Note

The following exercises can only be done if you are running your laboratory session on a system that is supported by terminal server software. Check with your instructor.

- Log in to the system using a legal user name and password.
- Practice the use of the Break key and various terminal server commands:
 - a. Use the Break key to establish another session at your terminal.
 - b. Use the SHOW SESSIONS command to see your two sessions.
 - c. Use terminal server commands to move back and forth between sessions.
 - d. Use the RESUME command to resume each session.
 - e. Use the LOCK command to lock your session.
 - f. Enter the password to unlock your terminal.
- Log out of the system on both sessions.

Lab Solution: Terminal Server

1. Log in to the system using a legal user name and password.

```
DECserver 500 Terminal Server V2.1.5 - LAT V5.1
ABC-1 LAB

Please type HELP if you need assistance

Enter username> KR
Local> CONNECT CD
Local -010- Session 1 to CD on node NEAT established

Welcome to OpenVMS V7.2

Username: USER1
Password:
Welcome to OpenVMS VAX version V7.2 on node NEAT
Last interactive login on Wednesday, 27-SEP-1999 11:25
Last non-interactive login on Wednesday, 27-SEP-1999 09:04
$
```

2. Practice the use of the Break key and various terminal server commands:

- a. Use the Break key to establish another session at your terminal.

```
$ [Break]
Local> C CD
Local -101- 1 other session(s) connected
Local -010- Session 2 to CD on node TIDY established

Welcome to OpenVMS V7.2

Username: USER1
Password:
Welcome to OpenVMS VAX version V7.2 on node TIDY
Last interactive login on Wednesday, 27-SEP-1999 15:01
Last non-interactive login on Wednesday, 27-SEP-1999 09:04
$
```

- a. Use the SHOW SESSIONS command to see your two sessions.

```
Local> SHOW SESSIONS

Port 16: KR          Local mode  Current session: 2
- Session 1: Connected Interactive CD      (NEAT)
- Session 2: Connected Interactive CD      (TIDY)
```

- b. Use terminal server commands to move back and forth between sessions.

```
Local> FORWARD
Local -012- CD session 1 resumed
$ [Break]
Local> BACKWARD
Local -012- CD session 2 resumed
$
```

- c. Use the RESUME command to resume each session.

```
Local> RESUME 2
Local -012- CD session 2 resumed
$ [Break]
Local> RESUME 1
$
```

- d. Use the LOCK command to lock your session.

```
Local> LOCK
Lock password>
Verification>
Local -019- Port 16 locked
```

- e. Enter the password to unlock your terminal.

```
Unlock password>
Local>
```

1. Log out of the system on both sessions.

```
$ LOGOUT
USER1 logged out at 27-SEP-1999 15:03:02.95
Local -218- Connection to CD terminated
Service user disconnect request
Local> FORWARD
Local -012- CD session 2 resumed
$ LOGOUT
USER1 logged out at 27-SEP-1999 15:04:23.17
Local -218- Connection to CD terminated
Service user disconnect request
Local>
```

Lab Exercise: Remote Session

1. Log in to your student account.
2. Create a remote session on the same system.
3. Log out of the remote session.
4. Create another remote session and save a log file of the session.
5. Log out of the remote session.
6. Display the log file at your terminal.

Lab Solution: Remote Session

1. Log in to your student account.

Enter your user name and password at the system prompts.

2. Create a remote session on the same system.

```
$ SET HOST 0
```

3. Log out of the remote session.

```
$ LOGOUT
```

4. Create another remote session on the same system and save a log file of the session.

```
$ SET HOST 0/LOG
```

5. Log out of the remote session.

```
$ LOGOUT
```

6. Display the log file at your terminal.

```
$ TYPE SETHOST.LOG
```

Written Exercise: Terminology

Match the definition on the left with the term below. Terms may be used more than once, and some definitions may match up with more than one term.

- | | |
|-----------------|---|
| a. [Break] | Command used to change a password. |
| b. Password | A word or phrase, not echoed when typed, that is required for access to a system. |
| c. SET PASSWORD | |

Written Solution: Terminology

__c__ Command used to change a password.

__b__ A word or phrase, not echoed when typed, that is required for access to a system.

Terms:

- a. [Break]
- b. Password
- c. SET PASSWORD

Lab Exercise: Logging In, Changing Passwords, and Logging Out

Complete the following activities at an interactive terminal.

1. Try to log in to the system using a name and password other than your own.

Observe what happens.

2. Now, try to log in to the system using your name but an incorrect password.

Keep trying, with the incorrect password.

What happens?

3. Log in to the system, using a legal user name and password.
4. Change your password using a system-generated password.
5. Change your password using a password that you create.
6. Log out of the session.

Lab Solution: Logging In, Changing Passwords, and Logging Out

1. Try to log in to the system using a name and password other than your own.

Observe what happens.

```
Welcome to OpenVMS V7.2
```

```
Username: STRANGER
```

```
Password:
```

```
User authorization failure
```

2. Now, try to log in to the system using your name but an incorrect password.

Keep trying, with the incorrect password.

What happens?

```
Welcome to OpenVMS V7.2
```

```
Username: USER1
```

```
Password:
```

```
User authorization failure
```

```
Username: USER1
```

```
Password:
```

```
User authorization failure
```

```
Username: USER1
```

```
Password:
```

```
User authorization failure
```

- After a set number of login failures, the user is not permitted to attempt another login for a period of time.
- The number of failures and the time period are determined by the system manager.

3. Log in to the system, using a legal user name and password.

```
Welcome to OpenVMS V7.2
```

```
Username: USER1
```

```
Password:
```

```
Welcome to OpenVMS VAX version V7.2 on node TIDY
```

```
Last interactive login on Wednesday, 20-SEP-1999 08:48
```

```
Last non-interactive login on Wednesday, 20-SEP-1999 08:50
```

```
$
```

4. Change your password using a system-generated password.

```
$ SET PASSWORD/GENERATE
Old password:
ayxtwiaw    ayx-twi-aw
kumtkane    kumt-kane
ejinassya   e-ji-nas-sya
rexaptjauz  rex-apt-jauz
alahaiqua   a-la-hai-qua
Choose a password from this list, or press [Return] to get a new list
New password:
Verification:
```

5. Change your password using a password that you create.

```
$ SET PASSWORD
Old password:
New password:
Verification:
$
```

6. Log out of the system.

```
$ LOGOUT
USER1      logged out at 17-SEP-1999 15:57:10.28
```

Issuing Commands

Module 2

Constructing a DCL Command

Written Exercise: DCL Command Components

Match the command line components on the left with the definitions on the right.

Command line:

```
$ TYPE MYFILE.001
```

Component	Definition
_____ \$	a. Command delimiter
_____ TYPE	b. Command parameter
_____ (space)	c. DCL command
_____ MYFILE.001	d. DCL prompt
_____ /	e. Qualifier
_____ PAGE	f. Qualifier delimiter

Written Solution: DCL Command Components

Component	Definition
__d__\$	a. Command delimiter
__c__TYPE	b. Command parameter
__a__(space)	c. DCL command
__b__MYFILE.001	d. DCL prompt
__f__ /	e. Qualifier
__e__PAGE	f. Qualifier delimiter

Using DCL to Interact with the System

Written Exercise: Understanding System Messages

Match the components of a system error message on the left with the definition on the right.

Error message:

\$ MAKE

%DCL-W-IVVERB, unrecognized command verb - check validity and s

\MAKE\

Component	Definition
_____ %	a. Facility code
_____ DCL	b. Message identification code
_____ W	c. Message text
_____ IVVERB	d. Severity level code
_____ unrecognized....	e. System message delimiter

Written Solution: Understanding System Messages

Component	Definition
__e__ %	a. Facility code
__a__ DCL	b. Message identification code
__d__ W	c. Message text
__b__ IVVERB	d. Severity level code
__c__ unrecognized....	e. System message delimiter

Lab Exercise: Error Messages

- Enter the following command line at your terminal:

```
$ PRODUCE NONESUCH.FIL
```

What happens?

- Enter the following command line at your terminal:

```
$ TYPE NONESUCH.FIL
```

What happens?

- Note the difference between the two error messages.
- Continue to experiment with entering incorrect commands.

Lab Solution: Error Messages

1. Enter the following command line at your terminal:

```
$ PRODUCE NONESUCH.FIL
```

What happens?

2. Since neither the command nor the parameter of the preceding command line exists, the operating system will display one or more error messages at your terminal:

```
$ PRODUCE NONESUCH.FIL
%DCL-W-IVVERB, unrecognized command verb - check validity and spelling
\PRODUCE\
```

Enter the following command line at your terminal:

```
$ TYPE NONESUCH.FIL
```

What happens?

The specified parameter names a file that does not exist. The operating system will display one or more error messages at your terminal:

```
$ TYPE NONESUCH.FIL
%TYPE-W-SEARCHFAIL, error searching for WORK3:[USER1]NONESUCH.FIL;
-RMS-E-FNF, file not found
```

3. Note the difference between the two commands.
 - The first error is from DCL. The verb PRODUCE does not exist.
 - The second error is in finding the file NONESUCH.FIL.
 - Notice that the facility code in the first is DCL and the facility code in the second is TYPE.
4. Continue to experiment with entering incorrect commands.

Results will vary.

Using DCL Features

Lab Exercise: Recalling and Editing Command Lines

The purpose of the exercise is to give you practice in entering commands, recalling and editing command lines, and using qualifiers.

1. Log in to the system.
2. Enter the command `SHOW TME`

What happens?

3. Enter the DCL command `SHOW USERS`
4. Enter the DCL command `SHOW USERS/FULL/OUTPUT=USERS.LIST`

Did anything appear on your screen?

5. Enter the DCL command `TYPE USERS.LIST`
6. Use the UP ARROW to recall the previous command and edit the command line to add the `/PAGE` qualifier to the `TYPE` command:

`TYPE/PAGE USERS.LIST`

Notice the difference in the display.

7. Enter the DCL command `RECALL/ALL`

What is displayed?

8. Use the `RECALL [number]` command to recall the `SHOW TME` command.

Use command line editing to correct the typing error:

`SHOW TIME`

Press Return to issue the corrected command.

Lab Solution: Recalling and Editing Command Lines

1. Log in to the system.

```
Welcome to OpenVMS VAX V7.2

Username: USER1
Password:
Welcome to OpenVMS VAX version V7.2 on node TIDY
Last interactive login on Wednesday, 27-SEP-1999 15:03
Last non-interactive login on Wednesday, 27-SEP-1999 09:04
$
```

2. Enter the command SHOW TME

What happens?

```
$ SHOW TME
%DCL-W-IVKEYW, unrecognized keyword - check validity and spelling
\TME\
```

3. Enter the DCL command SHOW USERS

```
$ SHOW USERS
VAX/VMS User Processes at 27-SEP-1999 15:13:59.77
Total number of users = 6, number of processes = 57

Username   Node      Interactive Subprocess  Batch
.
.
.
```

4. Enter the DCL command SHOW USERS/FULL/OUTPUT=USERS.LIST

```
$ SHOW USERS/FULL/OUTPUT=USERS.LIST
$
```

Did anything appear on your screen?

No. The output in this command is directed to the file USERS.LIST.

5. Enter the DCL command TYPE USERS.LIST

```
$ TYPE USERS.LIST
      VAX/VMS User Processes at 27-SEP-1999 15:18:23.41
      Total number of users = 6,  number of processes = 57

      Username      Node      Interactive  Subprocess   Batch
      .
      .
      .
```

6. Use the UP ARROW to recall the previous command and edit the command line to add the /PAGE qualifier to the TYPE command:

```
TYPE/PAGE USERS.LIST
```

Notice the difference in the display.

The /PAGE qualifier stops the output to the screen when the screen is full and waits for the Return key to be pressed before continuing.

7. Enter the DCL command RECALL/ALL

What is displayed?

```
$ RECALL/ALL
1 TYPE/PAGE USERS.LIST
2 TYPE USERS.LIST
3 SHOW USERS/FULL/OUTPUT=USERS.LIST
4 SHOW USERS/FULL
5 SHOW TME
```

8. Use the RECALL [number] command to recall the SHOW TME command.

Use command line editing to correct the typing error:

```
SHOW TIME
```

Press Return to issue the corrected command.

```
$ RECALL 5
$ SHOW TIME [Return]
27-SEP-1999 15:19:23
$
```

Naming and Storing Files

Module 3

Constructing an OpenVMS File Specification

Written Exercise: Constructing OpenVMS File Specifications

Match the file specification elements with the definitions below.

File specification:

```
TIDY::DUA3:[PERSONNEL]MONTHLY_REPORT.DATA;7
```

- | | |
|--------------|----------------------|
| a. Device | TIDY:: |
| b. Directory | DUA3: |
| c. File name | [PERSONNEL] |
| d. File type | MONTHLY_REPORT |
| e. Node name |DATA |
| f. Version | ;7 |

Written Solution: Constructing OpenVMS File Specifications

- | | | | |
|----|-----------|-----------|----------------|
| a. | Device | <u>e.</u> | TIDY:: |
| b. | Directory | <u>a.</u> | DUA3: |
| c. | File name | <u>b.</u> | [PERSONNEL] |
| d. | File type | <u>c.</u> | MONTHLY_REPORT |
| e. | Node name | <u>d.</u> | .DATA |
| f. | Version | <u>f.</u> | ;7 |

Creating and Identifying Subdirectories

Written Exercise: User Subdirectories

Next to each directory-related operation, write the letter of the OpenVMS command best suited to perform the job. You may use some commands more than once; others not at all.

- | | | | |
|----|---------------------|-------|--|
| a. | a. CREATE | | Display the name of your current directory. |
| b. | b. CREATE/DIRECTORY | | Display the contents of a directory hierarchy. |
| c. | c. DIRECTORY | | Add a directory to a directory hierarchy. |
| d. | d. SET DEFAULT | | Change your current default directory. |
| e. | e. SHOW DEFAULT | | Display the name of your current default device. |
| | | | Change your current default device. |

Written Solution: User Subdirectories

- | | |
|---------------------|--|
| a. CREATE | <u>e.</u> Display the name of your current directory. |
| b. CREATE/DIRECTORY | <u>c.</u> Display the contents of a directory hierarchy. |
| c. DIRECTORY | <u>b.</u> Add a directory to a directory hierarchy. |
| d. SET DEFAULT | <u>d.</u> Change your current default directory. |
| e. SHOW DEFAULT | <u>e.</u> Display the name of your current default device. |
| | <u>d.</u> Change your current default device. |

Lab Exercise: User Subdirectories

Be sure to perform these steps in numeric order.

1. Create a subdirectory called [.MEMO].
2. Set your default to the [.MEMO] subdirectory.
3. Obtain a directory listing of all files in the subdirectory in a file named MEMO.LIST.
4. Set your default to your login directory without typing the name of the directory.
5. Using only one command, display a directory listing of all of your files.
6. With your default still set to your login directory, display a directory listing of the files in the MEMO subdirectory.

Lab Solution: User Subdirectories

1. Create a subdirectory called [.MEMO].

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

Your login directory now contains a file called MEMO.DIR.

2. Set your default to the [.MEMO] subdirectory.

```
$ SET DEFAULT [.MEMO]
```

3. Obtain a directory listing of all files in the subdirectory in a file named MEMO.LIST.

```
$ DIRECTORY/OUTPUT=MEMO.LIST
```

4. Set your default to your login directory without typing the name of the directory.

```
$ SET DEFAULT [-]
```

5. Using only one command, display a directory listing of all of your files.

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

6. With your default still set to your login directory, display a directory listing of the files in the MEMO subdirectory.

```
$ DIRECTORY [.MEMO]
```

Displaying Information About Files

Written Exercise: Using Wildcards

Suppose your default directory contains the following files:

A.DAT;1	A.FOR;2	AREA.FOR;2	AREA.FOR;1
B.DAT;3	B.FOR;1	C.DAT;4	C.FOR;1
MAILD22.DAT;2	MAILF22.DAT;2	MAILIST.COB;1	MAILJ14.DAT;1

Refer to this list of files in responding to the following exercises.

1. List the files that are specified by the following file specifications:

a. *.FOR;2

.....

b. *.FOR

.....

c. A*.*;*

.....

d. A%%%.*.*

.....

e. %.DAT

.....

f. *.*.*

.....

2. Give a single file specification that describes the following lists of files:

a. A.DAT;1, A.FOR;2

.....

b. A.DAT;1, B.DAT;3, C.DAT;4

.....

c. MAILD22.DAT;2, MAILF22.DAT;2, MAILJ14.DAT;1

.....

d. A.DAT;1, MAILJ14.DAT;1

.....

Written Solution: Using Wildcards

1. List the files that are specified by the following file specifications:

a. *.FOR;2

A.FOR;2 AREA.FOR;2

b. *.FOR

A.FOR;2 AREA.FOR;1 AREA.FOR;2 B.FOR;1 C.FOR;1

c. A*.*;*

A.DAT;1 A.FOR;2 AREA.FOR;1 AREA.FOR;2

d. A%%%.*;*

AREA.FOR;1 AREA.FOR;2

e. %.DAT

A.DAT;1 B.DAT;3 C.DAT;4

f. *.*;*

The entire listing

2. Give a single file specification that describes the following lists of files:

a. A.DAT;1, A.FOR;2

A.*;*

b. A.DAT;1, B.DAT;3, C.DAT;4

%.DAT;* or *.DAT;*

c. MAILD22.DAT;2, MAILF22.DAT;2, MAILJ14.DAT;1

MAIL*.DAT;*

d. A.DAT;1, MAILJ14.DAT;1

*.DAT;1

Lab Exercise: DIRECTORY Command

Set your default to your login directory to perform these exercises.

1. Display a listing of the names of the files in your default directory.
2. Display the names of these files in a single column of output.
3. Display the names, creation dates, and sizes of these files.
4. Create a file called MYFILES.LIST, containing a listing of all files in all of your directories.
5. Display a full directory listing of any files with the file type LIST.

Lab Solution: DIRECTORY Command

1. Display a listing of the names of the files in your default directory.

```
$ DIRECTORY *.*  
(or)
```

```
$ DIRECTORY
```

2. Display the names of these files in a single column of output.

```
$ DIRECTORY/COLUMNS=1 *.*  
(or)
```

```
$ DIRECTORY/COLUMNS=1
```

3. Display the names, creation dates, and sizes of these files.

```
$ DIRECTORY/DATE/SIZE *.*  
(or)
```

```
$ DIRECTORY/DATE/SIZE
```

4. Create a file called MYFILES.LIST, containing a listing of all files in all of your directories.

```
$ DIRECTORY/OUTPUT=MYFILES.LIST [...]*. *  
(or)
```

```
$ DIRECTORY/OUTPUT=MYFILES.LIST [...]
```

5. Display a full directory listing of any files with the file type LIST.

```
$ DIRECTORY/FULL *.LIST  
(or)
```

```
$ DIRECTORY/FULL .LIST
```

Creating Memos, Reports, and Data Files

Module 4

Lab Exercise 1: Creating a File and Issuing Commands

Practice creating a file:

1. Use the EVE editor to create a file called TESTFILE.TEXT.
2. Type in the lines listed below, pressing **[Return]** at the end of each line.

```
This is an exercise that uses the EVE editor.  
This editor allows you to type text into a file.
```
3. Issue the TOP command.
4. Select all of the text in the file from the top to the bottom.
5. Remove the text.
6. Insert the text six times.
7. Press **[Prev Screen]** and **[Next Screen]** to move the cursor throughout the file.
8. Press **[Help]** to obtain help on the keypad keys.
9. End the editing session and save your work.

Lab Solution 1: Creating a File and Issuing Commands

Your practice session might look like this:

1. Use the EVE editor to create a file called TESTFILE.TEXT.

```
$ EDIT/TPU TESTFILE.TEXT
```

2. Type in the lines of text, pressing **[Return]** at the end of each line.
3. Issue the TOP command.

```
[Do]  
Command: TOP
```

4. Select all of the text in the file from the top to the bottom.

```
[Select]  
[Next Screen]
```

5. Remove the text.

```
[Remove]
```

6. Insert the text six times.

Press **[Insert Here]** six times.

7. Press **[Prev Screen]** and **[Next Screen]** to move the cursor throughout the file.

[Prev Screen] moves the cursor toward the beginning of the file.

[Next Screen] moves the cursor toward the end of the file.

8. Press **[Help]** to obtain help on the keypad keys.

The screen will clear and EVE will build a diagram of the keypad keys for you to study.

9. End the editing session and save your work.

```
[Do]  
Command: EXIT
```

Lab Exercise 2: Editing a File

1. Begin another editing session using TESTFILE.TEXT.

2. Modify the text of the first line to read:

`This is an example that uses the EVE editor.`

3. Insert another paragraph after the first sentence.

`This exercise is showing how to insert new text into
an existing text file.`

`By hitting Return twice I can space out paragraphs so they are
visually separated on the page.`

`Insert Return twice more after finishing this last sentence.`

4. End the editing session and save the file.
5. Enter the DIRECTORY TESTFILE.TEXT command.
6. How many files do you find? What is the difference in their file specifications?

Lab Solution 2: Editing a File

1. Begin another editing session using TESTFILE.TEXT.

```
$ EDIT/TPU TESTFILE.TEXT
```

2. Modify the text of the first line to read:

```
This is an example that uses the EVE editor.
```

- Move the cursor to the beginning of the word "exercise."
- Use the appropriate key to delete the word "exercise."
- Insert the word "example" in its place.

3. Insert another paragraph after the first sentence.

```
This exercise is showing how to insert new text into  
an existing text file.
```

```
By hitting Return twice I can space out paragraphs so they are  
visually separated on the page.
```

```
Insert Return twice more after finishing this last sentence.
```

- Press **[Return]** twice. Start typing.

4. End the editing session and save the file.

```
[Do]
```

```
Command: EXIT
```

5. Enter the DIRECTORY TESTFILE.TEXT command.

```
$ DIRECTORY TESTFILE.TEXT
```

6. How many files do you find? What is the difference in their file specifications?

- There should be two files:

```
TESTFILE.TEXT;1
```

```
TESTFILE.TEXT;2
```

- Version 2 of the file contains the edits that you just made.

Lab Exercise: Requesting Help

1. Invoke the EVE editor to edit any file.
2. Request help information about the REPLACE command.
3. Request help with the keypad.

Lab Solution: Requesting Help

1. Invoke the EVE editor to edit any file.

```
$ EDIT/TPU TESTFILE.TXT
```

2. Request help information about the REPLACE command.

Press the Do key and enter HELP REPLACE:

```
[Do]
```

```
Command: HELP REPLACE
```

3. Request help with the keypad.

Press the Help key:

```
[Help]
```

Then press any keypad key to see what help information is displayed.

Lab Exercise 1: Including a File in a Document and Issuing DCL Commands

1. Edit a new file called NEWFILE_2.TEXT.
2. Look in your directory to see what files are there without leaving EVE.
3. Include the file you created earlier (TESTFILE.TEXT).
4. Save the new file.

Lab Solution 1: Including a File in a Document and Issuing DCL Commands

1. Edit a new file called NEWFILE_2.TEXT.

```
$ EDIT/TPU NEWFILE_2.TXT
```
2. Look in your directory to see what files are there without leaving EVE.

```
[Do]  
Command: DCL DIRECTORY  
  
.  
.  
.  
[Do]  
Command: ONE WINDOW
```
3. Include the file you created earlier (TESTFILE.TEXT).

```
[Do]  
Command: INCLUDE FILE TESTFILE.TEXT
```
4. Save the new file.

```
[Do]  
Command: EXIT
```

Lab Exercise 2: Alternate Key Definitions

This is an optional exercise. If you are not familiar with the EDT editor, or if you do not currently use it at your site, you may not wish to explore the possibility of setting your EVE keypad to resemble EDT.

1. Edit the file you last created, NEWFILE_2.TXT.
2. Issue the command that will define your keypad as an EDT keypad.
3. Issue the key command that will enable you to enter an EVE command line.
4. Include the file TESTFILE.TEXT into the document again.
5. Display the keypad diagram for EDT.
6. Exit and save the file.

Lab Solution 2: Alternate Key Definitions

1. Edit the file you last created, NEWFILE_2.TXT.

```
$ EDIT/TPU NEWFILE_2.TXT
```

2. Issue the command that will define your keypad as an EDT keypad.

```
[Do]
```

```
Command: SET KEYPAD EDT
```

3. Issue the key command that will enable you to enter an EVE command line.

```
[GOLD] [KP7]
```

```
Command:
```

4. Include the file TESTFILE.TEXT into the document again.

```
Command: INCLUDE TESTFILE.TEXT
```

5. Display the keypad diagram for EDT.

```
[Help ]
```

6. Exit and save the file.

```
Press [Ctrl/Z]
```

Lab Exercise 2: Recovering an Editing Session

1. Begin an editing session, type a few commands, and then press **[Ctrl/Y]** to simulate a system interruption.
2. Back at DCL level, enter a DIRECTORY command and find the journal file in the listing. Note how the file specification was constructed.
3. Recover your editing session.

Lab Solution 2: Recovering an Editing Session

Your practice session might look like this:

1. Begin an editing session, type a few commands, and then press **[Ctrl/Y]** to simulate a system interruption.

```
$ EDIT/TPU/JOURNAL=SAMPLE.TJL SAMPLE.TXT
.
.
.
[Interrupt]
$
```

2. Back at DCL level, enter a DIRECTORY command and find the journal file in the listing. Note how the file specification was constructed.

```
$ DIRECTORY SAMPLE

Directory WORK3:[USER1]

SAMPLE.TJL;1

Total of 1 file.
$
```

3. Recover your editing session.

```
$ EDIT/TPU/RECOVER SAMPLE.TXT
.
.
.
```

Manipulating Files

Module 5

Making Copies of Files

Lab Exercise: Copying Files

1. Preparation:
 - Set default to your login directory.
 - Create a file by entering the following command:


```
$ DIRECTORY/OUTPUT=FILES.LIST
```
2. Copy FILES.LIST to a new file named DIRECTORY.LIST.
3. Copy FILES.LIST and DIRECTORY.LIST into a single file named DOUBLE.LIST
4. Copy all files with a file type of .LIST to a file with a file type of .OLD.

Lab Solution: Copying Files

1. Preparation:

- Set default to your login directory.

```
$ SET DEFAULT SYS$LOGIN
```

- Create a file by entering the following command:

```
$ DIRECTORY/OUTPUT=FILES.LIST
```

2. Copy FILES.LIST to a new file named DIRECTORY.LIST.

```
$ COPY FILES.LIST DIRECTORY.LIST
```

3. Copy FILES.LIST and DIRECTORY.LIST into a single file named DOUBLE.LIST

```
$ COPY FILES.LIST,DIRECTORY.LIST DOUBLE.LIST
```

4. Copy all files with a file type of .LIST to a file with a file type of .OLD.

```
$ COPY *.LIST *.OLD
```

Displaying the Contents of Files

Lab Exercise: Typing Files

1. Use `DIRECTORY/OUTPUT=filename` to create a file named `FILES.LIST` in your default login directory.
2. Display this file on your terminal screen.
3. Display this file one screen at a time.
4. Create a subdirectory called `[.DOWN-ONE]` and set your default to it.
5. Display the file again.

Lab Solution: Typing Files

1. Use DIRECTORY/OUTPUT=filename to create a file named FILES.LIST in your default login directory.

```
$ DIRECTORY/OUTPUT=FILES.LIST
```

2. Display this file on your terminal screen.

```
$ TYPE FILES.LIST
```

3. Display this file one screen at a time.

```
$ TYPE/PAGE FILES.LIST
```

4. Create a subdirectory called [.DOWN-ONE] and set your default to it.

```
$ CREATE/DIRECTORY [.DOWN_ONE]  
$ SET DEFAULT [.DOWN_ONE]
```

5. Display the file again.

```
$ TYPE [-]FILES.LIST
```

Printing Files

Lab Exercise: Printing Files

1. Use the DIRECTORY command to create a text file named DIRECTORY.TXT and the SHOW QUEUE/DEVICES command to create a text file named PRNT.TXT.
2. Print DIRECTORY.TXT.
3. Use a single PRINT command to print two copies of DIRECTORY.TXT.
4. Display a list of all printer queues on your system and all jobs in the queues.
5. Print DIRECTORY.TXT and PRNT.TXT so you get two copies of the first file and three copies of the second file.
6. Send DIRECTORY.TXT to the printer queue, requesting that the system notify you when the job completes.
7. Send the file PRNT.TXT to the printer using the /HOLD qualifier. Display the queue status of the job waiting to be printed. Delete this job from the queue. Show entry again.

Lab Solution: Printing Files

1. Use the DIRECTORY command to create a text file named DIRECTORY.TXT and the SHOW QUEUE/DEVICES command to create a text file named PRNT.TXT.

```
$ DIRECTORY/OUTPUT=DIRECTORY.TXT
$ SHOW QUEUE/DEVICES/OUTPUT=PRNT.TXT
```

2. Print DIRECTORY.TXT.

```
DIRECTORY (queue SYS$PRINT, entry 330) started on LPA0
```

3. Use a single PRINT command to print two copies of DIRECTORY.TXT.

```
$ PRINT/COPIES=2 DIRECTORY.TXT
Job DIRECTORY (queue SYS$PRINT, entry 331) started on LPA0
```

4. Display a list of all printer queues on your system and all jobs in the queues.

```
$ SHOW QUEUE/DEVICE=PRINTER/ALL
```

5. Print DIRECTORY.TXT and PRNT.TXT so you get two copies of the first file and three copies of the second file.

```
$ PRINT DIRECTORY.TXT/COPIES=2,PRNT.TXT/COPIES=3
Job DIRECTORY (queue SYS$PRINT, entry 337) started on LPA0
```

6. Send DIRECTORY.TXT to the printer queue, requesting that the system notify you when the job completes.

```
$ PRINT/NOTIFY DIRECTORY.TXT
Job DIRECTORY (queue SYS$PRINT, entry 357) started on LPA0
```

```
Job DIRECTORY (queue SYS$PRINT, entry 357) completed
$
```

7. Send the file PRNT.TXT to the printer using the /HOLD qualifier. Display the queue status of the job waiting to be printed. Delete this job from the queue. Show entry again.

```
$ PRINT PRNT.TXT/HOLD
Job PRNT (queue SYS$PRINT, entry 36) holding
$ SHOW ENTRY
Entry Jobname      Username    Blocks Status
-----
36 PRNT          USER1        13 Holding
On generic printer queue SYS$PRINT
$ DELETE/ENTRY=36

Job PRNT (queue SYS$PRINT, entry 36) terminated with error status
%JBC-F-JOBDELETE, job deleted before execution
%DELETE-I-DELETED, entry 36 aborting or deleted
$ SHOW ENTRY
%JBC-E-NOSUCHENT, no such entry
$
```

Renaming Files

Lab Exercise: Renaming Files

1. Set default to your login directory.
2. Rename FILES.LIST to USERS.LIST.
3. Request a directory listing of FILES.LIST. What do you find?
4. Rename all versions of DIRECTORY.LIST to USERS.LIST.
5. Request a directory of DIRECTORY.LIST. What do you find?

Lab Solution: Renaming Files

1. Set default to your login directory.

```
$ SET DEFAULT SYS$LOGIN
```

2. Rename FILES.LIST to USERS.LIST.

```
$ RENAME FILES.LIST USERS.LIST
```

3. Request a directory listing of FILES.LIST. What do you find?

If you had more than one version of FILES.LIST, there will still be a file or files named that on disk.

4. Rename all versions of DIRECTORY.LIST to USERS.LIST.

```
$ RENAME DIRECTORY.LIST;* USERS.LIST
```

5. Request a directory of DIRECTORY.LIST. What do you find?

```
$ DIRECTORY FILES.LIST
%DIRECT-W-NOFILES, no files found
$
```

There should be no files of that name left on disk. By specifying a wildcard in the version number, RENAME will rename all versions of the specified files.

Removing Files from Disk

Lab Exercise: Deleting and Purging Files

1. Use the DIRECTORY command to make at least three versions of a file named DIR.TXT.
2. Delete all versions of all files except the two latest versions.
3. Purge all files in the directory.
4. How could you save selected versions of files in the preceding command?
5. Delete DIR.TXT.

Lab Solution: Deleting and Purging Files

1. Use the DIRECTORY command to make at least three versions of a file named DIR.TXT.


```
$ DIRECTORY/OUTPUT=DIR.TXT  
$ DIRECTORY/OUTPUT=DIR.TXT  
$ DIRECTORY/OUTPUT=DIR.TXT  
$
```
2. Delete all versions of all files except the two latest versions.


```
$ PURGE/KEEP=2  
$
```
3. Purge all files in the directory.


```
$ PURGE
```
4. How could you save selected versions of files in the preceding command?
Use the PURGE/CONFIRM command.
5. Delete DIR.TXT.


```
$ DELETE DIR.TXT;*
```

Communicating with Other Users

Module 6

Using the Mail Utility

Lab Exercise 1: Sending and Reading Mail Messages

1. Invoke the Mail utility and send a single mail message to two other users.
2. Send yourself a mail message.
3. Read the message.
4. Forward it to yourself.
5. Edit and send a message to yourself.

Lab Solution 1: Sending and Reading Mail Messages

1. Invoke the Mail utility and send a single mail message to two other users.

```
$ MAIL
MAIL> SEND
To: LAUREL,HARDY
Subj: Test message
Enter your message below. Press [Ctrl/Z] when complete, or...

.
.
.
[ Exit ]
MAIL> EXIT

$
```

2. Send yourself a mail message.

```
$ MAIL
MAIL> SEND
To: USER1
Subj: Test Message
Enter your message below. Press [Ctrl/Z] when complete, or...

I am sending this mail message to myself...

[Exit]
MAIL> EXIT

$
```

3. Read the message.

```
$ MAIL
MAIL> READ

#1      20-SEP-1999  11:12:45.34      NEWMAIL
From:   NEAT::USER1
To:     NEAT::USER1
Subj:   Test Message

I am sending this mail message to myself...

New mail on node NEAT from NEAT::USER1 (1:04:56)
MAIL>
```

4. Forward it to yourself.

```
MAIL> FORWARD
To:    NEAT::USER1
Subj:   Forwarded message
```

```
New mail on node NEAT from NEAT::USER1 (11:04:56)
MAIL>
```

5. Edit and send a message to yourself.

```
$ MAIL> SEND/EDIT
To:    NEAT::USER1
Subj:   Edited message
```

```
[End of file]
```

You can now see the screen view of the text you type, as you are seeing this text. When you have completed your message, press **[Ctrl/Z]** to send the message, or **[Do]** and enter QUIT to quit and not send the message.

```
New mail on node NEAT from NEAT::USER1 (12:04:56)
MAIL>
```

Lab Exercise 2: MAIL Commands

1. Invoke the Mail utility and send a single mail message to some other user. If you do not know other users on your system, send a mail message to yourself.
2. Display a list of your mail messages.
3. Read only the second mail message.
4. Delete the first mail message.
5. Exit from the Mail utility.

Lab Solution 2: MAIL Commands

1. Invoke the Mail utility and send a single mail message to some other user. If you do not know other users on your system, send a mail message to yourself.

```
$ MAIL
MAIL> SEND
To:   STUDENT1,STUDENT2,STUDENT3
Subj: Test message
Enter your message below. Press CTRL/Z when complete, or...

.
.
.
[ Exit ]
```

2. Display a list of your mail messages.

```
MAIL> DIRECTORY
```

3. Read only the second mail message.

```
MAIL> READ 2
```

4. Delete the first mail message.

```
MAIL> DELETE 1
```

5. Exit from the Mail utility.

```
MAIL> EXIT
```

Lab Exercise 3: Working with Folders

1. Send two Mail messages to yourself.
2. Read the first message and delete it.
3. Read the second message and move it to a folder named TEST. Select the TEST folder and check to see if the message is there.
4. List the folders you have.
5. List the messages in the WASTEBASKET folder.
What is in this folder?
6. Exit and then enter MAIL again.
7. Select the WASTEBASKET folder.
What happens?
8. Select the folder TEST.
What happens?
9. Delete a message from this folder.
10. List the contents of the WASTEBASKET folder.
11. Move the contents of the WASTEBASKET folder to the TEST folder.
12. Exit from MAIL.

Lab Solution 3: Working with Folders

1. Send two Mail messages to yourself.

```
MAIL> SEND
To:   STUDENT1
Subj:  Test message
Enter your message below. Press CTRL/Z when complete...

.
.
.
[ Exit ]
```

(Do this twice)

2. Read the first message and delete it.

```
MAIL> DIRECTORY/NEW
MAIL> READ 1

.
.
.
MAIL> DELETE
```

3. Read the second message and move it to a folder named TEST. Select the TEST folder and check to see if the message is there.

```
MAIL> READ 2

.
.
.
MAIL> MOVE TEST
Folder TEST does not exist.
Do you want to create it (Y/N, default is N)? Y
%MAIL-I-NEWFOLDER, folder TEST created

MAIL> SELECT TEST
%MAIL-I-SELECTED, 1 message selected
```

4. List the folders you have.

```
MAIL> DIRECTORY/FOLDER
```

5. List the messages in the WASTEBASKET folder.

What is in this folder?

```
MAIL> DIRECTORY WASTEBASKET
```

The WASTEBASKET folder contains the message that was deleted.

6. Exit and then enter MAIL again.

```
MAIL> EXIT
```

```
$ MAIL
```

```
MAIL>
```

7. Select the WASTEBASKET folder.

What happens?

```
MAIL> SELECT WASTEBASKET
```

```
%MAIL-E-NOTEXIST, folder WASTEBASKET does not exist
```

8. Select the folder TEST.

What happens?

```
MAIL> SELECT TEST
```

```
%MAIL-I-SELECTED, 1 message selected
```

9. Delete a message from this folder.

```
MAIL> DELETE 1
```

10. List the contents of the WASTEBASKET folder.

```
MAIL> DIRECTORY WASTEBASKET
```

The WASTEBASKET folder contains the message that was just deleted from the TEST folder.

11. Move the contents of the WASTEBASKET folder to the TEST folder.

```
MAIL> READ 1
```

```
.  
. .  
.
```

```
MAIL> MOVE TEST
```

```
Folder TEST does not exist.
```

```
Do you want to create it (Y/N, default is N)? Y
```

```
%MAIL-I-NEWFOLDER, folder TEST created
```

12. Exit from MAIL.

```
MAIL> EXIT
```

Lab Exercise 4: Working with Files

1. Create a distribution list of all of the students in the class.
2. Create a file containing a short greeting.
3. Enter the Mail utility and send your greeting file to all of the students in your class using the distribution list you created.

Lab Solution 4: Working with Files

1. Create a distribution list of all of the students in the class.

```
$ CREATE CLASS.DIS
STUDENT1
STUDENT2
STUDENT3
[ Exit ]
$
```

Each line of the file should be the user name of one of the students.

This solution shows the use of the DCL CREATE command. Type Ctrl/Z (which echoes as [**Exit**]) when you are finished entering text. You can, of course, use an editor to create the file.

2. Create a file containing a short greeting.

```
$ CREATE GREETING.TXT
Greetings from a fellow student.
I am practicing the use of a distribution list.
[ Exit ]
$
```

The CREATE command was used here. For a more complicated message, use an editor.

3. Enter the Mail utility and send your greeting file to all of the students in your class using the distribution list you created.

```
$ MAIL
MAIL> SEND GREETING.TXT
To: @CLASS.DIS
Subj: Greetings
```

Because TXT is the default file type for a file being sent by the Mail utility, you could type **SEND GREETING** instead.

Because DIS is the default file type for a distribution list, you could type **@CLASS** instead.

Getting Help

Module 7

Using Online Help

Lab Exercise 1: HELP Command

Use the Help facility to:

1. Find a description of the login procedure.
2. Find a description of the /FULL qualifier of the LOGOUT command.
3. Find a description of the TIME option of the SHOW command.

Lab Solution 1: HELP Command

1. Find a description of the login procedure.

```
$ HELP LOGIN
```

2. Find a description of the /FULL qualifier of the LOGOUT command.

```
$ HELP LOGOUT/FULL
```

3. Find a description of the TIME option of the SHOW command.

```
$ HELP SHOW TIME
```

If you did not know that TIME was the actual name of the option for the SHOW command, you could have typed:

```
$ HELP SHOW
```

You then could have selected TIME from the additional information subtopic list that HELP would have displayed.

Lab Exercise 2: More Complicated Investigation

Here is a more advanced practice problem for using the Help facility.

Find out how to change your DCL prompt to match your name, and do so. (Hint: what general category of commands changes characteristics of things?)

Lab Solution 2: More Complicated Investigation

```
$ HELP SET
```

```
SET
```

```
.  
.
.
```

```
SET Subtopic? PROMPT
```

```
.  
.
.
```

```
$ SET PROMPT="Meg> "  
Meg>
```

Lab Exercise: Locating Information

Do this exercise if there are documentation volumes available for your use.

1. Look up "command lines" in the Master Index of the OpenVMS Documentation Set.
2. Locate the section listed under "editing."
3. Go to that manual and find the pages on editing.
4. Go back to the Master Index and look up "editing command lines."
5. Find those pages in the appropriate manual.
6. Experiment looking up topics covered in this course to get accustomed to using the Master Index and the manuals.

(No solution is given for this exercise.)

Protecting Your Data

Module 8

Using File Protection

Written Exercise 1: File Protection

Next to each operation, write the letter of the DCL command best suited to perform the job. You may use some commands more than once; others not at all.

You may supply more than one command for an operation.

- | | | | |
|----|----------------------------|-------|--|
| a. | DIRECTORY/PROTECTION | | Change the protection code of a file. |
| b. | DIRECTORY/SECURITY | | Display all protection information for a file. |
| c. | SET
SECURITY/PROTECTION | | Display only the protection code of a file. |
| d. | SHOW PROTECTION | | Display your default protection code. |
| e. | SHOW SECURITY | | |

Written Solution 1: File and Directory Protection

Next to each operation, write the letter of the DCL command best suited to perform the job. You may use some commands more than once; others not at all.

You may supply more than one command for an operation.

- | | | |
|-------------------------------|------------|--|
| a. DIRECTORY/PROTECTION | <u>c.</u> | Change the protection code of a file. |
| b. DIRECTORY/SECURITY | <u>b,e</u> | Display all protection information for a file. |
| c. SET
SECURITY/PROTECTION | <u>a.</u> | Display only the protection code of a file. |
| d. SHOW PROTECTION | <u>d.</u> | Display your default protection code. |
| e. SHOW SECURITY | | |

Written Exercise 2: UIC Protection

Suppose you wish to access a file whose protection code is:

S:RWE,O:RWED,G:RE,W:R

Complete the following chart showing the type of access you can have to this file if your UIC is the one specified.

File's UIC	Your UIC	Type of Access You Can Have
[132,10]	[2,25]	Read, Write, Execute (you are "System")
[5,15]	[100,10]	Read (you are "World")
[150,10]	[150,11]	Read, Execute (you are "Group")
[200,5]	[100,5]	
[300,5]	[300,10]	
[105,6]	[5,16]	
[10,5]	[105,6]	
[202,15]	[202,15]	
[307,5]	[2,25]	
		

Written Solution 2: UIC Protection

File's UIC	Your UIC	Type of Access You Can Have
[132,10]	[2,25]	Read, Write, Execute (you are "System")
[5,15]	[100,10]	Read (you are "World")
[150,10]	[150,11]	Read, Execute (you are "Group")
[200,5]	[100,5]	Read (you are "World")
[300,5]	[300,10]	Read, Execute (you are "Group")
[105,6]	[5,16]	Read, Write, Execute (you are "System")
[10,5]	[105,6]	Read (you are "World")
[202,15]	[202,15]	Read, Write, Execute, Delete (you are "Owner")
[307,5]	[2,25]	Read, Write, Execute (you are "System")

Lab Exercise 1: Protecting Files

1. Display the protection code for some file in your directory.
2. Change the protection code for this file to allow only read access for all users. Display the protection code to verify the change.
3. Try to delete this file. What happened and why?
4. Display your default protection code.
5. Change your default protection code to allow the following access:
 - System, Group, and World can only read
 - Owner has full access
6. Create a new file named NEWFILE.TXT. What protection code does this new file have, and why?

Lab Solution 1: Protecting Files

1. Display the protection code for some file in your directory.

```
$ DIRECTORY/PROTECTION TESTFILE.TEXT
```

```
Directory WORK3:[USER1]
```

```
TESTFILE.TEXT;1      (RWED,RWED,RE, )
```

```
Total of 1 file.
```

2. Change the protection code for this file to allow only read access for all users.

```
$ SET SECURITY/PROTECTION=(S:R,O:R,G:R,W:R) TESTFILE.TEXT
```

Display the protection code to verify the change.

```
$ DIRECTORY/PROTECTION TESTFILE.TEXT
```

```
Directory WORK3:[USER1]
```

```
TESTFILE.TEXT;1      (R,R,R,R)
```

```
Total of 1 file.
```

```
$
```

3. Try to delete this file. What happened and why?

```
$ DELETE TESTFILE.TEXT;
```

```
%DELETE-W-FILNOTDEL, error deleting WORK3:[USER1]TESTFILE.TEXT;1  
-RMS-E-PRV, insufficient privilege or file protection violation  
$
```

The file is protected against deletion.

4. Display your default protection code.

```
$ SHOW PROTECTION
```

```
SYSTEM=RWED, OWNER=RWED, GROUP=RE, WORLD=NO ACCESS  
$
```

5. Change your default protection code to allow the following access:

- System, Group, and World can only read
- Owner has full access

```
$ SET PROTECTION=(S:R,O:RWED,G:R,W:R)/DEFAULT
```

6. Create a new file named NEWFILE.TXT. What protection code does this new file have, and why?

```
$ DIRECTORY/OUTPUT=NEWFILE.TXT  
$ DIRECTORY/PROTECTION NEWFILE.TXT
```

```
Directory WORK3:[USER1]
```

```
NEWFILE.TXT;1      (R,RWED,R,R)
```

```
Total of 1 file.  
$
```

The file was created with the new default protection code.

Lab Exercise 2: Deleting a Subdirectory

1. Create a subdirectory called [.TEST] to use in this exercise.
2. Put at least one file in the [.TEST] subdirectory.
3. Use the DIRECTORY [...] command to verify that [.TEST] is part of your directory structure.
4. Delete the [.TEST] subdirectory.
5. With your default set to your login directory, enter the following command:

```
DIRECTORY [...]
```

What has changed?

Lab Solution 2: Deleting a Subdirectory

1. Create a subdirectory called [.TEST] to use in this exercise.

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

2. Put at least one file in the [.TEST] subdirectory.

Commands you might use:

```
$ DIRECTORY/OUTPUT=[.TEST]DIR.TEXT  
$ COPY/EXCLUDE=*.DIR *.* [.TEST]*.*  
$ SHOW TERMINAL/OUTPUT=[.TEST]TERMINAL.DAT
```

3. Use the DIRECTORY [...] command to verify that [.TEST] is part of your directory structure.

The [.TEST] subdirectory and its contents will be included in the listing.

4. Delete the [.TEST] subdirectory.

Remove all files from [.TEST], either by deleting them or renaming them to another directory.

```
$ DELETE [.TEST]*.*;*
(or)
$ RENAME [.TEST]*.*;* [.MEMO]*.*;*
```

Give yourself D (delete) access to the TEST.DIR file.

```
$ SET SECURITY/PROTECTION=0:RWED TEST.DIR
```

Delete the directory file.

```
$ DELETE TEST.DIR;
```

5. With your default set to your login directory, enter the following command:

```
DIRECTORY [...]
```

What has changed?

The [.TEST] subdirectory no longer exists.

Saving Copies of Files

Lab Exercise: Backing Up and Restoring Your Files

1. Make sure your directory contains at least three files that are not directory files.
2. Back up your non-directory files to a saveset called PRACTICE.BCK. Use the /EXCLUDE qualifier to exclude directory files from the operation, and the /LOG qualifier to request a log of the operation.
3. List the contents of the saveset.
4. Create a subdirectory called [.PRACTICE]. Restore one file from the saveset to this subdirectory. Request a log of the operation.

Lab Solution 1: Backing Up and Restoring Your Files

1. Make sure your directory contains at least three files that are not directory files.

Commands you might use to create more files:

```
$ DIRECTORY/OUTPUT=filename.type  
$ COPY filename.type *.*
```

2. Back up your non-directory files to a saveset called PRACTICE.BCK.

Use the /EXCLUDE qualifier to exclude directory files from the operation, and the /LOG qualifier to request a log of the operation.

```
$ BACKUP/EXCLUDE=*.DIR/LOG *.* PRACTICE.BCK /SAVE_SET
```

3. List the contents of the saveset.

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

4. Create a subdirectory called [.PRACTICE]. Restore one file from the saveset to this subdirectory. Request a log of the operation.

```
$ CREATE/DIRECTORY [.PRACTICE]  
$ BACKUP/LOG  
_From: PRACTICE.BCK/SAVE_SET/SELECT=SAMPLE.TXT  
_To: [.PRACTICE]*.*
```

Customizing Your Working Environment

Module 9

Using Logical Names

Lab Exercise: Logical Names

1. Create a logical name MEMOS to represent your [.MEMO] subdirectory.
2. Use the SHOW LOGICAL command to display the equivalence for this logical name.
3. Enter the command DIRECTORY MEMOS. What happens?
4. Use the logical name MEMOS to set your default to the [.MEMO] subdirectory.
5. Enter the command SHOW LOGICAL SYS\$LOGIN.
6. Use the logical name SYS\$LOGIN to set your default to your login directory.
7. Delete the logical name MEMOS.
8. Enter the command DIRECTORY MEMOS. What happens?

Lab Solution: Logical Names

1. Create a logical name MEMOS to represent your [.MEMO] subdirectory.

```
$ ASSIGN [STUDENT1.MEMO] MEMOS
```

(or)

```
$ DEFINE MEMOS [STUDENT1.MEMO]
```

Note that if you do not include the name of your login directory in the logical name definition, the logical names will only work when your default is set to your login directory.

2. Use the SHOW LOGICAL command to display the equivalence for this logical name.

```
$ SHOW LOGICAL MEMOS
"MEMOS" = "[STUDENT1.MEMO]" (LNM$PROCESS_TABLE)
$
```

3. Enter the command DIRECTORY MEMOS. What happens?

A directory listing of the files in the [.MEMO] subdirectory is displayed.

4. Use the logical name MEMOS to set your default to the [.MEMO] subdirectory.

```
$ SET DEFAULT MEMOS
$ SHOW DEFAULT
WORK3:[STUDENT1.MEMO]
$
```

5. Enter the command SHOW LOGICAL SYS\$LOGIN.

```
$ SHOW LOGICAL SYS$LOGIN
"SYS$LOGIN" = "WORK3:[STUDENT1]" (LNM$JOB_808F4710)
```

Note that the equivalence includes both the login directory and the login device.

6. Use the logical name SYS\$LOGIN to set your default to your login directory.

```
$ SET DEFAULT SYS$LOGIN
$ SHOW DEFAULT
  WORK3:[STUDENT1]
$
```

7. Delete the logical name MEMOS.

```
$ DEASSIGN MEMOS
```

8. Enter the command DIRECTORY MEMOS. What happens?

```
$ DIRECTORY MEMOS
%DIRECT-W-NOFILES, no files found
$
```

MEMOS is treated as a file name, not a logical name.

There are no files with the name MEMOS in this directory.

Using Symbols

Lab Exercise: Creating Symbols

1. Create a global symbol that equates the word HOME with setting your default to your default directory at login.
2. Set your default directory to some other directory and type the command HOME at the \$ prompt. Display your default directory. What happened?
3. Create local symbols that allow you to add A and B to get 6.
4. Create an integer symbol, T, that is equal to the number 2.
5. Create a string symbol, TWO, that is equal to the character 2.
6. Compare the equivalence of T and TWO. What do you find?
7. Type the command: `$ SUM = TWO + TWO` . Then display the value of SUM using the SHOW SYMBOL command. What do you get?

Lab Solution: Creating Symbols

1. Create a global symbol that equates the word HOME with setting your default to your default directory at login.

```
$ HOME ::= SET DEFAULT SYS$LOGIN
```

2. Set your default directory to some other directory and type the command HOME at the \$ prompt. Display your default directory. What happened?

When you type the word HOME, DCL translates the symbol HOME and substitutes the value SET DEFAULT SYS\$LOGIN for it. In effect, you typed SET DEFAULT SYS\$LOGIN.

3. Create local symbols that allow you to add A and B to get 6.

```
$ A = 2
$ B = 4
$ C = A+B
$ SHOW SYMBOL C
```

```
C = 6  Hex = 00000006 Octal = 0000000006
```

You will notice that the SHOW SYMBOL command displays all available values for the symbol C: decimal, hexadecimal, and octal. For now, the decimal value is all that concerns us.

4. Create an integer symbol, T, that is equal to the number 2.

```
$ T = 2
```

5. Create a string symbol, TWO, that is equal to the character 2.

```
$ TWO := 2
```

6. Compare the equivalence of T and TWO. What do you find?

```
$ SHOW SYMBOL T
T = 2  Hex = 00000002 Octal = 0000000002
$ SHOW SYMBOL TWO
TWO = "2"
```

The first symbol has an integer value. The second has a character string value.

7. Type the command: `$ SUM = TWO + TWO` . Then display the value of `SUM` using the `SHOW SYMBOL` command. What do you get?

```
$ SHOW SYMBOL SUM
```

```
SUM = "22"
```

What you are adding here are two character strings, 2 and 2. If you added the letters T and W and O, you would get TWO. This example did the same process, since TWO was created as a character string, not as an integer.

Using Keypad Definitions

Lab Exercise: Defining a Key to Recall Commands

1. Define the **[PF3]** key to display the last 255 commands issued without having to press **[Return]** .
2. Define the **[PF4]** key to allow you to select one of the recalled commands.
3. Test these keys.

Lab Solution: Defining a Key to Recall Commands

1. Define the **[PF3]** key to display the last 255 commands issued without having to press **[Return]**.

```
$ DEFINE/KEY PF3 "RECALL/ALL"/TERMINATE/NOECHO/NOLOG
```

2. Define the **[PF4]** key to allow you to select one of the recalled commands.

```
$ DEFINE/KEY PF4 "RECALL"/NOTERMINATE/ECHO/NOLOG
```

3. Test these keys.

```
❶ $
❷ 1 DEFINE/KEY PF4 "RECALL"/NOTERMINATE/ECHO/NOLOG
2 DEFINE/KEY PF3 "RECALL/ALL"/TERMINATE/NOECHO/NOLOG
3 SET DEFAULT [SMITH.DOCUMENT]
4 SHOW TIME

.
.
.

❸ $ RECALL 4
❹ $ SHOW TIME
20-SEP-1999 09:44:15
$
```

Notes on the practice session:

- ❶ Type the PF3 key without pressing Return. The command does not echo.
- ❷ The previous commands (up to 255) are listed on the screen.
- ❸ Press the PF4 key. The system displays RECALL and waits for you to select a command by number. Press the number 4 to recall the fourth command in the recall buffer.
- ❹ The SHOW TIME command is redisplayed. Press [Return] to issue this command.

Lab Exercise: Writing a LOGIN.COM

1. Write a LOGIN.COM that performs the following actions:
 - a. Defines a logical name that points to one of your subdirectories.
 - b. Defines global symbols to be used as command synonyms. The command synonyms should perform the following actions:
 - 1) Set default
 - 2) Display your current directory
 - 3) Set your default to your login disk and directory
 - 4) Redefine the DELETE command so it logs its actions
 - c. Enables the terminal for line editing
 - d. Defines the following terminal keys:
 - ♦ PF1 = "SET TERMINAL/APPLICATION_KEYPAD"
 - ♦ PF4 = "RECALL/ALL"
 - ♦ COMMA = "DIRECTORY"
 - ♦ And any other key definitions that you want
 - e. Displays the current default device and directory
2. Test your LOGIN.COM interactively to make sure it does not generate errors.
3. Log out and log back in. Have your customizations taken effect?
4. Log out and log back in using the /NOCOMMAND qualifier with your user name. Are your customizations in effect?

Lab Solution: Writing a LOGIN.COM

1. Write a LOGIN.COM that performs the following actions:

```
$! LOGIN.COM
$
$! Assign logical name to subdirectory
❶ $ ASSIGN SYS$LOGIN_DEVICE:[STUDENT1.MEMO] MEMOS
$
❷ $! Create global symbols to be used as command synonyms
$ TO == "SET DEFAULT" ! Reset default
❸ $ WHE*RE == "SHOW DEFAULT" ! Display current directory
$! Resets default to login values
$ HOME == "SET DEFAULT SYS$LOGIN"
$ WHEN == "SHOW TIME" ! Display current date and time
$ DEL*ETE == "DELETE/LOG" ! Log any delete operations
$ ! Set up for line editing
❹ $ SET TERMINAL/LINE_EDITING/INSERT
$
$! Define some terminal keys
❺ $ DEFINE/KEY PF1 "SET TERMINAL/APPLICATION_KEYPAD"/TERMINATE
$ DEFINE/KEY PF4 "RECALL/ALL"/TERMINATE
$ DEFINE/KEY COMMA "DIRECTORY"/TERMINATE
$ DEFINE/KEY MINUS "SET DEFAULT [-]"/TERMINATE
❻ $ DEFINE/KEY PF2 "SHOW KEY/ALL"/TERMINATE/NOECHO
$ DEFINE/KEY PF3 -
"MAIL/EDIT=(FORWARD,SEND,REPLY=EXTRACT)"/TERMINATE/NOECHO
$
$ WHERE ! Display current default
$
❼ $ EXIT ! End in an orderly way
```

Notes on the sample LOGIN.COM:

- ❶ You could also use:
\$ DEFINE MEMOS SYS\$LOGIN_DEVICE:[STUDENT1.MEMO]
Use SYS\$LOGIN_DEVICE rather than a device name, so if your login directory is moved to another device your LOGIN.COM will not have to be modified.
- ❷ Make sure these are **global** symbols. Local symbols will not remain after LOGIN.COM finishes executing.
- ❸ Note the abbreviation point, indicated by the asterisk (*).
- ❹ This command changes the line editing default from overstrike mode to insert mode.
- ❺ The use of /TERMINATE means that you do not have to press Return to enter the command after you press the defined key.
- ❻ The use of /NOECHO means that you will not see the keypad definition on your screen when you press the defined key.
- ❼ This is the "proper" way to end a command procedure.

2. Test your LOGIN.COM interactively to make sure it does not generate errors.

```
$ @LOGIN
%DCL-I-DEFKEY, DEFAULT key PF1 has been defined
%DCL-I-DEFKEY, DEFAULT key PF4 has been defined
%DCL-I-DEFKEY, DEFAULT key COMMA has been defined
%DCL-I-DEFKEY, DEFAULT key MINUS has been defined
%DCL-I-DEFKEY, DEFAULT key PF2 has been defined
%DCL-I-DEFKEY, DEFAULT key PF3 has been defined
  WORK3:[STUDENT1]
$
```

- To suppress the "DEFKEY" messages, include /NOLOG in your DEFINE/KEY commands.
3. Log out and log back in. Have your customizations taken effect?
 - Does DIRECTORY MEMOS list the files in your MEMO subdirectory?
 - If any symbols do not work, check to be sure that you created them with two equal signs (==).
 - If any key definitions do not work, press PF1 or enter the SET TERMINAL/APPLICATION_KEYPAD command and try again.
 4. Log out and log back in using the /NOCOMMAND qualifier with your user name. Are your customizations in effect?

```
Username: STUDENT1/NOCOMMAND
Password:
  Welcome to OpenVMS VAX version E7.0 on node TIDY
  Last interactive login on Monday, 20-SEP-1999 13:16
  Last non-interactive login on Monday, 20-SEP-1999 09:05
$ DIRECTORY MEMOS
%DIRECT-W-NOFILES, no files found
$ WHERE
%DCL-W-IVVERB, unrecognized command verb - check validity and spelling
```

- No, because LOGIN.COM did not execute.