Chapter Four

NAMING AND MAINTAINING FILES

Introduction

Almost every operation that a user performs in the course of a terminal session involves the use of files. To use a file, you must be able to specify its name and location. When you have created many files, it is helpful to organize them into groups so that they are easily accessed. These groups are called directories in OpenVMS.

This lesson discusses the creation and use of OpenVMS file specifications and the OpenVMS directory structure.

Objectives

To name and store files, a user should be able to:

- Construct an OpenVMS file specification.
- Create and identify subdirectories.
- Obtain a listing of files.

Constructing an OpenVMS File Specification

Overview

When you create or name a file, you must specify certain information so the system can locate and identify it. This information is called a ***file specification***.

This section discusses OpenVMS file specifications.

OpenVMS File Specifications

A full file specification completely describes the access path that the operating system uses to locate and identify a file. The following illustrates the structure of a full file specification:

NODE::DEVICE:[DIRECTORY]FILENAME.TYPE;VERSION

This full file specification is repeated on the top of the next page, followed by a table which describes each of its components.

NODE::DEVICE:[DIRECTORY]FILENAME.TYPE;VERSION

This component...	Supplies this information
Node	What system the file is located on in the network. A maximum of six characters can be used. In the OpenVMS cluster systems and networks, files do not have to be on the same computer system to be used by a command or a program.
Device	The device or disk where the file physically resides. Many systems have several disks or storage areas where files are kept.
Directory	The directory that contains a file. The OpenVMS operating system provides a hierarchical method of organizing a storage area on disk called directories. These directories identify locations on a specific disk.
File Name	The actual name of a file. File names can have from 1 to 39 characters. Valid characters are: A through Z (always treated as uppercase); 0 through 9; underscore (_); hyphen (-); and dollar sign ($). The dollar sign is generally avoided because of possible confusion with names generated or used by the OpenVMS operating system.
File Type	Information about the contents of the file, preceded by a period (.). Many types of files have default file types that allow users to immediately identify the function of a file. Examples include: • Command procedures --- .COM • Executable Images --- .EXE
Version	A number preceded by a semicolon (;). It specifies the version number of a file relative to other copies of the same file. The highest version number possible is 32767.

Figure 4-1 - Parts of an OpenVMS File Specification

Sample File Specifications

A simple file specification consists of a file name and file type.
```
   FEBRUARY_REPORT.TXT
```

A complete file specification consists of a file name and file type plus other information that locates and identifies the file.
```
   NEAT::DUA0:[SMITH]FEBRUARY_REPORT.TXT;3
```

Guidelines For Naming Files

Suggested guidelines for naming files include:

- Use a file name that reflects the contents or usage of the file.
- Use standard default file types.
- Avoid the use of the dollar sign ($) so personal or project files are not easily confused with system files in which the “$” often occurs.
- Avoid overuse of special characters.
- Do not use a hyphen as the first or last character in a file name.

Naming SuRPAS Directories and Files

The standard for naming SuRPAS directories and files will be discussed in depth, in Chapter Nine.

Guidelines For Naming File Types

File Type	Description	Source/Destination
Common OpenVMS File Types		
.COM	Command Procedure	Built with an editor / Use @
.EXE	**Executable Image (Program)**	**Output of the Linker / Use RUN**
.FOR	Fortran Source Code File	Built with an editor / To be compiled
.MAC	**MACRO Source Code File**	**Built with an editor / To be assembled**
.INC	Source Code Include File	Built with an editor / To be compiled
.JNL	**Journal File**	**Output of an editor / For file recovery**
.LIS	Listing File	Report output of a compiler
.OBJ	**Object File**	**Output of a compiler / Input to Linker**
.OLB	Object Library File	Output of the Librarian / Input to Linker
.TLB	**Text Library File**	**Provided by OpenVMS or an editor**
.HLB	Help Library File	Provided by OpenVMS or an editor
.OPT	**Linker Options File**	**Built with an editor / Input to Linker**
.MAP	Linker Map File	Report output of the Linker
.BCK	**Backup File**	**Output of Backup utility / For Restore**
.DAT	Data File	RMS READ/WRITE sequential Data File
Common SuRPAS File Types		
.SCR	Screen Files	
.FRM	**Forms File**	
.RPT	Report File	
.JDL	**Data Layout FIle**	
.CFD	Configuration & File Definition	
.CRD	**Configuration & Record Definition**	
.FDL	Data Layout FIle	
.FIL	**Code Management System**	

Creating and Identifying Subdirectories

Overview

Each disk contains a main directory called the ***master file directory (MFD)***, which contains a list of ***user file directories (UFD's)***. In most cases, a user file directory exists for each user on the system.

A ***subdirectory*** is any directory file that is not a master file directory or user file directory. Subdirectories enable users to organize their files into meaningful groups. For example, a user might have one subdirectory for memos and another subdirectory for command procedures.

This section discusses:

- The OpenVMS directory structure
- Creating subdirectories
- Using subdirectories

OpenVMS Directory Structure

System and User Directories

The OpenVMS operating system stores information in a hierarchical structure.

- At the top of this hierarchy is the ***master file directory***.
- The master file directory contains a list of all ***user file directories***. The user file directories of each user is often called the ***login directory*** or ***default directory;*** this is the directory used by default when that user logs in.
- Below the user file directory, the user can create personal subfile directories called subdirectories.
- Each user file directory can have a maximum of seven levels of subdirectories below it.

The following figure shows a sample directory hierarchy:

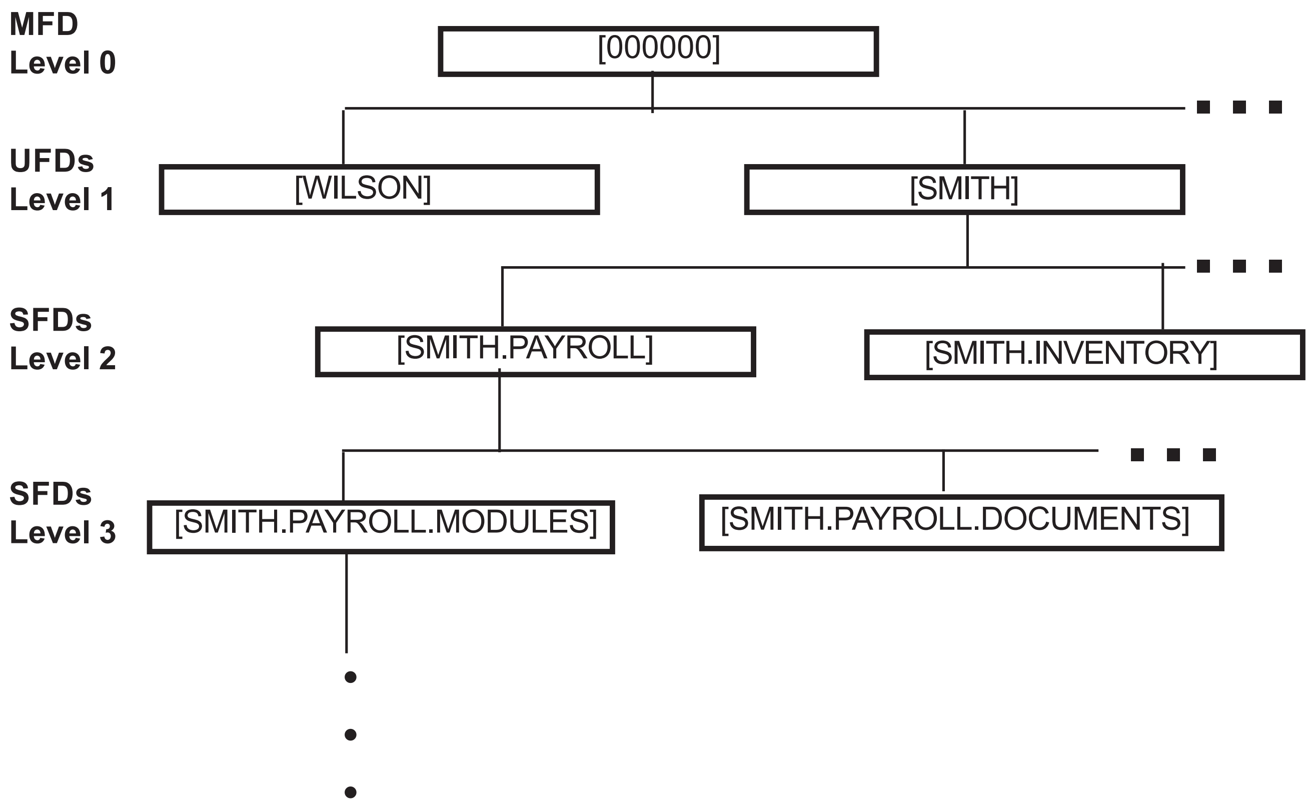


Figure 4-2 - OpenVMS Directory Structure

Notes: OpenVMS Directory Structure

- MFD is the Master File Directory, which catalogs all of the user directories.
- UFD is the User File Directory.
- SFD refers to subfile directories or subdirectories.
  - WILSON has no subdirectories.
  - SMITH has two levels of subdirectories.
  - While SMITH.PAYROLL has at least one level of subdirectories, SMITH.INVENTORY has none.

The Default Directory

There is always a default directory path known to the OpenVMS operating system. It is defined by the system manager for each user and is in place when the user logs in. The user can then change the default within restrictions set by the operating system and the system manager.

Creating Subdirectories

Identifying Subdirectories

The format for directory and subdirectory specifications is:

[DIRECTORY.SUBDIRECTORY.SUBDIRECTORY...]

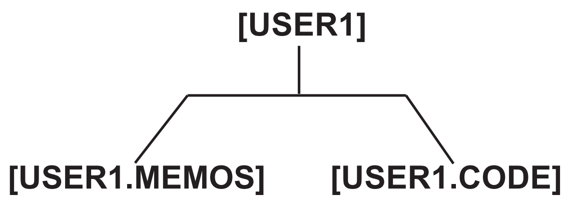
A period (.) between two directory names identifies the rightmost directory as a subdirectory of the one that is named immediately to its left.

A subdirectory file has a file type of .DIR and is always assigned a version number of 1 by the system. You can identify subdirectories easily by issuing the following command, which will list all of the directory files in your current directory:

```
$ DIRECTORY *.DIR
```

Rules for Naming Subdirectories

- The subdirectory name must be enclosed in brackets.
- Use a period (.) to separate directory and subdirectory names.
- The subdirectory name includes the directory name where it is located.



The complete name of the subdirectory .MEMOS under the UFD [USER1] is [USER1.MEMOS]. If the default is set to [USER1], the subdirectory may be simply identified by [.MEMOS].

Creating A Subdirectory

Use the **CREATE/DIRECTORY** command to create a subdirectory, using the following syntax:

CREATE/DIRECTORY   [directory.subdirectory]

This command creates a subdirectory under the specified directory regardless of the current default directory.

In the following example, the optional /LOG qualifier is used to cause the system to display an informational message when the subdirectory is created. By default, the system just returns the $ prompt to indicate that it has successfully executed the command.

```
$   CREATE/DIRECTORY/LOG  [USER1.TRAVEL]
%CREATE-I-CREATED,  $1$DUA1:[USER1.TRAVEL]  created
```

Example 4-1 - Using the CREATE/DIRECTORY Command

A common error when creating a subdirectory is the omission of the period (.) in the subdirectory specification, as in the following example:

```
$  CREATE/DIRECTORY/LOG  [TRAVEL]
%CREATE-E-DIRNOTCRE,  [TRAVEL]  directory  file  not  created
-  SYSTEM-F-NOPRIV,  no  privilege  for  attempted  operation
$
```

Example 4-2 - Error in Creating a Subdirectory

Without the period, this command fails to create a user file directory. To create a TRAVEL subdirectory under [USER1], use one of the following commands:

```
$  CREATE/DIRECTORY    [USER1.TRAVEL]

$  CREATE/DIRECTORY    [.TRAVEL]
```

Using Subdirectories

Uses For Subdirectories

Use subdirectories to:

- Organize the directory structure
- Protect files from accidental modification or loss
- Make it easier for the user to find files
- Organize FTK's

Files are usually grouped by:

- Function (e.g. all report files)
- Application(e.g. all files for a given project)
- Type (e.g. all command procedures)

Displaying Your Default Device or Directory

Use the SHOW DEFAULT command to display your current default device and directory.

```
$ SHOW DEFAULT
    WORK3:[USER21]
$
```

Example 4-3 - Using the SHOW DEFAULT Command

Navigating a Directory Structure

Use the SET DEFAULT command to change the default device and/or the directory name of your process with the following syntax:

$ SET DEFAULT ***device-name:*****[*****directory-name*****]**

- If the new default is on the same device, it is not necessary to specify a value for device-name.
- A physical device name must be terminated with a colon (:).
- The directory-name might include a subdirectory specification.
- Enclose the directory or subdirectory name in square brackets.

Specifying a Higher Directory Level

Use the hyphen (-) to specify a directory level above your current default without specifying the upper directory name.

This command...	Performs this function:
`$ SET DEFAULT [-]`	Sets your default to the directory one level above your current default directory.
`$ SET DEFAULT [--]`	Sets your default to the directory two levels above your current default directory.

The following examples show the use of the SET DEFAULT and SHOW DEFAULT commands. Try each one at your terminal and see if you achieve the same results.

Display the initial default device and directory:

```
$ SHOW DEFAULT
    WORK3:[USER1]
$
```

Example 4-4 - Displaying Initial Default Device and Directory

Change the default directory to the COMMANDS subdirectory. The default device is not changed.

```
$ SET DEFAULT [USER1.COMMANDS]
$ SHOW DEFAULT
    WORK3:[USER1.COMMANDS]
$
```

Example 4-5 - Changing Default Directory to COMMANDS Subdirectory

Use a hyphen to set the default directory to one directory level above the current default. The default device does not change.

```
$ SET DEFAULT [-]
$ SHOW DEFAULT
    WORK3:[USER1]
$
```

Example 4-6 - Setting Default Directory to One Directory Above Current Default

Change the default directory to the commands subdirectory again, this time using a shortcut (leave out the directory name).

```
$ SET DEFAULT [.COMMANDS]
$ SHOW DEFAULT
    WORK3:[USER1.COMMANDS]
$
```

Example 4-7 - Shortcut

It may be difficult to try the next two examples yourself, but you can get the idea by examining them in the manual.

- Change the default device and directory to device WORK7 and the BUILD subdirectory in directory USER2.

```
$ SET DEFAULT
WORK7:[USER2.BUILD]
$ SHOW DEFAULT
    WORK7:USER2.BUILD]
$
```

- Use the hyphen to reset the directory default; then set the new default to the MEMOS subdirectory.

```
$ SET DEFAULT [-.MEMOS]
$ SHOW DEFAULT
    WORK7:[USER2.MEMOS]
$
```

Some Common Errors

You can set your default to a nonexistent directory. No error is returned until you attempt to access the directory. See the example that follows:

```
$ SET DEFAULT [NONEXISTENT]
$ SHOW DEFAULT
    WORK3:[NONEXISTENT]
%DCL-I-INVDEF, WORK3:[NONEXISTENT] does not exist
$ DIRECTORY
%DIRECT-I-OPENIN, error opening WORK3:[NONEXISTENT]*.*;* as input
-RMS-E-DNF, directory not found
-SYSTEM-W-NOSUCHFILE, no such file
$
```

Example 4-8 - Setting Default to a Nonexistent Directory

Issue the following command to get back to your home directory from anywhere!

$ SET DEFAULT SYS$LOGIN

A common error in accessing a subdirectory is to use an incomplete subdirectory specification. In the following example, the use of [.MEMOS] causes an error because the system looks for that subdirectory below the [.COMMANDS] subdirectory.

```
$ SHOW DEFAULT
    WORK3:[USER1]
$ DIRECTORY *.DIR

Directory WORK3:[USER1]

COMMANDS.DIR;1     MEMOS.DIR;1

Total of 2 files.
$ SET DEFAULT [.COMMANDS]
$ DIRECTORY

Directory WORK3:[USER1.COMMANDS]
        .
        .
        .
$ SET DEFAULT [.MEMOS]
$ DIRECTORY
%DIRECT-E-OPENIN, error opening
WORK3:[USER1.COMMANDS].MEMOS]*.*;* as input
-RMS-E-DNF, directory not found
-SYSTEM-W-NOSUCHFILE, no such file
$
```

Example 4-9 - Using an Incomplete Subdirectory Name

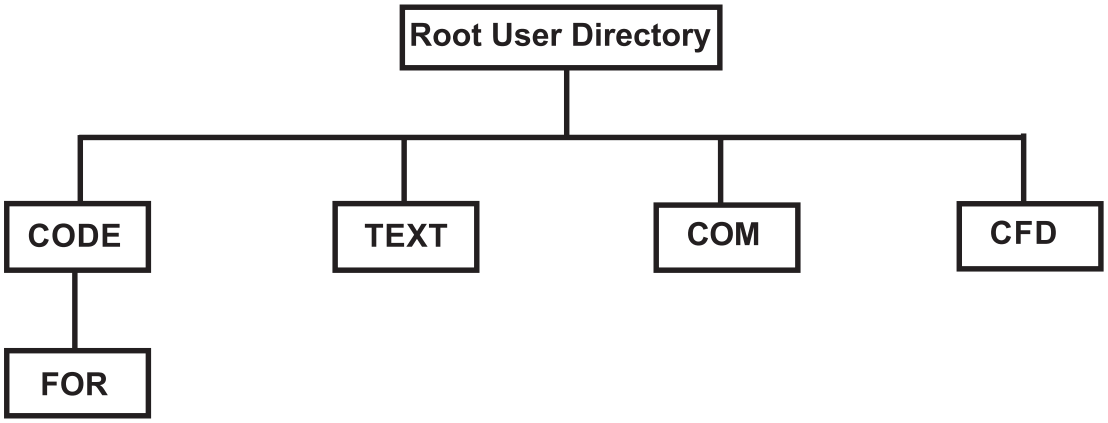
Specifying Many Directory Levels

Use an ellipsis (...) to specify many levels in a directory structure. This can be used with a variety of DCL commands, including DIRECTORY, PURGE, DELETE, and others.

This command...	Performs this task
`$ DIRECTORY [...]`	Searches the current default directory and all subdirectories below it.
`$ DIRECTORY [REPORT...]`	Searches all subdirectories below the directory called REPORT.
`$ DIRECTORY [.MEMOS...]`	Searches files in the MEMOS subdirectory and all subdirectories below it.

Interactive Exercise

Each student has an OpenVMS user account. Your account has a root user directory. Follow the steps in this exercise to create the appropriate subdirectories necessary to support the directory hierarchy diagram below.



Follow these steps to create the directory hierarchy diagram above:

1. Issue the **`SET DEFAULT SYS$LOGIN`** command so you point to your Root User Directory
2. Issue the **`SHOW DEFAULT`** command to verify the name of your Root User Directory
3. Issue the **`CREATE/DIRECTORY`** command to create each of the necessary subdirectories.
4. Issue a **`DIRECTORY`** command on the directory directly above each newly created directory to verify that the appropriate **`.DIR`** file has been created.
5. Issue a **`DIRECTORY`** command (**`DIRECTORY [...]`**) at your root login directory to view the entire directory structure.

Displaying Information About Files

Overview

There are many situations in which you need to display information about files. For example, you may need to:

- Locate a particular file so you can print its contents.
- Find the date you created a file so you can determine if it needs to be updated.
- Determine the number and size of your files to monitor disk usage.
- List the names of all files with a particular file type because you have forgotten the name of the file you want.

This section discusses:

- The DIRECTORY command
- Using wildcards

The DIRECTORY Command

Listing Files

Use the DIRECTORY command to locate a particular file or get information about files. The syntax is as follows:

DIRECTORY [file-spec [, ...]]

- The file specification determines which files will be listed.
- If you omit the file specification, you will get directory information about all of the files in your current default directory.
- If you specify more than one file, you must separate the file specifications with either commas (,) or plus signs (+) .
- You can use wildcard characters in the directory specification, file name, file type, or version number fields of a file specification to list all files that satisfy the fields you specify.

The default DIRECTORY display gives the file name, file type, and version number of each file. The following example shows the format of the output from the DIRECTORY command.

```
$ DIRECTORY

Directory  USER_DISK:[USER1]

DEBUG.LOG;3    DECTERM_ERROR.LOG;223         DECTERM_ERROR.LOG;222
DECW$TERMINAL_DEFAULT.DAT;3  DEC$TEST.PS;1  GREET.COM;1
LIBRARY.SAVE;1 LOGIN.COM;25   LOGIN.COM;1    LOGIN.LST;1
LOGIN.SAVE;1   LWK_PERSONAL.LINKBASE;1       MAIL.MAI;2
     •
     •
     •
Total of 73 files.
$
```

Example 4-10 - Multi-column Directory Display

Controlling the DIRECTORY Display

You can use a variety of qualifiers to control the display of the DIRECTORY command. The following table lists some commonly used qualifiers.

To display this information...	Use this qualifier:
Size of each file	/SIZE
Files in a specified number of columns	/COLUMNS=n
The file creation date	/DATE
Detailed information about each file	/FULL

A popular directory display produced by **`DIRECTORY/SIZE/COLUMNS=1`** can be seen in the following example:

```
$  DIRECTORY/SIZE/COLUMNS=1  LOGIN.*

Directory  USER_DISK:[USER1]

LOGIN.COM;25   3
LOGIN.COM;1    2
LOGIN.LST;1    2
LOGIN.SAVE;1   2

Total  of  4  files.
$
```

Example 4-11 - Single column Directory Display

Displaying Detailed Directory Information

Use the DIRECTORY/FULL command to display the following file information:

- File name, type, and version number
- File identification number (FID)
- Number of blocks used and allocated
- Identification code of the file's owner
- Dates of creation, last modification, expiration, last backup, effective usage, recording on media
- File owner's user identification code (UIC)
- File organization and other file attributes
- Journaling information
- Protection information

Other DIRECTORY qualifiers display subsets of this information.

At your terminal, use the DIRECTORY/FULL command to display the file information about the LOGIN.COM file in your default user account directory. The display will look similar to the example below:

```
$ DIRECTORY/FULL LOGIN.COM

Directory USER_DISK:[ROBBINSB]

LOGIN.COM;1                          File ID:    (135,1,0)
Size:                7/8             Owner:    [12,210]
Created:        24-DEC-2001 14:38:16.87
Revised:        26-DEC-2001 12:24:45.22
Expires:        <None Specified>
Backup:         <No Backup Recorded>
Effective:      <None specified>
Recording:      <None specified>
File Organization:   Sequential
Shelved state:       Online
File attributes:     Allocation: 8, Extend: 0, Global buffer count:0
                     No version limit
Record format:       Variable length, maximum 0 bytes, longest 122
bytes
Record attributes:   Carriage return carriage control
RMS attributes:      None
Journaling enabled:  None
File protection:     System: RWED,  Owner: RWED,  Group: RE,   World:
Access Cntrl List:   None

Total of 1 file, 7/8 blocks
$
```

Example 4-12 - Displaying Detailed Directory Information

Writing a Directory Listing to a File

Use the /OUTPUT qualifier to send the output from a directory operation to a file with the following syntax:

DIRECTORY/OUTPUT=*file-spec*

```
$ DIRECTORY/OUTPUT=HOME_USER_DIRECTORY_25DEC01.LIST
```

Example 4-13 - Using the DIRECTORY/OUTPUT Command

Displaying File Security Using the DIRECTORY Command

There are times when it is important to be aware of the security and protection information about one or more files. The DIRECTORY command supports the **`/SECURITY`** qualifier to control whether information about file security is displayed. Using the `/SECURITY` qualifier is equivalent to using the **`/ACL`**, **`/OWNER`**, and **`/PROTECTION`** qualifiers together (three additional qualifiers discussed below). The default is **`/NOSECURITY.`**

The **`/ACL`** qualifier controls whether the ***access control list (ACL)*** for each file is displayed. By default, the DIRECTORY command does not display the ACL for each file. An access control list defines the kinds of access granted or denied to users of an OpenVMS object. These objects can include files, devices, and logical name tables.

The **`/OWNER`** qualifier controls whether the file owner's user identification code (UIC) is displayed. If selected, only the first 20 characters are displayed. The default is **`/NOOWNER`**.

The **`/PROTECTION`** qualifier controls whether the file protection for each file is displayed. The default is **`/NOPROTECTION`**. When file protection is displayed it is in the form of four sets of four characters, each set separated by a comma and all four sets surrounded in parentheses. As will be discussed in the Chapter Ten on File Security, each file has four types of access: Read(R), Write(W), Execute(E), and Delete(D). Each file also has four protection categories: System(S), Owner(O), Group(G), and World(W). For each protection category a file can have no access or up to four types of access. From this we can build the protection code.

Examples of each of these four DIRECTORY qualifiers can be found in Chapter Ten on the section discussing Displaying File Security Information, Pages 10-9 and 10-10.

Using Wildcards

You can use wildcard characters to apply a DCL command to multiple files rather than to one file at a time. The command applies to all files that match the portion of the file specification entered using the wildcard character(s).

The asterisk (*) is used to replace an entire field or a multiple character portion of a field in a file or directory specification. The percent sign (%) is used to replace a single character in a file or directory specification.

You can use the asterisk or the percent sign as a wildcard in a directory name, file name, and/or file type. You can also use the asterisk (but not the percent sign) in version numbers.

The following DIRECTORY command uses the asterisk to represent a multi-character portion of a file name specification.

```
$ DIRECTORY M*

Directory USER_DISK:[USER1]

MAIL.MAI;2     MARBER.KEEP;1   MARBER.SAVE;1   MESSAGES.DIR;1
MONITOR.SUM;1

Total of 5 files.
$
```

Example 4-14 - Using Wildcard Characters - the Asterisk

The following command uses the percent sign to represent three characters of the file name. Only files having four character file names are displayed.

```
$ DIRECTORY M%%%

Directory USER_DISK:[USER1]

MAIL.MAI;2

Total of 1 file.
$
```

Example 4-15 - Using Wildcard Characters - the Percent Sign

Summary

Concepts

Constructing an OpenVMS File Specification

- A file specification provides the system with the information it needs to identify a file.
- The parts of a file specification are:
  NODE::DEVICE:[DIRECTORY]filename.filetype;version

Creating and Identifying Subdirectories

- The OpenVMS operating system stores information in a hierarchical structure.
- Use the CREATE/DIRECTORY command to create a subdirectory.
- Use subdirectories to organize your files.
- Use the SHOW DEFAULT command to display your current default device and directory.
- Use the SET DEFAULT command to change the default device and/or the directory name for your process.
- Use the ellipsis mark (...) and the hyphen (-) as shortcuts in specifying directory levels.

Displaying Information About Files

- To locate a particular file or get information about files, use the DIRECTORY command.
- When you want information about several files with similar file names or file types, use wildcards to represent matching characters.
- Use the asterisk (*) or the percent sign (%) as a wildcard in a file specification.

Commands

Creating and Identifying Subdirectories

CREATE/DIRECTORY
Creates a directory in an OpenVMS system.

SHOW DEFAULT
Displays the current default device and directory.

SET DEFAULT
Sets a default device and directory.

Displaying Information about Files

DIRECTORY
Displays a listing of files.