

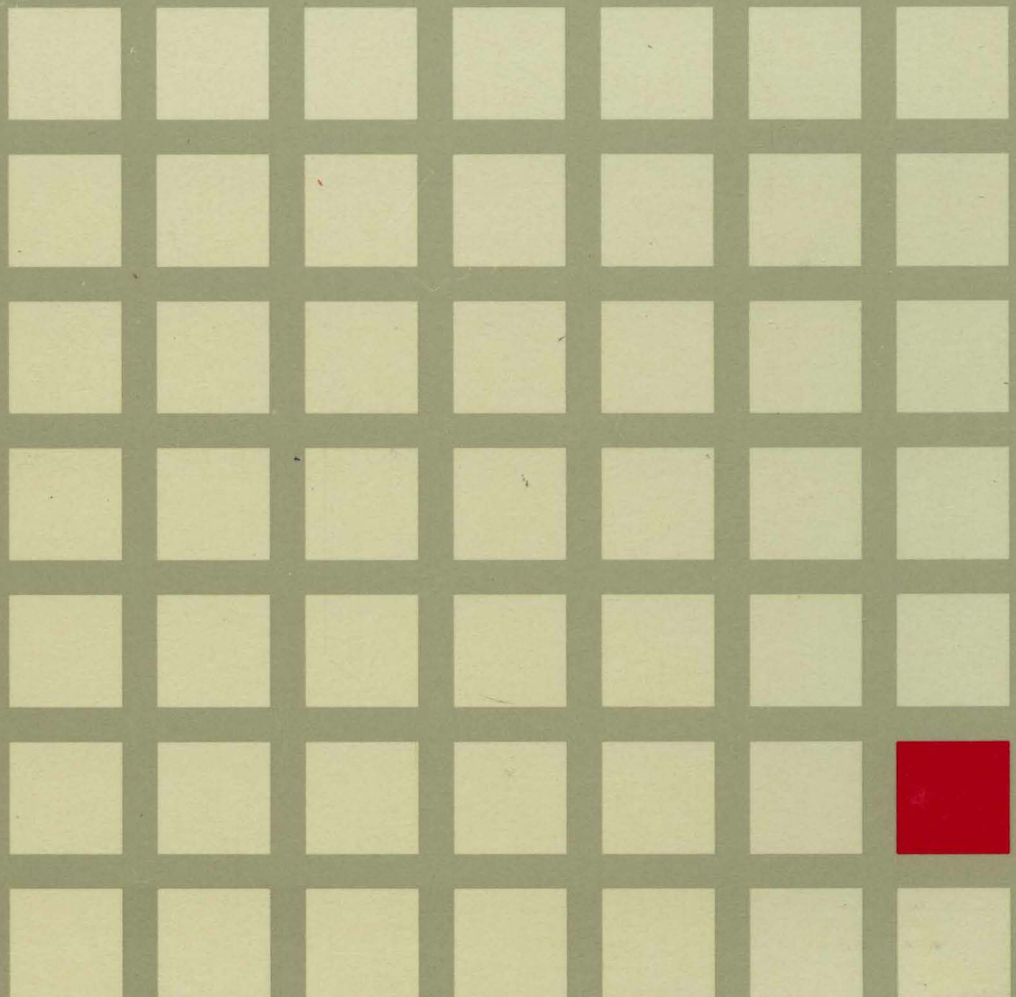


Virtual Machine/System Product

SC19-6210-05

CMS User's Guide

Release 6





Virtual Machine/System Product

SC19-6210-05

CMS User's Guide

Release 6

Sixth Edition (July 1988)

This edition, SC19-6210-05, is a major revision of SC19-6210-04 and applies to the IBM Virtual Machine/System Product Release 6, program number 5664-167, and to all subsequent releases of this product until otherwise indicated in new editions or Technical Newsletters. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, 4300, and 9370 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

Summary of Changes

For a list of changes, see "Summary of Changes" on page 365.

Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

In this manual are illustrations in which names are used. These names are fanciful and fictitious; they are used solely for illustrative purposes and not for identification of any persons or company.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent program may be used instead.

Ordering Publications

Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. Publications are *not* stocked at the address given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Dept. G60, P.O. Box 6, Endicott, NY, U.S.A. 13760. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

The form for readers' comments provided at the back of this publication may also be used to comment on the VM/SP online HELP facility.

Preface

Who is This Book For?

This book is intended for the general VM/SP user. It contains information describing the interactive facilities of VM/SP and includes examples showing you how to use VM/SP.

How is This Book Organized?

This book contains four parts, plus appendices.

“Part 1: Getting to Know VM/SP” contains sections that describe, in general terms, the CMS facilities and the CP and CMS commands that you can use to control your virtual machine. If you are an experienced programmer who has used interactive terminal systems before, you may be able to refer directly to the *VM/SP CMS Command Reference* book to find specific details about CMS commands that are summarized in this part. Otherwise, you may need to refer to later sections of this publication to gain a broader background in using CMS.

The topics discussed in Part 1 are:

- Introduction to VM/SP
- VM/SP Environments and Mode Switching
- CMS File System
- Using the Shared File System
- Storing Your Files on Minidisks
- What You Can Do with CMS Commands
- Using the HELP Facility.

“Part 2: Working with VM/SP” discusses the procedures to be followed for performing routine CMS tasks.

The topics discussed in Part 2 are:

- Editing Your Files
- Using Real Printers, Punches, Readers and Tapes
- Communicating with Other Computer Users
- Looking at VM/SP Through Windows
- Using the CMS Batch Facility.

“Part 3: Learning To Use Execs” gives detailed information on creating exec procedures to use with CMS.

The topics discussed in Part 3 are:

- Introduction to the Exec Processors
- Creating System Product Interpreter Execs
- Creating a PROFILE EXEC
- Commands Used with System Product Interpreter Execs.

“Part 4: Tailoring Your System” contains information on how to use the various functions of VM/SP to customize your virtual machine.

The topics discussed in Part 4 are:

- Customizing Full-Screen CMS
- Tailoring the HELP Facility.

“Appendix A: Summary of CMS Commands” briefly describes the commands available in the CMS command environment. Special CMS commands that can be used from a full-screen environment are listed in a separate table in this section.

“Appendix B: Summary of CP Commands” describes the CP command privilege classes and summarizes the commands available for general users in the CP command environment.

“Appendix C: Considerations for Full-Screen CMS and Windowing” provides a basic overview of windowing and full-screen CMS functions. It also gives some specific information regarding CP, CMS, System Product Editor, and application interactions with windowing and full-screen CMS.

The “Glossary of Terms and Abbreviations” lists and defines terms used in this manual.

Contents

Part 1: Getting to Know VM/SP	1
Chapter 1. Introduction to VM/SP	3
How You Communicate With VM/SP	3
What You Must Know to Use VM/SP	6
Beginning Your Terminal Session	6
Logging On at a 3270-Type Terminal	6
Logon Exceptions	8
Getting Into CMS	8
Ending Your Terminal Session	9
Entering Commands	10
RETRIEVE Function	11
Setting Program Function Keys	12
Using Full-Screen CMS	13
Display Screen Characteristics	13
Messages	13
VM Status Notices	14
Full-Screen CMS Status Notices	16
Additional Display Screen Capabilities	17
How VM/SP Responds to Your Commands	17
Working with CMS	19
Storing Your Files	22
Accessing Your Directories or Minidisks	23
Releasing Directories or Minidisks	24
Console Output	24
Chapter 2. VM/SP Environments and Mode Switching	27
CP Environment	28
The CMS Environment	29
Protected Application Environment	30
XEDIT and CMS Subset	30
CMS/DOS	31
Interrupting Program Execution	32
Using the Attention Key	32
Interrupting Your Programs	32
Virtual Machine Interruptions	33
Control Program Interruptions	34
Address Stops and Breakpoints	35
Using the 3270 Text Feature	35
Error Situations	36
Chapter 3. CMS File System	37
CMS File Formats	37
How CMS Files Get Their Names	37
Duplicate File Names or File Types	38
What Are Reserved File Types?	40
File Types for CMS Commands	40
Output Files: TEXT and LISTING	46
File Types for Temporary Files	46
File Types for Documentation	48

File Mode Letters and Numbers	48
When To Specify File Mode Letters: Reading Files	50
When To Specify File Mode Letters: Writing Files	52
How File Mode Numbers Are Used	53
File Mode Numbers in SFS	54
File Mode Numbers for Minidisks	55
Commands Used to Change File Mode Numbers	57
Managing Your File Space	58
Managing Your Minidisks	59
CMS Command Search Order	59
CMS Command Execution Characteristics	60
Displaying a List of Your CMS Files	61
Finding Files in Your FILELIST List	62
Erasing Files from FILELIST	63
Listing Your Files with the LISTFILE Command	63
Comparing Contents of Files	64
Copying Files	65
Renaming Files	65
Changing the Record Format of a File	65
Using Synonyms	66
Using Translations	67
Chapter 4. Using the Shared File System	69
What is the Shared File System?	69
Getting Started	71
Organizing Your Files	72
Working with Directories	73
Using the Abbreviated Form of Your Top Directory	73
Accessing Another User's Directory	74
Specifying a Directory Identifier	75
Listing the Structure of a Directory with DIRLIST	76
Using the DIRLIST PF Keys	77
Using the LISTDIR Command	78
Creating a Directory	79
Putting Files Into a Directory	81
Copying Files to a Directory	81
Creating Files with XEDIT	83
Renaming Your Files and Directories	83
Relocating Your Files and Directories	84
Erasing a Directory	87
Navigating Through Your Directories	88
Sharing Files	94
Creating Aliases to Files	95
Using the QUERY ALIAS Command	102
Using ALIALIST	103
Erasing Your Base Files	105
Authorizing Others to Access Your Files and Directories	107
Granting Authority	109
Revoking Authority	110
Determining Who Has Authority on a File or Directory	112
Using AUTHLIST	112
Determining Ownership of a File or Directory	115
Using Aliases to Share Files	116
Creating a Bulletin Board or Shared Disk	117
Locking Files and Directories	119
Determining If a File or Directory is Locked	122

Determining How Much of Your File Space You Have Used	124
Application Considerations	124
Using Multiple File Pools	125
Sharing Files with Users on Other Systems	126
Chapter 5. Storing Your Files on Minidisks	127
Minidisks and How They Are Defined	127
Defining Temporary Minidisks	127
Formatting Minidisks	128
Sharing Minidisks: Linking	128
Linking to Your Minidisks	130
Detaching Minidisks	130
Minidisk File Directories	130
Chapter 6. What You Can Do with CMS Commands	133
Beginning and Ending Your Terminal Session	133
Tailoring Your System	134
Requesting Information	136
Communicating with Other Computer Users	137
Controlling Terminal Output	138
Sharing Minidisks	139
Sharing Your Files and Directories	140
What You Can Do to the Files in Your Virtual Reader	141
Receiving or Loading Files into Your Directory or onto Your Minidisk	142
Erasing Files From Your Directory or Minidisk	142
Creating and Modifying Files	143
Moving Files	143
Chapter 7. Using the HELP Facility	145
Getting HELP on Commands	146
BRIEF HELP	147
DETAIL HELP	148
RELATED HELP	149
Getting HELP on SET and QUERY	150
Facts About Command HELP	150
Getting HELP on Messages	151
Menus	151
TASK Menus in HELP	152
Component Menus	153
Using the PA2 and PF Keys	153
Toggling	154
Using the MOREHELP Command	156
System Product Editor	156
Printing HELP Screens	157
Working with Your HELP Files	157

Part 2: Working with VM/SP 159

Chapter 8. Editing Your Files	161
System Product Editor	161
Beginning an Editing Session	162
Saving Your Changes	164
We Made Changes to the File at the Same Time!	165
Canceling Your Changes	165
What To Do When You Run Out of Space	166

If You Run Out of Virtual Storage	166
If Your Minidisk Is Full	168
If Your File Space is Full	169
If Your Storage Group is Full	169
Using the Editor in Line-Mode	170
Editing on a Remote 3270	170
Editing from an Exec File	170
Chapter 9. Using Real Printers, Punches, Readers, and Tapes	171
CMS Unit Record Device Support	171
Using the CP Spooling System	171
Spool File Characteristics	171
Altering Spool Files	174
Using Your Card Punch and Card Reader in CMS	175
Handling Tape Files in CMS	177
Using the CMS TAPE Command	178
Tape Labels in CMS	180
User Responsibilities	181
Label Processing in OS Simulation	181
Label Processing in CMS/DOS	188
CMS TAPESL Macro	190
Tape Label Processing by CMS Commands	191
LABELDEF Command	192
End-of-Volume and End-of-Tape Processing	193
Error Processing	196
MOVEFILE Command	197
OS Utility Programs	197
Specifying Special Tape Handling Options	198
Moving Data Between Shared and Nonshared Tape Subsystems	199
Chapter 10. Communicating with Other Computer Users	201
What Is a Names File?	201
Creating a Names File	202
Sending Messages	207
Receiving Messages	208
Sending Notes and Files	210
Composing Notes	210
Sending a Note	211
Sending Files	211
Sending One File	213
Receiving Notes and Files	214
Chapter 11. Looking at VM/SP Through Windows	219
What Are Windows and Virtual Screens?	219
What Can You Do With a Window?	220
Using Full-Screen CMS	221
Status Information	223
Location Information	224
Your Default Windows and Virtual Screens	224
Special Keys	226
Messages in Full-Screen CMS	231
Reentering Commands	233
Working with Border Commands	235
Using the WM Window	242
Chapter 12. Using the CMS Batch Facility	249

Submitting Jobs to the CMS Batch Facility	249
Input to the Batch Machine	249
Batch Considerations for Shared File System (SFS) Files	250
Submitting Virtual Card Input to the CMS Batch Facility	251
Other Input Records	254
How the Batch Facility Works	254
Preparing Jobs for Batch Execution	254
Restrictions on CP and CMS Commands in Batch Jobs	255
Batch Facility Output	257
Purging and Reordering Batch Jobs	257
Using Exec Files for Input to the Batch Facility	258
Sample System Procedures for Batch Execution	259
A Batch Exec for a Non-CMS User	262

Part 3: Learning to Use Execs 263

Chapter 13. Introduction to the Exec Processors	265
The System Product Interpreter	265
EXEC 2 Processor	266
The CMS EXEC Processor	266
Relationship of the Exec Interpreters	267
Running Execs	267
Attributes of EXEC Files	267
Chapter 14. Creating System Product Interpreter Execs	269
Running Your Exec Files	269
Sample System Product Interpreter Execs	270
Chapter 15. Creating a PROFILE EXEC	273
Chapter 16. Commands Used with System Product Interpreter Execs	277

Part 4: Tailoring Your System 287

Chapter 17. Customizing Full-Screen CMS	289
Tailoring System Defaults	289
POSITION WINDOW	293
SIZE WINDOW	294
MAXIMIZE WINDOW and RESTORE WINDOW	294
Using the SET Command	297
SET BORDER	297
SET RESERVED	299
SET WINDOW	301
Exploring Other Commands	303
Window and Virtual Screen Tables	303
Considerations When Disconnecting and Reconnecting	306
Message Routing Tables	308
Chapter 18. Tailoring the HELP Facility	309
Creating HELP Files	309
File Names for HELP Files	309
File Types for HELP Files	310
Examples of Naming Conventions	312
Creating Menus for HELP Files	312

Creating Command HELP Files	316
Creating HELP Files for Messages	318
Highlighting Words within a File	319
Using Command Abbreviations	319
Adding Your Own HELP Components	320
Using HELPCONV to Create HELP Files	320
Changing Existing HELP Files	330
Adding HELP Files	330
Deleting HELP Files	330
Altering Existing HELP Files	330
Changing Menus	331
Appendix A. Summary of CMS Commands	333
Appendix B. Summary of CP Commands	343
Privilege Classes for CP Commands	343
CP Commands	345
Appendix C. Considerations for Full-Screen CMS and Windowing	349
Windowing Commands	349
General Information on Full-Screen CMS	352
Virtual Screens and Windows	352
Screen Organization	353
Default Settings	354
WM Environment	356
Migration Considerations	357
CMS Considerations	358
CP Considerations	360
System Product Editor Considerations	360
Considerations for Writing Applications	361
Summary of Changes	365
Glossary of Terms and Abbreviations	375
Bibliography	383
Prerequisite Publications	383
Corequisite Publications	383
Quick References	383
Related VM/SP Publications	383
Related Publications for OS Users	384
Related Publications for VSAM and Access Method Services Users	384
Related Publications for CMS/DOS Users	384
Index	389

Part 1: Getting to Know VM/SP

Learning how to use CMS is not an end in itself; you have specific tasks to do, and you need to use the computer to perform them. The information contained in Part 1 of the *VM/SP CMS User's Guide* is organized to help you quickly familiarize yourself with VM/SP, so that you can learn how to take advantage of VM/SP to simplify your work.

Chapter 1: Introduction to VM/SP introduces you to VM/SP and its conversational component, CMS. It should help you to get a picture of how you, at a terminal, use and interact with the system.

Chapter 2: VM/SP Environments and Mode Switching. During a terminal session, commands and requests that you enter are processed by different parts of the system. This chapter describes how and when you can communicate with these different programs.

Chapter 3: CMS File System. Data and programs that you create are stored in *files*. These files are stored in Shared File System (SFS) directories or on minidisks. This chapter introduces you to the creation and handling of CMS files.

Chapter 4: Using the Shared File System provides details on the functions you can perform and the commands available to you when you store your files in a Shared File System (SFS) file pool.

Chapter 5: Storing Your Files on Minidisks provides details on how to manage files stored on minidisks.

Chapter 6: What You Can Do with CMS Commands contains a sampling of commands in various functional areas to give you a general idea of the kinds of things you can do and the commands available to help you do them.

Chapter 7: Using the HELP Facility. The CMS HELP facility provides an online display of documentation and an online list of tasks that guide you to the appropriate HELP file for the command or commands. This chapter describes the HELP facility and shows you how to use HELP to assist you in performing CMS tasks.



Chapter 1. Introduction to VM/SP

Virtual Machine/System Product (VM/SP) is a program product that controls *virtual machines*. A virtual machine is the functional equivalent of a real computer that you control from your terminal, using a command language of verbs and nouns.

The command languages correspond to the components of VM/SP. CP (Control Program) controls the resources of the real machine; that is, the physical machine in your computer room. CP also manages the communications among virtual machines and between a virtual machine and the real system. CMS (Conversational Monitor System) is the operating system that runs under CP; it can simulate many of the functions of OS (Operating System) and DOS (Disk Operating System), so that you can run many OS and DOS programs in a conversational environment.

Although this publication is primarily concerned with using CMS, it also contains examples of CP commands with which you, as a CMS user, should be familiar.

How You Communicate With VM/SP

When you are running your virtual machine under VM/SP, each command, or request for work, that you enter on your terminal is processed as it is entered; usually, you enter one command at a time and commands are processed in the order that you enter them.

You can enter CP commands from either the CP or CMS environment, but you cannot enter CMS commands while in the CP environment. The concept of *environments* in VM/SP is discussed in Chapter 2, "VM/SP Environments and Mode Switching."

After you have typed or keyed in the line you wish to enter, you press the Return or ENTER key on the keyboard. When you press this key, the line you have entered is passed to the command environment you want to have process it. If you press this key without entering any data, you have entered a *null line*. Null lines sometimes have special meanings in VM/SP.

If you make a mistake entering a command, VM/SP tells you what your mistake was, and you must enter the line again. The examples in this publication assume that the commands are correctly entered.

You can enter commands using any combination of uppercase and lowercase characters; VM/SP translates your input to uppercase. Examples in this publication show all user-entered input lines in blue type; system responses are shown in black type.

CP Command Language

You use CP commands to communicate with the control program. CP commands control the devices attached to your virtual machine and their characteristics.

Allocating Space: For example, if you want to increase the virtual address space assigned to your virtual machine, use the CP command DEFINE. CP takes care of the space allocation for you and then lets your virtual machine use it.

Receiving Messages: Or if, for example, you are receiving printed output at your terminal and do not want to be interrupted by messages from other VM/SP users, you can use the CP command SET MSG OFF to refuse messages, because it is CP that handles communication among virtual machines. The CP QUERY SET command displays the status of the CP SET MSG function and other CP SET command functions.

Sending Messages: CP commands let you send messages to the system operator and to other users, or modify the configuration of devices in your virtual machine. CP commands are available to all virtual machines using VM/SP. You can enter these commands when you are in the virtual machine environment using CMS (or some other operating system) in your virtual machine.

The CP commands and command privilege classes (not all commands are available to all users) are listed in Appendix B, "Summary of CP Commands." The CP Commands applicable to the general user are discussed in detail in the *VM/SP CP General User Command Reference*. However, because many CP commands are used with CMS commands, some of the CP commands you will most frequently use are discussed in this publication, in the context of their usefulness for a CMS application.

CMS Command Language

The CMS command language lets you create, modify, and debug problem or application programs and, in general, manipulate data files. Your CMS files are stored within directories, groups of related files that are organized hierarchically, or on minidisks.

Many OS language processors are executed under CMS: the assembler, VS BASIC, OS FORTRAN, VS FORTRAN, OS/VS COBOL, and OS PL/I Optimizing and Checkout Compilers. In addition, the DOS/VS COBOL, DOS PL/I, and VS APL Program Products are supported. You can find a comprehensive list of language processors that are executed under CMS and relevant publications in the *VM/SP Introduction* book. CMS executes the assembler and the compilers when you invoke them with CMS commands.

Using XEDIT, the Editor: When you enter the XEDIT command, you invoke the System Product Editor to create, modify, or manipulate CMS files. Once the VM/SP System Product Editor has been invoked, you may execute XEDIT subcommands and use the System Product Interpreter or EXEC 2 macro facility. The System Product Interpreter, CMS EXEC interpreter, and the EXEC 2 interpreter provide execution procedures consisting of CP and CMS commands; they also provide the conditional execution capability of a macro language. A macro language is a facility that lets you simplify your work by expanding the basic subcommand language, eliminating repetitive tasks, and much more.

Using Virtual Devices: Other CMS commands allow you to read cards from a virtual card reader, punch cards to a virtual card punch, and print records on a virtual printer. Many commands are provided to help you manipulate your directories, minidisks, and files.

Storing Files: You can take advantage of two methods for storing your files; you can store them in a file pool or on minidisks. A file pool is a large amount of DASD (direct access storage device) space containing the files for many users. Within a file pool, you are assigned an individual file space in which you can

organize your files. The part of CMS that manages file pools is called the Shared File System (SFS).

If you wish, you can store files on minidisks, which are areas of DASD assigned to individual users.

When you store files in a file pool, you will be able to perform the same functions that you can by using minidisks. In addition, because of the way SFS handles file spaces, you will be able to better organize your files and easily share them with others.

Depending on your system configuration, you may have the option to use both methods of storing files. You could store those files that you may want to share in your SFS file space; other files could be stored on minidisks.

Using Full-Screen CMS: CMS also allows you to use windowing commands and full-screen CMS to help you manage the data on your physical screen. When you set full-screen on, you can type commands from almost anywhere on the physical screen. Full-screen CMS also allows you to scroll forward and backward through your CMS session to see commands you entered previously and CMS responses to those commands.

Using HELP: You use the HELP command to display information on how to use CP commands and CMS commands, subcommands, and EXECs, and explanations of CP and CMS messages. You can issue the HELP command when a brief explanation of syntax, a parameter, or function is sufficient, thereby avoiding interrupting your terminal session to refer to a manual.

Using CP from CMS: You can also enter CP commands from within the CMS virtual machine environment.

Using Non-English Languages: If your VM/SP system supports a language other than English, you can receive messages, view productivity aid panels (like the FILELIST screen), and enter various CMS commands in that language.

You can use the QUERY LANGLIST command to find out the languages that your virtual machine supports. You can also find out what language environment you are currently working in with two QUERY commands:

1. The QUERY CPLANG command tells you the language environment for CP.
2. The QUERY LANGUAGE command tells you the language environment for CMS.

VM/SP allows you to change the language you are working in without having to quit your session. The SET LANGUAGE command automatically gets all the information you need to interact with VM/SP in another language. SET LANGUAGE also allows you to add language information for applications.

Refer to the *VM/SP CMS Command Reference* for more detailed information about the SET LANGUAGE, QUERY LANGLIST, and QUERY LANGUAGE commands. Refer to the *VM/SP CP General User Command Reference* for information on the QUERY CPLANG command.

If you want to know more about the languages available on your VM/SP system, contact your system administrator.

What You Must Know to Use VM/SP

Before using CP and CMS, you should know:

1. How to operate your terminal
2. Your user ID (user identification) and password.

The Terminal: Your Virtual Console

There are many types of terminals you can use as a VM/SP virtual console. Before you can conveniently use any of the commands and facilities described in this publication, you have to familiarize yourself with the terminal you are using. Generally, you can find information about the type of terminal you are using and how to use it with VM/SP in the *VM/SP Terminal Reference*. If your terminal is a 3767, you also need the *IBM 3767 Communication Terminal Operator's Guide, GA18-2000*.

In this publication, examples and usage notes assume that you are using a display terminal (such as a 3277). If you are using a typewriter style terminal (such as a 2741) consult "Logical Line Editing Symbols" in the *VM/SP CP General User Command Reference* for a discussion of special techniques that you can use to communicate with VM/SP.

Your User ID and Password: Keys into the System

Your user ID identifies your virtual machine to VM/SP and allows you to gain access to the system. Your password functions as a protective device ensuring that only those allowed can use your virtual machine. The user ID and password are usually defined by the system programmer for your installation.

Beginning Your Terminal Session

To establish contact with VM/SP, you switch on the terminal device and VM/SP responds with a logo and some form of the message:

VIRTUAL MACHINE/SYSTEM PRODUCT

to let you know that VM/SP is running and that you can use it. If you do not receive the "VIRTUAL MACHINE/SYSTEM PRODUCT" message, see the *VM/SP Terminal Reference* for specific directions.

Note: If your terminal is not a 3270-type, use the LOGON procedures described in the section entitled "Logon Exceptions" on page 8.

Logging On at a 3270-Type Terminal

If you are using a 3270-type terminal, you may log on directly from the logo screen.

In the following figure, notice that below the actual VM/SP logo on the logo screen are two lines instructing you to fill in your user ID and password. Following these instructions are three input lines labeled USERID, PASSWORD, and COMMAND. The cursor is placed at the input line for USERID.

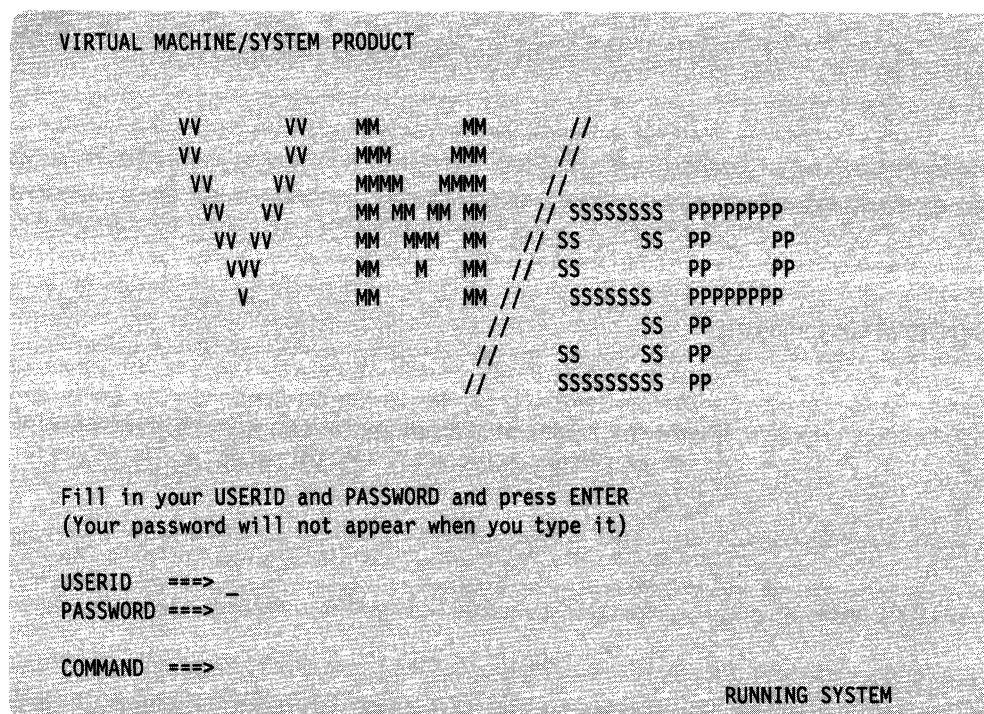


Figure 1. Sample of VM/SP Logo on a 3270-Type Terminal

You may type your user ID and password in the **USERID** and **PASSWORD** input areas and press **ENTER**. If all of the information is entered correctly, the logo is cleared from the screen, no further prompts will appear, and you will be logged on to the system. If an invalid user ID or password is entered, the logo is cleared from the screen, and the following message and prompt will appear:

```
DMKLOG050E LOGON unsuccessful--incorrect password
```

or

```
DMKLOG053E userid not in CP directory
```

Enter one of the following commands:

```

LOGON userid          (Example: LOGON VMUSER1)
DIAL userid          (Example: DIAL VMUSER2)
MSG userid message   (Example: MSG VMUSER2 GOOD MORNING)
LOGOFF

```

If you enter only your **PASSWORD** in the input area, or if your **USERID**, as entered, contains one or more blanks (for example, **V MUSER1**), the following error message will be issued, followed by the **LOGON** prompts:

```
DMKCFM288E LOGON from the initial screen was unsuccessful
```

Introduction

You may also enter your user ID in the USERID input area, without your password or enter the LOGON command, followed by your user ID, in the COMMAND input area. The following prompt will appear:

```
Enter password (it will not appear when typed):
```

If you have entered the information correctly, the logo is cleared from the screen and you will be logged on to the system.

Logon Exceptions

If your terminal is not a 3270-type, and the VM/SP logo screen is displayed, you can now press the ENTER key (or equivalent) on your terminal to clear the display. Now, enter your first command to identify yourself to VM/SP, the CP LOGON command. If your user ID is TIGER, then you type:

```
logon tiger
```

and press the ENTER key. You only need to type L, because L is short for LOGON.

If VM/SP accepts your user ID, it responds by asking you for your password:

```
Enter password (it will not appear when typed):
```

Now, carefully type your password, and press the ENTER key. You may not see your password as you type it. This is a security measure; it prevents others from learning your password. If you receive the message:

```
DMKLOG050E LOGON unsuccessful--incorrect password
```

followed by:

```
Enter one of the following commands:
```

```
LOGON userid      (Example: LOGON VMUSER1)
DIAL userid       (Example: DIAL VMUSER2)
MSG userid message (Example: MSG VMUSER2 GOOD MORNING)
LOGOFF
```

you will have to start over, beginning with the CP LOGON command. You will also receive a prompting message to help you restart if you make a mistake in entering other LOGON information.

Getting Into CMS

After a successful LOGON, your next step is to load CMS in your virtual machine using the CP IPL command. IPL stands for Initial Program Load,

```
ipl cms
```

where "cms" is assumed to be the saved system name for your installation's CMS. VM/SP responds by displaying a message such as:

```
VM/SP CMS - 05/16/86 12:54
```

to indicate that the IPL command executed successfully. Press the ENTER key again. VM responds with a message. The last line, known as the ready message, may look like this:

```
Ready; T=0.01/0.01 08:05:50
```

At this point you have loaded CMS, and you can now enter both CP and CMS commands.

Your user ID may be set up for an automatic IPL, so that you receive a message, indicating that you are in the CMS command environment, without having to issue the IPL command.

Many of the commands you enter work with data files. Typically, those commands operate within your file space. If you receive the following message at LOGON time,

```
DMSSDM1153E File pool filepoolid is unavailable or unknown
```

the server that manages your file pool is not available. At this point, you must wait until the file pool becomes available to complete your work. You can periodically try to access your top directory to determine the status of your file pool. You will see how to do this in Chapter 4, "Using the Shared File System." Your top directory is the directory assigned to you by the system administrator at the time you are enrolled in a file pool.

If you are enrolled in another file pool, or if you have access to minidisk storage, you can use these alternative storage methods until your primary file pool becomes available.

If your files are stored on minidisks and this is the first time you are using a new minidisk, you may receive the following message:

```
DMSACC112S A(191) device error
```

At this point, you must "format" the minidisk, that is, prepare it for use with CMS files. See the section entitled "Formatting Minidisks" on page 128.

Ending Your Terminal Session

To end your terminal session, use the CP LOGOFF command. Enter:

```
logoff
```

and press the ENTER key, or just enter:

```
log
```

and press the ENTER key, because LOG is short for LOGOFF.

At times you may be running a long program under one user ID and wish to use your terminal for some other work. Then, you can disconnect your terminal using the CP command DISCONN:

```
#cp disconn
```

or

```
#cp disconn hold
```

Your virtual machine continues to run while disconnected. Disconnected virtual machines are automatically logged off after 15 minutes if a CP READ or VM READ status is pending, that is, if the machine is waiting for you to enter a response. If you want your virtual machine to be logged off the system 15 minutes after your program has finished executing, issue SET AUTOREAD ON. AUTOREAD ON is the default for some terminal types; see the *VM/SP CMS Command Reference* for a description of the SET AUTOREAD command. Refer to the *VM/SP Terminal Reference* information on specific terminal types and how to reconnect to your virtual machine.

If you want to regain terminal control of your virtual machine after disconnecting, log on as you would to begin your terminal session. Your virtual machine is placed in the CP environment, and to resume its execution, you use the CP command BEGIN. You should not disconnect your virtual machine if a program requires an operator response, because the console read request cannot be satisfied.

Entering Commands

To enter a command from a display terminal when you use it as a virtual machine console under VM/SP, type the command and press the ENTER key. Since the keyboard is never locked during the execution of a command or program, you can enter successive commands without waiting for the completion of the previous command. This stacking function can be combined with the other methods of stacking lines, such as using the logical line end symbol (#) to stack several commands. To stack commands with the logical line end symbol, type the commands on the command line, separate them with #'s, but do not press ENTER until you have typed all the commands you want done at one time. For example, you might enter the following commands:

```
cp query time # cp query reader all # receive
```

First the system will display the time, then the contents of your virtual reader, and then will read in the first file in your virtual reader.

If, however, you enter more lines than your terminal can accommodate, you receive the status message NOT ACCEPTED and you must wait until the buffer is cleared before you can enter the line.

You will find, as you become accustomed to using a 3270, that the #CP function is very useful. The #CP function allows you to pass a command to the control program immediately, bypassing any processing by the virtual machine (CMS). The #CP function can be used in any VM/SP environment, and you can enter it even when a program is executing. You do not have to interrupt a program's execution to enter a command such as:

```
#cp query reader all
```

to display the contents of your reader, or:

```
#cp spool printer class s
to spool your virtual printer.
```

RETRIEVE Function

One of the most common user difficulties is typing errors. The RETRIEVE function provides a convenient and time-saving method of correcting errors without retyping the entire input. You can use this function by defining a program function (PF) key for it, using a command such as:

```
set pf6 retrieve
```

If you define a PF key for the RETRIEVE function, VM/SP remembers input lines entered at the terminal. When you press the PF key, VM/SP redisplay the latest input line in the input area, so that you can modify and reenter the data. This allows you to correct errors, change your input, or repeatedly reissue a command.

VM/SP actually remembers several input lines. The number of lines remembered depends on the length of the lines; VM/SP remembers more short lines than long lines, but it can always remember at least one full input line. Duplicate input lines (lines that are the same as the previous input) are not remembered because it is not useful to remember the same line twice. For security reasons, input lines that are not displayed at the terminal, such as passwords, are never remembered.

You can set two PF keys for the RETRIEVE command— one to retrieve in a FORWARD direction, and one to retrieve in a BACKWARD (the default) direction. When you first press the RETRIEVE BACKWARD program function key, VM/SP displays the latest input line. If you press the RETRIEVE BACKWARD key again, VM/SP displays the previous input line. As the key is pressed, VM/SP steps back through the input lines, displaying them one at a time. When VM/SP reaches the oldest line that it has remembered, it cycles back to the latest one again. When an input line is entered, VM/SP resets itself so that the RETRIEVE program function key starts with the latest input line.

You can also set a PF key to RETRIEVE FORWARD. For example, if you are using the RETRIEVE BACKWARD PF key to find a line of input, and you inadvertently pass over that line, you can press RETRIEVE FORWARD. The search will change directions, allowing you to recover the line you just passed, without having to cycle back through all of the older lines to get to the latest one again.

If you are using full-screen CMS, you will not need to define PF keys for the RETRIEVE function, since the default setting for CMS PF6 is RETRIEVE. (In full-screen CMS, the RETRIEVE function only works in a backward direction.) Full-screen CMS also provides you with another method of retrieving commands that you previously entered. Simply scroll your screen back to a command you wish to reenter, position the cursor over the command, update any portion of the command if necessary, and press ENTER. The (updated) command will be reentered.

Setting Program Function Keys

If there are commands that you use frequently, you can set the program function (PF) keys on your terminal to execute them. Although there is one set of function keys (1 through 24) on your terminal, these keys can have different settings in various environments.

For example, when you first LOGON, you might set your PF keys to perform certain functions. Then, when you enter different CMS environments, your PF keys may have entirely different settings. Chapter 11, "Looking at VM/SP Through Windows," provides details on PF keys in full-screen CMS and in the WM environment. The remainder of this section will concentrate on setting PF keys for use when full-screen CMS is set off.

Some examples of commands you might wish to catalog on your CP and CMS PF keys are:

```
#CP QUERY READER ALL
#CP QUERY PRINTER ALL
QUERY ACCESSED
```

To set function keys 1, 2, and 3 to perform these command functions, enter:

```
cp set pf1 immed "#cp query reader all
cp set pf2 immed "#cp query printer all
cp set pf3 immed query accessed
```

Note: When you want to execute a #CP function with a PF key, or you want a PF key to execute a series of commands, you must use the logical escape symbol (^) when you enter the SET command.

You can change a PF key setting any time during a terminal session, according to your needs. This example shows how you can change the setting of the PF5 key:

```
cp set pf5 immed xedit test file"#bo"#input line"#file
```

sets the PF5 key as:

```
XEDIT TEST FILE#BO#INPUT LINE#FILE
```

Then, when you press PF5, VM/SP will XEDIT the file TEST FILE, input the word "line", and write the file to file mode A.

Note: Throughout this book, you may see references to the term *file mode A*. This term is used to refer to a Shared File System (SFS) directory or to a minidisk that is accessed with a file mode of A. You may also see references to *A-disk*, which is another term sometimes used to refer to the directory or minidisk accessed with a file mode of A.

You can also set all of your program function keys in your PROFILE EXEC so they are set each time you load CMS. To change the setting of the PF5 key in your PROFILE EXEC, you could add to your PROFILE EXEC the line:

```
CP SET PF5 IMMED XEDIT TEST FILE #BO# INPUT LINE #FILE
```

Then, the next time you load CMS, the PF5 key will be set to perform this function. In this instance, you would not need to include the logical escape characters because the command was entered from a file.

The above examples use the IMMED operand of the SET command, which specifies that the function is performed as soon as you press the PF key. You can also set a key so that it is delayed; that is, the command or data line is placed in the user input area. Then, you must press the ENTER key to execute the command. You may modify the line before you enter it. This is the default setting (DELAY) for program function keys.

For example, you might set a key as:

```
QUERY ACCESSED X@
```

When you press this PF key, the command is placed in the user input area, with the cursor positioned following the "@" logical character delete symbol; you can enter the mode letter of the directory or minidisk you are querying before you press the ENTER key to execute the command. If you enter "A", the "X" is deleted, and the resulting command as seen by CMS is QUERY ACCESSED A. For more information on using the logical character delete symbol, refer to "Logical Line Editing Symbols" in the *VM/SP CP General User Command Reference*.

For more details on setting PF keys, see the *VM/SP CP General User Command Reference* and the *VM/SP Terminal Reference*.

Using Full-Screen CMS

Another way to facilitate your work is to use full-screen CMS. You can do this by entering on the command line, SET FULLSCREEN ON or by putting this command in your PROFILE EXEC. Your system administrator may also put this command in your system profile (SYSPROF) EXEC. In this case, you will automatically be in full-screen CMS when you IPL CMS. For more information on SYSPROF, refer to *VM/SP Administration*.

You can take advantage of full-screen CMS capabilities to perform several functions such as entering commands from almost anywhere on your physical screen and displaying messages and other information in windows on your screen.

For more information on how to use full-screen CMS and windowing support, see Chapter 11, "Looking at VM/SP through Windows."

Display Screen Characteristics

Messages

During a CP or CMS session (other than an edit session) messages and warnings from the system operator or other users are highlighted. This distinguishes these messages from other output and lessens the possibility of important messages being lost or ignored.

A major feature of a 3270 display screen is the screen status area, which indicates, at all times that you are logged on, the current operating condition your virtual machine is in. Understanding the status conditions can help you use CMS on a 3270 more effectively.

VM Status Notices

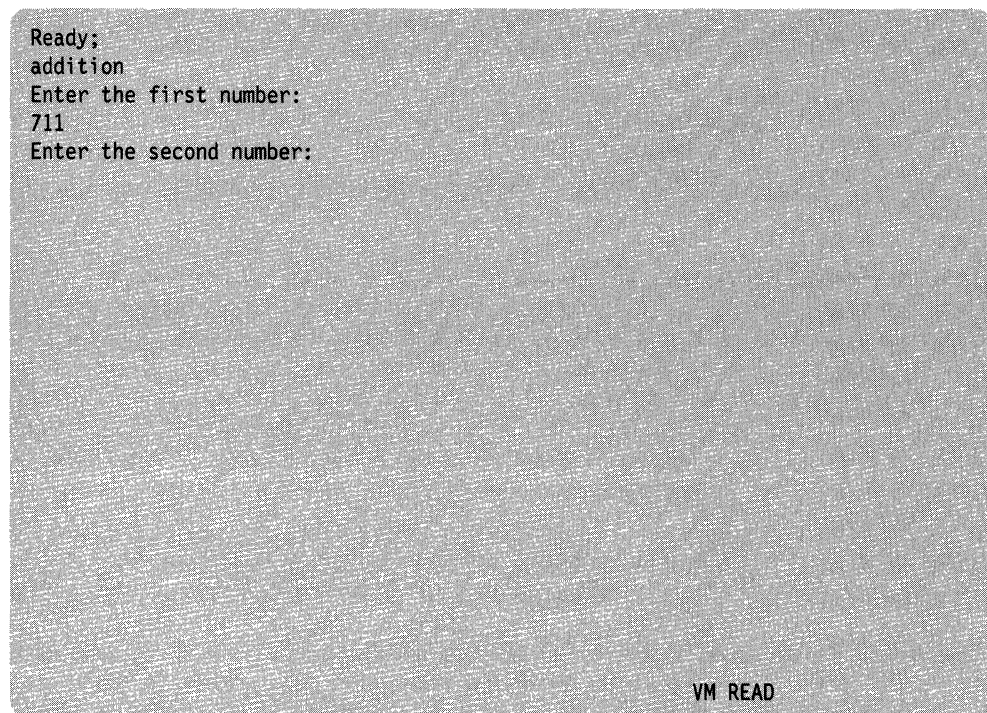
The screen status area indicates one of the following conditions:

CP READ

This status notice is the first one you see; it indicates that the terminal is waiting for a line to be read by the control program. You can enter only CP commands when the screen status area indicates a CP READ.

VM READ

This status notice indicates that your terminal is waiting for a line to be issued to your virtual machine; you may be in the CMS environment, in the edit environment, or you may be executing a program or an exec that has issued a read to the console. Following is an example of this status notice:



```
Ready;
addition
Enter the first number:
711
Enter the second number:

VM READ
```

Figure 2. Sample Status Notice in VM

While in VM READ, you can pass null lines or Immediate commands to CMS. This procedure is particularly useful when you wish to stop execution of an exec that is issuing VM READs to the terminal. You can type HI for Halt Immediate or HX for Halt Execution and have it passed to CMS without being read and interpreted as data by the EXEC. At the VM READ, use the cursor movement key to move the cursor back one space from its current position at the command line. (This results in the cursor being positioned in the lower right corner of the screen, three lines up from the bottom). Press the ENTER key. If your cursor is not returned to the command line (for example, when using VTAM), use the cursor movement key to move the cursor back to the command line. Your terminal will remain in VM READ status. At this point, you can enter HI, HX, or any Immediate command, or if you wish to resume execution of the exec, enter the next line of data.

Refer to the *VM/SP CMS Command Reference* for more information on using HI or HX.

RUNNING

This status notice indicates that your virtual machine is operating. Once you have loaded CMS and are using the CMS environment, this status is almost continually in effect, even when you are not currently executing a command or program.

You can alter the way this works by using the AUTOREAD function of the SET command. When the AUTOREAD setting is OFF, (the default for display terminals), your terminal displays a RUNNING status after the execution of each CMS command. If you want the terminal to be in a VM READ status following each command, enter:

```
set autoread on
```

The ON setting is the default for typewriter terminals, because a read on a typewriter terminal must be accompanied by the unlocking of the keyboard.

The advantage of keeping your virtual machine in a running status even when it is not actually executing a program is that it makes your terminal ready to receive messages. If your terminal is waiting for a read, either from CP or from the virtual machine, and if a user or a program sends a message to your virtual console, then the message is not displayed until you use the ENTER key to enter a command or null line. When your machine is in a running status, the terminal console is always ready to accept messages.

MORE...

This status notice indicates that your display screen is full, but that there is more data to be displayed. This message, in addition to indicating that there is more data, gives you a chance to freeze your screen's current display so you can continue to examine it, if necessary.

When you see the screen is in a MORE... status, you can either:

- Press the Clear, Cancel, or PA2 keys to clear the screen and see the next screen, or
- Press the ENTER key to put the screen in HOLDING status.

If you do not do either, then after 60 seconds, the screen is cleared and the next screen is displayed.

HOLDING

This status notice indicates that you have pressed the ENTER key to freeze the screen. You must use the Cancel, Clear, or PA2 keys to erase this screen and go on to the next display.

A holding status also results if you have received a message that appeared on this screen. When the screen becomes full, it does not automatically pass to the next display after 60 seconds, but waits until you specifically clear the screen. (This feature ensures that any important messages you receive are not lost.)

NOT ACCEPTED

This status notice indicates that you are trying to enter a command line but the terminal buffer is full and cannot accept it. This message is also issued when you attempt to use the 3270 COPY function and a printer is either not available or not ready.

For more information on full-screen CMS, refer to Chapter 11, “Looking at VM/SP Through Windows.”

Additional Display Screen Capabilities

Both extended highlighting and the seven-color feature are available on the 3279 Models 2 and 3. If you are using 3278 Model 2, 3, 4, or 5, the options for both features will be accepted. However, only the highlight feature is available.

The CP SCREEN command (with its operands) allows you to choose one of three highlighting features (blinking, underscore, or reverse video) and one of seven different colors (red, green, blue, pink, turquoise, yellow, or white) for each screen area.

If you want the input area to be turquoise without highlighting, you should enter:
screen inarea turquoise none

Or, if you want the input area pink and the status area yellow with the blinking highlight, you should enter:

```
screen inarea pink statarea yellow blink
```

The CP QUERY SCREEN command displays the color and extended highlight values currently in effect. For more details on the CP SCREEN command, see the *VM/SP CP General User Command Reference* and for more details on the terminal display areas, see the *VM/SP Terminal Reference*.

You can tailor your XEDIT screen colors with the XEDIT subcommand SET COLOR. Refer to the *VM/SP System Product Editor Command and Macro Reference* for a description of the SET COLOR XEDIT subcommand.

When you are using CMS windowing support, you can tailor your screen for color, extended highlighting, Programmable Symbol Sets (PSS), character attributes, and Double-Byte Character Sets (DBCS). For more information, refer to the *VM/SP System Product Editor Command and Macro Reference* and the *VM/SP CMS Command Reference*.

How VM/SP Responds to Your Commands

CP and CMS respond differently to different types of requests. All CMS command responses (and all responses to CP commands that are entered from the CMS environment) are followed by the CMS ready message. The form of the ready message can vary, because it can be changed using the SET command. The long form of the ready message is:

```
Ready; T=7.36/19.89 09:26:11
```

If you have issued the command:

```
set rdymsg smsg
```

meaning, set the ready message to the short form, the ready message looks like:

```
Ready;
```

Introduction

When you enter a command incorrectly, you receive a message describing the error. The ready message contains the last 5 digits (4 digits for a negative return code) from the command. For example:

```
Ready(00028);
```

indicates that the return code from the command was 28.

A ready message from the command may contain a negative return code; for example:

```
Ready(-0001);
```

indicates that the return code from the command was -1.

Some Sample CP and CMS Command Responses

If you enter a CP or CMS command that requests information about your virtual machine, the response should be the information requested. For example, if you enter the command:

```
query reader all
```

CP responds by showing you the contents of your reader:

```
ORIGINID FILE CLASS RECORDS CPY HOLD DATE TIME NAME TYPE DIST
BROWNL 1725 A PUN 00000009 001 NONE 05/22 10:59:10 BROWNL NOTE G67/33
DEBBIEB 0711 A PUN 00000016 001 NONE 05/21 13:02:54 DEBBIEB NOTE G42/02
```

Similarly, if you issue the CMS command:

```
listfile * assemble c
```

you might receive the following information:

```
JUNK ASSEMBLE C1
MYPROG ASSEMBLE C1
```

If you enter a CP command to alter your virtual machine configuration or the status of your spool files, CP responds by telling you that the task is accomplished. The response to:

```
purge reader all
```

might be:

```
0004 FILES PURGED
```

Some CP commands, those that alter some of the characteristics of your virtual machine, give you no response at all. If you enter:

```
spool e class x hold
```

you receive no response from CP.

Certain CMS commands may issue prompting messages, to request you to enter more information. The SORT command, which sorts CMS files, is an example. If you enter:

```
sort in file a1 out file a1
```

you are prompted with the message:

```
DMSSRT604R Enter sort fields:
```

and you can then specify which fields you wish the input records to be sorted on.

Working with CMS

If you have just logged on for the first time, and you want to try a few CMS commands, enter:

```
query accessed
```

After you enter the command, you will see a response similar to the following:

```
Mode Stat Files Vdev Label/Directory
A      R/W      3  DIR  VMSYSU:VMUSER.
C/A    R/O     765 19C  19CSP6
S      R/O    1321 190  CMS6.0
Y/S    R/O     337 19E  19ESP6
```

Check to see if the listing for file mode A shows **DIR** in the *Vdev* column. If so, you are enrolled in a Shared File System (SFS) file pool, and your system administrator has set up an SFS top directory for you. In SFS, you will have a file space to store your files. Your file space, along with the file spaces of many other users, will reside in a file pool. It is easy to share files with users who belong to your file pool, as well as users in other file pools.

If, instead, the display shows **191** in the *Vdev* column, your files are stored on a 191 minidisk.

Depending on your installation and on your needs as a user, you may complete your work with CMS using either the Shared File System, minidisks, or a combination of both. Unless otherwise stated, the term *CMS file* refers to a file that can be stored in a directory or on a minidisk.

In a situation where you can use both SFS and minidisks, you can choose which files you might wish to share or organize hierarchically and store these files in an SFS file pool; other files could be stored on your minidisks. At any time, you could move files from a minidisk to SFS by using the COPYFILE command. For details on how to do this, see to Chapter 4, "Using the Shared File System."

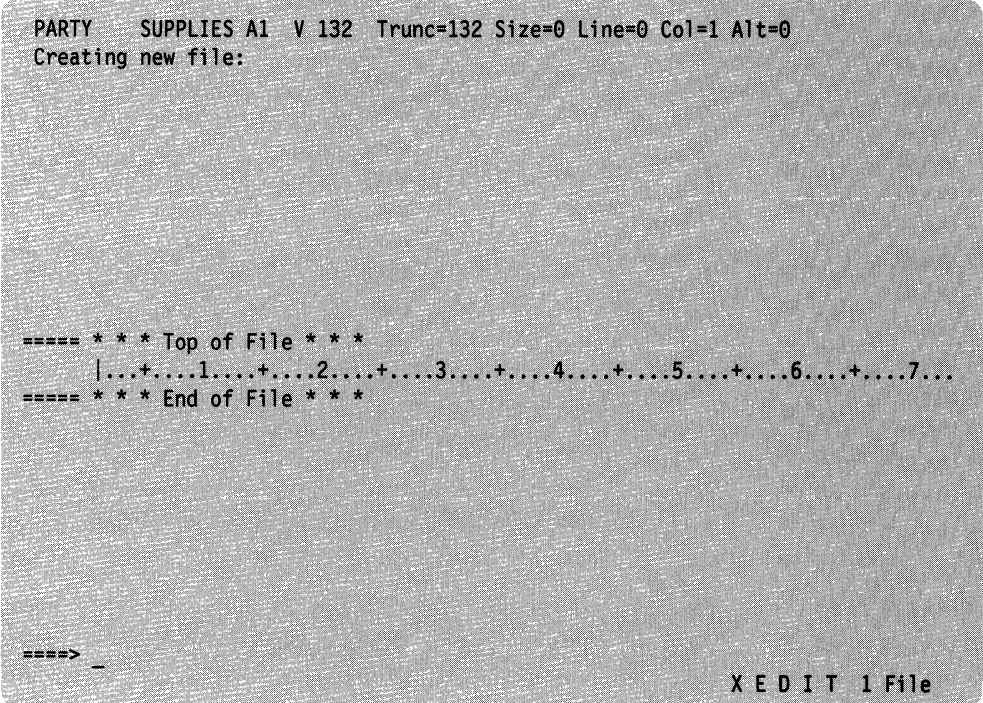
Introduction

Whether your files are stored in an SFS file pool or on minidisks, CMS commands will generally work the same.

You will also be able to use the System Product Editor to create and modify files. For example, you can use the XEDIT command to create a file named PARTY SUPPLIES. Enter:

```
xedit party supplies
```

The display will look like Figure 4.



```
PARTY  SUPPLIES A1  V 132  Trunc=132 Size=0 Line=0 Col=1 Alt=0
Creating new file:

===== * * * Top of File * * *
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...
===== * * * End of File * * *

=====> _
```

X E D I T 1 File

Figure 4. Sample XEDIT Screen

On the command line (next to the arrow) type **INPUT** and press the **ENTER** key. The file is placed in input mode. The cursor is placed automatically on the first line in the input zone, where you enter your data. You are writing input lines that are eventually going to be written onto file mode A.

Enter the following data:

```
balloons
cake
party hats
ice cream
guests
```

```

PARTY  SUPPLIES A1  V 132  Trunc=132  Size=12  Line=0  Col=1  Alt=0
Input mode:

* * * Top of File * * *
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...
balloons
cake
party hats
ice cream
guests

====> * * * Input Zone * * *

Input-mode 1 File

```

Figure 5. Sample XEDIT Screen in INPUT Mode

Now, press the ENTER key, the screen moves up so that you can enter more data.

When you are finished entering data, press the ENTER key again to return to edit mode.

To keep this file in permanent storage, you type FILE on the command line and press the ENTER key. You should see a message that looks something like this:

```
Ready;
```

Even though the file has disappeared from your screen, the editor has saved it.

Let's check and see if the file was really saved. We will use the LISTFILE command to list the files on file mode A with the file name of PARTY. Enter:

```
listfile party
```

you should see the following:

```
PARTY SUPPLIES A1
```

Request a display of the file, using the TYPE command. Enter:

```
type party supplies
```


You should see the following:

```
BALLOONS  
CAKE  
PARTY HATS  
ICE CREAM  
GUESTS
```

Because you really do not need this file, you can erase it from your permanent storage using the ERASE command. Enter:

```
erase party supplies
```

When you receive the ready message (Ready;), you know that the file was erased. You can check to see if it really was erased. Use the LISTFILE command again to list the files on file mode A with the file name of PARTY. Because you just erased the file, you will receive the following message:

```
File not found  
Ready(00028);
```

Most CMS commands create or reference files, which are stored within your file space in an SFS file pool or on minidisks. These commands are as easy to use as the commands shown above.

Storing Your Files

You will be using CMS on a *virtual machine*. A virtual machine is similar to a real machine in that it has access to direct access storage device (DASD) storage. This storage is either within a Shared File System file pool or on a minidisk.

A *storage group* is a subset of minidisks within an SFS file pool. Within a storage group, potential space is allocated for a user. This individual allocation of space is called a user's *file space*. In each user's file space, individual files are organized in hierarchical structures called *directories*. File spaces can contain CMS files or CMS simulations of OS data sets.

A minidisk is a location on a real DASD that has been allocated for storage of a user's files. Minidisks can also be formatted for use as an OS or DOS disk.

For CMS applications, it does not matter whether your files are stored within a file space or on a minidisk. Regardless of the location of your data, CMS commands will generally function the same.

If you use SFS, your files will be stored in a file space allocated for you by your system administrator. For information on the size of your SFS file space, you can use the QUERY LIMITS command. (See QUERY LIMITS in the *CMS Command Reference*.)

To share SFS files, you grant other users authority to the files or directories of your choice. They do not need to link to your minidisks to share your files.

Accessing Your Directories or Minidisks

When you use the Shared File System, the directories within your file space are usually accessed so that you can manipulate them using CMS commands. By accessing directories and establishing a file mode letter for them, you can also save time typing many commands. If a command accepts a file mode, you can simply specify the file mode, rather than the entire directory name, to execute the command.

If your files are stored on minidisks, which are portions of a physical volume defined using CP, you must link to the minidisks before you can access them. For information on how to link to minidisks refer to Chapter 5, "Storing Your Files on Minidisks."

By accessing your directories or minidisks and establishing file mode letters for them, you can control:

- The command search order for programs executing in your virtual machine (Refer to Chapter 3, "CMS File System" for further information on CMS command search order.)
- Which directories or minidisks are to contain the new files that you create.

If you want to know which file modes (directories or minidisks) you currently have access to, enter:

query accessed

You will see a display like this:

Mode	Stat	Files	Vdev	Label/Directory
A	R/W	3	DIR	VMSYSU:VMUSER.
C/A	R/O	765	19C	19CSP6
S	R/O	1321	190	CMS6.0
Y/S	R/O	337	19E	19ESP6

The *Mode* column indicates the mode letter used to access the directory or minidisk.

Stat gives the status of the directory or minidisk: R/W (read/write) or R/O (read-only). When a directory is accessed, the status of the directory is determined by its ownership:

- If the user is the directory owner, then the directory is accessed as read/write, unless the directory is accessed as an extension of the file mode. (Different rules apply for extensions. These conditions are discussed in Chapter 3, "CMS File System").
- If the user is not the owner of the directory, the directory is *always* accessed as read-only, regardless of whether or not the user has read or write authority on the directory.

Files displays the number of entries in the directory or the number of files on a minidisk. This total can include subdirectories, revoked or erased aliases, and files. (These terms will be discussed in Chapter 4, "Using the Shared File System.")

The *Vdev* column displays **DIR** for directories or the virtual address for minidisks.

Introduction

Label/Directory shows the complete name of the directory or the label assigned to the minidisk when it was formatted. (In the following example, VMSYSU is the file pool name, and VMUSER is the directory.)

Suppose you are authorized to access another directory, and VMSYSU:JACK. was its name. To access it with a file mode of B, you would enter:

```
access vmsysu:jack. b
```

Similarly, if your files are stored on minidisks and you had linked to minidisk 194, you could access it with a file mode of B by entering:

```
access 194 b
```

Releasing Directories or Minidisks

When you no longer need a directory or minidisk that you temporarily accessed, you can release it. Enter the CMS RELEASE command:

```
release b
```

When you want to assign a currently active mode letter to another directory or minidisk, enter the ACCESS command to reassign that mode letter. It is not necessary to release an accessed directory or minidisk before accessing another with the same mode.

When you logoff, any directories or minidisks that you temporarily accessed are automatically released. For further information on accessing and releasing directories, refer to Chapter 4, "Using the Shared File System."

If you are experiencing poor response time and have many directories or minidisks accessed that you do not need, you may want to release some of these directories or minidisks.

Console Output

When you use a 3270 terminal as your virtual machine console, you do not ordinarily retain a console log, as you do on a typewriter terminal. There are many circumstances in which you may want a printed record of your console output. It could be to obtain a copy of program-generated output, or to retain a record of CP and/or CMS commands that resulted in an error condition. There are two techniques you can use in VM/SP to obtain hard copy representations of display terminal sessions: spooling console output and the 3270 copy function.

Spooling Console Output

The CP SPOOL command provides the CONSOLE operand, which lets you begin and end console spooling. Enter

```
spool console start
```

when you want to begin recording your terminal session, and

```
spool console stop
```

when you have finished. In between, you can periodically close the console file to release for printing whatever has been spooled thus far:

```
spool console close
```

Other operands that you can enter are the same as you might specify for any printer file, such as CLASS, COPY, CONT, and HOLD.

An alternate technique is to spool your console to your own virtual reader:

```
spool console start * class a
```

Then, when you close the console file, instead of being released to the CP printer spool file queue, it is routed to your virtual reader, and you can load it onto file mode A as a CMS file. You would do this by receiving the console file using the RDRLIST command. (For more information on the RDRLIST command, see "Receiving a File" on page 215.) You can then use the editor to examine it (or to delete sections you do not need) and use the PRINT command to obtain a printed copy.

If you are using full-screen CMS, you will need to use logfiles to spool information from your terminal. When you set full-screen on, by default, messages and warnings are logged for you. Messages are logged to a file with the file name and file type of MESSAGE LOGFILE; warnings are logged to WARNING LOGFILE.

If you wish, you can create a log of your CMS output or other output. To do this, after you are in full-screen CMS, enter the command SET LOGFILE CMS ON. Your output is then sent to a file with the file name and file type of CMS LOGFILE. Later, you could XEDIT or print this file to obtain a hard copy of the work you completed. See the *VM/SP CMS Command Reference* for more information on the SET LOGFILE command.

Copying Your Screen

If you are using a 3270 display terminal, and you have a 3284, 3286, 3287, 3288, or 3289 printer, you copy the entire screen display currently appearing on the screen. To copy the screen, you have to assign the copying function to a program function key, with the SET command:

```
set pf9 copy
```

Note: The PF key copy function is not available if the printers are dedicated (not to be shared among users).

Then, whenever you want to copy a screen display, you can press the PF9 key (or whichever key you set). The display is printed on any 3270 display printer that you specified using the SET PFnn COPY command. If, when you press the PF key, the screen status area indicates NOT ACCEPTED, it means that the printer is either not ready or not available. When you press the PF key and receive no response, it means that the screen has been copied.

There is a print matrix available to the 3274 and 3276 user that allows control of the display to printer operations. In addition, a local print key is provided on the display terminal that is used for copy operations.

Figure 6 on page 26 is an example of a 3270 screen display that could be copied on the printer. When you use the copy function to copy a screen, all 24 lines of the display screen are copied; the screen status area (indicated as RUNNING in Figure 6) is blank if the 3270 is locally attached. If the 3270 is remotely attached, the entire screen including the screen status area, is copied. You can use the user input area of your screen to key in comments, or your name or user ID, if several users are spooling copy files.

Chapter 2. VM/SP Environments and Mode Switching

When you are using VM/SP, your virtual machine can be in one of two possible *environments*, the control program (CP) environment or the virtual machine environment, which may be CMS. The CMS environment has several subenvironments, sometimes called *modes*. Each environment or subenvironment accepts particular commands or subcommands, and each environment has its own entry and exit paths, responses and error messages. If you have a good understanding of how the VM/SP environments are related, you can learn to quickly change environments and efficiently use your virtual machine.

This chapter introduces the CP and CMS environments that you use and describes:

- Entry and exit paths
- Command subsets that are valid as input.

Figure 7 summarizes the VM/SP command environments and lists the commands and terminal paths that let you from one environment to another.

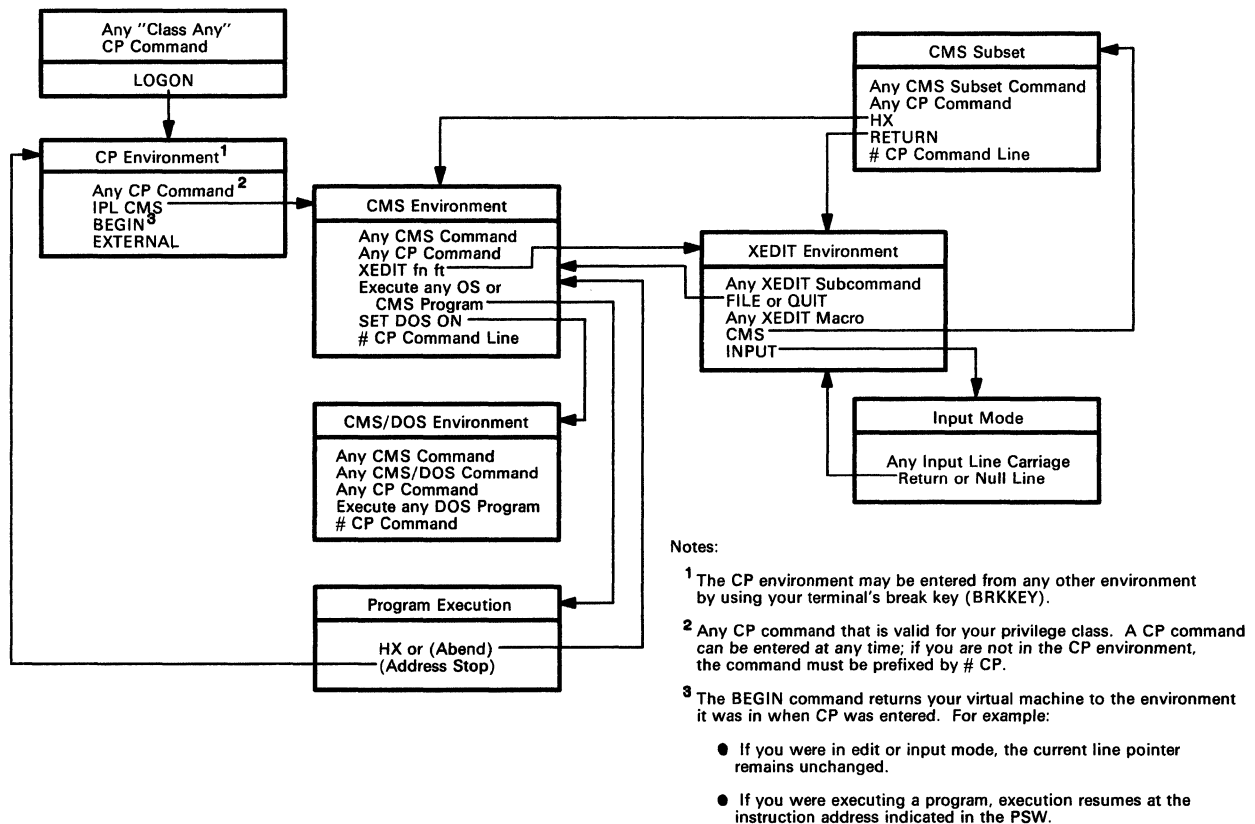


Figure 7. VM/SP Environments and Mode Switching

With the exception of input mode in the edit environment, you can always determine which environment your virtual machine is in by pressing the Return or ENTER key on a null line. The responses you receive and the environments they indicate, are:

Response	Environment
CP	CP
CMS	CMS
CMS (DOS ON)	CMS/DOS
CMS SUBSET	CMS Subset

CP Environment

Once you log on to VM/SP, your virtual machine is in the CP environment. In this environment, you can enter any CP command that is valid for your privilege class. This publication assumes that you are a general, or class G, user. You can find information about the commands in the *VM/SP CP General User Command Reference*.

Only CP commands are valid terminal input in the CP environment. You can, however, preface a CP command with the characters *CP* or *#CP*, followed by one or more blanks, although it is not necessary. These functions are described in "The CMS Environment" on page 29.

You can enter CP commands from other VM/SP environments. There may be times during your terminal session when you want to enter the CP environment to enter one or more CP commands. You can do this from any other environment by doing either of two things:

1. Enter the command:

```
#cp
```

2. Use your terminal's Attention key (PA1 or equivalent). On a 2741 terminal, you usually press the Attention key twice, quickly, to enter the CP environment.

The following message indicates that your virtual machine is in the CP environment:

```
CP
```

After entering whatever CP commands you need to use, you return your virtual machine to the environment or mode that it came from by entering the CP command

```
begin
```

which begins execution of your virtual machine.

Note: Once you set full-screen CMS on, the PA1 key no longer serves as an Attention key but performs a windowing function. If you want to use an Attention key to enter the CP environment, you will need to set another key to the attention function. When you set full-screen CMS off, the CP TERMINAL BRKKEY remains as NONE. To reset it to PA1 (the default setting), use the CP TERMINAL command. For more information, see the BRKKEY option of the CP TERMINAL command in the *VM/SP CP General User Command Reference*.

The CMS Environment

You enter the CMS environment from CP by entering the IPL command, which loads CMS into your virtual storage area. If you are planning to use CMS for your entire terminal session, you should not have to IPL again unless a program failure forces you into the CP environment.

When you enter the IPL command, specify the named system CMS at your installation. For example:

```
ipl cms
```

When your virtual machine is in the CMS environment, you can enter any CMS command and any of the CP commands that are valid for your user privilege class. You can also execute many of your own OS or DOS programs. For more information, see the *VM/SP Application Development Guide for CMS* book.

You can enter CP commands from CMS in any of the following ways:

- Using the implied CP function of CMS (See *Note*.)
- With the CP command
- With the #CP function.

Note: For the most part, you can enter any CP command directly from the CMS environment. This implied CP function is controlled by an operand of the CMS SET command, IMPCP. You can determine whether the implied CP function is in effect for your virtual machine by entering the command:

```
query impcp
```

If the response is:

```
IMPCP      = OFF
```

you can change it by entering:

```
set impcp on
```

When the implied CP function is set off, you must use either the CP command or the #CP function to enter CP commands from the CMS environment. CP commands that you execute in EXEC procedures should be prefaced by the CP command, regardless of the implied CP setting. An example of using the CP command is:

```
cp close punch
```

When you enter CP commands from the CMS environment either implicitly or with the CP command, you receive, in addition to the CP response (if any), the CMS ready message. If you use the #CP function, discussed next, you do not receive the ready message.

You can preface any CP command with the characters #CP, followed by one or more blanks. When you enter a CP command this way, the command is immediately processed by CP; it is as if your virtual machine were actually in the CP environment.

Protected Application Environment

A protected application environment provides special enhancements to prevent users with no need to interact with CP from accidentally entering the CP environment.

When you are operating in a protected application environment:

- Pressing the Attention key will not cause you to enter CP mode.
- The terminal break key is set to NONE.
- CP attempts to initiate an automatic re-IPL upon encountering an error that causes CMS to abnormally end (abend).

A protected application environment is entered by using the CONCEAL option of the SET command or the directory OPTION control statement. Your system administrator sets the directory OPTION control statement so that you would be placed in a protected application at LOGON time. You can use the CONCEAL option of the SET command to enter a protected application at any time during your terminal session.

To issue SET CONCEAL, you would simply enter:

```
set conceal on
```

Either way a protected application is entered, the terminal break key will always be set to NONE. SET CONCEAL OFF returns the break key to its default setting of PA1, for local, remote, and VM/VTAM 3270 graphics terminals.

For more information on the SET CONCEAL command, see the *VM/SP CP General User Command Reference*.

XEDIT and CMS Subset

The System Product Editor is a VM/SP facility that lets you create and modify data files that reside on minidisks or Shared File System directories. The editor environment, more commonly called the edit environment, is entered when you enter the CMS command XEDIT, specifying the identification of a data file you want to create or modify. Complete information about the System Product Editor, invoked via the XEDIT command, is found in the *VM/SP System Product Editor Command and Macro Reference* and the *VM/SP System Product Editor User's Guide*. For introductory tutorial information about editing, see the *VM/SP CMS Primer*. For example

```
xedit party supplies
```

is the command you would use to enter the edit environment to either create a file called PARTY SUPPLIES or to make changes to a file that already exists under that name.

When you enter the edit environment, your virtual machine is automatically in edit mode, where you can enter XEDIT subcommands, CMS commands, or CP commands. After you enter the XEDIT subcommand

```
input
```

data lines that you enter are considered input to the file. To return to edit mode, you must enter a null line.

If you enter the XEDIT subcommand

CMS

the editor responds

CMS subset

and your virtual machine is in CMS subset mode. At this point, you can enter any CMS command that is allowed in CMS subset mode. Commands that execute in the user area are not allowed in the CMS subset environment. You can also enter CP commands. To return to edit mode, you use the special CMS subset command, RETURN. If you enter the Immediate command HX, your editing session abnormally terminates and your virtual machine returns to the CMS environment.

For more information on CMS subset, see the *VM/SP Application Development Guide for CMS*.

When you are finished with an edit session, you return to the CMS environment by issuing the FILE subcommand, which indicates that all modifications or data insertions that you have made should be written onto a minidisk or Shared File System directory. Otherwise, you can enter the subcommand QUIT, which tells the editor not to save any modifications or insertions made since the last time the file was written.

CMS/DOS

If you are a VSE user, the CMS/DOS environment provides you with all the CMS interactive functions and facilities, as well as special CMS/DOS commands that simulate VSE functions. The CMS/DOS environment becomes active when you enter the command:

```
set dos on
```

When your virtual machine is in the CMS/DOS environment you can enter any command that would be valid in the CMS environment, including the facilities of XEDIT, DEBUG, and EXEC. You cannot enter CMS commands or program modules that load and/or execute programs using OS macros or functions.

The following commands are provided in CMS/DOS to test and develop DOS programs, and to provide access to VSE libraries:

ASSGN	DOSPLI	LISTIO
CATCHECK	DSERV	OPTION
DLBL	ESERV	PSERV
DOSLIB	FETCH	RSERV
DOSLKED	FCOBOL	SSERV

Your virtual machine leaves the CMS/DOS environment when you enter the command:

```
set dos off
```

If you reload CMS (with an IPL command) during a terminal session, you must also reenter the SET DOS ON command to return to the CMS/DOS environment. For more information about the CMS/DOS environment, see the *VM/SP Application Development Guide for CMS*.

Interrupting Program Execution

When you are executing a program under CMS or executing a CMS command, your virtual machine is not available for you to enter commands. There are, however, ways that you can interrupt a program and halt its execution: temporarily, (in which case you can resume its execution), or permanently (in which case your virtual machine returns to the CMS environment). In both cases, you interrupt execution by creating an *attention interruption*, which may take two forms:

- An attention interruption to your virtual machine operating system
- An attention interruption to the control program.

Using the Attention Key

Attention interrupts result in what are known as VM or CP “reads” being presented to your virtual console. The two keys on your 3270 keyboard that signal interruptions are the PA1 key -- REQ key on a 3278 Model 2A -- and the ENTER key. Throughout this publication, interruption signaling has been described in terms of the Attention key.

You can enter the CP environment by pressing the PA1 key. Whenever you press this key, your virtual machine is placed in a CP READ status, and you can enter any CP command. From the CP environment, you must enter the CP command BEGIN to resume execution of your virtual machine. On a typewriter terminal, the keyboard unlocks when a read occurs.

Whether you have to press the Attention key once or twice depends on the terminal mode setting in effect for your virtual machine. This setting is controlled by the CP TERMINAL command:

```
terminal mode vm
```

The setting VM is the default for virtual machines; you do not need to specify it. The VM setting indicates that one depression of the Attention key sends an interruption to your virtual machine, and that two depressions result in an interruption to CP.

The CP setting for terminal mode, which is the default for the system operator, indicates that one depression of the Attention key results in an interruption to CP. If you are using your virtual machine to run an operating system other than CMS, you might wish to use this setting. Enter the command:

```
terminal mode cp
```

Note: Once you set full-screen CMS on, the PA1 key no longer serves as an Attention key but performs a windowing function. If you want to use a break key, you will need to set another key to the attention function. When you set full-screen CMS off, the CP TERMINAL BRKKEY remains as NONE. To reset it to PA1 (the default setting), enter the CP TERMINAL command.

Interrupting Your Programs

On a typewriter terminal, the Attention key, pressed once, causes a virtual machine interruption (if the terminal mode is set to VM); you must use it when you want to enter an Immediate command, such as HT or HX. On a display terminal, you can enter these commands whenever your virtual machine is in a running status, without having to signal an interruption before you enter the command.

Sometimes, however, if your terminal is rapidly displaying output you must wait until the screen is full and the screen status area indicates a MORE... status before you attempt to enter the HT or HX command.

The ENTER key can also be used as an interruption signaling key. If you press it once when your virtual machine is running, you will place your virtual machine in the VM READ status, so you can enter a command.

While in VM READ, you can pass null lines or Immediate commands to CMS. This procedure is particularly useful when you wish to stop execution of an Exec that is issuing VM READs to the terminal. You can type HI or HX and have it passed to CMS without being read and interpreted as data by the Exec. At the VM READ, use the cursor movement key to move the cursor back one space from its current position at the command line. (This results in the cursor being positioned in the lower right corner of the screen, three lines up from the bottom). Press the ENTER key. If your cursor is not returned to the command line (for example, when using VTAM), use the cursor movement key to move the cursor back to the command line. Your terminal will remain in VM READ status. At this point, you enter HI, HX, or any Immediate command, or if you wish to resume execution of the exec, enter the next line of data.

If you are in either line mode or full-screen CMS and wish to interrupt execution of an exec issuing a read (indicated by a VM READ status in line mode or Enter your response in vscreen CMS in full-screen CMS), type any Immediate command prefixed by a #. Instead of being passed to the exec, your command will be interpreted immediately. For instance, to halt interpretation of the exec, you would type #HI.

When you are entering an Immediate command prefixed by #, nothing else may be entered along with the Immediate command. For example, if you type ABCD#HI (where ABCD is any data) then ABCD will be passed to the exec and HI will be placed in the console stack. The HI Immediate command will not be executed (not even when the exec issues another read). If you type HI#ABCD, then HI will be passed to the exec and ABCD will be placed on the console stack.

Note: The Immediate command HT is the only exception to this rule. You are permitted to follow a line of data with #HT. For example, if you enter ABCD#HT in response to a read by an exec, then the HT command will be executed first and ABCD will be passed to the exec.

Virtual Machine Interruptions

While a command or program is executing, if you press the ENTER key on a 3270 (or the Attention key once on a 2741), you create a virtual machine interruption.

Halting Screen Displays

When your terminal is displaying successive screens of output from a program or a CMS command, use the HT or HX Immediate commands to halt the display or the execution of the command, respectively. If your terminal is writing the information at a very rapid rate, you can have difficulty entering the HT or HX command. In these circumstances, press the ENTER key twice to force your terminal to a CP READ status. Then, you use either the CP REQUEST or ATTN command to signal a virtual machine read. When the screen status area indicates VM READ, you enter HX or HT. The program halts execution, your terminal accepts an input line, and you can:

- Terminate the execution of the program by issuing an Immediate command to halt execution:

```
hx
```

The HX command causes the program to abnormally terminate (abend).

Note: If you get no response after using the HX command, you should enter #CP, and then enter IPL CMS to reload CMS. After you receive the VM READ status in the lower-right corner of your screen, press the enter key once more.

- Enter a CMS command. The command is stacked in a console stack and is processed by CMS when your program is finished executing and the next virtual machine read occurs. For example:

```
print abc listing
```

After you enter this line, the program resumes execution.

- If the program is directing output to your terminal and you wish only to halt the terminal display, use the Immediate command:

```
ht
```

The program resumes execution. Terminal output can also be immediately suppressed when you enter a command by placing #HT after the command. For example, you would enter

```
tape dump #ht
```

to suppress output from the TAPE DUMP command. The logical line end character (#) lets the Immediate command HT be accepted; program execution proceeds without typing.

You can, if you want, cause another interruption and request that typing be resumed by entering the RT (resume typing) command:

```
rt
```

- Enter a null line; your program continues execution. The null line is stacked in the console stack and read by CMS as a stacked command.

HX, HT, and RT are three of the CMS Immediate commands. They are *immediate* because they are executed as soon as they are entered. Unlike other commands, they are not stacked in the console stack. You can only enter an Immediate command following an attention interruption.

Immediate commands that are entered while a command or program is running (the status is not VM READ or Enter your response in vscreen CMS) should not be prefixed with #. If they are, they will not be executed at the time they are entered. Instead, they will be executed when the next read is issued.

Control Program Interruptions

You can interrupt a program and directly enter the CP environment by pressing the PA1 key on a 3270 or by pressing the Attention key twice, quickly, on a 2741. Then, you can enter any CP command. To resume the execution of the program, enter the CP command:

```
begin
```

If your terminal is operating with the terminal mode set to CP, pressing the Attention key once places your virtual machine in the CP environment.

Note: Remember that in full-screen CMS mode, PA1 pops the WM window. If you wish to override this default setting, use the BRKKEY option of the CP TERMINAL command. When you set full-screen CMS off, the CP TERMINAL BRKKEY remains as NONE. To reset it to PA1 (the default setting), use the CP TERMINAL command. For more information, see the *VM/SP CP General User Command Reference*.

Address Stops and Breakpoints

You can halt program execution after an instruction at a particular address with the CP PER command. Entering the command

```
per i r 201ea
```

causes program execution to halt after the instruction at location X'201EA'. A program may also be interrupted by an instruction address stop, which you specifically set by the CP command ADSTOP. For example, if you issue the command:

```
adstop 201ea
```

an address stop is set at virtual storage location X'201EA'. When your program reaches this address during its execution, it is interrupted and your virtual machine is placed in the CP environment, where you can enter any CP command, including another ADSTOP command, before resuming the execution of your program with the CP command BEGIN.

Using the 3270 Text Feature

If you have a 3277 or 3278 display station equipped with the Data Analysis Text keyboard, you can key in, as well as display, all of the special text characters. For a description of these characters, see the *VM/SP Terminal Reference*. These characters are in addition to those available with standard EBCDIC 3270 terminals. If you have a suitably equipped printer attached to your 3270, you can use the SET PFnn COPY function to obtain a printed copy of the screen.

Note: This procedure for obtaining a printed copy of your screen pertains to sessions when you *are not* using full-screen CMS. See "Copying Your Screen" on page 25 for information on how to copy your screen in full-screen CMS.

When you want to activate the text feature, and use the special characters, enter the command:

```
terminal text on
```

The TERMINAL TEXT ON command automatically forces the TERMINAL APL OFF command. Now, you can use any of the special characters when you enter, change, or locate text lines in a file.

You leave the special text environment by entering the command:

```
terminal text off
```

See the *VM/SP CMS Command Reference* and *VM/SP System Product Editor Command and Macro Reference* for more information on using the SET TEXT commands to select appropriate translation tables for special characters.

Error Situations

If you do not have the appropriate text hardware feature on your 3270, but attempt to display a file that contains the characters, the characters appear as blanks on a 3277, and as hyphens on a 3276 and a 3278.

If you inadvertently enter the `TERMINAL TEXT ON` command while using a terminal that does not have the text capability, you must do the following to return to regular operating procedures:

1. Press the PA1 key to enter the CP environment.
2. Enter, in uppercase letters only, the command:

```
TERMINAL TEXT OFF
```

Notes:

1. The 3270 text hardware feature is activated by a key, not a switch. Each time you press the `TEXT On/Off` key, you reverse its setting. If your terminal has a red light on the text keyboard, it will be illuminated when the text feature is on.
2. Compound characters, such as a character/-backspace/-character combination, are still entered and displayed as three characters. The screen position occupied by the backspace character appears as a blank because the character (X'16') is nondisplayable. For information on displaying nondisplayable characters, see the description of the `NONDISP` commands in the *VM/SP CMS Command Reference* and *VM/SP System Product Editor Command and Macro Reference*.

Chapter 3. CMS File System

The file is the essential unit of data in CMS. Files in CMS are unique and cannot be read or written using other operating systems. When you create a file in CMS, you name it using a file identifier. The file identifier consists of three fields:

- File name (fn)
- File Type (ft)
- File Mode (fm) or Directory Name (dirname).

When you use CMS commands and programs to modify, update, or reference files, you must identify the file by using these fields. Some CMS commands allow you to enter only the file name, or the file name and file type; others require you to enter the file mode or directory name as well. This chapter contains information about the things you must consider when you give your CMS files their identifiers and notes on the file system commands that create and modify CMS files.

There are two ways to manage files. Your files can be stored within a Shared File System (SFS) file space or on minidisks. Depending on your system configuration, you may have the option to use both methods of storing files. In this situation, you could store those files that you may want to share in your SFS file space; other files could be stored on minidisks.

This chapter provides general information on how to manage your CMS files whether they reside in SFS directories, minidisks or both. In fact, the term *CMS file* refers to a file that can be stored in a directory or on a minidisk. Chapter 4, "Using the Shared File System," provides further information on managing your files in SFS. For specific minidisk information, refer to Chapter 5, "Storing Your Files on Minidisks."

CMS File Formats

The CMS file management routines write CMS files in fixed physical blocks, regardless of whether they have fixed- or variable-length records. For most of your CMS applications, you never need to specify either a logical record length and record format or block size when you create a CMS file.

When you create a file using one of the CMS editors, the file has certain default characteristics, based on its file type. The special file types recognized by the editor, and their applications, are discussed in "What Are Reserved File Types?" on page 40.

How CMS Files Get Their Names

When you create a CMS file, you can give it any file name and file type you wish. The rules for forming file names and file types are:

- The file name and file type can each be from one to eight characters.
- The valid characters are A-Z, a-z, 0-9, \$, #, @, +, - (hyphen), : (colon), and _ (underscore).

Note: Lowercase letters within a file ID are valid for use within the CMS file system. However, some CMS commands do not support file IDs that contain lowercase letters.

When you enter a command into the VM/SP system, VM/SP translates your input line by either the user-defined input table or by the uppercase table. See the CMS SET INPUT command in the *VM/SP CMS Command Reference*. If you do not have an input table, you can just enter the command in lowercase and VM/SP translates your input line into uppercase characters.

Note: When defining input characters be sure that you will not end up with a file ID containing invalid characters.

The # and @ characters are line editing symbols in VM/SP; when you use them to identify a file, you must precede them with the logical escape symbol ("). See the *VM/SP CP General User Command Reference* for a list of logical line editing symbols.

The third field in the file identifier is either the file mode (fm) or the directory name (dirname). The first character of a directory name must be a letter (A-Z) and the remaining can be A-Z and 0-9. Lowercase letters are allowed, as they are translated to uppercase. You will learn more about directory names in Chapter 4, "Using the Shared File System." Also, the *VM/SP CMS Command Reference* provides more information.

The file mode indicates the file mode letter (A-Z) currently assigned to the SFS directory or minidisk where you want the file to reside. When you use the editor to create a file, and you do not specify this field, the file you create is written to the directory or minidisk accessed with a file mode of A.

The file mode letter, for any file, can change during a terminal session. Suppose when you log on, your top directory is accessed with a file mode of A. (Your top directory is the directory assigned to you by the SFS administrator at the time you are enrolled in a file pool.) A file in that directory named SPECIAL EVENTS would have a file identifier of:

```
SPECIAL EVENTS A
```

If, however, you later access another directory or minidisk with a file mode of A, and access your top directory with a file mode of B, then the SPECIAL EVENTS file will have a file identifier of:

```
SPECIAL EVENTS B
```

Duplicate File Names or File Types

On a given directory or minidisk, you can give the same file name to as many files as you want, providing you assign them different file types. Or you can create many files with the same file type but different file names. But you cannot have files with the same file name and file type on one minidisk or in one directory. That is, you could not have two files named COST ESTIMATE on file mode A.

For the most part, file names that you choose for your files have no special significance to CMS. If, however, you choose a name that is the same as the name of a CMS command, and the file that you assign this name to is an executable module or exec procedure, then you may encounter difficulty if you try to execute the CMS command whose name you duplicated.

For an explanation of how CMS identifies a command name, see "CMS Command Search Order" later in this chapter.

Using Asterisks (*) and Percent Signs (%) in File IDs

Some CMS commands that manipulate files allow you to enter the file name and/or file type fields as an asterisk (*), indicating that all files of the specified file name/file type are to be modified. Following is a list of some of these commands:

COPYFILE	RENAME	FILELIST
ERASE	TAPE DUMP	LISTFILE

For example, if you enter

```
erase * test a
```

all files with a file type of TEST on file mode A are erased.

Several commands let you perform operations on a group of files that have a file name or file type that begin with the same character string. These commands are shown in the following list:

LISTFILE	CREATE LOCK	GRANT AUTHORITY	CREATE ALIAS
FILELIST	DELETE LOCK	REVOKE AUTHORITY	

For example,

```
listfile t* assemble
```

produces a list of all files on file mode A with file names beginning with the letter T and having the file type of ASSEMBLE. The command

```
listfile tr* a*
```

produces a list of all files on file mode A with file names beginning with the letters TR and having file types beginning with the letter A.

These same commands allow you to use the percent sign (%) as a place holder to mean any single character. For example, if you enter

```
listfile %%% stock
```

you will see a list of all the files on file mode A whose file name is three characters in length and whose file type is STOCK.

```
listfile t% cat
```

produces a list of all the files on file mode A with a three-character file name beginning with the letter T and having a file type of CAT, such as:

```
top cat
the cat
tom cat
```

Equal Signs in Output File IDs

The COMPARE, COPYFILE, RENAME, RECEIVE, and CREATE ALIAS commands let you enter output file identifiers as equal signs (=), to indicate that it is the same as the corresponding input file identifier. For example:

```
copyfile myprog assemble b = = a
```

copies the file MYPROG ASSEMBLE from file mode B to file mode A, and uses the same file name and file type as specified in the input file ID for those positions in the output file ID.

Similarly, if you enter the command

rename temp * b perm = =

all files with a file name of TEMP are renamed to have file names of PERM; the existing file types of the files remain unchanged.

What Are Reserved File Types?

For the purposes of most CMS commands, the file type field is used merely as an identifier. Some file types, though, have special uses in CMS; these are known as *reserved file types*.

Nothing prevents you from assigning any of the reserved file types to files that are not being used for the specific CMS function usually associated with that file type.

Some reserved file types also have special significance to the System Product Editor. When you use the XEDIT command to create a file with a reserved file type, the editor assumes various default characteristics for the file, such as record length and format, tab settings, translation to uppercase, truncation column, and so on.

File Types for CMS Commands

Reserved file types sometimes indicate how the file is used in CMS: the file type ASSEMBLE, for example, indicates that the file is to be used as input to the assembler; the file type TEXT indicates that the file is in relocatable object form, and so on. Many CMS commands assume input files of particular file types, and require you to enter only the file name on the command line. For example, if you enter

```
synonym test
```

CMS searches for a file with a file type of SYNONYM and a file name of TEST. A file named TEST that has any other file type is ignored.

Some CMS commands create files of particular file types, using the file name you enter on the command line. The language processors do this as well; if you are recompiling a source file, but wish to save previous output files, you should rename them before executing the command.

Table 1 on page 41 lists the file types used by CMS commands and describes how they are used.

In addition to these CMS file types, there are special file types reserved for use by the language processors, which are IBM licensed programs. These file types, and the commands that use them, are:

File Types	Commands
COBOL, SYMDMP, TESTCOB	COBOL, FCOBOL, TESTCOB
FORTRAN, FREEFORT, FTnn001,	FORTRAN, FORTGI, FORTHX,
TESTFORT	GOFORT, TESTFORT
PLI, PLIOPT	DOSPLI, PLIC, PLICR, PLIOPT
RPGII	RPGII
VS BASIC, VSBDATA	VS BASIC

For details on how to use these file types, consult the documentation for the appropriate licensed program.

Table 1 (Page 1 of 6). File Types Used by CMS Commands		
File Type	Command	Comments
AMSERV	AMSERV	Contains VSAM access method services control statements executed with the AMSERV command.
ASM3705	ASM3705 GEN3705	Used by system programmers to generate the 3704/3705 control program.
ASSEMBLE	ASSEMBLE	Contains source statements for assembler language programs.
AUTOSAVE	XEDIT	Contains an automatically saved copy of the edited file.
AUXxxxx	UPDATE	Points to files that contain UPDATE control statements for multiple updates.
CNTRL	UPDATE	Lists files that either contain UPDATE control statements or point to additional files.
COPY	MACLIB	Can contain COPY control statements and macros or copy files to be added to MACLIBS.
CSLCNTRL	CSLGEN	Contains control information for building a callable services library (CSL). This control file indicates the routine names, TEXT files, template files, and other CSL control files that are to be used in building the library. Note: CSLCNTRL is a default file type; a different file type may be specified by the user on the CSLGEN command and in the CSL control files.
CSLLIB	CSLGEN CSLLIST RTNLOAD	Contains a callable services library (CSL), generated by CSLGEN, for use on DASD.
CSLSEG	CSLGEN CSLLIST RTNLOAD	Contains a callable services library (CSL), generated by CSLGEN, for use in a logical saved segment.
DIRECT	DIRECT	Contains entries for the VM/SP user directory file. The system programmer controls this file.
DCSSMAP	LANGGEN	Shows the location of each application text deck that was loaded into the discontinuous saved segment.
EXEC	EXEC GEN3705 LISTFILE	Can contain sequences of CMS or user-written commands, with execution control statements.
EXPAND	EXPAND	Contains a list of control sections and the size of the expansion.
GCS	EXEC	Can contain sequences of GCS or user-written commands, with execution control statements.
GLOBALV	DEFAULTS GLOBALV	Contains variables used by GLOBALV.

Table 1 (Page 2 of 6). File Types Used by CMS Commands		
File Type	Command	Comments
GROUP	GROUP	Contains entries for the following data blocks used to describe a GCS virtual machine group: CONFIGURATION, SEGMENT, and AUTHUSERS.
HELPABBR HELPAVS HELPCMS HELPCMSQ HELPCMSS HELPCP HELPCPOT HELPCPQU HELPCPSE HELPEDIT HELPEXC2 HELPEXEC HELPGROU HELPHelp HELPIPCS HELPMACR HELPMENU HELPMMSG HELPPREF HELPPVM HELPQUER HELPREXX HELPROUT HELPRSCS HELPSET HELPSQLD HELPSRPI HELPTASK HELPTSAF HELPXEDI	HELP	Contains descriptive information for the following: CP, CMS, IPCS, TSAF, and AVS commands; CMS macros; CMS routines; messages; REXX, EXEC 2, and exec statements; CMS editor and System Product Editor subcommands; and menu, task, and command abbreviation lists. Information is also provided for the SQL/Data System Program Product (5688-004) and the Remote Spooling Communications Subsystem (RSCS) Networking Program Product (5664-188 Version 2.3) if they are installed on your system. Note: The HELP facility allows you to create HELP files on a minidisk or within any accessed Shared File System (SFS) directory. However, the HELP system disk cannot be replaced by an SFS directory in a file pool.

Table 1 (Page 3 of 6). File Types Used by CMS Commands

File Type	Command	Comments
\$HLPABBR \$HLPAVS \$HLPCMS \$HLPCMSQ \$HLP CMSS \$HLP CP \$HLP CPOT \$HLP CPQU \$HLP CPSE \$HLP EDIT \$HLP EXC2 \$HLP EXEC \$HLP GROU \$HLP HELP \$HLP IPCS \$HLP MACR \$HLP MENU \$HLP MSG \$HLP PREF \$HLP PVM \$HLP QUER \$HLP REXX \$HLP ROU \$HLP RSCS \$HLP SET \$HLP SQLD \$HLP SRPI \$HLP TASK \$HLP TSAF \$HLP XEDI	HELPCONV	<p>Contains descriptive information for the following: CP, CMS, IPCS, TSAF, and AVS commands; CMS macros; CMS routines; messages; REXX, EXEC 2, and exec statements; CMS editor and System Product Editor subcommands; and menu, task, and command abbreviation lists. Information is also provided for the SQL/Data System Program Product (5688-004) and the Remote Spooling Communications Subsystem (RSCS) Networking Program Product (5664-188 Version 2.3) if they are installed on your system.</p> <p>Note: The HELP facility allows you to create HELP files on a minidisk or within any accessed Shared File System (SFS) directory. However, the HELP system disk cannot be replaced by an SFS directory in a file pool.</p>
LANGGCTL	LANGGEN	The control file used by LANGGEN.
LANGMAP	LANGMERG	Shows where language information is stored within a text deck.
LANGMCTL	LANGMERG	The control file used by LANGMERG.
LIST38PP	PRINT	Is a file that is suitable for printing on a 3800 Model 3 page printer. The PRINT command assumes that the file contains a carriage control character in column one and also assumes the OVERSIZE option.
LIST3800	PRINT	Is a file that is suitable for printing on a 3800 printer. The PRINT command assumes that the file contains a carriage control character in column one and a TRC (Table Reference Character) in column two.
LIST3820	PRINT	Is a file that is suitable for printing on a 3820 page printer. The PRINT command assumes that the file contains a carriage control character in column one and also assumes the OVERSIZE option.

Table 1 (Page 4 of 6). File Types Used by CMS Commands		
File Type	Command	Comments
LISTCPDS	PRINT	Is a file that is suitable for printing on a page printer. The PRINT command assumes that the file contains a carriage control character in column one and also assumes the OVERSIZE option.
LISTING	AMSERV ASSEMBLE ASM3705 COBOL DOSPLI FCOBOL GENMSG LOADLIB PLIOPT PRINT	Listings are produced by the assembler, the language processors, and the AMSERV, GENMSG, and LOADLIB commands.
LKEDIT	LKED	Contains the printer output from the LINK EDIT of a CMS text file or OS object module.
LOADLIB	FILEDEF GLOBAL LKED LOADLIB NUCXLOAD OSRUN QUERY ZAP	Is a library created by the LKED command or the LOADLIB utility command. The GLOBAL or FILEDEF command identifies the libraries that should be searched for program execution. NUCXLOAD loads a member of a CMS LOADLIB library or an OS module library. OSRUN executes a member of a CMS LOADLIB library or an OS module library. Query indicates the libraries that were affected by the GLOBAL command. ZAP modifies an existing LOADLIB member.
LOGFILE	SET LOGFILE	Contains a log of information written to a virtual screen.
LSEG	SEGGEN	Contains records defining the contents of a logical segment.
LSEGMAP	SEGGEN	Shows the name, starting address, minidisk mode, and other information about each object in a logical segment. Created by the SEGGEN command for each logical segment in a physical segment.
MACLIB	GLOBAL MACLIB MACLIST	Library members contain macro definitions or copy files; the MACLIB command creates the library, and lists, adds, deletes, or replaces members. The GLOBAL command identifies which macro libraries should be searched during an assembly or compilation. The MACLIST command lists all members in a specified MACLIB which can edit and enter commands from the list.
MACRO	MACLIB	Contains macro definitions to be added to a CMS macro library (MACLIB).

Table 1 (Page 5 of 6). File Types Used by CMS Commands		
File Type	Command	Comments
MAP	INCLUDE LOAD MACLIB TAPE TXTLIB	Maps created by the LOAD and INCLUDE commands indicate entry point locations; the MACLIB, TXTLIB, and TAPE commands produce MAP files.
MODULE	GENMOD LOADMOD MODMAP NUCXLOAD	module files created by the GENMOD command are nonrelocatable executable programs. The LOADMOD command loads a module file for execution; the MODMAP command displays a map of entry point locations. NUCXLOAD loads a module into free storage and defines it as a nucleus extension.
NAMES	NAMEFIND NAMES	Contains information regarding users with whom you communicate.
NETLOG	RECEIVE SENDFILE	Contains records that log the transmission of files sent by or received by you.
NOTEBOOK	RECEIVE SENDFILE	Contains notes sent to you or sent by you to other users.
PSEG	SEGGEN	Contains records defining the logical segments to be contained in a physical segment.
PSEGMAP	SEGGEN	Shows the starting address and name of the logical segments in the physical segment. Created by the SEGGEN command for each physical segment built.
SYNONYM	SYNONYM	Contains a table of synonyms for CMS commands and user-written exec and module files.
TEMPLATE	CSLGEN	Contains the template information used by CSLGEN to create callable services libraries (CSLs). Template files define the type, length, and usage of each parameter expected by a CSL routine. Note: TEMPLATE is a default file type; a different file type may be specified by the user in CSL control files.
TEXT	ASSEMBLE INCLUDE LOAD TXTLIB	TEXT files contain relocatable object code created by the assembler and compilers. The LOAD and INCLUDE commands load them into storage for execution. The TXTLIB command manipulates libraries of TEXT files.
TXTLIB	GLOBAL TXTLIB	Library members contain relocatable object code. The TXTLIB command creates the library and lists or deletes existing members. The GLOBAL command identifies TXTLIBs to search.
TXTxxxxx	GENMSG CONVERT COMMANDS	Contains the object code for language files (message repositories).
UPDATE	UPDATE	Contains UPDATE control statements for single updates applied to source programs.

Table 1 (Page 6 of 6). File Types Used by CMS Commands		
File Type	Command	Comments
UPDLOG	UPDATE	Contains a record of additions, deletions, or changes made with the UPDATE command.
UPDTxxxx	UPDATE	Contains UPDATE control statements for multilevel updates.
XEDIT	XEDIT	Can contain sequences of XEDIT subcommands or user-written commands, with execution control statements.
ZAP	ZAP ZAPTEXT	Contains control records for the ZAP and ZAPTEXT commands, which are used by system support personnel.

Output Files: TEXT and LISTING

Output files from the assembler and the language processors are logically related to the source programs by their file names. Some of these files are permanent and some are temporary. For example, if you enter the command

```
assemble myfile
```

CMS locates a file named MYFILE with a file type of ASSEMBLE and the system assembler assembles it. If the file is on file mode A, then when the assembler completes execution, the permanent files you have are:

```
MYFILE ASSEMBLE A1
MYFILE TEXT      A1
MYFILE LISTING  A1
```

where the TEXT file contains the object code resulting from the assembly, and the LISTING file contains the program listing generated by the assembly. If any TEXT or LISTING file with the same name previously existed, it is erased. The source input file, MYFILE ASSEMBLE A1, is neither erased nor changed.

Because these files are CMS files, you can use the editor to examine or modify their contents. If you want a printed copy of a LISTING file, you can use the PRINT command to print it. If you want to examine a TEXT file, you can use the TYPE or PRINT command specifying the HEX option.

Note: If a TEXT file contains control characters meaningful to the terminal, the lines in the file may not be displayed in their true form. Therefore, it is suggested you do not use the editor for TEXT files because the results are unpredictable. Instead, use the TYPE or PRINT commands with the HEX option to display TEXT decks. Use ZAPTEXT to modify the TEXT decks.

File Types for Temporary Files

The file types of files created by the assembler and language processors for use as temporary workfiles are:

Table 2. File Types for Temporary Work Files

SYSUT1	SYS001	SYS004
SYSUT2	SYS002	SYS005
SYSUT3	SYS003	SYS006
SYSUT4		

CMS handles all SYSUTx and SYS00x files as temporary files.

The CMS AMSERV command, executing VSAM utility functions, uses two workfiles that have file types of LDTFDI1 and LDTFDI2.

The CMS SET LANGUAGE command creates a temporary file with a file name of SET\$TEMP and a file type of TEXT. This file is created when making user language additions to the message repository, user parser table, or synonym table.

Space is allocated for temporary files on an as-needed basis. They are erased when processing is complete. If a program you are executing is terminated before completion, these workfiles can remain on your minidisk or Shared File System directory. You can erase them.

CMSUT1 Files

The CMSUT1 file name or file type is used by CMS commands that create files. The CMSUT1 file is used as a workfile and is erased when processing is complete. When a command fails to complete execution properly, the CMSUT1 file may not be erased. CMSUT1 files are reserved for system usage, and use of these files may cause unpredictable results.

Note: CMSUT1 files should not be shared.

The commands, and the file names or file types they assign to files they create, are listed as follows:

Command	File Name
AMSERV	fn (file name supplied on the AMSERV command)
CONVERT COMMANDS	COMMANDS
COPYFILE	COPYFILE
DISK LOAD	DISK
DOSLIB	libname (file name of the CMS/DOS phase library)
EDIT	EDIT
ESERV	fn (file name of the ESERV file)
EXECUPDT	X\$EUPD\$X
EXPAND	CMSUT1 (filetype = TEXT)
HELP	HELP
HELPCONV	HELPCONV
INCLUDE	DMSLDR
LANGMERG	'Application ID'NLS
LOAD	DMSLDR
MACLIB	DMSLBM
READCARD	READCARD
RECEIVE	RECEIVE
SENDFILE	SENDFILE
TAPE LOAD	TAPE
UPDATE	fn (the file name of the UPDATE file)
XEDIT	XEDTEMP
ZAPTEXT	CMSUT1 (filetype = TXTLIB)

Note: CONVERT COMMANDS also produces temporary CMSUT2 files with a file name of COMMANDS. LANGMERG produces temporary CMSUT2 files with the a file name composed of the application ID followed by a one-to-five character language id. ESERV produces temporary CMSUT2 files with a file name of the ESERV file.

File Types for Documentation

There are two CMS reserved file types for which the System Product Editor accepts (by default) uppercase and lowercase input data. These are MEMO and SCRIPT.

- You can use MEMO files to document program notes or to write reports
- The SCRIPT file type is used by the SCRIPT command. This command invokes a text processor that is part of the IBM Document Composition Facility program product.

File Mode Letters and Numbers

The file mode field of a CMS file identifier has two characters: the file mode letter and the file mode number.

- The file mode letter is established by the ACCESS command and specifies the directory or minidisk where a file resides: A through Z.
- The file mode number is a number from 0 to 6, which you can assign to the file when you create it or rename it; if you do not specify it, the value defaults to 1.

How you use file mode letters depends on how your files are stored and how you want to use them. For most of the *reading* and *writing* you do of files, you use your directory or minidisk accessed with a file mode of A. File mode A typically has read/write status.

You may access other directories or minidisks in read-only or read/write status. Remember, to access a directory, you must either be the owner of it or be granted authority to it by the owner. To access a minidisk, you must first be linked to it. (For information on linking to minidisks, refer to Chapter 5, "Storing Your Files on Minidisks.")

You load CMS with the IPL command. At this point, if you are enrolled in an SFS file pool, your top directory is accessed with a file mode of A. If your files are stored on minidisks, you may instead have a minidisk at virtual address 191 accessed as your "A-disk."

It is possible that you could be enrolled in a file pool and also be able to use minidisk storage. In this case, if your top directory is accessed with a file mode of A and the file server is down, CMS will try to access your 191 minidisk with a file mode of A. If you do not have a 191 minidisk, you will need to wait for the file server to become available in order to continue your work.

Also, during CMS initialization, the minidisks that control your virtual machine are accessed. Your minidisk at address 190 (the system disk) is accessed with a file mode of S; the minidisk at 19E is accessed as an extension of file mode S, with a file mode letter of Y. File mode S and file mode Y are accessed for only S2 and Y2 files. This is because these are the only files on the file mode S and file mode Y that you, as a user, can see. Therefore, the access commands for these file modes are:

```
access 190 S/S * * S2
access 19E Y/S * * Y2
```

In addition, if you have a minidisk defined at address 192, it is accessed for you as file mode D. If the 192 minidisk has not been formatted, CMS will do it automatically and label the minidisk SCRTCH.

The actual letters you assign to any other minidisk or SFS directory (and you may reassign the letters A, D, and Y), is arbitrary; but it does determine the CMS search order, which is the order of file modes CMS searches when it is looking for a file. The order of search (when all file modes are being searched) is alphabetical: A through Z. If you have duplicate file identifiers on different file modes, you should check your search order before entering commands against that file name to be sure that you will get the file you want. You can find out the current search order by entering the command

```
query search
```

You can also access file modes as logical extensions of other file modes; for example:

```
access 235 b/a
```

The “B/A” indicates that file mode B is to be a read-only extension of file mode A; file mode A is considered the “parent” of file mode B. A file mode may have many extensions, but only one level of extension is allowed.

How Extensions Are Used

If you have a minidisk accessed as an extension of a file mode, the extension is automatically read-only, and you cannot write on it. Therefore, you might access a minidisk as its own extension to protect the files on it from being accidentally overwritten.

For example:

```
access 235 b/b
```

If your files are stored in an SFS file pool, accessing a directory as read-only restricts writing to files in that directory for all commands that use file modes, even if you have write authority to specific files. The only exceptions to this are COPYFILE, XEDIT, and CREATE ALIAS. With these three commands, you will be able to write to a file if you have authority to do so.

To protect a file in an SFS directory from being overwritten, you may want to create a lock. A SHARE lock will allow the owner, as well as other users, only the ability to read the file. To change the file, the owner must enter the DELETE LOCK command. For detailed information on locking files, see Chapter 4, “Using the Shared File System.”

Another use of file extensions is to extend the CMS search order. If you enter a command request to read a file, for example:

```
type alpha plan
```

CMS searches file mode A for the file named ALPHA PLAN and if it does not find it, searches any extensions to file mode A. If you have a file named ALPHA PLAN on file mode B but have not accessed it as an extension of file mode A, CMS will not find the file, and you will have to reenter the command:

```
type alpha plan b
```

Some commands, however, handle file mode extensions differently. The TYPE command defaults to a file mode of A. Therefore, in the example above, when you did not specify a file mode, CMS would search only file mode A and all extensions of file mode A. See the *VM/SP CMS Command Reference* for information on the search order for specific commands.

Note that if you specify a file mode of asterisk (*) or if a particular command defaults to a file mode of asterisk, CMS uses the standard search order (A through Z, without regard to any extensions you have defined). For example, if a certain file was on your file mode W, which you had previously defined as an extension of file mode A (W/A), CMS would alphabetically search file mode A, followed by file mode B, and any remaining file modes until it reached file mode W where the file would be found.

The same applies for commands that search for a file based on a default file type, such as the LOAD command, which defaults to a file type of TEXT. Because these commands usually have a default file mode of asterisk, CMS would use the standard search order.

Additionally, for some CMS commands that read and write a file, if you enter the command and the file to be read is on an extension of a file mode having read/write status, the output file is written to the parent read/write file mode. The COPYFILE command is a good example of this type of command. If you have a file named FINAL LIST on file mode B extension of a read/write file mode A, and if you want to make a copy of the file with a different name, enter

```
copyfile final list a final newlist a
```

after the command is completed, the copy is written onto file mode A. The file on file mode B remains unchanged.

Accessing and Releasing Read-Only Extensions

When you access a file mode as a read-only extension, it remains an extension of the parent file mode as long as both file modes are still accessed. If either is released, the relationship is terminated.

If the parent file mode is released, the extension remains accessed and you may still read files on it. If you access another directory or minidisk at the file mode letter of the original parent file mode, the parent/extension relationship remains in effect.

If you release a read-only extension and access another directory or minidisk with the same file mode letter, it is not an extension of the original parent file mode unless you access it as such. For example, if you enter

```
access 198 c/a
release c
access 199 c
```

file mode C (virtual address 199) is not an extension of file mode A

When To Specify File Mode Letters: Reading Files

When you request CMS to access a file, you have to identify it so that CMS can locate it for you. The commands that expect files of particular file types (reserved file types) let you enter only the file name of the file when you enter the command.

When you execute any of these commands or execute a module or exec file, CMS uses the standard search order to search all of your accessed file modes (directories and minidisks) to locate the file. Some CMS commands that perform this type of search are:

AMSERV	GLOBAL	MODMAP
ASSEMBLE	LOAD	RUN
DOSLIB	LOADMOD	TXTLIB
EXEC	MACLIB	

Some CMS commands require you to enter the file name and file type to identify a file. You may specify the file mode letter; if you do not specify the file mode, CMS searches only file mode A and its extensions when it looks for the file. If you do specify a file mode letter, the file mode you specify and its extensions are searched for the file. Some commands you can use this way are:

FILEDEF	SYNONYM	UPDATE
PRINT	TAPE	
PUNCH	TYPE	

For the STATE command, if you specify the file name, file type and file mode, CMS searched the specified disk and its extensions. If no file mode is specified, CMS searches all accessed disks in CMS search order.

There are some CMS commands that do not search extensions of file modes when looking for files. They include:

- CREATE ALIAS
- CREATE LOCK
- DELETE LOCK
- ERASE
- FILELIST
- GRANT AUTHORITY
- LISTFILE
- QUERY ALIAS
- QUERY AUTHORITY
- QUERY LOCK
- RELOCATE
- RENAME
- REVOKE AUTHORITY

You must explicitly enter the file mode if you want to use these commands to process files that are on extensions. Note that ERASE and RENAME only operate on file modes accessed as read/write if a file mode letter is used. If a directory name is specified instead of a file mode letter, you can ERASE a file regardless of how the directory is accessed.

The following commands search every accessed file mode, regardless of whether they have read-only status or read/write status.

- NAMES
- NAMEFIND

Using Asterisks and Equal Signs

For some CMS commands, if you specify the file mode of a file as an asterisk, it indicates that you either do not know or do not care what file mode the file is on and you want CMS to locate it for you. For example, if you enter

```
listfile myfile test *
```

the LISTFILE command responds by listing all files named MYFILE TEST on your accessed file modes. When you specify an asterisk for the file mode of the COPYFILE, ERASE, or RENAME commands, CMS locates all copies of the specified file. For example,

```
rename temp sort * good sort =
```

renames all files named TEMP SORT to GOOD SORT on all of your accessed file modes having a read/write status. An equal sign (=) is valid in output file IDs for the RENAME and COPYFILE commands.

For some commands, when you specify an asterisk for the file mode of a file, CMS stops searching as soon as it finds the first copy of the file. For example:

```
type myfile assemble *
```

If there are files named MYFILE ASSEMBLE on file mode A and file mode C, then only the copy on file mode A is displayed. The commands that perform this type of search are:

COMPARE	PUNCH	SYNONYM
DISK DUMP	RUN	TAPE DUMP
FILEDEF	SORT	TYPE
PRINT	STATE	XEDIT

For the COMPARE, COPYFILE, RENAME, and SORT commands, you must always specify a file mode letter, even if it is specified as an asterisk.

The CREATE ALIAS and RELOCATE commands require an output file mode or a directory identifier, *dirid*. For more information on the syntax of these commands, see Chapter 4, "Using the Shared File System."

When To Specify File Mode Letters: Writing Files

When you enter a CMS command that writes a file, and you specify the output file mode, CMS writes the file onto the directory or minidisk accessed with that file mode. Some of the commands that require you to specify the output file mode are:

```
COPYFILE
RENAME
SORT
```

Some of the commands that let you specify the output file mode, but do not require it, are:

FILEDEF	TAPE LOAD
GENMOD	TAPPDS
READCARD	UPDATE

When you do not specify the file mode on these commands, CMS writes the output files onto file mode A.

Some CMS commands that create files always write them onto file mode A. The LOAD and INCLUDE commands write a file named LOAD MAP A5. The LISTFILE command creates a file named CMS EXEC on file mode A.

Other commands that do not allow you to specify the file mode write output files either:

- To the file mode from which the input file was read, or
- To file mode A, if the file was read from a read-only file mode.

These commands are:

AMSERV
 MACLIB
 TXTLIB
 UPDATE

The SORT command also functions this way if you specify the output file mode as an asterisk (*).

In addition, many of the language processors, when creating work files and permanent files, write onto the first file mode in your search order having a read/write status, if they cannot write on the source file's file mode or its parent.

How File Mode Numbers Are Used

Every CMS file, regardless of whether it resides in a file space or on a minidisk, has a file mode number associated with it. The file mode number is established when the file is created. All CMS mechanisms that create files (such as the XEDIT command, the FSnnnn macros, or the COPYFILE command) let you specify a file mode number. If you do not specify a file mode number, CMS assigns a default file mode number of 1.

You can change a file mode number using any CMS command that changes or rewrites a file.

Note: When changing a file name with the RENAME command, these rules concerning file mode numbers apply:

- If you use RENAME on an alias, only the file name and file type of the alias are changed; you cannot change the file mode number of the alias.
- If you use RENAME on a base file, you can change the file mode number of the base file, which in turn will cause the file mode number of each alias to the file to be altered.

To change the file mode number of your files, you can use COPYFILE with the REPLACE option:

```
copyfile temp script a1 = = a0 (replace
```

If you use COPYFILE with the REPLACE option to change the file mode number of a base file, the file mode number of all aliases to that base file will also be changed.

If you use COPYFILE with the REPLACE option on an alias, the file mode number of its base file and all other aliases to its base file (including the one specified in the command) are changed.

You can change the file mode number of a file you are editing by entering the XEDIT subcommand:

```
====> ffile temp script a0
```

Here, also, any changes to the file mode number of a base file or an alias results in the same change to all related aliases and the base file.

Other commands used to change file mode numbers are covered in the following section, "Commands Used to Change File Mode Numbers."

In SFS, you use the ACCESS command to access entire directories. For example, the following command is valid:


```
access .payroll b/a
```

You cannot define a subset of files in a directory using the ACCESS command the same way you can with minidisks (see "File Mode Letters and Numbers" on page 48).

The proper way to access a subset of files is to create a subdirectory with those files in it, and then access the subdirectory. You will see how to do this in Chapter 4, "Using the Shared File System."

Although you cannot access a subset of a directory, you can use the file mode numbers in other CMS commands that operate on files in a file space. For example, to list only those files in an accessed directory that have a file mode of B2, you can enter:

```
filelist * * b2
```

Similarly, you can use file mode numbers to copy a subset of files:

```
copyfile * * b4 = = a4
```

So, while file mode numbers are not useful accessing subsets of directories, they are useful for identifying subsets of files in many other CMS commands.

File Mode Numbers in SFS

Following is a description of file mode numbers 0 through 6 for Shared File System (SFS) directories. If your files are stored on minidisks, see the next section, "File Mode Numbers for Minidisks."

File Mode Number 0

File mode number 0 has no special meaning for SFS. In a minidisk environment, file mode number 0 is used to make files private. Because all SFS files are private unless you explicitly grant authority to someone for the file, file mode A0 is not meaningful for SFS files. If you do use file mode number 0, SFS treats the file as though it had a file mode number of 1.

Note that if you grant read authority on a directory that contains files with a file mode number of 0, the files are not entirely private because other users to whom you have granted authority can see the names of the files. By granting someone *read authority* on a directory, you grant them authority to see every file name in the directory regardless of the file mode. If you do not want users to see the file names, either revoke authority for the directory, or move the files into some other directory.

File Mode Number 1

File mode number 1 is used for reading and writing files. It is the default file mode number.

File Mode Number 2

SFS treats file mode number 2 the same as file mode number 1.

File Mode Number 3

Files with a file mode number of 3 are erased after they are read. If you create a file with a file mode number of 3 and then request that it be printed, the file is printed, and then erased. You can use this file mode number when writing a program or exec procedure to create files which you do not want to retain in your file space. You can create the files, print them, and not have to worry about erasing them later.

The language processors and some CMS commands create work files and give these work files a file mode number of 3.

Note: A file mode number of 3 should not be used in the file ID when naming an exec; depending on what commands are executed within the exec, an exec with a file mode number of 3 may be erased before it completes execution.

File Mode Number 4

Files with a file mode number of 4 are stored in the SFS file pool in OS simulated data set format. These files are created by OS macros in programs running in CMS. You specify that a file created by a program is to have OS simulated data set format by specifying a file mode number of 4 when you enter the FILEDEF command for the output file. If you do not specify a file mode number of 4, the output file is created in CMS format.

You can find more information about OS simulated data sets in the *VM/SP Application Development Guide for CMS*.

Note: There are no file mode numbers reserved for DOS or VSAM data sets, because CMS does not simulate these file organizations.

File Mode Number 5

File mode number 5 is used for reading and writing files, as is file mode number 1.

File Mode Number 6

SFS treats this file mode number the same as file mode number 1.

File mode numbers 7 through 9 are reserved for IBM use.

File Mode Numbers for Minidisks

If your files are stored on minidisks, file mode numbers 0 through 6 will have the following meanings:

File Mode Number 0

A file mode number of 0 assigned to a file makes that file private. No other user may access it unless they have read/write access to your minidisk. Under normal circumstances, if someone has a read-only link to your minidisk and requests a list of all the files on your minidisk, the files with a file mode number of 0 are not listed.

The DDR command lets you copy from one minidisk to another, and therefore, the file mode number 0 files. Use a read share password to protect minidisks with private files when using ACCESS.

File Mode Number 1

File mode number 1 is used for reading and writing files. It is the default file mode number.

File Mode Number 2

File mode number 2 is essentially the same, for the purposes of reading and writing files, as file mode number 1. Usually a file mode number of 2 is assigned to files that are shared by users who link to a common minidisk, like the system disk. Because you can access a minidisk and specify which files on that minidisk you want to access, files with a file mode number of 2 provide a convenient subset of all files on a minidisk. For example, if you enter the command:

```
access 489 e/a * * e2
```

you can only read files with a file mode number of 2 on the minidisk at virtual address 489.

File Mode Number 3

Files with a file mode number of 3 are erased after they are read. If you create a file with a file mode number of 3 and then request that it be printed, the file is printed, and then erased. You can use this file mode number if you write a program or exec procedure that creates files that you do not want to maintain copies of on your minidisks. You can create the file, print it, and not have to worry about erasing it later.

The language processors and some CMS commands create work files and give these work files a file mode number of 3.

Note: A file mode number of 3 should not be used for execs. Depending on what commands are entered within it, an exec with a file mode number of 3 may be erased before it completes execution.

File Mode Number 4

Files with a file mode number of 4 are in OS simulated data set format. These files are created by OS macros in programs running in CMS. You specify that a file created by a program is to have OS simulated data set format by specifying a file mode number of 4 when you enter the FILEDEF command for the output file. If you do not specify a file mode number of 4, the output file is created in CMS format.

You can find more details about OS simulated data sets in *VM/SP Application Development Guide for CMS*.

Note: There are no file mode numbers reserved for DOS or VSAM data sets, because CMS does not simulate these file organizations.

File Mode Number 5

This file mode number is the same, for purposes of reading and writing, as file mode number 1. You can assign a file mode number of 5 to files that you want to maintain as logical groups, so that you can manipulate them in groups. For example, you can reserve the file mode number of 5 for all files that you are retaining for a certain period of time; then, when you want to erase them, you could enter the command:

```
erase * * a5
```

File Mode Number 6

The file mode number 6 indicates that the *update-in-place* attribute of a CMS file is in effect. This means that the existing records of a file are written back to their previous location on the minidisk rather than in a new slot. This only applies to files located on 512-, 1K-, 2K-, or 4K-byte block formatted minidisks. To take advantage of the *update-in-place* capability, the FSWRITE macro must be used, whether explicitly by the user or implicitly by the system.

Warning: It is possible to destroy the integrity of a minidisk if all of the following conditions are true:

- Updates were made to a file mode number 6 file that altered either the number or length of the records in the file.
- One or more output files remain open on the same minidisk.
- A system crash or CMS re-IPL occurs.

Note: For a variable format file, *update-in-place* applies only if a record is replaced by a record of the same length.

File mode numbers 7 through 9 are reserved for IBM use.

Commands Used to Change File Mode Numbers

You can assign file mode numbers when you use the following commands:

COPYFILE	You can assign a file mode number when you create a new file with the COPYFILE command.
DLBL, FILEDEF	When you assign file definitions to files for programs or CMS command functions, you can specify a file mode number.
GENMOD	You can specify a file mode number with the GENMOD command. To change the file mode number of an existing module file, use the COPYFILE command.
READCARD	You can assign a file mode number when you specify a file identifier with the READCARD command or on a READ control card.
RECEIVE	You can assign a file mode number when receiving a file from your virtual reader.
RENAME	<p>You can use RENAME to change the file mode number of a base file, which will cause the file mode number of all aliases to the base file to be changed.</p> <p>For example, if you rename the base file TEST RESULTS:</p> <pre>rename test results a1 = = a2</pre> <p>all aliases to the base file TEST RESULTS now have a file mode number of 2.</p> <p>If you use RENAME on an alias, only the file name and file type of the alias may be changed; you cannot change the file mode number of the alias.</p>
SORT	You can specify file mode numbers for the input and/or output file IDs with the SORT command.
XEDIT	You can assign a file mode number when you create a file with the System Product Editor. To change the file mode number of an existing file, use the RENAME or COPYFILE commands, or use the SET FMODE subcommand when you are in the edit environment.

Managing Your File Space

This section discusses how you can manage your SFS file space and determine its limits. If your files are stored on minidisks, see the following section, "Managing Your Minidisks."

In the beginning of this chapter, we discussed that a file space is an allocation of space within a file pool where you can store files. File spaces are not infinite but contain a certain amount of space that your system administrator allocates to you. When your user ID was added to a file pool, the system administrator also allotted you a certain amount of space. If necessary, you can ask your system administrator to change your space allocation.

At any time, you can determine the amount of space you have used and how much more is available. To determine what proportion of your file space you have used, you would enter the QUERY LIMITS command:

```
query limits *
```

Your output will look like this:

Userid	Storage Group	4K Block Limit	4K Blocks Committed	Threshold
yourid	3	1000	820-82%	90%

The first column shows your user ID (*yourid* in this example). The column labeled *Storage Group* shows the storage group within your file pool where your system administrator has assigned you.

The third and fourth columns contain information regarding the size of the file space. The third column shows that you have been allocated 1000 4K block units. The *4K Blocks Committed* column shows that of the 1000 4K blocks you were allocated, you have used 820, which is 82% of the total.

The column labeled *Threshold* shows when you will receive a warning from the system informing you that your file space is almost full. The default threshold is 90%. When your file space is 90% full, you will receive a warning. If you wish to change the threshold, you can do so by issuing the SET THRESHOLD command. For details on the SET THRESHOLD command, see the *VM/SP CMS Command Reference*.

If, while using CMS, your file space becomes 100% full, you will receive an error message. At this point, you can use the FILELIST command to list the files in your file space, then use the DISCARD command to erase any unwanted files.

If you cannot erase any of the files in your file space, there are several alternate recovery paths you can take:

1. If you are able to store any of your files on minidisks, you may be able to use the COPYFILE command to move files from your file space to a read/write minidisk. After copying the files, erase the original copy in your file space.
2. If you do not have any read/write minidisks in your virtual machine, you may be able to transfer some of your files to another user, using either the SENDFILE, PUNCH or DISK commands. When the files have been read into the other user's file space, you can erase them from your file space.

3. You may contact your file system administrator to request that more storage be added to your file space.

Managing Your Minidisks

The number of files you can write on a minidisk depends on both the size of the minidisk and the size of the files that it contains. You can find out how much space is being used on a minidisk by using the `QUERY DISK` command. For example, to see how much space is on a minidisk with a file mode of `A`, you would enter:

```
query disk a
```

The response may be something like this:

LABEL	VDEV	M	STAT	CYL	TYPE	BLKSIZE	FILES	BLKS USED-(%)	BLKS LEFT	BLK TOTAL
MYDISK	191	A	R/W	5	3330	1024	171	1221-92	107	1328

The first column, *LABEL*, shows the label assigned to the disk when it was formatted. *VDEV* is the virtual device address. *M* is the access mode letter. The next column, *STAT*, indicates whether disk status is read/write or read-only.

Next, *CYL* is the number of cylinders available on the disk. The *TYPE* column shows the device type of the disk. The *BLKSIZE* is the CMS disk block size when the minidisk was formatted. The *FILES* column shows the number of CMS files on the disk.

BLKS USED indicates the number of CMS disk blocks in use. The percentage of blocks in use is also displayed. The *BLKS LEFT* column indicates the number of disk blocks left. The last column, *BLK TOTAL*, contains the total number of disk blocks.

When a minidisk is becoming full, you should erase whatever files you no longer need, or dump to tape files that you need to keep but do not need to keep active on the minidisk.

When you are executing a command or program that creates and stores a file, and the minidisk becomes full in the process, you will receive an error message. You must then try to clear some space on the minidisk before you can attempt to execute the command or program again. To avoid the delays that such situations cause, you should try to maintain an awareness of the usage of your minidisks. If you cannot erase any more files from your minidisks, you should contact installation support personnel about obtaining additional read/write minidisk space.

Note: For more information on minidisks, see Chapter 5, "Storing Your Files on Minidisks."

CMS Command Search Order

When you enter a command in the CMS environment, CMS must locate the command to execute it. If you have execs in storage or on an accessed file mode, or if you have module files in a saved segment or on any of your accessed file modes, CMS treats them as commands; they are known as *user-written* commands.

As soon as the command name is found, the search stops and the command is executed. The list below outlines the search order followed each time you specify a command name:

1. Search for an exec with the specified command name.
2. Search for a translation or synonym for the specified command name. If one is found, repeat Step 1 to search for an exec, using the translation or synonym.
3. Search for a CMS command with the specified command name.
4. Search for a translation or synonym for the specified command name. If one is found, repeat Step 3 to search for a command, using the translation or synonym.
5. Pass the command to CP for execution.

For example, if you enter the command:

```
x sauces cookbook
```

CMS would complete the following search:

1. First, CMS searches for X EXEC. For this example, assume that an X EXEC would not be found.
2. CMS then searches the translation and synonym tables and finds that X is a synonym for XEDIT. Then, CMS repeats Step 1, searching for XEDIT EXEC. Again, assume that an XEDIT EXEC would not be found.
3. Next, CMS searches for a CMS command with the name X. It would not be found.
4. CMS again searches the translation and synonym tables and finds that X is a synonym for XEDIT. Then, CMS repeats Step 3, searching for XEDIT. The XEDIT command would be found and executed. As a result, you would be able to XEDIT the file SAUCES COOKBOOK.

For more information on the CMS command search order, see the *VM/SP CMS Command Reference*.

CMS Command Execution Characteristics

Following is an alphabetical list of the CMS commands that require special consideration when invoked from a user program. For example, a program running in the user area (the storage available to the user) cannot call a CMS command that also runs in the user area. To avoid conflicts with non-relocatable CMS commands, you should ensure that your user programs are relocatable.

Any commands that are listed in the *VM/SP CMS Command Reference* but are not in this table, are nucleus resident and will not interfere with the execution of a user program.

The *Code* column indicates the execution characteristics of the command.

Table 3. CMS Command Execution Characteristics

Code	Meaning
E	Indicates that this command is an exec. It may execute one or more CMS commands that run in the user free storage or transient areas. (The transient area is the storage area used for temporary storage of programs or routines.)
T	Indicates that this command executes in the transient area.
U	Indicates that this command executes in the user free storage or program area. All OS free storage pointers are reset.

Command	Code	Command	Code	Command	Code
ALIALIST	E	FCOBOL	E	OSRUN	U
AMSERV	U	FILELIST	E	PEEK	E
ASSEMBLE	U	FORMAT	U	PSERV	U
ASSGN	T	GENDIRT	T	PUNCH	T
AUTHLIST	E	GENMSG	U	RDR	T
CATCHECK	U	GLOBAL	T	RDRLIST	E
CMSBATCH	U	HELPCONV	T	READCARD	T
CMSSERV	E	IOCP	U	RECEIVE	E
COMPARE	T	LABELDEF	T	RESERVE	T
CONVERT COMMANDS	E	LANGGEN	E	RSERV	U
CSLGEN	E	LANGMERG	E	RUN	E
CSLLIST	E	LISTDS	U	SENDFILE	E
DDR	U	LISTIO	T	SETPRT	T
DEFAULTS	E	LKED	U	SORT	U
DIRLIST	E	LOADLIB	U	SSERV	U
DISCARD	E	MACLIB	U	SVCTRACE	T
DOSLIB	U	MACLIST	E	SYNONYM	T
DOSLKED	U	MODMAP	T	TAPE	T
DOSPLI	E	MOREHELP	E	TAPEMAC	U
DSERV	U	MOVEFILE	U	TAPPDS	U
EDIT	E	NAMES	E	TELL	E
ESERV	E	NOTE	E	TXTLIB	U
EXECMAP	T	NUCXDROP	T	TYPE	T
EXECUPDT	E	OPTION	T	UPDATE	U

Displaying a List of Your CMS Files

Use the FILELIST command to display information about your CMS files. If your files are stored in a Shared File System (SFS) file pool, you can use FILELIST for files in accessed directories.

In a full-screen environment, FILELIST provides you with the same information as the LISTFILE command, but also lets you edit files and enter commands from the list. You can enter XEDIT subcommands to manipulate the list itself.

When using SFS, there are several options of the FILELIST command. The default FILELIST screen is the FILELIST STATS screen. The FILELIST STATS screen follows:


```

ZOOKEEP FILELIST  A0 V 108 Trunc=108 Size=8  Line=1 Col=1 Alt=0
Directory = VMSYSU:ZOOKEEP.
Cmd Filename Filetype Fm Format Lrecl Records Blocks  Date      Time
ANIMAL  DATA    A1 V      95      34      1 10/04/88 21:12:04
BANANA  DATA    A1 V      95      29      1 10/04/88 20:58:07
BEAR    NOTE     A1 V     107     281     5 10/04/88 17:59:00
HONEY   DATA    A1 V      92     101     2 10/02/88 15:33:05
LION    NOTE     A2 V      75      28      1  9/25/88 12:10:03
ALL     NOTEBOOK A0 V     120     277     4  9/24/88  9:14:02
TIGER   NOTE     A1 V      26       7      1  9/23/88 16:50:06
ZOOKEEP NETDATA  A1 V      80     489    10  8/26/88 16:05:08

1= Help    2= Refresh  3= Quit    4= Cancel    5= Sort(dir) 6= Sort(size)
7= Backward 8= Forward  9= FL /n  10= Share    11= XED/FILEL 12= Cursor

====> _

X E D I T  1 File

```

Figure 8. Sample FILELIST STATS Screen

SFS also provides you with SHARE and SEARCH options of FILELIST to display different FILELIST screens to see other information about your files and directories. Each screen provides you with a set of PF keys to enter commands or get more information. See Chapter 4, "Using the Shared File System," for a description of the various options of FILELIST and examples using these options.

Finding Files in Your FILELIST List

If you have many files in your list, the list may take up more than one screen. To find files in your FILELIST list, you can do any of the following:

1. Scroll through the list using the PF keys.

Key	Function
PF7	Scrolls backward one screen.
PF8	Scrolls forward one screen.

2. Use any of the appropriate PF keys to sort the displayed output. (See Chapter 4, "Using the Shared File System," for information on the PF key definitions for the STATS, SHARE, and SEARCH options of FILELIST.)

3. Use the XEDIT subcommand LOCATE if you know the file name and/or file type of the file that you are looking for. You enter the LOCATE command at the bottom of the screen and then press the ENTER key. For example:

```
====> locate/banana data/
```

If BANANA DATA is located, the line containing it becomes the first line on the screen.

4. Rearrange the list by entering one of the following synonyms on the command line:

SNAME	Sorts the list alphabetically by file name, file type, and file mode.
STYPE	Sorts the list alphabetically by file type, file name, and file mode.
SDIR	Sorts the list by directory name, file name, and file type.
SMODE	Sorts the list by file mode, file name, and file type.
SRECF	Sorts the list by record format, file name, file type, and file mode.
SLREC	Sorts the list by logical record length and then by size (greatest to least).
SSIZE	Sorts the list by number of blocks and number of records (greatest to least).
SDATE	Sorts the list by year, month, day, and time (most recent to oldest).

Using FILELIST to List Some of Your Files

FILELIST lets you obtain various lists of your files and subdirectories (lower-level directories). We will discuss subdirectories in Chapter 4, "Using the Shared File System." You can ask for a list of files or subdirectories that have the same file name or file type or all of the ones that begin with a certain letter. The abbreviation for FILELIST is FILEL. Following are various ways that you might use the FILELIST command:

<code>filelist</code>	Displays a list of the files and subdirectories on file mode A.
<code>filelist * * b</code>	Displays a list of the files and subdirectories on file mode B.
<code>filelist bear *</code>	Displays a list of the files on file mode A with a file name of BEAR.
<code>filelist * data</code>	Displays a list of files on file mode A with DATA as the file type.
<code>filelist * * a1</code>	Displays a list of the files with a file mode number 1 on file mode A.

Erasing Files from FILELIST

Use the DISCARD command to erase a file or subdirectory that is displayed in the list. DISCARD is equivalent to the CMS command ERASE. DISCARD can either be entered in the command area of the line that describes the file you want discarded, or it can be entered from the command line (at the bottom of the screen). DISCARD can only be used while in FILELIST, DIRLIST, RDRLIST, MACLIST, and PEEK command environments.

Listing Your Files with the LISTFILE Command

You can use the LISTFILE command to list information about your CMS files. For example, entering:

```
listfile * data
```

lists the files on file mode A with the file type of DATA. For example:

```
ANIMAL DATA A1
BANANA DATA A1
HONEY DATA A1
```

If you want more information than just the file IDs, you can use one of the options for LISTFILE. For example, when you enter:

```
listfile * data (share)
```

Your output will look like this:

```
FILENAME FILETYPE FM OWNER TYPE R W
ANIMAL DATA A1 CROCKETT BASE X X
BANANA DATA A1 BREEZY ALIAS X X
HONEY DATA A1 STONE ALIAS X -
```

In addition to the file name, file type, and file mode, this display shows you the owner of the file, whether it is a base file or an alias, and what type of authority you have to the file (read or write). This will be explained in detail in Chapter 4, "Using the Shared File System."

Like FILELIST, the LISTFILE command has various options that allow you to display different information about your files and directories. For more information, see the LISTFILE command in the *VM/SP CMS Command Reference*.

As with the FILELIST command, you can vary what you list with the LISTFILE command. Remember you only need to enter L, the minimum truncation for LISTFILE. There are various ways that you might use the LISTFILE command:

```
listfile           Lists the files on file mode A.
listfile * * b     Lists the files on file mode B.
listfile bear *    Lists the files on file mode A with a file name of BEAR.
listfile * data    Lists the files on file mode A with DATA as a file type.
listfile * * a1    Lists the files with a file mode number 1 on file mode A.
```

Comparing Contents of Files

To compare the contents of two files to see if they are identical, use the COMPARE command. For example:

```
compare labor stat a1 labor stat b1
```

Any records in these files that do not match are displayed at your terminal. The format of the COMPARE command is found in the *VM/SP CMS Command Reference*.

Copying Files

The COPYFILE command copies a file from one directory to another, from one minidisk to another, or between directories and minidisks.

For example,

```
copyfile linda assemble b pat assemble a
```

would create a copy of the LINDA ASSEMBLE file, name it PAT ASSEMBLE, and store it on file mode A.

Note: If you wish to copy a file into a file that already exists in a directory, and you want to use the REPLACE option, you need write authority to the existing file.

If you wish to copy a file into a directory by creating a new file, you need write authority to the directory in which you are creating the file.

Renaming Files

You can change the file identifier of a file with the RENAME command. For example,

```
rename test file a1 good file a1
```

You can use RENAME on a base file (not on an alias) to modify file mode numbers. For example,

```
rename news report a1 = = a2
```

changes the file mode number of the base file NEWS REPORT, along with the file mode numbers of all aliases to that file to 2.

You cannot use the RENAME command to move a file from one minidisk or directory to another. (You can, however, use the COPYFILE command to copy a file from one directory or minidisk to another.)

If your files are stored in an SFS file pool, you can use the RELOCATE command to move a file from one directory to another. See the RELOCATE command in the *VM/SP CMS Command Reference* for details.

You can also rename a file in another user's directory if the user has granted you write authority to the file and to the directory.

Changing the Record Format of a File

Files can either have fixed- or variable-length record formats. You can change the record format of a file with the COPYFILE command and the RECFM option;

```
copyfile data file a (recfm f lrecl 130
```

converts the file DATA FILE A1 to fixed-length 130-character records.

If you want to keep the original file intact, you can specify an output file id, for example:

```
copyfile data file a fixdata file a (recfm f lrecl 130
```

The file FIXDATA FILE A contains the converted records.

If the records in a file being copied are variable-length, each output record is padded with blanks to the specified record length. If any records are longer than the record length, they are truncated.

When you convert files from fixed-length records to variable-length records, you can specify the COPYFILE TRUNC option to ensure that all trailing blanks are truncated:

```
copyfile data file a (recfm v trunc
```

If you specify the COPYFILE LRECL option and RECFM V, the LRECL option is ignored and the output record length is taken from the longest record in the input file.

When you convert a file from variable-length to fixed-length records, you may also specify a fill character to be used for padding instead of a blank. If you specify:

```
copyfile short recs a (recfm f fill *
```

then each record in the file SHORT RECS is padded with asterisks to the record length. Assuming that SHORT RECS originally was a variable-length file, the record length is taken from the longest existing record. Note that if SHORT RECS is already fixed-length, it is not altered.

Similarly, when you are converting back to variable-length a file that was padded with a character other than a blank, you must specify the FILL option to indicate the pad character, so that character is truncated.

Using Synonyms

By using the SYNONYM and the SET ABBREV commands, you can control what command names, synonyms, or truncations are valid in CMS. For example, you could create a file named MYSYN SYNONYM that contains the following records:

```
PRINT   PRT   1
RELEASE LETGO 4
FILELIST FL   2
```

The first column specifies an existing CMS command, module, or exec name. The second column specifies the alternate name or synonym that you want to use. The third column is a count field that indicates the minimum number of characters of the synonym that can be used to truncate the name. Using this file, after you enter the command:

```
synonym mysyn
```

you can use PRT, LETGO, and FL in place of the corresponding CMS command names. Also, if the ABBREV function is in effect, (it is the default; you can make sure it is in effect by issuing the command SET ABBREV ON), you can truncate any of your synonyms to the minimum number of characters specified in the count field of the record (that is, you could enter P for PRINT and LETG for RELEASE). To invoke your synonym table at the beginning of every terminal session, enter the SYNONYM MYSYN command (or your own synonym table name) into your PROFILE EXEC.

Notes:

1. An exec procedure having a synonym defined for it can be invoked by its synonym if implied EXEC (IMPEX) function is on. However, within an exec procedure, only the exec file name can be used. A synonym for an exec is not recognized within an exec because the synonym tables are not searched during exec processing.
2. You cannot define translations or translation synonyms using the SYNONYM command. Translations must be defined in the Definition Language for Command Syntax (DLCS) file. You can truncate any translation or translation synonym to the minimum number of characters specified in the count field of the record if you entered SET ABBREV ON. See the *VM/SP Application Development Guide for CMS* book for more information about DLCS.
3. If either TAPE or VMFPLC2 is a synonym of the other, the synonym may not be used to call that function from within an exec. You may use any name other than TAPE or VMFPLC2 as a synonym of the other function. For example, from within an exec, TAPE is not a valid synonym for VMFPLC2; TAP, however, would be valid.

Using Translations

Once you have defined translations and translation synonyms for commands in your DLCS file, you can use the SET TRANSLATE command to control whether or not they are recognized by CMS. The SET ABBREV command controls whether or not the abbreviations of these translations will be recognized.

Note: The translation synonyms defined in a DLCS file are synonyms of command name translations. Do not confuse them with synonyms defined with the SYNONYM command. In the following paragraphs, the term *translation* is used to mean both the command name translations and translation synonyms defined in DLCS. The term *synonym* refers to synonyms defined with the SYNONYM command and abbreviations of system language command names.

When you enter a command in CMS, the command name you use and all of the keywords in it must be in the same language. If you use a translation of the command name, all of the keywords you use with that command will be translated. If you specify a synonym for a command name, the keywords will not be translated. Therefore, you must specify them so that the command will recognize them.

It is possible for the translation of a command or keyword to be the same as the original command or keyword. If the command name you specify is the same as the original command, but you specify keywords for that command in a different language, CMS would determine which language to use upon encountering the first keyword that is different. The subsequent keywords must be in the same language as the first keyword to be successfully translated.

There are ways you can code programs and execs so that you can choose whether or not to allow translations. CMS only recognizes translations for commands entered from the command line or those invoked with the System Product Interpreter command search function (ADDRESS CMS) or the equivalent search function in EXEC 2 (&PRESUME &SUBCOMMAND CMS).

CMS does not translate your command name or keywords if you SET TRANSLATE OFF. Also, a command will not be translated if it is invoked from another program using the search hierarchy for SVC 202 or using the System

Product Interpreter SVC 202 search hierarchy (ADDRESS COMMAND) or the equivalent search function in EXEC 2 (&PRESUME &COMMAND CMS). Refer to *VM/SP Application Development Guide for CMS* for further details on the CMS command search function for translations.

For more details of how CMS uses the translation and synonym tables to find commands, see the *VM/SP CMS Command Reference*.

Chapter 4. Using the Shared File System

What is the Shared File System?

The *Shared File System (SFS)* helps you manage and store your CMS files. Files you create reside in a *file pool*, a large amount of DASD space containing the files for many users. You will be enrolled in a file pool, and given a *file space* within it, where you can store your files.

There are two ways you can become enrolled in a file pool. You may be enrolled, along with other users, by an ENROLL PUBLIC command. In this case, you will have the authority to read from, write to, or lock the files and directories for which you are authorized. However, you will not have space to create files of your own. You may also be enrolled in a file pool by name. Here, you may have space allocated specifically for your use. Your system administrator is responsible for enrolling you in a file pool. This chapter assumes you are enrolled in a file pool by name.

Within your SFS file space you can organize your files into directories. A *directory* is a group of files. SFS directories can be arranged to form a hierarchy in which one directory can contain one or more subdirectories as well as files. Within a directory you can store closely related files. For example, you could create a directory to store all the files for a particular project. You can then define subdirectories beneath that directory to contain files related to major parts of the project. SFS lets you define up to eight levels of subdirectories.

Suppose there is a file pool named BOOKPOOL, which was created for and is shared by a group of writers. You and other writers have a file space in the BOOKPOOL file pool. Your file space could contain chapters of your book and any other research you are working on. You could set up directories in the BOOKPOOL file pool that organize your book files.

Your top directory, the directory from which all files and subdirectories will branch, could contain files such as letters and memos, as well as a NOVEL directory. You might wish to create more subdirectories of the NOVEL directory to organize files dealing with the major parts of the project such as the setting, the plot, or the ending. If you wanted, you could create even more subdirectories to organize individual chapters of the book.

Figure 9 on page 70 shows how this might look:

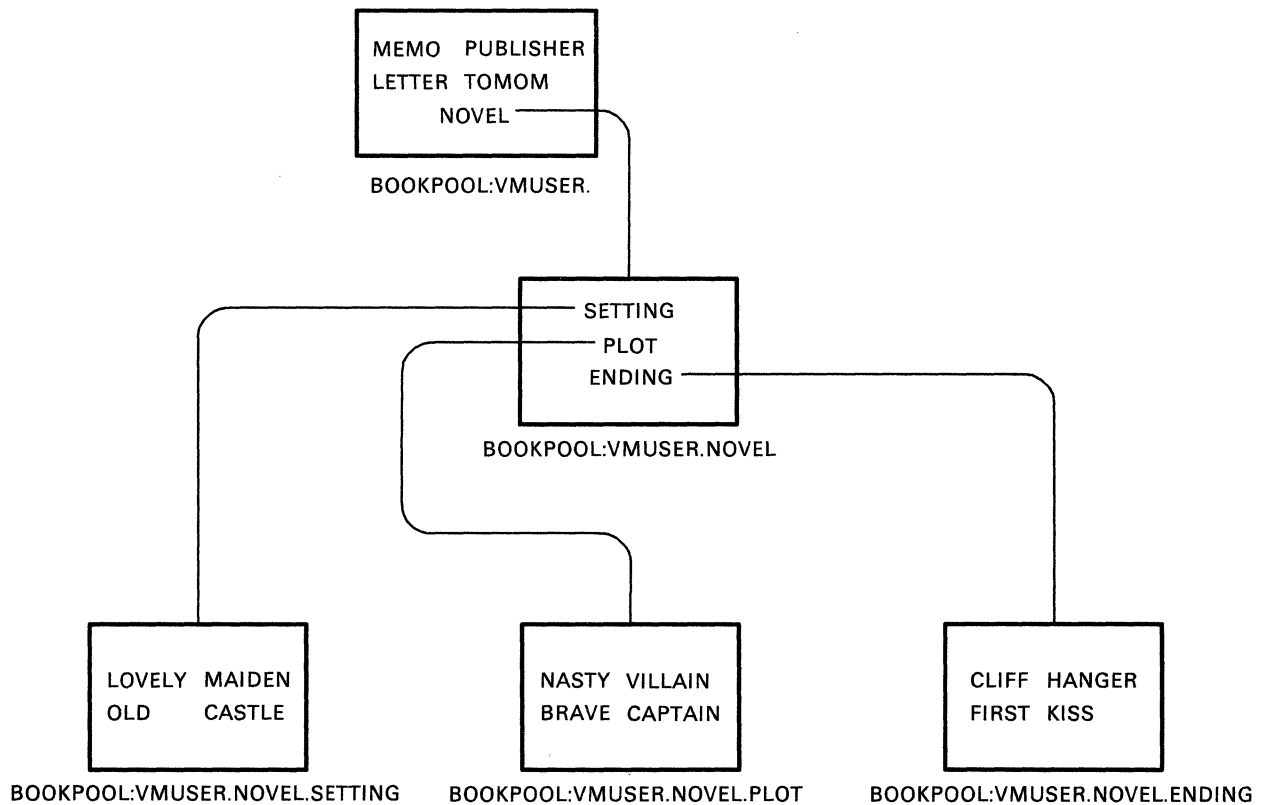


Figure 9. Sample Hierarchical Directory

Now that you understand how files and directories can be organized, you need to know how they are shared between users.

To share files or directories, you simply grant other users authority to them. You can choose to grant authority on one or more individual files, or you can grant authority on all the files currently in a directory. You can also grant users authority to a directory only, and not to the files it contains.

You have the option of letting other users write to these files or directories or granting them read authority only. As the owner of the files and directories, you can also choose to revoke the authorities you have granted.

This chapter provides information on how to manage the files and directories you store within an SFS file pool. Depending on your system configuration, your files can be stored on minidisks rather than in an SFS file pool. If this is the case, you may want to skip this chapter and instead see Chapter 5, "Storing Your Files on Minidisks," for additional information on how to manage your CMS files on minidisks.

With certain system configurations, you may have the option to store files both on minidisks and in SFS file pools. In this case, you can decide which files you want stored on minidisks, and which you want stored (and shared) in SFS file pools.

Getting Started

To use directories and share files, you need to be enrolled in a file pool and be given a file space. Only a system administrator can do this. When the administrator assigns you a file space within a file pool, SFS automatically defines one directory within that file space. This directory is called your *top directory* because under it you can create a hierarchy of subdirectories in which to arrange your files.

The name of the top directory is always the same as your user ID. It cannot be changed. If, for example, your user ID is HENRY and you become enrolled in a file pool, *HENRY.* would be the name of your top directory. Notice the period after the top directory name. In SFS, the period is used to indicate a directory; here the top directory belongs to the user ID HENRY. In the following examples, *yourid* is shown. When you enter each command, of course, your user ID will replace *yourid*.

Before you begin the exercises in this chapter, you will need to determine if you are enrolled in a file pool. If you do not know, see your system administrator now, and obtain your file pool ID. You may be enrolled in the IBM-supplied VMSYSU file pool or another file pool supplied by your system administrator. The file pool identifier VMSYSU appears in the examples. If this is not the file pool you are enrolled in, you will see your file pool identifier in place of VMSYSU when comparing your screen to these examples. Regardless of which file pool you are enrolled in, you can use the examples in this chapter.

Once you are enrolled, you should determine whether or not your top directory is accessed as A. To find out, enter the QUERY ACCESSED command. QUERY ACCESSED will show you the modes for all the directories and minidisks you have accessed. Enter:

```
query accessed
```

You will see a display similar to one of the following two samples. If your top directory has been accessed with a file mode of A, you will see a display similar to this:

Mode	Stat	Files	Vdev	Label/Directory
A	R/W	3	DIR	VMSYSU:yourid.
S	R/O	1321	190	MNT190
Y/S	R/O	337	19E	MNT19E

The first line of the display shows the virtual device (Vdev) is *DIR*. This means that a directory (your top directory) is accessed with a file mode of A. The string *VMSYSU:yourid.* tells you the complete name of your top directory. It shows that your user ID is assigned to the VMSYSU file pool.

If your top directory is not accessed as A, you will see a display similar to this:

Mode	Stat	Files	Vdev	Label/Directory
A	R/W	3	191	AMC191
S	R/O	1321	190	MNT190
Y/S	R/O	337	19E	MNT19E

In this case, to complete the examples in this chapter, enter the following commands, substituting your file pool ID:

```
set filepool filepoolid:  
access . a
```

The SET FILEPOOL command establishes the name of your default file pool. The file pool may be VMSYSU, the file pool shipped with your system, or any other file pool to which you are assigned. In subsequent commands, you will not need to type the file pool ID. Unless you specify a different file pool, SFS will use the default file pool ID you specified on the SET FILEPOOL command.

The ACCESS command accesses your top directory with a file mode of A. The . (period) is an abbreviation of your top directory, which is your user ID. Whenever you are referring to *yourid.*, you can substitute a . instead. Commands that default to file mode A take input from or send output to your top directory. For example, if you send a file to another user and do not specifically state the file mode of the file, SENDFILE defaults to file mode A. If the file is not located in the directory you have accessed with a file mode of A, you get an error message telling you that the file was not found.

If you did not have your top directory accessed as A, you may want to add the two commands in the previous example to your PROFILE EXEC on your 191 minidisk. Otherwise, if you log off the system, and later log back on, you will have to reenter the commands, because they are not saved after a terminal session.

Note: Depending on your installation, you are enrolled in a file pool, use minidisk storage, or have the capability to use both. If you have space in a file pool and no minidisk storage, your top directory is accessed with a file mode of A. If you have minidisk storage and no file pool space, your 191 minidisk is accessed with a file mode of A. If you have both space in a file pool and minidisk storage, one of the following statements is true:

- Your top directory is accessed with a file mode of A. Your minidisk storage is not automatically accessed. This minidisk storage, which is at virtual address 191 or any other available virtual address, is accessed with the file mode of your choice when you want to use it.
- Your 191 minidisk is accessed with a file mode of A. If you want to use your file pool storage, you can enter the SET FILEPOOL and ACCESS commands to access it with the file mode letter of your choice.

If you want to change the default, see your system administrator.

Organizing Your Files

SFS lets you keep your files organized because you can place groups of related files in their own directories. Your directories are arranged hierarchically. Your top directory is always the first level; subdirectories of your top directory branch out to a lower level. Both the top directory and subdirectories can contain CMS files.

For example, assume that Tony, a manager of a maintenance department, keeps employee records in a set of CMS files. The manager might create a directory for each employee. If his file space is in the POOLQ file pool, his directory structure might look like this:

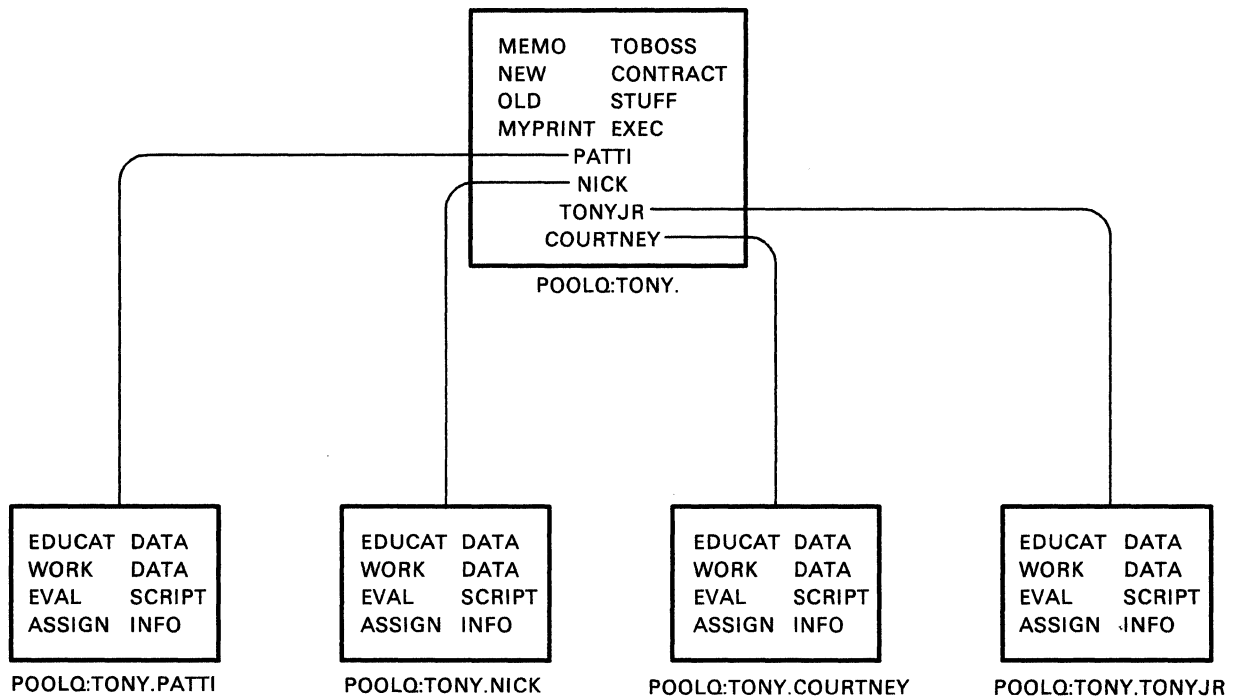


Figure 10. Another Sample Directory

The first four items in Tony's top directory are CMS files. His top directory also contains four subdirectories, one for each of his employees. The subdirectory for each employee contains four files.

Tony can use the same file identifier, such as EDUCAT DATA, for each of his employees because the files are in different subdirectories. The name

EDUCAT DATA POOLQ:TONY.COURTNEY

is the complete name of the EDUCAT DATA file in Tony's .COURTNEY directory. The name

EDUCAT DATA POOLQ:TONY.NICK

is the complete name of the EDUCAT DATA file in the .NICK directory.

Working with Directories

Before you begin working with SFS directories, you will need to understand how to specify the names of your directories within commands and how to list the contents of your directories. This section explains these topics and includes exercises using the SFS directories and files provided with your system.

Using the Abbreviated Form of Your Top Directory

In the previous example, Tony could refer to any directory by using the file pool identifier (POOLQ:), followed by the name of his top directory (TONY.), followed by the name of the other directories. The file pool identifier (*filepoolid*) must always be followed by a colon (:). Directory names must be separated by periods.

To refer to the .NICK directory in a command, Tony could specify the following directory name (*dirname* for short):

POOLQ:TONY.NICK

If POOLQ is Tony's default file pool, he can omit it from the directory name and refer to the directory as follows:

TONY.NICK

SFS will assume that the TONY.NICK directory is located within Tony's default file pool, POOLQ.

Tony can also omit his user ID from the directory name because your top directory is always the same as your user ID. He must, however, be sure to retain the period to indicate his top directory. Therefore, Tony could simply refer to the .NICK directory in a command as follows:

.NICK

In executing a command, SFS would begin with Tony's top directory (designated by the period) and move down one level to the .NICK directory. Notice that there is not a space between the period and the word NICK.

Accessing Another User's Directory

After your system administrator has set up the SFS files and directories that were shipped with your system, you will have automatic read authority to the files and directories that are owned by the MAINT user ID. Usually, other users will need to grant you authority on a directory or a file before you can access or use it.

Before you can work with the MAINT. top directory and its subdirectories, you must access it. You can access the MAINT. top directory in any available file mode letter. Your top directory is accessed with a file mode of A. Access the MAINT. top directory with a file mode of B.

A sample format of the ACCESS command follows:

```
access filepoolid:userid. fm
```

The MAINT user ID is assigned to the VMSYSU file pool; if VMSYSU is also your default file pool, you do not need to specify the file pool ID. Because MAINT is the name of a top directory, be sure to follow it with a period. Leave a blank space before the file mode letter, B. To access the top directory owned by the MAINT user ID as B, enter:

```
access maint. b
```

In cases where you are not sure if a user is enrolled in your file pool, you can use the QUERY FILEPOOL command. For more information on QUERY FILEPOOL, see the *VM/SP CMS Command Reference*.

Although you have now accessed another user's directory, your hierarchy of directories is not affected. That is, this directory becomes part of your CMS search order but is not part of your directory structure. Your directory structure will remain the same until you create new directories of your own.

To verify that the directory was accessed, enter the following command:

```
query accessed
```

After you enter the command, your screen will look something like this:

Mode	Stat	Files	Vdev	Label/Directory
A	R/W	2	DIR	VMSYSU:yourid.
B	R/O	543	DIR	VMSYSU:MAINT.
S	R/O	1321	190	MNT190
Y/S	R/O	337	19E	MNT19E

You will notice the MAINT. directory is accessed with a file mode of B.

The MAINT. directory only remains accessed for the duration of your CMS session; a LOGOFF command will automatically release it. If you wish to release a directory at any time during your CMS session, you can do so by entering the RELEASE command. For more information on the RELEASE command, see the *VM/SP CMS Command Reference*.

Specifying a Directory Identifier

You need to refer to directories often when using CMS commands. When you use commands that accept a directory identifier, a *dirid*, you can reference a directory several ways. A directory identifier can be a complete directory name, such as a file pool identifier followed by the name of a directory. (For example, VMSYSU:MAINT.) It could be an abbreviated form of the directory, as we discussed in “Using the Abbreviated Form of Your Top Directory” on page 73.

Because some directory names can be quite long, it could be tedious to have to type (or remember) the directory name each time you wanted to enter a command. For this reason, there are shorter methods of identifying the directory or subdirectory. One such method is plus (+) and minus (-) file mode notation, and another is the use of file mode letters.

SFS commands that accept a directory identifier accept plus (+) and minus (-) file mode letter notation as an abbreviated way of referring to a directory. Rather than typing out the entire directory name, you can use the plus sign (+) to move down one level lower in the hierarchy and the minus sign (-) to move up one level. You will find this short-cut syntax particularly helpful in writing execs. To learn about plus and minus notation, see the *VM/SP CMS Command Reference*.

You can also use file mode letters to refer to accessed directories in commands that accept a directory identifier. This chapter provides examples of the use of the file mode letter as an abbreviated form of the directory identifier.

Following is a list of commands that accept directory identifiers:

ACCESS	CSLLIST	QUERY AUTHORITY
ALIALIST	DELETE LOCK	QUERY LOCK
AUTHLIST	DIRLIST	RELEASE
CREATE ALIAS	DISCARD	RELOCATE
CREATE DIRECTORY	ERASE	RENAME
CREATE LOCK	GRANT AUTHORITY	REVOKE AUTHORITY
CREATE NAMEDEF	LISTDIR	RTNLOAD
CSLGEN	QUERY ALIAS	

Note: There are some rules to remember about the use of file mode numbers when specifying a file mode as a directory identifier. Generally, you do not want to use file mode numbers unless you are strictly specifying a file or set of files. Since file mode numbers are attributes of files, and not SFS directories or minidisks, you

cannot specify file mode numbers on commands that operate on an entire minidisk or SFS directory. For example, do not use file mode numbers on such commands as ACCESS, which requires a directory identifier, *dirid*, unless you are specifying a file or set of files. The same is true for the commands FORMAT, RELOCATE, DIRLIST, and RELEASE, which are also discussed in this chapter.

Listing the Structure of a Directory with DIRLIST

You can use the DIRLIST command to see what the MAINT. top directory contains. DIRLIST is a very useful command because it lets you easily see the “big picture” of what subdirectories are contained within a directory. A sample format of the DIRLIST command follows:

```
dirlist dirid
```

If you do not specify a directory identifier when entering the DIRLIST command, your top directory is assumed.

Because you accessed MAINT. with a file mode of B, CMS finds the correct directory and performs the DIRLIST command when you simply specify the file mode letter. Enter:

```
dirlist b
```

Your screen will look like this:

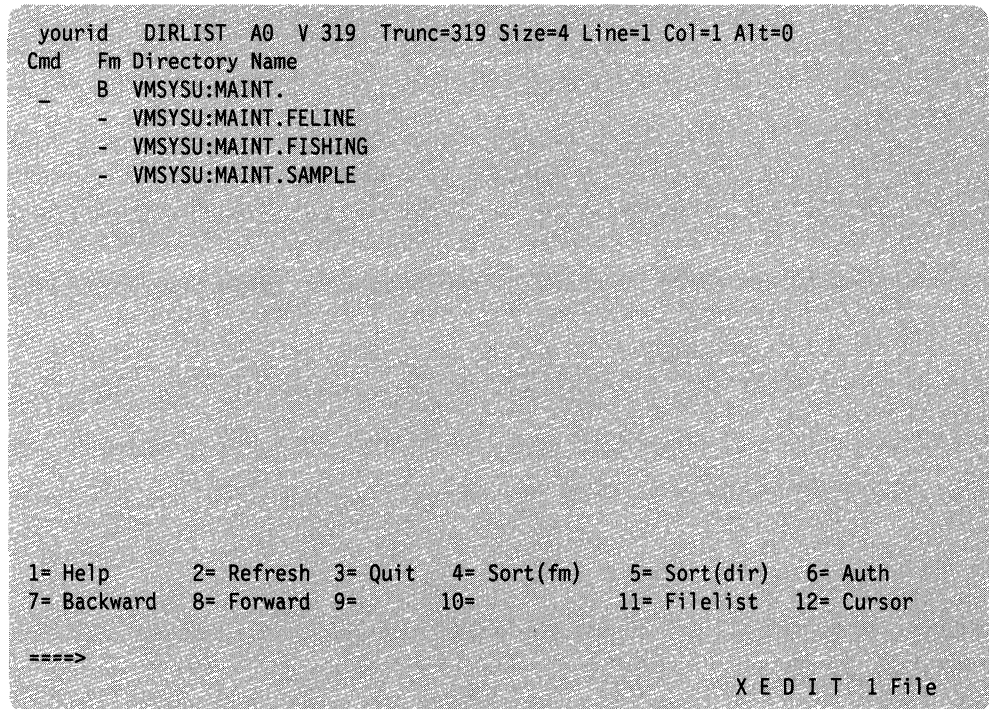


Figure 11. Entering the DIRLIST Command

With DIRLIST, directories are listed in a full-screen display similar to the output you receive when you enter the FILELIST or RDRLIST commands.

The first column of the display, labeled *Cmd*, is where you can enter commands to be executed against any of the directories listed. The *Fm* column indicates the file

mode letter you used to access the directory. The column labeled *Directory Name* lists the complete name of the directory.

The DIRLIST command lists the directory you specify plus all its subdirectories. DIRLIST displays these directories whether or not they are accessed. To list only accessed directories, use the ACCESSED option of the DIRLIST command. See the *VM/SP CMS Command Reference* for more information.

You can tell whether a directory is accessed by referring to the column labeled *Fm*. If the column displays a file mode letter, then the directory listed on that line is accessed with the file mode letter shown. If the line shows a dash (—) in the *Fm* column, then that directory has not yet been accessed. You must access a directory to make it part of the CMS search order.

Accessing a directory also lets you refer to the directory by its file mode letter (for those commands that only accept file modes). Check to see if the command you want to use is in the list of commands in the section “Specifying a Directory Identifier.” If it is not, then the command will only accept file modes. If this is the case, you must first access the directory to make it part of the CMS search order.

If you have authority to subdirectories under another user’s top directory, but do not remember their names, you can use the DIRLIST command to display a list of all subdirectories to which you have authority. Use the same command format, except substitute the *userid* of the other user for the *dirid*.

Using the DIRLIST PF Keys

From the DIRLIST display, you can use your PF keys to obtain additional information for any of the directories listed.

Following is a list of the DIRLIST PF keys and their meanings:

Key	Meaning	Usage
PF1	Help	Use PF1 to display the main HELP menu.
PF2	Refresh	When you press PF2, the screen display is refreshed. The results of any recent commands are shown.
PF3	Quit	This key lets you quit from the DIRLIST environment and remove the displayed output from your screen.
PF4	Sort (fm)	The PF4 key sorts the output currently displayed on your screen alphabetically by file mode.
PF5	Sort (dir)	The PF5 key sorts the output currently displayed on your screen alphabetically by directory name.
PF6	Auth	Pressing the PF6 key executes the AUTHLIST command. You will see your authority to the directory that is on the line where your cursor is placed when you press PF6. Also, if you are the owner of the directory, you will see a list of users who have been granted read or write authority. (See “Using AUTHLIST” on page 112 for more information.)
PF7	Backward	This key scrolls the DIRLIST display backward one screen.
PF8	Forward	This key scrolls the DIRLIST display forward one screen.

Table 4 (Page 2 of 2). DIRLIST PF Keys

Key	Meaning	Usage
PF9		Undefined
PF10		Undefined
PF11	Filelist	Pressing the PF11 key brings you into a FILELIST display of the files or directories contained in the directory indicated (the directory on the line where your cursor is currently located.)
PF12	Cursor	This key causes the cursor to move from the file area to the command line. If the cursor is on the command line, it moves to its previous location in the file area.

You can use these PF keys to find useful information about directories and files. For now, press PF3 to quit the DIRLIST display.

Using the LISTDIR Command

You can also list the structure of a directory by using the LISTDIR command. LISTDIR provides the same information as DIRLIST, but while DIRLIST provides a full-screen display, the output from LISTDIR appears in line-mode format. A sample format of the LISTDIR command follows:

```
listdir dirid
```

Like DIRLIST, the easiest way to enter the LISTDIR command is to type the command followed by the file mode. For example, to list the structure of the MAINT. top directory, enter:

```
listdir b
```

Your screen will display the following information:

```
Fm Directory Name
B VMSYSU:MAINT.
- VMSYSU:MAINT.FELINE
- VMSYSU:MAINT.FISHING
- VMSYSU:MAINT.SAMPLE
Ready;
```

LISTDIR displays the same information as DIRLIST. The column labeled *Fm* shows the file mode letter where the directory is accessed, and the column labeled *Directory Name* shows the complete name of the directory.

If you are enrolled in VMSYSU you will be given read authority to the IBM-supplied files and directories owned by the MAINT user ID. You will be able to see a list of the directories using DIRLIST or LISTDIR. You can see a list of the files and directories using FILELIST or LISTFILE. You can see the actual contents of the files using XEDIT.

In the previous examples, we used DIRLIST and LISTDIR to list the structure of another user's top directory. Both DIRLIST and LISTDIR are also useful to see all of the directories below your own top directory. If you specify either command without a directory identifier, the command will default to your top directory and list all the subdirectories it contains.

Creating a Directory

Now that we have seen how to access and list the structure of existing directories, let us create a new directory. To create a directory of your own, use the CREATE DIRECTORY command. A sample format of the command follows:

```
create directory dirid
```

Remember, you will always have a top directory whose name is the same as your user ID, followed by a period. Any new directories you create will be subdirectories of your top directory. You cannot change the name of your top directory, and you cannot create new directories that are the same level as your top directory.

For example, to create the directories shown in Figure 10 on page 73, Tony would have used the following series of commands:

```
create directory .patti
create directory .tonyjr
create directory .courtney
create directory .nick
```

In the first command, the period specifies Tony's top directory. Therefore, the command actually tells SFS to create a directory below Tony's top directory, with the name PATTI. The subsequent commands set up the .TONYJR, .COURTNEY, and .NICK directories.

To create a directory of your own, enter the following command:

```
create directory .party
```

This command will create a directory called .PARTY below your top directory. The following diagram represents your directory structure after this command:

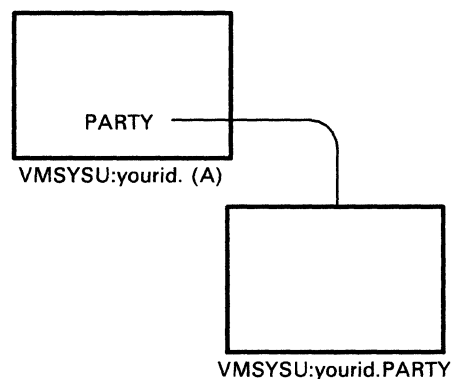


Figure 12. The .PARTY Directory

To create another directory called .PARTY.FOOD below the .PARTY directory, enter:

```
create directory .party.food
```

This diagram represents your directory structure now:

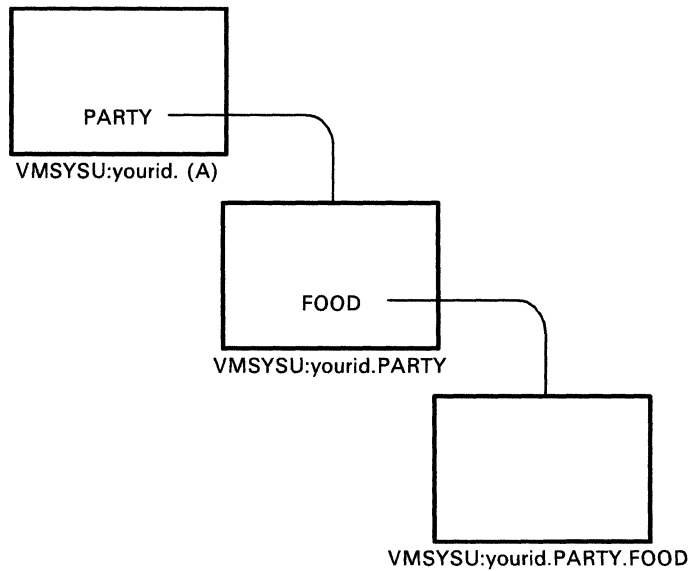


Figure 13. The .PARTY.FOOD Directory

The .PARTY directory is a subdirectory of your top directory; the .PARTY.FOOD directory is a subdirectory of the .PARTY directory.

To see a list of all of your directories, use the DIRLIST command with the name of your top directory or the file mode of your top directory. If you do not specify a directory identifier, DIRLIST defaults to your top directory and lists all the subdirectories it contains. Enter:

```
dirlist
```

Your screen will appear as follows:

```

yourid  DIRLIST  A0  V 319  Trunc=319  Size=3  Line=1  Col=1  Alt=0
Cmd    Fm Directory Name
-      A  VMSYSU:yourid.
-      -  VMSYSU:yourid.PARTY
-      -  VMSYSU:yourid.PARTY.FOOD

1= Help      2= Refresh  3= Quit    4= Sort(fm)  5= Sort(dir)  6= Auth
7= Backward  8= Forward   9=         10=          11= Filelist  12= Cursor

====>
X E D I T  1 File

```

Figure 14. Using DIRLIST to List All Directories

Press PF3 to quit the DIRLIST display.

The .PARTY and .PARTY.FOOD directories and any other directories you create will remain in your hierarchy until you explicitly erase them using the ERASE command. For more information on erasing directories, see the *VM/SP CMS Command Reference*.

Putting Files Into a Directory

Although you have just created the .PARTY and .PARTY.FOOD directories, they do not contain any files. You will now want to add files to your directories. You can do this by copying existing files into the directory and by creating new files.

Copying Files to a Directory

One of the ways you can put existing files into a directory is by copying existing files using the COPYFILE command. First, access the .PARTY directory you just created. Enter the QUERY ACCESSED command to determine which file modes you have used:

query accessed

Your screen will display information similar to this:

Mode	Stat	Files	Vdev	Label/Directory
A	R/W	2	DIR	VMSYSU:yourid.
B	R/O	543	DIR	VMSYSU:MAINT.
S	R/O	1321	190	MNT190
Y/S	R/O	337	19E	MNT19E

You used file mode A to access your top directory and file mode B to access the MAINT. top directory. File modes S and Y were assigned to access some of the disks that control your virtual machine. Choose file mode D to access the .PARTY directory. Enter:

```
access .party d
```

This command will access the .PARTY directory, which is one level below your top directory, with a file mode of D. To check that the directory has been accessed, you could enter the DIRLIST command. In the column labeled *Fm*, you would see the file mode letter D.

Now, you are ready to use the COPYFILE command. A sample format of the command follows:

```
copyfile fn1 ft1 fm1 fn2 ft2 fm2
```

The first file name, file type, and file mode refer to the original file you are copying; the second file name, file type, and file mode refer to the copy you wish to create.

The MAINT. top directory (accessed with a file mode of B) contains the files INVITE SCRIPT and CAKE SCRIPT. You can copy the INVITE SCRIPT and CAKE SCRIPT files into your .PARTY directory. Keep the same file names and file types. Enter the following commands, pressing ENTER after each one:

```
copyfile invite script b = = d
```

```
copyfile cake script b = = d
```

The following diagram represents your directory structure now:

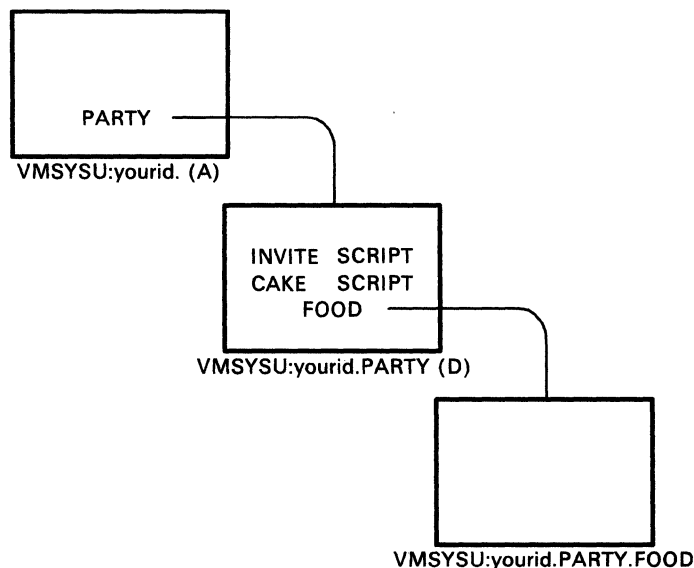


Figure 15. Files Within the .PARTY Directory

You can use the COPYFILE command to copy files from one directory to another, as we did in this example, or to copy files from a minidisk to a directory (or vice versa). You simply need to know the file mode of the directory or minidisk where the file is located and the file mode of the destination directory or minidisk.

Creating Files with XEDIT

Another way to add files to your directories is to create new files using XEDIT. Once you have accessed a directory, you can simply use the file mode to create new files in that directory.

For example, to create a file called FUNTIMES SCRIPT within the .PARTY directory (accessed as D), you would enter the command, XEDIT FUNTIMES SCRIPT D.

If you need more information on using XEDIT to create files, see Chapter 8, "Editing Your Files."

Renaming Your Files and Directories

In Chapter 3, "CMS File System," we briefly discussed how to rename CMS files. You can use the RENAME command to rename your own files. You can also rename a file in another user's directory if you have write authority to the base file (the original file) and the directory where the file resides. (We will discuss write authority to files and directories later in this chapter.) Also, you can use the RENAME command to rename a directory.

To rename your own files, use the RENAME command as follows:

```
rename fn1 ft1 dirid fn2 ft2 dirid
```

To rename a file, specify the original file name, original file type and the directory identifier followed by the new file name and new file type. Repeat the directory identifier.

Note: If you are renaming a file in another user's directory, you must use the directory name, not the file mode, when using the RENAME command.

You can only rename files within the same directory. You cannot specify a new directory name because it is not possible to use the RENAME command to move files to other directories. If you wish to move files between directories, you can do so with the RELOCATE command, which we will discuss in the section, "Relocating Your Files and Directories."

To rename the INVITE SCRIPT file (in your .PARTY directory—accessed as D) to GUESTS SCRIPT, enter the following command:

```
rename invite script d guests script d
```

If you wish to rename a directory, use the following format:

```
rename dirid1 dirid2
```

Simply specify the original directory identifier and the new directory identifier. You can use the RENAME command to rename directories; however, you cannot use RENAME to move a directory to another parent directory. Also, you cannot rename directories you do not own.

Rename your .PARTY.FOOD directory to .PARTY.TREATS by entering the following command:

```
rename .party.food .party.treats
```

You could not rename the .PARTY.TREATS directory to make it a subdirectory of your top directory. That is, you *cannot* enter this command:

```
rename .party.food .treats
```

If you want to relocate your files or directories, see the following section for information on how to do so.

Relocating Your Files and Directories

If you do not wish to create new files in your directories, you can relocate files from other directories by using the RELOCATE command. A sample format of the RELOCATE command follows:

```
relocate [fn ft] dirid1 TO dirid2
```

To relocate a file, you specify the file name and file type, followed by the name of the directory where the file is currently located (dirid1), the word TO, and the name of the destination directory (dirid2).

For example, you can relocate the CAKE SCRIPT file, which is currently in your .PARTY directory, to your .PARTY.TREATS directory. The easiest way to relocate files is to access the directories and use file modes. At this point, enter:

```
dirlist
```

You previously accessed the .PARTY directory as D, so you can use E to access the .PARTY.TREATS directory.

Currently, the DIRLIST display is on your screen. Move your cursor to the *Cmd* column on the line across from the listing for the VMSYSU:yourid.PARTY.TREATS directory. Enter:

```
access/ e
```

Be sure to leave a space after the slash. This is a short-cut method of specifying the directory name when you access a directory from DIRLIST.

You can enter several commands from the *Cmd* area of DIRLIST. The same is true of the FILELIST screen display. See the descriptions of DIRLIST and FILELIST in the *VM/SP CMS Command Reference* for information on the commands that can be entered from their screen displays.

Press PF3 to remove the DIRLIST display from your screen.

Both directories are now accessed and you can use file mode letters to relocate the CAKE SCRIPT file. Enter:

```
relocate cake script d to e
```

RELO is short for RELOCATE. Once you have entered this command, the CAKE SCRIPT file will be in the .PARTY.TREATS directory. CAKE SCRIPT yourid.PARTY.TREATS is the long version of the new name of the file. (It is now located in the .PARTY.TREATS directory, a subdirectory of .PARTY, which is a subdirectory of your top directory).

The following diagram shows your directory structure now:

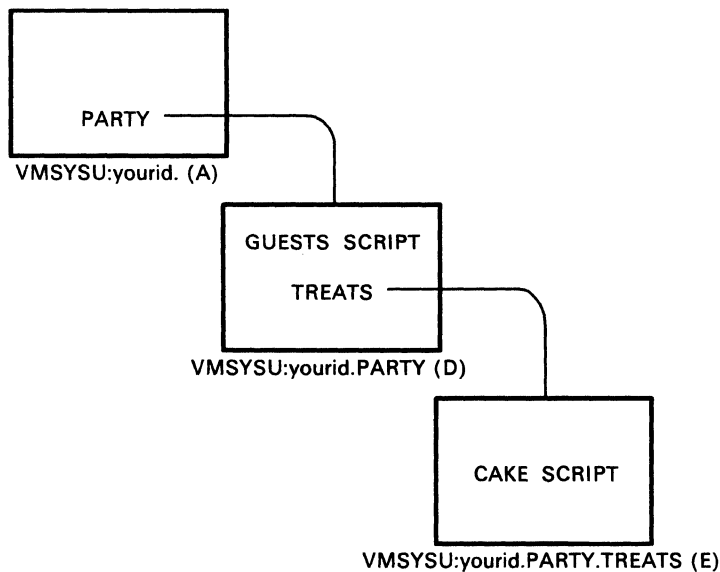


Figure 16. Moving a File to the .PARTY.TREATS Directory

You can also use the **RELOCATE** command to relocate an entire directory and all the files it contains. To do so, you would simply specify the current directory name, followed by the word **TO**, and the destination directory name.

Create a new directory under the **.PARTY** directory:

```
create directory .party.favors
```

The following diagram represents your directory structure now:

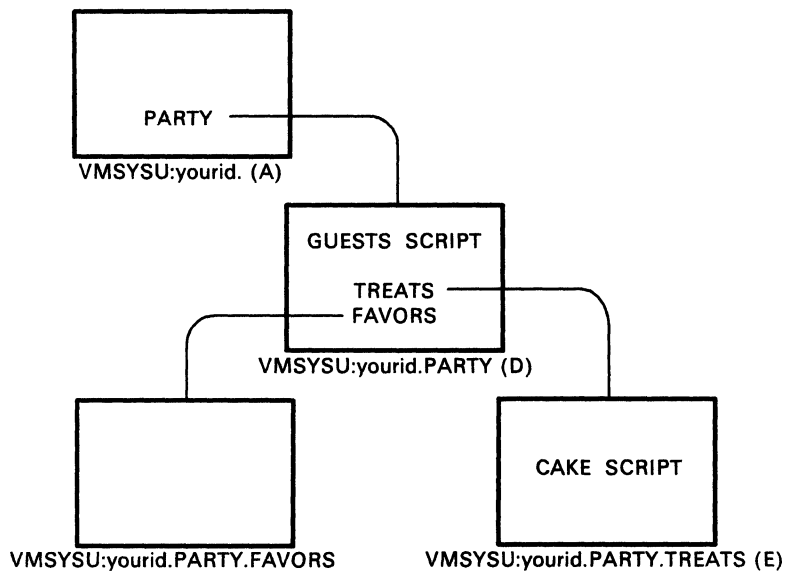


Figure 17. Creating the .PARTY.FAVORS Directory

You can relocate the .PARTY.FAVORS directory to make it a subdirectory of the .PARTY.TREATS directory. Because you have not yet accessed the new directory, you must relocate it by using the directory name. Enter:

```
relocate .party.favors to .party.treats
```

Your directory structure now looks like this:

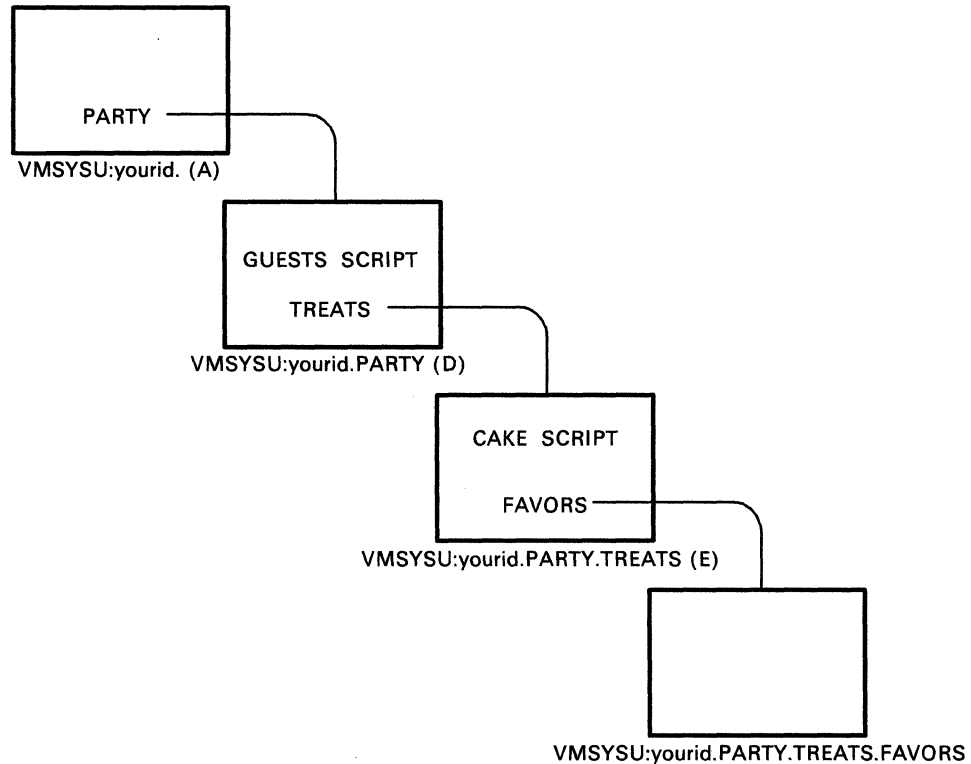


Figure 18. Relocating the .PARTY.FAVORS Directory

Once you have entered this command, the .PARTY.FAVORS directory and any files it contains (currently none) would be relocated to below the .PARTY.TREATS directory. It would now be the .PARTY.TREATS.FAVORS directory because it is one level below the .PARTY.TREATS directory.

Create one more directory below the .PARTY.FAVORS directory:

```
create directory .party.treats.favors.games
```

To check that the commands worked, enter:

```
dirlist
```

With no specified directory identifier, the DIRLIST will default to your top directory and list all the directories below it. The output will look like this:

```

yourid  DIRLIST  A0  V 319  Trunc=319  Size=5  Line=1  Col=1  Alt=0
Cmd  Fm  Directory Name
-   A  VMSYSU:yourid.
    D  VMSYSU:yourid.PARTY
    E  VMSYSU:yourid.PARTY.TREATS
    -  VMSYSU:yourid.PARTY.TREATS.FAVORS
    -  VMSYSU:yourid.PARTY.TREATS.FAVORS.GAMES

1= Help      2= Refresh  3= Quit    4= Sort(fm)  5= Sort(dir)  6= Auth
7= Backward  8= Forward  9=         10=          11= Filelist  12= Cursor

====>
X E D I T 1 File

```

Figure 19. Listing All Your Directories

You now have your top directory (accessed as A), your .PARTY directory (accessed as D), your .PARTY.TREATS directory (accessed as E), your .PARTY.TREATS.FAVORS directory (not yet accessed), and your .PARTY.TREATS.FAVORS.GAMES directory (not yet accessed).

Unlike COPYFILE, which creates a duplicate copy of the file in a new location, RELOCATE moves the file from one place to another. However, any aliases or authorities you may have created earlier are unchanged. Later in this chapter, we will discuss aliases and authorities, and you will see why this can be important for your work.

When you are using the RELOCATE command, remember that you cannot relocate a file or directory to another user's file space or to another file pool. Also, to relocate a file or directory, you must be the owner.

Press PF3 to remove the DIRLIST display from your screen.

Erasing a Directory

When you erase a directory, you must first determine what to do with the files (if there are any) in the directory. If you wish to keep the files, but erase the directory, you must first relocate these files to another directory. The RELOCATE command is detailed in the next section.

A sample format of the ERASE command follows:

```
erase fn ft dirid (nofiles
```

The NOFILES option of the ERASE command tells CMS that you have emptied the directory of all files and aliases before entering the command. (You will learn about aliases later.) NOFILES is the default for the ERASE command.

If you wish to erase not only the directory but also the file(s) it contains, specify the FILES option when using the ERASE command. If FILES is not specified and the directory contains one or more files, the erase is not performed. Additionally, if any subdirectories branch from the directory, the directory is not erased.

Navigating Through Your Directories

You can use the options of the FILELIST command and the PF keys that appear on your screen to navigate through your directories and see the files they contain.

Before you begin, copy some additional files from the MAINT. top directory to your .PARTY and .PARTY.TREATS directories. Enter the following commands:

```
copy theme script b = = d
copy music script b = = d
copy drink script b = = e
copy cookies script b = = e
```

The diagram that follows represents your directory structure now:

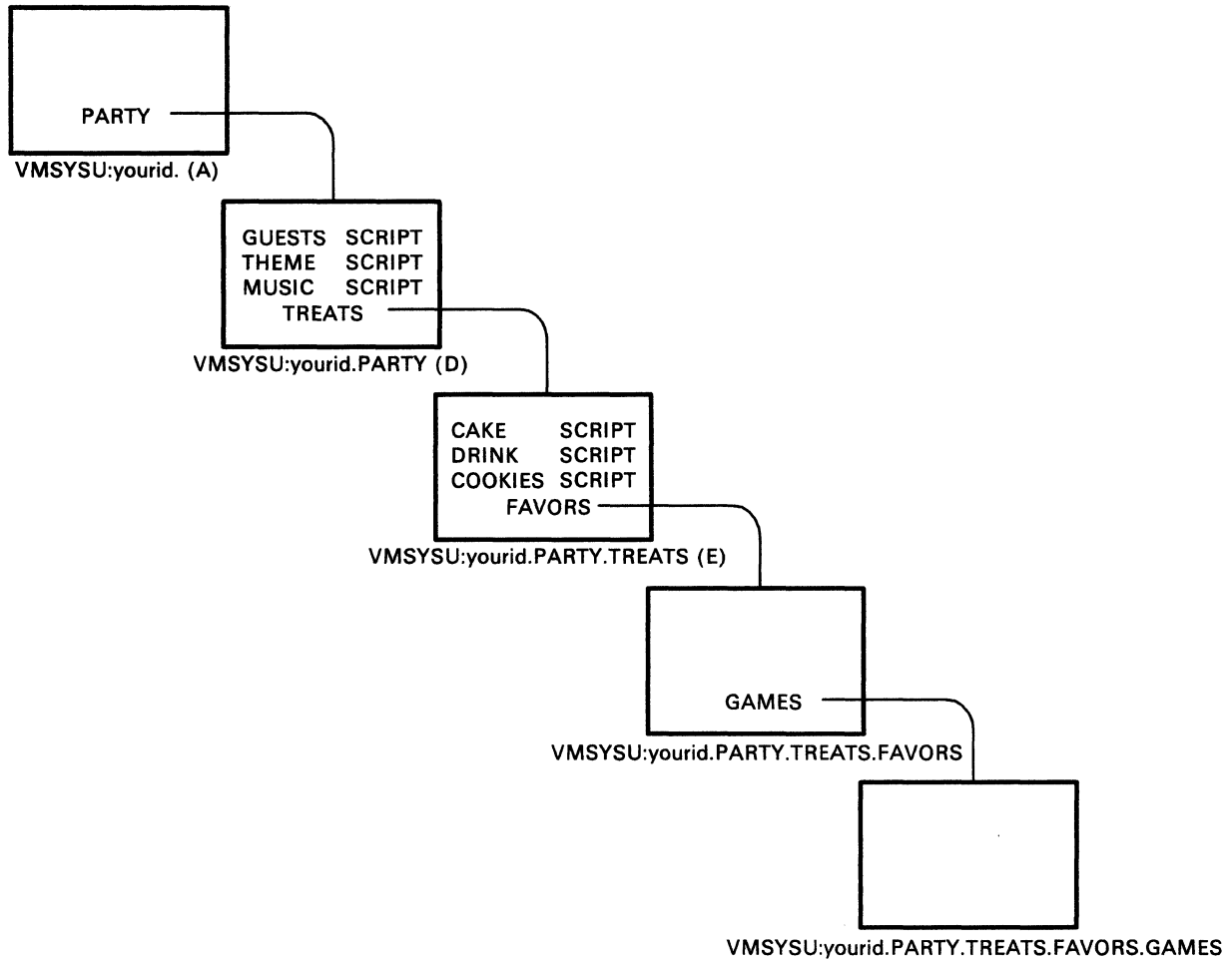


Figure 20. Copying More Files to .PARTY and .PARTY.TREATS

Once again, DIRLIST is a good starting point for working with your files. Enter:
dirlist

To see a FILELIST display of the .PARTY.TREATS directory, position your cursor on the line for .PARTY.TREATS and press PF11.

Your screen will look like this:

```

yourid FILELIST A0 V 108 Trunc=108 Size=4 Line=1 Col=1 Alt=0
Directory = VMSYSU:yourid.PARTY.TREATS
Cmd  Filename Filetype Fm Format Lrecl  Records  Blocks  Date    Time
-    COOKIES  SCRIPT  E1 V      37      6        1  1/06/88 16:03:52
     DRINK   SCRIPT  E1 V      13      8        1  1/06/88 16:03:42
     FAVORS  E DIR   -        -        -   1/06/88 15:59:38
     CAKE   SCRIPT  E1 V      55     13        1  1/06/88 15:56:16

1= Help      2= Refresh  3= Quit    4= Cancel    5= Sort(dir)  6= Sort(size)
7= Backward  8= Forward  9= FL /n  10= Share    11= XED/FILEL 12= Cursor

====>
X E D I T 1 File

```

Figure 21. Using PF11 from DIRLIST

This display is called the FILELIST STATS screen. It shows a listing of all the files contained in the directory and a listing of all the subdirectories that are one level below the .PARTY.TREATS directory.

The first column in the FILELIST display, labeled *Cmd*, is where you would enter commands for a specific file or directory listed. The next two columns show the file name and file type for files. Names of subdirectories are also listed in the *Filename* column, but the *Filetype* column is blank.

The column labeled *Fm* shows the file mode where the parent directory (the directory name listed at the top of the screen) is accessed.

The next column shows the *Format*. For files, this column will show an **F** for fixed format files; variable format files will be indicated by a **V**. Directories are indicated by **DIR**.

The columns, *Lrecl*, *Records*, and *Blocks*, carry information on the size of the files shown. Listings for directories show a dash (—) in the columns for *Lrecl*, *Records*, and *Blocks*.

The *Date*, and *Time* columns show when files were last updated. For directories, these columns show when the directories were created.

FILELIST shows you that the .PARTY.TREATS directory contains the files CAKE SCRIPT, DRINK SCRIPT, and COOKIES SCRIPT, and the .PARTY.TREATS.FAVORS directory (a subdirectory of the .PARTY.TREATS directory).

The .PARTY.TREATS.FAVORS directory is shown with file mode E, the file mode of the parent directory (.PARTY.TREATS). Notice that subdirectories do not have file mode numbers.

To see what the .PARTY.TREATS.FAVORS directory contains, you can move one level further down and display another FILELIST screen. Move your cursor to the line for the FAVORS directory and press PF11. Your screen will look like this:

```

yourid FILELIST A0 V 108 Trunc=108 Size=1 Line=1 Col=1 Alt=0
Directory = VMSYSU:yourid.PARTY.TREATS.FAVORS
Cmd  Filename Filetype Fm Format Lrecl  Records   Blocks   Date     Time
-    GAMES      Z DIR      -          -         -    1/06/88 16:01:29

1= Help      2= Refresh  3= Quit    4= Cancel   5= Sort(dir) 6= Sort(size)
7= Backward  8= Forward  9= FL /n 10= Share 11= XED/FILEL 12= Cursor
Directory has been temporarily accessed as filemode Z
====>
X E D I T 1 File

```

Figure 22. Using PF11 from FILELIST

The .PARTY.TREATS.FAVORS directory contains the .PARTY.TREATS.FAVORS.GAMES subdirectory. Now the .PARTY.TREATS.FAVORS directory is the parent directory. Notice the message at the bottom of the screen. Because you have not yet accessed it, FILELIST temporarily accesses the .PARTY.TREATS.FAVORS directory with the first available file mode letter, starting at the end of the alphabet (Z).

The .PARTY.TREATS.FAVORS.GAMES directory is shown here with a file mode of Z (the file mode of its parent). Because FILELIST temporarily accesses subdirectories that are not already accessed, if the .GAMES directory contained any files, you could use PF11 to move further down the directory structure and list them.

Move your cursor to the line for the .PARTY.TREATS.FAVORS.GAMES directory and press PF11. Because the directory does not contain any files, you will see the following message at the bottom of your screen:

```
Directory is empty
```

At this point, press PF3 to return to the original FILELIST screen.

You can always tell where you are in the hierarchy by referring to the line of the FILELIST screen labeled *Directory*. Currently, this line shows the following:

```
VMSYSU:yourid.PARTY.TREATS
```

This means that the files (and directories) you are viewing are in the .PARTY.TREATS directory, a subdirectory of the .PARTY directory. .PARTY is, in turn, a subdirectory of your top directory (indicated by your user ID). VMSYSU is the name of your file pool.

On the FILELIST screen, you have several PF keys to provide you with information about the files and directories listed:

Table 5. FILELIST STATS PF Keys		
Key	Meaning	Usage
PF4	Cancel	The PF4 key lets you exit all the way out of FILELIST, regardless of where you are within FILELIST and how many times you pressed PF11 to enter new displays.
PF5	Sort(dir)	The PF5 key sorts the output currently displayed on your screen. Directories are listed alphabetically, followed by files, which are listed by date and time.
PF6	Sort(size)	When you press PF6, the items shown in the FILELIST display will be sorted by block size, from largest to smallest.
PF9	FL /n	When you place your cursor next to a file and press PF9, your screen displays a list of all files with that same file name and any other file type and file mode.
PF10	Share	Pressing the PF10 key is the same as entering the FILELIST command with the SHARE option. A new FILELIST display is shown providing more data and another set of PF keys. You can press PF10 again to toggle back to the FILELIST STATS screen. The FILELIST SHARE screen and PF keys are discussed later.
PF11	XED/FILEL	When you press PF11, if your cursor is located on a file, PF11 will XEDIT the file. If your cursor is on a directory, PF11 will show a FILELIST display of the files within that directory. You can press PF3 to return to the FILELIST STATS screen.

You can press PF10 to display the FILELIST SHARE screen. This screen shows more information about your files. PF10 is a toggle key. If you press it repeatedly, you will toggle between the FILELIST STATS screen (the default FILELIST screen) and the FILELIST SHARE screen.

Press PF10 now. Your screen will look like this:

```

yourid  FILELIST A0 V 149 Trunc=149 Size=4 Line=1 Col=1 Alt=18
Directory = VMSYSU:yourid.PARTY.TREATS
Cmd  Filename Filetype Fm Owner  Type  R W
-   COOKIES  SCRIPT  E1 yourid  BASE  X X
   DRINK    SCRIPT  E1 yourid  BASE  X X
   FAVORS   E  yourid  DIR   X X
   CAKE     SCRIPT  E1 yourid  BASE  X X

1= Help      2= Refresh  3= Quit    4= Cancel   5= Sort(dir)  6= Auth
7= Backward  8= Forward  9= Alias  10= Stats   11= XED/FILEL 12= Cursor

====>
                                         X E D I T  1 File

```

Figure 23. The FILELIST SHARE Screen

Like the FILELIST STATS screen, the FILELIST SHARE screen contains a line labeled *Directory*, which shows you exactly where you are in the hierarchy. The screen also contains columns labeled *Cmd*, *Filename*, *Filetype*, and *Fm*, which contain the same information as on the FILELIST STATS screen.

The SHARE display also contains a column labeled *Owner*, which lists the owner of each file or directory displayed.

The next column, labeled *Type*, shows the type of item displayed, such as directory, minidisk, base file, alias, erased, or revoked. We will discuss base files and aliases in the section “Creating Aliases to Files” on page 95. For more information on erased and revoked files and aliases, see “Erasing Your Base Files” on page 105 and “Revoking Authority” on page 110.

The last two columns indicate whether you have read or write authority on each file or directory displayed. An X indicates that you have authority; a dash (-) means that you do not.

A quick way to bring up the FILELIST SHARE screen is to enter the FILELIST command with the SHARE option. See the *VM/SP CMS Command Reference* for more information.

The FILELIST SHARE screen also contains several PF keys to provide information on the data displayed. While some of these keys are the same as those on the FILELIST STATS screen, you will also be able to use these keys:

Table 6. FILELIST SHARE PF Keys

Key	Meaning	Usage
PF6	Auth	Pressing the PF6 key executes the AUTHLIST command and displays the authority you have on the file or directory where your cursor is located. If you are the owner, it also lists information on any authority you have granted to other users. The AUTHLIST command is discussed in detail later.
PF9	Alias	When you press PF9, you will execute the ALIALIST command and see information regarding the file where the cursor is located. If the file is a base file that you own, you will see a list of users who have an alias to it. If the file is a base file that someone else owns, you will see a list of your aliases to it. If the file is an alias, you will see the owner of the base file. The ALIALIST command is discussed in detail later.
PF10	Stats	Pressing PF10 is the same as entering the FILELIST command with the STATS option. Your screen will display a new screen showing more data about your files and directories.

Press PF10 to toggle back to the FILELIST STATS screen. Then, press PF3 to exit from the FILELIST STATS screen and return to the DIRLIST screen. Press PF3 one more time to remove the DIRLIST display from your screen.

Sharing Files

You can share your files with other users by:

- Accessing directories
- Creating aliases
- Granting authority

We have already discussed accessing other users' directories to share files. This section will discuss sharing files by creating aliases and granting authority. Whenever you store your files in an SFS file pool, you have the option to share any of your files or directories with other users or to share none at all.

To determine who is connected to your file pool, you can use the QUERY FILEPOOL CONNECT command. A sample format of the command follows:

```
query filepool connect for userid
```

Specify a user ID or nickname to determine if a specific user is enrolled. For example, to determine if Mary is connected to the VMSYSU file pool, you could specify:

```
query filepool connect for mary
```

Your output might look like this:

```
Userid  Connected
jonesm  YES
```

To see a list of all the users currently connected to your file pool, enter:

```
query filepool connect for all
```

This chapter provides information on how to share files with users enrolled in your file pool. However, you can also use the QUERY FILEPOOL CONNECT command to determine which users are connected to other file pools. (See “Sharing Files with Users on Other Systems” on page 126. For more information, see the *VM/SP CMS Command Reference*.

Creating Aliases to Files

When you create a file, this original file is known as a *base file*. Later, you can create an *alias* to the file and place it in another directory. The alias simply serves as a pointer to the base file; the base file does not move, and you are not creating a copy of it.

Aliases allow you to reference a single file in more than one directory, or more than once in one directory. Aliases also let two different users reference the same file using different names.

When entering most CMS commands, you do not need to be concerned with whether a file is a base file or an alias. All CMS commands will work on the file name you specify, regardless of whether it is a base file or an alias.

You can create an alias to your own file if you want to point to the same information from two directories, or from two different places within the same directory. For example, assume that Jim is the owner of a file called PRICE LIST. PRICE LIST is within the directory Jim uses for the files for Project A1. When Jim is assigned a second project, he creates a separate directory to contain files for the new project.

If Jim needs to use the same pricing information for both Project A1 and his new project, Project EZ, he may find it useful to create an alias for PRICE LIST in the new directory. He can name the file EZ PRICES in the new directory. EZ PRICES is then an alias to the base file, PRICE LIST. If he wanted to XEDIT the price information, he could specify either name. The advantage to making an alias to the base file, instead of a copy of the file, is that he could make changes to either file, and the change would be reflected in both files.

An alias does not have to be created from a base file; you can create an alias to an alias. Once Jim creates the alias EZ PRICES, he can create an alias to EZ PRICES if he needs the same information for a third project.

Aliases are most useful when sharing files. The other user can erase, rename, or relocate aliases to your file without affecting your base file. Also, if you change the name of your base file, a user who has an alias to it will still be able to share the file. CMS will automatically update the pointer so that the alias still refers to the same base file.

For example, Terry and Mike are working together on a project. Mike has created a file called LOTS TODO. Terry needs to share the file and would like to have her own pointer to it. There are two ways Terry and Mike can accomplish this:

- Mike can grant Terry read or write authority on the LOTS TODO file. Terry could then create an alias to the file in one of her own directories. She could call it MIKE JOBS. Then every time she wanted to work with the file, she could XEDIT her alias MIKE JOBS. While Terry would have the alias called MIKE JOBS, the base file, LOTS TODO, is still owned by Mike.

- Terry could grant Mike write authority on one of her directories, and Mike could create an alias for Terry within that directory. He could create for her an alias called TERRY JOBS as a pointer to the LOTS TODO file. When Terry wanted to work with the file, she would XEDIT her alias, TERRY JOBS.

Either way, once Terry has an alias to the file, she and Mike would be able to XEDIT the same information.

Creating an Alias to Your Own File

Previously, you used the COPYFILE command to copy the files GUESTS SCRIPT, THEME SCRIPT, and MUSIC SCRIPT from the MAINT. directory to the .PARTY directory that you created and accessed with a file mode of D.

Suppose you also wanted to group your GUESTS SCRIPT file in the directory with the files for your party favors. You can create an alias for the GUESTS SCRIPT file in the .PARTY.TREATS.FAVORS directory by using the CREATE ALIAS command.

A sample format of the CREATE ALIAS command follows:

```
create alias fn1 ft1 dirid1 fn2 ft2 dirid2
```

The first file name, file type, and directory identifier refer to the source file (the base file or the alias you would like to create an alias to); the second file name, file type, and directory identifier refer to the alias you wish to create.

Note: When specifying the directory identifier for the alias you are creating, you cannot specify a different file mode number; the alias always has the same file mode number as the base file. Any change to the file mode number of the base file results in the same change to all the aliases to the file.

As we discussed previously, a directory identifier can be a directory name, file mode, or any other way of referring to the specific directory. One of the easiest ways to use many commands is to access the directories and use file mode letters.

Previously, you accessed directories with the file mode letters A, B, D, and E. (If you were not sure which file modes you had used, you could enter the QUERY ACCESSED command to check). Access the .PARTY.TREATS.FAVORS directory with a file mode of F:

```
access .party.treats.favors f
```

Because your .PARTY directory is already accessed as D, you are ready to create the alias. You can choose whether to use the same file name and file type for the alias and base file. For this example, use the same name as your base file in your .PARTY directory for the alias you are creating in your .PARTY.TREATS.FAVORS directory:

```
create alias guests script d = f
```

To check to see that the CREATE ALIAS command worked, do a FILELIST of all of your directories, beginning with your top directory, to look for occurrences of the GUESTS SCRIPT file. You can do this by using the SEARCH option of the FILELIST command. Enter:

```
filelist guests script a (search
```

This command example tells SFS to display every occurrence of the GUESTS SCRIPT file in any directory, beginning with your top directory (accessed with a file

mode of A). You must give the command a starting point to search from. In this case, you start from your top directory, so that the search will begin at the top directory and search all of its subdirectories.

The FILELIST command will search only the files in your own directory structure. Any minidisk files you have will not be searched. It will also not find files you have in other user's directories.

Your output will look like this:

```

yourid  FILELIST A0  V 355  Trunc=355 Size=2 Line=1 Col=1 Alt=0
Cmd  Filename Filetype Fm Directory Name
-   GUESTS  SCRIPT  D1 VMSYSU:yourid.PARTY
    GUESTS  SCRIPT  F1 VMSYSU:yourid.PARTY.TREATS.FAVORS

1= Help      2= Refresh  3= Quit    4= Dirlist  5= Sort(name) 6= Auth
7= Backward  8= Forward  9= Alias  10= Filelist 11= XEDIT     12= Cursor

====>
XEDIT 1 File

```

Figure 24. The FILELIST SEARCH Screen

The FILELIST SEARCH screen contains a column labeled *Cmd*, where you can enter commands to be executed on particular files displayed. The next columns show the *Filename* and *Filetype* for each file. The *Fm* column shows the file mode letter where the file resides (the file mode letter you used to access the directory containing that file). The last column contains the entire directory name.

Note: If you see a dash (—) in the *Fm* column of any of the displayed files, the file contained on that line resides in a directory that is not accessed. You can still XEDIT the file from this screen by pressing the PF11 key; CMS temporarily accesses the directory containing the file for the duration of the XEDIT session.

GUESTS SCRIPT appears in your .PARTY directory (currently accessed as D) and your .PARTY.TREATS.FAVORS directory (currently accessed as F). Press PF3 to quit the FILELIST display.

In the previous example, you were searching for the occurrence of a specific file. To list all of your files in a specific directory, you would replace the file name and file type with asterisks and specify the file mode letter you used to access that directory.

Therefore, FILELIST * * D would list all of your files in the .PARTY directory (accessed as D) and will also list the names of any subdirectories of the .PARTY directory, such as the .PARTY.TREATS directory.

To list all of your files, in all of your directories, enter this command:

```
filelist * * a (search)
```

This command will begin with your top directory (accessed as A) and list the files in the top directory and each directory below.

Your output will look like this:

```
yourid  FILELIST A0  V 355  Trunc=355 Size=7 Line=1 Col=1 Alt=0
Cmd  Filename Filetype Fm Directory Name
-   GUESTS  SCRIPT  D1  VMSYSU:yourid.PARTY
    MUSIC  SCRIPT  D1  VMSYSU:yourid.PARTY
    THEME  SCRIPT  D1  VMSYSU:yourid.PARTY
    CAKE   SCRIPT  E1  VMSYSU:yourid.PARTY.TREATS
    COOKIES SCRIPT  E1  VMSYSU:yourid.PARTY.TREATS
    DRINK  SCRIPT  E1  VMSYSU:yourid.PARTY.TREATS
    GUESTS SCRIPT  F1  VMSYSU:yourid.PARTY.TREATS.FAVORS

1= Help      2= Refresh  3= Quit    4= Dirlist   5= Sort(name)  6= Auth
7= Backward  8= Forward  9= Alias  10= Filelist 11= XEDIT     12= Cursor

====>
X E D I T  1 File
```

Figure 25. Using FILELIST SEARCH to List All Your Files

The FILELIST SEARCH screen contains several PF keys to determine information about the data on your screen. While some of these keys are the same as those on the FILELIST STATS or FILELIST SHARE screens, FILELIST SEARCH also provides you with these keys:

Table 7 (Page 1 of 2). FILELIST SEARCH PF Keys		
Key	Meaning	Usage
PF4	Dirlist	The PF4 key lets you enter the DIRLIST command for the directory on the line where your cursor is located.
PF5	Sort(name)	The PF5 key sorts the output currently displayed on your screen alphabetically by file name and file type.
PF10	Filelist	When you press PF10, the directory where your cursor is located is temporarily accessed (if it is not already accessed), and you will see a FILELIST display of all the files in the directory.

Table 7 (Page 2 of 2). FILELIST SEARCH PF Keys

Key	Meaning	Usage
PF11	XEDIT	PF11 will let you XEDIT the file on the line where your cursor is located when you press the key. You can still XEDIT a file residing in a directory that is not accessed; CMS gives you temporary access for the duration of the XEDIT session.

Press PF3 to quit the FILELIST SEARCH display.

Creating an Alias to Another User's File

If you have authority to another user's file, there are several reasons you may find it useful to create an alias to the file. As we discussed earlier, an alias provides another way of referencing information in a file.

For example, the MAINT. top directory contains the MAINT.SAMPLE subdirectory. Within the MAINT.SAMPLE subdirectory is a file called GIFTS SCRIPT. You have been granted read authority to this file. If you have read authority to the MAINT.SAMPLE directory, you could access the directory to see what the GIFTS SCRIPT file contains. However, instead of accessing the entire directory, you may find it easier and quicker to create an alias to the specific file you want to use.

Another reason you may want to create an alias to another user's file is that this lets you place the alias wherever you like within your own directory structure. For example, because you already have a directory of party favors (your .PARTY.TREATS.FAVORS directory), you could create the alias in that directory. Otherwise, if you accessed the entire MAINT.SAMPLE directory, you would have to work with the directory structure the owner had organized.

An additional reason you may find it useful to create an alias to another user's file is in a situation where you have authority to a file, but not to the directory in which the file resides. In this case, you will not be able to access the directory and establish a file mode for it. Therefore, you would not be able to use any of the commands that require file modes, such as XEDIT, FILELIST, and COPYFILE. The CREATE ALIAS command would then provide you with the means to reference the information in the file through one of your own directories.

Even if you have write authority to a directory where a file resides, you may not be able to work with the file you need. This is because SFS accesses other users' directories as read-only. Many commands that require file modes will only work on your own directories, where access is read/write. To use those commands, you must create an alias in your own directory and access the directory as read/write. (SFS functions this way so that programs and execs written before Release 6 will operate correctly.)

Note: Remember, you cannot share files across file pools. If you want to create an alias to another user's file, that user must be in your file pool.

Following is a list of commands that can be used to modify the directory regardless of whether directory access is read/write or read-only. If you wish to use a command that writes to a file and the command is **not** in this list, you should create an alias in your own directory for the file you wish to work with.

COPYFILE	ERASE*
CREATE ALIAS	RELOCATE
CREATE DIRECTORY	RENAME*
DISCARD*	XEDIT

*You must use the *dirname* with the command if using it on a read-only directory.

Create an alias to the GIFTS SCRIPT file in the MAINT.SAMPLE directory and put it in your .PARTY.TREATS.FAVORS directory, which you previously accessed with a file mode of F. Give your alias the name PRIZES SCRIPT:

```
create alias gifts script maint.sample prizes script f
```

After you enter the CREATE ALIAS command, you have a pointer to the data in the GIFTS SCRIPT file, but the actual data still resides in the MAINT.SAMPLE directory. After the owner of the GIFTS SCRIPT file updates and stores the file, you will be able to see the updated information through your alias, PRIZES SCRIPT.

To see that the CREATE ALIAS command worked, do a FILELIST on your .PARTY.TREATS.FAVORS directory to see all the files it contains:

```
filelist * * f
```

The asterisks indicate that you wish to list all the files, regardless of file name and file type.

Your output will look like this:

```
yourid  FILELIST A0 V 108 Trunc=108 Size=3 Line=1 Col=1 Alt=0
Directory = VMSYSU:yourid.PARTY.TREATS.FAVORS
Cmd  Filename Filetype Fm Format Lrecl  Records  Blocks  Date  Time
-   GAMES      F  DIR      -      -      -   1/06/88 16:01:29
    GUESTS    SCRIPT  F1 V      29      13      1   1/06/88 15:56:04
    PRIZES    SCRIPT  F1 V      16      6       1   1/06/88 14:12:36

1= Help      2= Refresh  3= Quit    4= Cancel  5= Sort(dir)  6= Sort(size)
7= Backward  8= Forward  9= FL /n 10= Share 11= XED/FILEL 12= Cursor

====>
XEDIT 1 File
```

Figure 26. Listing All the Files in a Directory

Move your cursor to the line for the PRIZES SCRIPT file, and press PF11 to XEDIT the file. Your screen will look like this:

```
PRIZES SCRIPT F1 V 132 Trunc=132 Size=6 Line=0 Col=1 Alt=0
Warning: Not authorized to lock file PRIZES SCRIPT F1

===== * * * Top of File * * *
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...
===== Gifts For Guests
=====
===== Balloons
===== Candies
===== Party Hats
===== Noisemakers
===== * * * End of File * * *

=====> -
X E D I T 1 File
```

Figure 27. Using PF11 to XEDIT a File

Notice the message at the top of the screen, warning you that you are not authorized to lock the file. File locking will be discussed later. You can view the contents of the PRIZES SCRIPT file, but you cannot change them because you only have read authority to the base file. If you had write authority and were able to make changes to your alias, the owner of the file and other users who had authority to the base file would see your changes through their files.

Press PF3 to quit from the PRIZES SCRIPT file. You will be returned to the FILELIST STATS screen.

You can also find out information about the GIFTS SCRIPT base file from the FILELIST SHARE screen. Move your cursor to the entry for the PRIZES SCRIPT file, and press PF10 to display the FILELIST SHARE screen. Your screen will look like this:


```

yourid  FILELIST A0  V 149  Trunc=149 Size=3 Line=1 Col=1 Alt=5
Directory = VMSYSU:yourid.PARTY.TREATS.FAVORS
Cmd  Filename Filetype Fm Owner   Type  R W
-   GAMES      Fm yourid  DIR   X X
   GUESTS     SCRIPT  F1 yourid  ALIAS X X
   PRIZES     SCRIPT  F1 MAINT   ALIAS X -

1= Help      2= Refresh  3= Quit    4= Cancel   5= Sort(dir)  6= Auth
7= Backward  8= Forward  9= Alias   10= Stats   11= XED/FILEL 12= Cursor

====>
X E D I T  1 File

```

Figure 28. Using PF10 for Information on Aliases

You can see that the PRIZES SCRIPT file is an alias to which you have read authority. The user ID MAINT is listed in the column labelled *Owner* because MAINT owns the base file.

Press PF10 to toggle back to the FILELIST STATS display. Press PF3 to remove the FILELIST display from your screen.

Using the QUERY ALIAS Command

The QUERY ALIAS command is useful for determining who has created aliases to your file. If you are the owner of a base file, and you enter the QUERY ALIAS command on that base file, you will see a list of the users who have an alias to your file and the number of aliases they have. Also, if the aliases reside in one of your own directories or in another user's directory to which you have read authority, QUERY ALIAS will show you the file name and file type of each alias.

If you enter the QUERY ALIAS command on a base file that another user owns, you will see the names of any aliases you have to that base file.

If you enter the QUERY ALIAS command on an alias, you will see who owns the base file. If you own or have read or write authority to the directory where the base file resides, you will also be able to see the name of the base file.

You can enter the QUERY ALIAS command from the command line. A sample format of the command follows:

```
query alias fn ft dirid
```

The file name, file type and directory identifier specify the file you wish to query. If you do not specify a directory identifier, the command will default to the directory accessed as A.

In the previous exercises, you created an alias, GUESTS SCRIPT, in your .PARTY.TREATS.FAVORS directory. The base file, with the same name, is located in your .PARTY directory. Enter the QUERY ALIAS command on the base file:

```
query alias guests script .party
```

Your screen will look like this:

```
Directory = VMSYSU:yourid.PARTY
Filename Filetype Fm T Userid   Num Filename Filetype Directory
GUESTS   SCRIPT   D1 B yourid   1 GUESTS   SCRIPT   .PARTY.TREATS.FAVORS
```

The QUERY ALIAS display shows you have an alias to the GUESTS SCRIPT base file (the base file is indicated by a **B** in the fourth column), and that the alias is located in your .PARTY.TREATS.FAVORS directory.

Your user ID is shown in the *Userid* column because you are the owner of the alias. The *Num* column indicates the number of aliases to the file by the user. The file name and file type of your alias, GUESTS SCRIPT, are shown to the right, as well as .PARTY.TREATS.FAVORS, the directory name of the directory where the alias resides.

You could also have entered the QUERY ALIAS command on your alias in the .PARTY.TREATS.FAVORS directory. To do so, you would enter:

```
query alias guests script .party.treats.favors
```

This command would show you that GUESTS SCRIPT is an alias and that the base file by the same name resides in the .PARTY directory. You would be able to see the name of the directory where the base file resides because it is your own directory—you automatically have read and write authority to it.

If you entered a QUERY ALIAS command on an alias, and the base file resided in a directory to which you do not have any authority, you would not be able to see the file name, file type, or directory name of the base file.

Using ALIALIST

There is also another way to determine who has an alias to a file. You can use one of the PF keys that appears on both the FILELIST SHARE and SEARCH screens to enter the ALIALIST command.

For example, to see if there are any aliases to the GUESTS SCRIPT file in your .PARTY directory, enter:

```
filelist guests script d (share
```

Your screen will look like this:

```

yourid  FILELIST A0 V 149 Trunc=149 Size=1 Line=1 Col=1 Alt=0
Directory = VMSYSU:yourid.PARTY
Cmd  Filename Filetype Fm Owner  Type  R W
-   GUESTS  SCRIPT  D1 yourid  BASE  X X

1= Help      2= Refresh  3= Quit    4= Cancel   5= Sort(dir)  6= Auth
7= Backward  8= Forward  9= Alias  10= Stats   11= XED/FILEL 12= Cursor

====>
X E D I T 1 File

```

Figure 29. Using the PF Keys on the FILELIST SHARE Screen

Position your cursor on the line for the GUESTS SCRIPT file and press PF9 to enter to the ALIALIST command. Your screen will look like this:

```

yourid  ALIALIST A0 V 190 Trunc=190 Size=1 Line=1 Col=1 Alt=0
Base file = GUESTS SCRIPT VMSYSU:yourid.PARTY
Userid  Num Filename Filetype Directory
yourid  1 GUESTS  SCRIPT  .PARTY.TREATS.FAVORS

1= Help      2= Refresh  3= Return  4= Sort(type) 5= Sort(name) 6= Sort(dir)
7= Backward  8= Forward  9= S(user) 10=          11=          12=

====> _
X E D I T 1 File

```

Figure 30. Entering the ALIALIST Command

As you can see, the information you receive from ALIALIST is similar to the information you received from the QUERY ALIAS command. The first column, *Userid* shows the user ID of anyone with an alias to the file. Here, it shows your user ID because you have an alias to GUESTS SCRIPT.

The *Num* column shows how many aliases the user has—in this case 1. The next two columns show the file name and file type of the alias, GUESTS SCRIPT. The final column shows the directory where the alias resides—your .PARTY.TREATS.FAVORS directory.

If another user has an alias to your file that resides in a directory to which you do not have authority, you would see only the user ID and the number of aliases; the directory name would not appear. For example, if Jay had two aliases to your GUESTS SCRIPT in his PEOPLE directory (to which you do not have authority), you would only see one listing with his user ID, and the number 2.

The ALIALIST display provides the following PF keys to let you sort the entries shown on your screen:

Table 8. ALIALIST PF Keys		
Key	Meaning	Usage
PF4	Sort(type)	The PF4 key sorts the files displayed, first by file type, then by file name.
PF5	Sort(name)	PF5 sorts the files displayed, first by file name, then by file type.
PF6	Sort(dir)	PF6 sorts the display alphabetically, first by directory name, then by file name, then by file type.
PF9	S(user)	PF9 sorts the display alphabetically by user ID.

Press PF3 to return to the original FILELIST screen. Press PF3 again to quit FILELIST.

Note: If you find the SHARE or SEARCH options of FILELIST to be more helpful for your work, you can set up either as the default FILELIST display. For more information on how to do this, see the DEFAULTS command in the *VM/SP CMS Command Reference*.

Erasing Your Base Files

When you erase a base file, users with an alias to it will see that the status of the file is changed when they use such commands as FILELIST or QUERY ALIAS. The same is true, of course, if other users erase base files to which you have an alias.

For example, assume that you have an alias to CURTISJ's file, PALS SCRIPT. Your alias is called FRIENDS SCRIPT and is in a directory you have accessed as P. If Jay erases the base file, when you entered the command FILELIST * * P (SHARE, your screen would look like this:

```

yourid  FILELIST A0 V 149 Trunc=149 Size=3 Line=1 Col=1 Alt=0
Directory = VMSYSU:yourid.PARTY.PEOPLE.ATTEND
Cmd  Filename Filetype Fm Owner   Type   R W
-   NEIGHBOR SCRIPT  P1 yourid  BASE   X X
    FAMILY  SCRIPT  P1 yourid  ALIAS  X X
    FRIENDS SCRIPT  P1 CURTISJ ERASED - -

1= Help      2= Refresh  3= Quit    4= Cancel   5= Sort(dir)  6= Auth
7= Backward  8= Forward  9= Alias   10= Stats   11= XED/FILEL 12= Cursor

====>
X E D I T 1 File

```

Figure 31. Erased Indicator on the FILELIST SHARE Screen

You would see “ERASED” in the column labelled *Type*. This tells you that the owner, CURTISJ, has erased the base file for your FRIENDS SCRIPT alias.

If, instead of entering the FILELIST command, you had entered the QUERY ALIAS command on your alias, FRIENDS SCRIPT, your screen would look like this:

```

Directory = VMSYSU:yourid.PARTY.PEOPLE.ATTEND
Filename Filetype Fm T Userid  Num Filename Filetype Directory
FRIENDS  SCRIPT  P1 E CURTISJ  1

```

There is an **E** in the *Type* column to indicate that the base file for the FRIENDS SCRIPT alias has been erased, and the file name, file type, and directory name for the base file are not shown, even if you have authority to the directory where the base file previously resided.

In either case, when you see that the PALS SCRIPT base file no longer exists, your alias to it is no longer valid. You can use the ERASE command to remove the reference to the erased file. For more information on erasing files, see the ERASE command in the *VM/SP CMS Command Reference*.

Authorizing Others to Access Your Files and Directories

Users can share your files or directories if you grant them authority to do so. You can grant another user read or write authority on any of your base files or directories; other users can also choose to share any of their files or directories with you.

When you create a file or subdirectory in one of your own directories, you are the owner of the new file or directory. As the owner, you automatically have the authority to read from or write to the file or directory, rename it, or relocate it. You can erase any of your directories, and you can create or erase files within directories you own. You cannot erase, rename, or relocate your top directory.

You can grant other users read or write authority to your directories or to any of the base files in your directories. Also, if you grant a user authority to an alias of a base file you own, you are, in reality, granting authority to the base file. To grant authority on files and directories that you own, use the **GRANT AUTHORITY** command. You can grant either *read* or *write* authority. When you grant authority, the default is read authority. Write authority always implies read authority.

Other users cannot read or modify your files or directories, unless you authorize them to do so. If you no longer want to share a file or directory, you can revoke the authority you have granted.

It is important to remember that only the person who owns a file or directory can grant a user authority on it. (The only exception to this is that an SFS administrator can grant authority to another user. See “Administrator Authority” on page 108 for more information.) If, for example, you grant authority on one of your files to a user named Bill, he cannot grant that authority to someone else.

You can avoid confusion about authorization if you keep these rules in mind:

1. An authority on a file does not imply you have any authority on the directory where the file resides.
2. An authority on a directory does not imply that you have any authority on any file within the directory.
3. Only the person who owns the file or directory can grant and revoke authority on it.

Read Authority on a File

Read authority on a file means that the user can read, but not change, the contents of the file. If you give read authority to another user, that user can make a copy of the file, print it, browse through it, define aliases for it, and create a **SHARE** lock for it. (We will discuss locking later in this chapter.) If that user tries to write to the file, however, SFS prevents the write and returns an error message.

Read Authority on a Directory

Read authority on a directory means that the user can read the names of objects within the directory (base files, aliases, and subdirectories). This might lead you to believe that the person can read the contents of all the files within the directory, but this is not the case.

While it is convenient to think of directories as containing files, all they really contain are the *names* of files and subdirectories. When considering authorizations, think of directories and files as separate entities.

When you have read authority on a directory, you do not have read authority on a set of files. All you can do is see the names of its base files and aliases and the names of its subdirectories. (Only file names and directory names for that immediate level are shown—items at deeper levels of the hierarchy are not shown unless you also have authority to a lower-level directory.)

Read authority to a directory does not let you rename or relocate any base files or aliases within the directory. Only the owner of the directory is allowed these functions. Also, read authority lets you lock the directory in SHARE mode only.

Write Authority on a File

Write authority on a file means you can modify the file or erase it. However, you cannot modify or erase any other files within the directory without authority on each individual file.

If you have write authority on a file, you can lock the file in any mode (EXCLUSIVE, UPDATE, or SHARE).

Write Authority on a Directory

Write authority on a directory gives the grantee a fair amount of authority over the contents of the directory. If you are granted write authority on another user's directory, you can see the names of objects within the directory (which is implied with read authority). Also, you can create aliases or new files in that directory. However, you cannot read the contents of any files unless you have read authority on the file, and you cannot change, erase or rename any file unless you have write authority on the file.

While you can create new files in the directory, the directory owner is considered the owner of any files you create in their directory. You automatically have write authority to files you create, but you cannot grant authority to other users. Only the owner of the file can do so.

With write authority on a directory, you can also create an alias in that directory to one of your own base files or to a base file of another user. Although you still own the base file, the other user would have the authority to use the alias and erase it at any time.

If you have write authority to individual files within the directory, you can modify them. If you have write authority on both the directory and an individual file, you can erase or rename the file.

You cannot delete another user's directory to which you have write authority, nor can you rename or relocate it or create or delete subdirectories to the directory. However, you can lock the directory in any mode (EXCLUSIVE, UPDATE, or SHARE).

Administrator Authority

The Shared File System administrator is the person responsible for generating file pools and managing their operation and use. An SFS administrator can do anything to a base file, alias, or directory that the owner can do, such as:

- Create files and directories
- Erase, rename, relocate, and copy files and directories
- Grant authority

- Revoke authority
- Create locks
- Delete locks

The administrator can only do things that the owner can do, unless he or she performs additional tasks first. For example, the administrator cannot create an alias for a user if that user does not already have authority on the base file. However, the administrator can grant that authority and then create the alias.

To determine who has administrator authority in your file pool, you can enter the `ADMINISTRATOR` parameter of the `QUERY ENROLL` command. For more information, see the *VM/SP CMS Command Reference*.

Granting Authority

To grant another user read and/or write authority on one of your files, use the `GRANT AUTHORITY` command. A sample format of the command follows:

```
grant authority [fn ft] dirid TO userid
```

or

```
grant authority [fn ft] dirid TO PUBLIC
```

If you wish to grant another user authority to a file, you would specify the file name, file type, and directory identifier. If you wish to grant authority on the entire directory, you would specify only a directory identifier. The word `TO` is shown in capital letters because it must be included in the command. Following the word `TO`, you would specify the user ID or nickname of the person or persons to whom you wish to grant authority.

If you wish to grant authority to everyone with access to your file pool, replace *userid* with `PUBLIC`.

To specify what type of authority you want to grant, you follow the `GRANT AUTHORITY` command with the options `READ` or `WRITE`. Read authority is the default.

Craig is working on a project with Debbie. She needs to modify his file, `SPECIAL PROJECT` file, which is in a directory Craig has accessed as `Q`. He could grant her write authority to that file as follows:

```
grant authority special project q to debbie (write
```

After Craig entered this command, Debbie would have read and write authority to the `SPECIAL PROJECT` file.

If Debbie does not have authority to the directory on which the file resides, to access the file, she could create an alias to it, then `XEDIT` the alias.

As the owner of the file, Craig can revoke the authority he has granted at any time by using the `REVOKE AUTHORITY` command. For more information on `REVOKE AUTHORITY`, see the following section.

Revoking Authority

If you no longer want another user to have authority to one of your files or directories, you can revoke the authority you granted previously with the **REVOKE AUTHORITY** command.

A sample format of the **REVOKE AUTHORITY** command follows:

```
revoke authority [fn ft] dirid FROM userid
```

To revoke authority on a file, specify the file name, file type, and directory identifier, followed by the word **FROM**, and the user ID; to revoke authority on a specific directory, specify the directory identifier, followed by **FROM**, and the user ID.

For example, if Craig later decided he no longer wanted to share the **SPECIAL PROJECT** file with Debbie, he could revoke her authority to the file by specifying the following command:

```
revoke authority special project q from debbie
```

If Craig wanted to revoke write authority from Debbie, but let her continue to read the **SPECIAL PROJECT** file, he could use the **KEEPREAD** option of the **REVOKE AUTHORITY** command:

```
revoke authority special project q from debbie (keep
```

KEEP is short for **KEEPREAD**. This command specifies that Debbie's authority should be changed from write (which Craig originally granted) to read.

Use the **PUBLIC** parameter of the **REVOKE AUTHORITY** command to revoke **PUBLIC** authority you granted earlier. You cannot revoke authority individually if you used **PUBLIC** to **GRANT AUTHORITY**. Similarly, the **PUBLIC** parameter will not revoke individual authority that you granted.

You can revoke authority individually if you grant authority individually. The **ALL** parameter revokes **PUBLIC** authority as well as individual authorities that you may have granted.

When you revoke another user's authority to one of your files, the user will see that the status of the file is changed when they enter the **FILELIST** or **QUERY AUTHORITY** commands. The same is true, of course, if other users revoke your authority on a file.

For example, assume that Mark granted you read authority to his **COOKING HINTS** file. At that time, you created an alias to the file called **CHEF TIPS** and placed the alias in your directory accessed with a file mode of **M**.

If Mark decides he no longer wants you to be able to read the file and revokes your authority, when you enter **FILELIST * * M (SHARE**, your screen would look like this:

```

yourid FILELIST A0 V 149 Trunc=149 Size=1 Line=1 Col=1 Alt=0
Directory = VMSYSU:yourid.KITCHEN
Cmd  Filename Filetype Fm Owner  Type  R W
-    CHEF     TIPS     M1 MARKD  REVOKED - -

1= Help      2= Refresh  3= Quit    4= Cancel   5= Sort(dir) 6= Auth
7= Backward  8= Forward  9= Alias  10= Stats   11= XED/FILEL 12= Cursor

====>
X E D I T 1 File

```

Figure 32. Revoked Indicator on the FILELIST SHARE Screen

The word *REVOKED* in the column labelled *Type* tells you that the owner, MARKD, has revoked your authority to the base file for your CHEF TIPS alias.

If, instead of entering the FILELIST command, you had entered the QUERY AUTHORITY command on your alias, CHEF TIPS, your screen would look like this:

```

Directory = VMSYSU:yourid.ATTEND
Filename Filetype Fm Type  Grantee R W
CHEF     TIPS     M1 REVOKED yourid - -

```

The word *REVOKED* in the *Type* column indicates that your authority to the base file for the CHEF TIPS alias has been revoked. The file name, file type, and directory name for the base file are not shown, even if you still maintain authority to the directory where it resides.

In either case, when you see that you no longer have authority to the COOKING HINTS file, your alias to it is no longer valid. You can use the ERASE command to remove the reference to the revoked file.

Note: An external security manager is a program that either augments or completely replaces the authorization checking done by file pool server processing. If there is an external security manager active on your system, the GRANT and REVOKE AUTHORITY commands you need to enter may be different from the commands discussed in the previous sections. You will need to refer to the external security manager documentation for the actual commands you would use to grant or revoke authority. Check with your system administrator to see if you have an external security manager active on your system.

Determining Who Has Authority on a File or Directory

To determine what authority has been granted on a file or directory, you can enter the QUERY AUTHORITY command. If you are the owner of a file or directory, QUERY AUTHORITY will show you a list of user IDs (including your own) for users who have authority to your file or directory and will show what type of authority you have granted each user.

You can also enter the QUERY AUTHORITY command if you are not the owner of a file or directory. Here, the command output would show only the authority you have been granted to the specific file or directory.

A sample format of the command follows:

```
query authority [fn ft] dirid
```

To determine who has authority on a specific file or files, specify the file name and file type, and the directory to be queried. To determine the authorities granted on a directory, specify only the directory identifier.

To determine if any other user has authority to the CAKE SCRIPT file in your .PARTY.TREATS directory (currently accessed as E), enter:

```
query authority cake script e
```

Your output will look like this:

```
Directory = VMSYSU:yourid.PARTY.TREATS
Filename Filetype Fm Type   Grantee R W
CAKE      SCRIPT    E1 BASE   yourid X X
```

The QUERY AUTHORITY display shows that the file has the file name and file type CAKE SCRIPT, and a file mode of E. Your user ID is listed under the Grantee column and both R for read authority and W for write authority are marked with an "X." Because you are the owner of the CAKE SCRIPT file, you automatically have read and write authority to it. You did not grant anyone else authority on the file, so no other user IDs are listed.

Although you originally used the COPYFILE command to copy the CAKE SCRIPT file from the MAINT. top directory (to which you had read authority), the MAINT user ID does not maintain any authority on the new file. New files created using COPYFILE have none of the authorities or aliases associated with the original file. When you copied the CAKE SCRIPT file to your own directory, you became the owner of the new file.

Using AUTHLIST

Another way to find out information on authorizations for a file or directory is by using PF keys on the DIRLIST and FILELIST screen to enter the AUTHLIST command. As discussed in "Navigating Through Your Directories" on page 88, you can display the FILELIST STATS screen and then press PF10 to display the SHARE screen, or you can specify the SHARE option of FILELIST to display the SHARE screen directly.

For example, assume that you had previously granted authority on the COOKIES SCRIPT file in your .PARTY.TREATS directory (accessed as E) to MAINT and to

RALPHW. To determine whether you gave each user read or write authority to the file, you could use the AUTHLIST command.

First, you could enter:

filelist cookies script e (share

Your screen would look like this:

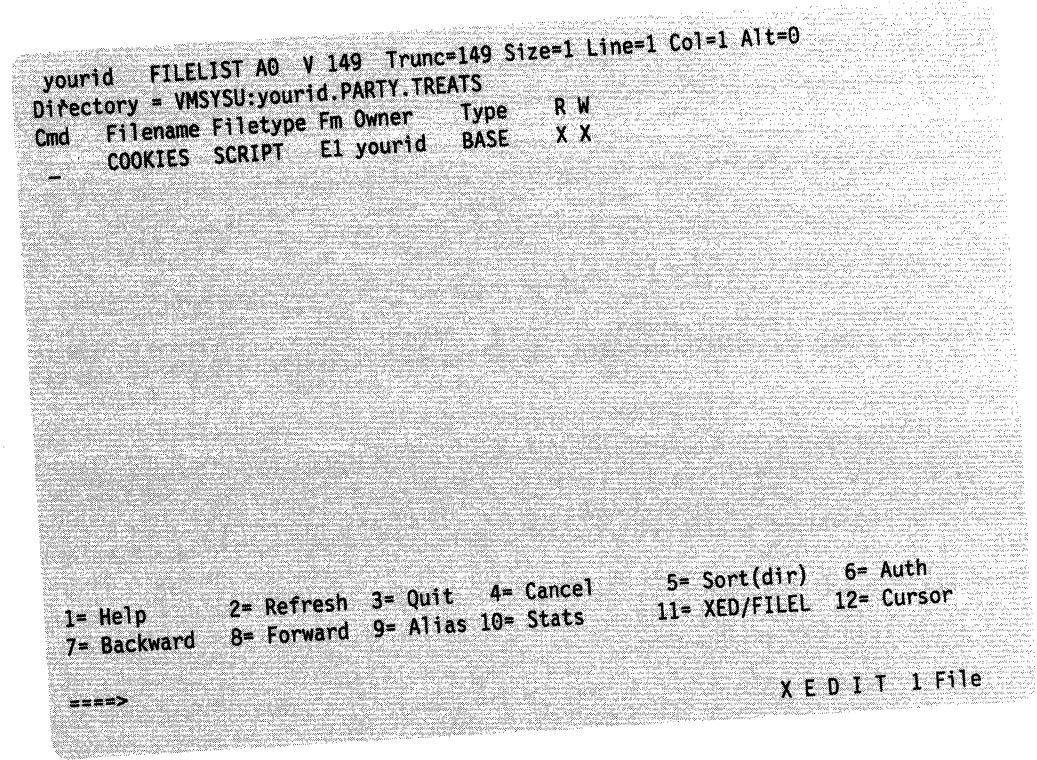


Figure 33. Using the PF Keys on the FILELIST SHARE Screen
Position your cursor on the line for the COOKIES SCRIPT file and press PF6.

Your screen would look like this:

```

yourid  AUTHLIST A0  V 165  Trunc=165 Size=3 Line=1 Col=1 Alt=0
File = COOKIES SCRIPT VMSYSU:yourid.PARTY.TREATS
Grantee R  W
yourid  X  X
MAINT   X  -
RALPHW  X  X

1= Help      2= Refresh  3= Return  4= S(Grantee)  5= Sort(W)  6=
7= Backward  8= Forward  9=         10=          11=         12=

====> _
X E D I T  1 File

```

Figure 34. Entering the AUTHLIST Command

The information you receive from the AUTHLIST command is similar to the kind of information you received when you entered the QUERY AUTHORITY command.

The first line of the display will show:

```
File = COOKIES SCRIPT VMSYSU:yourid.PARTY.TREATS
```

This is the complete name for your COOKIES SCRIPT file. Under the column labeled *Grantee*, your user ID is listed, along with the MAINT and RALPHW user IDs. There is an **X** in the columns labeled **R** and **W** to indicate that you have read and write authority to the base file. MAINT has read authority only, while RALPHW has both read and write authority.

Note: Remember that the file pool administrator will also have authority to your files and directories. However, his or her user ID will not appear in the output of the QUERY AUTHORITY or AUTHLIST commands.

The AUTHLIST display contains the following PF keys that you can use to sort the information displayed on your screen:

Table 9. AUTHLIST PF Keys

Key	Meaning	Usage
PF4	S(Grantee)	PF4 sorts the files displayed alphabetically by grantee.
PF5	Sort(W)	When you press PF5, grantees with write authority to the file are listed alphabetically, then other grantees are listed alphabetically.

Press PF3 to return to the FILELIST display.

Note: If the QUERY AUTHORITY or AUTHLIST command output shows **XP** or **-P** in the *Read* or *Write* column, the file shown is protected by an external security manager that is active on your system. Here, you will need to enter special external security manager commands to determine who has authority to the file. Contact your system administrator to obtain these commands.

Determining Ownership of a File or Directory

The FILELIST SHARE screen is also quite useful for determining the owner of a file. When you enter the FILELIST command with the SHARE option, the FILELIST display shows a column labeled *Owner*. You can use this information to determine the owner of a specific file. You can also use the SHARE option of the FILELIST command (or LISTFILE command) to determine the owners of any files or subdirectories contained within any directory.

The FILELIST SHARE display for the COOKIES SCRIPT file is currently displayed on your screen. As you can see, your user ID is listed in the column labelled *Owner*. Press PF3 to quit the FILELIST screen.

If you had a directory accessed with a file mode of T that contained several base files and aliases, you could determine the owner of any file by entering the command:

```
filelist * * t (share
```

Your output would look like this:

```

yourid FILELIST A0 V 149 Trunc=149 Size=7 Line=1 Col=1 Alt=0
Directory = VMSYSU:yourid.GOODIES.EAT
Cmd  Filename Filetype Fm Owner   Type   R W
-    FRENCH   BREAD   T1 yourid  BASE   X X
    MEAT    CAKES   T1 STANLEY ALIAS   X X
    FRIED   RICE    T1 ROMANOWS ALIAS   X -
    CHILI   BEANS   T1 yourid  BASE   X X
    PIZZA   SAUCE   T1 ACOUGHLN ALIAS   X -
    RECIPES        T1 yourid  DIR    X X
    ICE     CREAM   T1 HARRIS  ERASED - -

1= Help      2= Refresh  3= Quit   4= Cancel   5= Sort(dir)  6= Auth
7= Backward  8= Forward  9= Alias  10= Stats   11= XED/FILEL 12= Cursor

====>
X E D I T 1 File

```

Figure 35. Determining the Owner of a File

The *Owner* column shows you the owner of each base file, alias, or directory.

Using Aliases to Share Files

Sometimes you may want to share a file with other users, but you may not want them to know the file by its original name. For example, Dave has a program entitled AUTO4 ASSEMBLE, but he would like others to know it as SAMPLE ASSEMBLE.

There are a few ways Dave could do this. One way is for him to create a subdirectory and grant read authority on it to others. This subdirectory could contain only the file Dave wants to share with other users. To do this, Dave might enter commands such as these:

```

create directory .programs
grant authority .programs to mydept (read

```

MYDEPT, in this example, could be a group identifier. Dave could put the nickname MYDEPT in his *userid* NAMES file to include the nicknames or user IDs of all those with whom he wishes to share the file. By using MYDEPT in the GRANT AUTHORITY command, Dave can grant read authority to the entire group using a single command.

Next, he could grant all the people within the MYDEPT group read authority on AUTO4 ASSEMBLE. (Assume that the file resides in a directory called .ASSEMBLER.SOURCE.)

```

grant authority auto4 assemble .assembler.source to mydept (read

```

Finally, he could create an alias named SAMPLE ASSEMBLE in the shared .PROGRAMS subdirectory he created:

```
create alias auto4 assemble .assembler.source sample = .programs
```

Once Dave had completed these steps, all the users to whom he granted authority would know the file as SAMPLE ASSEMBLE. They would have access only to Dave's .PROGRAMS subdirectory, where he would have placed the SAMPLE ASSEMBLE alias. They would not know that the base file was AUTO4 ASSEMBLE nor would they have any authority on anything in Dave's ASSEMBLER.SOURCE directory.

Dave could have reversed the order of the GRANT AUTHORITY and CREATE ALIAS commands and still have achieved the same result. He could have first created the alias, and then granted authority on it or the base file.

Granting authority on an alias is the same as granting authority on the base file. Therefore, if Dave erases the alias, none of the authorizations on the base file would disappear, even if he had used the alias name on the GRANT AUTHORITY command.

If another user listed the contents of the .PROGRAMS directory between the CREATE ALIAS and GRANT AUTHORITY commands, he or she would be able to see the alias name, but would not be able to access the base file.

Another way to share a file is to have the users grant you write authority on one of their directories. This would let you create an alias for your file in their directories. You can also use the COPYFILE or XEDIT commands to create a base file in the other user's directory.

In this example, assume Dave already granted the users read authority on the file. Now, he can enter a CREATE ALIAS command to create an alias in each of their directories. For example, he might enter commands similar to the following:

```
create alias auto4 assemble .assembler.source sample = jones.programs
create alias auto4 assemble .assembler.source sample = bill.shared
create alias auto4 assemble .assembler.source sample = mary.toolstuff.programs
create alias auto4 assemble .assembler.source sample = tim.languages.assemble
```

These commands would create the alias SAMPLE ASSEMBLE in each of their directories (in Jones' .PROGRAMS directory, in Bill's .SHARED directory, in Mary's .TOOLSTUFF.PROGRAMS directory, and in Tim's .LANGUAGES.ASSEMBLE directory).

This technique for sharing files is useful when you are sharing with a few users. If you were sharing files with many users it would be impractical to have to enter a separate CREATE ALIAS command for each user. The following section, will explain ways to share a file with many users.

Creating a Bulletin Board or Shared Disk

If you want to create a "bulletin board" of files that are available to a group, you can do so by granting write authority on a directory. For example, suppose a user named Eric was coordinating shared files in your department. Your department writes programs that are used by everyone enrolled in the file pool.

To let everyone easily use the files, Eric might create a directory and grant write authority on that directory to every member of your department. Eric could also grant read authority on that directory to everyone else enrolled in the file pool so that everyone can view the programs your department writes.

Your department members could create aliases in Eric's directory whenever they created a new file or program they wished to share with the file pool users. At that time, they would also grant read authority on the base file to everyone in the file pool.

File pool users could periodically review the directory for new additions. To do this, they could add an ACCESS command for Eric's directory to their PROFILE EXECs. They could then occasionally enter a FILELIST for Eric's directory, and sort the listings by date to see what was new.

For example, suppose Eric wants to create a programming tools disk called VMTOOLS to contain short-cut programs and productivity aids for use by everyone in the file pool. First, he would create a directory:

```
create directory design:eric.vmtools
```

Next, Eric would enter the NAMES command:

```
names toolprog
```

For the nickname TOOLPROG, he would list those people who were authorized to write and distribute tools. To grant this group of people write authority on the VMTOOLS directory, Eric would enter the following command:

```
grant authority .vmtools to toolprog (write
```

Eric would then grant read authority on the directory to everyone in the file pool:

```
grant authority .vmtools to public (read
```

The GRANT AUTHORITY command with the PUBLIC parameter lets you grant authority to each user assigned to your file pool.

That is all Eric needs to do. His involvement in maintaining the VMTOOLS is over. Whenever a programmer develops a tool, the programmer would just grant read authority on the EXEC or MODULE to PUBLIC, and would create an alias in Eric's VMTOOLS directory.

Suppose Denny wants to make the new tool PROJTRAK MODULE (which is in his .GOODSTUFF directory) available to everyone. He would enter the command:

```
grant authority projtrak module .goodstuff to public (read  
create alias projtrak module .goodstuff = = eric.vmtools
```

This would give all the users in the file pool read authority to the new tool, PROJTRAK MODULE, in Denny's .GOODSTUFF directory and would create an alias for PROJTRAK MODULE in Eric's .VMTOOLS directory.

To access the currently updated list of tools, users would add this ACCESS command to their PROFILE EXECs:

```
access design:eric.vmtools g
```

This command would let them access Eric's .VMTOOLS directory each time they logon. Whenever users want to review what's available, they would enter:

```
filelist * * g
```

This would show them a FILELIST display of the .VMTOOLS directory so that they would know if a new program was added.

To run the PROJTRAK program, a user would enter:

projtrak

There is another way to do this that would let Eric have more control over what the user community sees. In this variation, Eric could have his programmers create files instead of aliases in his directory. That would make Eric the owner of the base files. Once the programmers create a file in Eric's directory, they would be unable to grant any authority on that file. Eric would be the only one who could grant read authority on those files to the user community.

You may find this technique useful for organizing shared disk files created before VM/SP Release 6, such as *tools* disks, *forum* disks, and *bulletin board* disks.

For huge forum disks that contain several files on any given topic, you might consider creating a subdirectory on each topic. When a user enters a FILELIST on the higher level directory, all the user will see is a list of subdirectories, each containing files devoted to a particular topic. This technique protects users from being overwhelmed by hundreds of unrelated files.

The Transparent Services Access Facility (TSAF) is a component of VM/SP that handles communication between systems by letting APPC/VM paths span multiple VM systems. TSAF lets a source program connect to a target program by specifying a name that the target has made known, instead of specifying a user ID and node ID. A collection is a group of up to eight VM/SP systems that can share resources.

If you are participating in a TSAF collection, you could ask your system administrator to make this kind of shared data available to everyone in the collection by entering the ENROLL PUBLIC command. This would let anyone in the collection enter an ACCESS command for the directory and have access to the tools. For more information about sharing data in a TSAF collection, see the *VM/SP CMS Shared File System Administration* book.

Locking Files and Directories

To share files and directories between users, you need to understand how CMS handles the simultaneous use of shared files. Also, you will need to know how to ensure that you and another user do not simultaneously edit a file and accidentally overwrite changes.

Whenever you are actively reading or writing a file or directory, CMS acquires a lock for the file or directory. This particular lock is called an *implicit* lock. An implicit lock allows multiple readers and only one writer to use a file or directory. CMS acquires and frees implicit locks automatically. They are usually short-term locks. If a user tries to read or write to a file or directory, SFS first checks to see whether it is locked before allowing access.

While SFS automatically acquires and releases implicit locks, there are situations in which you may want to purposely lock a file or directory with an *explicit* lock. For example, you may want to perform a series of tasks on a file and do not want anyone manipulating the file for a certain period of time. CMS provides you with a means of explicitly locking files for the duration of your CMS session, or until you further specify you want them unlocked.

Explicit locks are useful when you want to control the activity on your files or directories without revoking authority. For example, suppose Dr. Roman is performing various tasks (such as editing, renaming, and so on) on a file called MEDICAL HISTORY that many other doctors have access to, and he does not

want them to see the file until he has completed his work. Dr. Roman can create a lock which will prevent other users from reading or modifying the file until he deletes the lock.

He would specify this type of explicit lock using the `CREATE LOCK` command with the appropriate options. A sample format of the `CREATE LOCK` command follows:

```
create lock [fn ft] dirid share|exclusive|update session|lasting
```

To lock a file, specify the file name, file type, and directory identifier; to lock a directory, specify the directory identifier only.

You can create an `EXCLUSIVE`, `SHARE`, or `UPDATE` lock on a file or directory. Also, you must specify the duration of the lock: `SESSION` or `LASTING`. The following table describes the meaning of each lock as it pertains to a file or directory:

	File	Directory
Share	Other users can read the file while you are reading it. No one, including the person who issued the lock, can update the file until it is unlocked. Also, they cannot rename, relocate, or erase the file. A <code>SHARE</code> lock on a base file also prevents other users from issuing an <code>EXCLUSIVE</code> lock on the file's parent directory.	Other users and the person who locked the directory can read from files in the directory (if they are authorized for the files). No users (including the issuer of the lock) may update any files or subdirectories in the directory. Nor can they create, erase, rename, or relocate any base files, aliases, or subdirectories until the directory is unlocked. Also, a <code>SHARE</code> lock prevents anyone from issuing an <code>EXCLUSIVE</code> or <code>UPDATE</code> lock on any base file, alias, or subdirectory within the directory.
Exclusive	Other users cannot read or change the file. An <code>EXCLUSIVE</code> lock on a base file also prevents anyone from issuing any type of lock (<code>SHARE</code> , <code>EXCLUSIVE</code> , or <code>UPDATE</code>) on the file's parent directory.	Other users cannot read from or write to any of the base files or aliases in the directory. Also, they may not create, delete, rename, or relocate any base files or aliases until the directory is unlocked. They can manipulate the contents of subdirectories if they are authorized to do so.
Update	Other users may read the file while you are reading or updating it. They cannot rename, relocate, or erase the file. An <code>UPDATE</code> lock on a base file also prevents anyone from issuing any type of lock (<code>SHARE</code> , <code>EXCLUSIVE</code> , or <code>UPDATE</code>) on the file's parent directory.	The person who locked the directory may read from or write to any files for which he or she is authorized. Other users may only read files in the directory; they cannot write to the files (even if they have write authority to them). Nor can they create, delete, rename, or relocate any base file, alias, or subdirectory in the directory. Also, they may not lock the directory in any mode; base files, aliases, or subdirectories can be locked in <code>SHARE</code> mode only.

The lock that Dr. Roman would create for our previous example would be:

```
create lock medical history a exclusive lasting
```

Dr. Roman has just created an `EXCLUSIVE` lock to prevent others from reading or writing to the `MEDICAL HISTORY` file contained in directory A. Because he

specified a **LASTING** lock, the lock will remain until he issues a command to delete it. A **LASTING** lock lasts across CMS sessions, and can be removed only with a **DELETE LOCK** command. The **DELETE LOCK** command is discussed in the section "Deleting Explicit Locks."

The alternative to the **LASTING** lock is the **SESSION** lock. A **SESSION** lock is removed when the **DELETE LOCK** command is entered, or when the CMS session is terminated, whichever comes first.

If you have read authority to a file or directory, you can only create a **SHARE** lock on that file or directory. If you have write authority, you can issue any type of lock (**SHARE**, **EXCLUSIVE**, or **UPDATE**) on the file or directory.

Note: If you create a lock on an alias, it is the same as locking the base file—the owner of the base file and other users with aliases to the file will be affected by the lock.

To allow others to read a file while you are updating it, you can create an **UPDATE** lock on the file or on the directory where the file resides.

For example, if Stan shared with other users a file called **BRUSHING TIPS** in his **.DENTAL.HYGIENE** directory, he could create an **UPDATE** lock on the file for the duration of his CMS session. To do so he would enter the following command:

```
create lock brushing tips .dental.hygiene update session
```

After he enters this command, Anne, who shares the file with Stan, would be able to read the file but not write to it. To read the file, she could use the **TYPE** command. She could also use the **NOLOCK** option of the **XEDIT** command to view the contents of the file. She could not, however, enter the **XEDIT** commands **SAVE** or **FILE**, to store an updated copy of the file.

When you use **XEDIT** with the default **LOCK** option, you do not need to specifically create and delete a lock; **XEDIT** will take care of this for you. When you use **XEDIT** with the **LOCK** option, an **UPDATE SESSION** lock is automatically placed on your file. It is only when you specify the **NOLOCK** option of **XEDIT** that you need to be concerned with creating and deleting locks on files and directories. The **NOLOCK** option of **XEDIT** should only be used when you are not going to make any changes to the file, or if you are going to save your changes under a different name.

Once Stan completes his work with the file, he can use the **DELETE LOCK** command to remove the lock. Because he specified a **SESSION** lock, if he does not delete the lock, it will automatically be removed when he ends his CMS session.

Another way to lock a file or directory is to create a **SHARE** lock. With a **SHARE** lock, users with write authority could **XEDIT** the file with the **NOLOCK** option, but they would be unable to enter **SAVE** or **FILE**. Not even the person who creates the lock can update the file.

Deleting Explicit Locks

Use the DELETE LOCK command to delete explicit locks placed on files or directories with the CREATE LOCK command. A sample format for the DELETE LOCK command follows:

```
delete lock [fn ft] dirid
```

If Stan wanted to delete the UPDATE SESSION lock he placed on the BRUSHING TIPS file, he would enter the following command:

```
delete lock brushing tips a
```

To delete a lock on a directory, you specify the name of the directory to be unlocked, instead of the file name, file type and directory identifier.

Using the SET FILEWAIT Command

If you want your program to continue waiting for a file or directory that is implicitly locked, you can use the SET FILEWAIT command. A sample format of the command follows:

```
set filewait {ON | OFF}
```

If you use SET FILEWAIT ON, this means you do not want a request to fail because you cannot obtain immediate control of a file. The request will wait until the required files become available or until you re-IPL or logoff. For example, at the end of the day you could choose to start a program and disconnect your virtual machine.

If you do not wish to wait for an implicitly locked file or directory, use SET FILEWAIT OFF. This means that requests will fail immediately if a file or directory is not available.

If the file or directory is explicitly locked, you will not wait, regardless of the SET FILEWAIT setting.

Determining If a File or Directory is Locked

There will be times while using CMS commands when you may receive a message stating that the file or directory you want to work with is locked. You will need to find out who created the lock, so that you can get that person to delete or change the lock and allow you to read or write to the file. The next sections show you how to find explicit or implicit locks on files and directories.

Finding an Explicit Lock

The QUERY LOCK command displays the type of explicit lock and the user who created the lock on a file or directory. A sample format of the QUERY LOCK command follows:

```
query lock [fn ft] dirid
```

To query the lock for a file, you would specify the file name and file type, and the directory identifier. To query the lock for a directory, specify only the directory identifier.

For example, suppose you are responsible for keeping track of system problems that affect your department. For this purpose, you have created a file called PROBLEM LOG in your .PROBLEMS directory.

To let the other members of your department log a description of any system problems they experience, you have given each member write access to the PROBLEM LOG file. Before editing PROBLEM LOG to update the problems listed, you could enter the following command to determine if any other user has locked the file for editing:

```
query lock problem log .problems
```

Your output might look like this:

```
Directory = VMSYSU.yourid.PROBLEMS
Filename Filetype Fm Type Userid Lock Duration
PROBLEM LOG N1 BASE ELLENG UPDATE SESSION
```

As you can see, Ellen has entered an update lock for the file. Therefore, you should wait until Ellen completes her editing and unlocks the file before you try to edit it.

Finding an Implicit Lock

Because implicit locks are short term, the QUERY LOCK command does not display information about them. Even if it did so, the information might be invalid by the time the output was displayed. To check for implicit locks:

1. Issue the command SET FILEWAIT ON.

This command tells CMS to wait for the file or directory to become free if a lock conflict occurs.

2. Reenter the command that was causing the lock problem.

If the command succeeds, the implicit lock was just freed. Enter SET FILEWAIT OFF and continue your work.

If the command fails and produces an error message, another user may have just acquired an explicit lock. Otherwise, the file or directory may have been erased, or your authority to it may have been revoked.

3. If the command execution time is lengthy, ask another user to enter the QUERY FILEPOOL CONFLICT command for you.

For example, if the user CROCKETD suspected there was an implicit lock on the file he wants to access, he would ask another user to enter the following command:

```
query filepool conflict crocketd poolq:
```

The result would look like this:

```
Requestor Holder Wait Lock Lock Type
BRISEED SMITH Lock File Share
MIKEB SMITH Lock File Share
CROCKETD SMITH Lock File Share
```

The first two columns show the most important information: who is requesting the lock and who is holding the lock. CROCKETD, along with two other users, is waiting for SMITH. The other users are in the queue ahead of CROCKETD. When SMITH frees the lock, BRISEED will be the next user in line for the file.

The third column, *Wait*, indicates the wait state of the Holder. In this example, each of the users is waiting for a lock to be freed. The *Lock* column indicates the

type of resource for which the Requestor has requested a lock. The *Lock Type* column displays the type of lock that the Requestor wants placed on the resource.

Note: The QUERY FILEPOOL CONFLICT command does not show conflicts caused by explicit locks because CMS never waits for an explicit lock, even if FILEWAIT is on.

Once you find out who is holding the lock, you can call or send that user a message to see when he or she will be done. If you choose not to continue waiting, enter:

```
#cp ip1 cms
```

When you IPL CMS, SFS realizes that you have ended your CMS session and stops waiting to process your command. During CMS initialization, FILEWAIT is automatically reset to OFF.

Determining How Much of Your File Space You Have Used

As we discussed earlier, your SFS files are stored in a file space. A file space contains a certain amount of space that your system administrator allocated to you when your user ID was added to a file pool.

At any time, you can determine what proportion of your file space you have used by entering the QUERY LIMITS command. For more information on QUERY LIMITS, see Chapter 3, "CMS File System" or see the description of the QUERY LIMITS command in the *VM/SP CMS Command Reference*.

Application Considerations

The following considerations should be taken into account when using applications to operate in an SFS environment. For information on writing applications, see the *VM/SP Application Development Guide for CMS*.

- Many system facilities and applications, such as the DASD Dump Restore Service Program (DDR), use CMS files. These files, like any other CMS files, can reside in an SFS directory or minidisk. However, before sharing these files with other users (or before using the files of others), you should be aware of the implications of doing so. For example, if you grant authority to another user, and that user locks the file, the facility may not be able to use the file. The same is true if another user grants you authority and later revokes the file. In these cases, the facility may receive a non-zero return code and terminate because it cannot use a file it needs.
- If you are running an application program and it fails, it could be a result of accessing directories in multiple file pools in read/write mode. Some applications try to write to more than one file mode. If those file modes are associated with directories in different file pools, the program may fail.

Using Multiple File Pools

It is possible for you to be enrolled in more than one file pool. If this is the case, when you log on, you will have a top directory in each file pool. Of these multiple file pools, one of them will be the default. Use the `QUERY FILEPOOL PRIMARY` command to find out your default file pool. Generally, the default file pool is accessed with a file mode of A. You can access your file space in the other file pool by explicitly naming the file pool ID on any command that accepts a directory identifier.

For example, for user Alyson to create a directory named `NEWDIR` in the file pool `DEVELOP`, which is not her default file pool, she would enter:

```
create directory develop:alyson.newdir
```

If Alyson wanted to make the directory part of her CMS search order, she could do so by accessing the directory:

```
access develop:alyson.newdir w
```

Such `ACCESS` commands could be in her `PROFILE EXEC` so that the directory in the `DEVELOP` file pool is automatically accessed each time Alyson logs on.

If file mode A is a minidisk rather than a directory, you should decide which file pool will be your default file pool. You would then use the `SET FILEPOOL` command in your `PROFILE EXEC` to make that file pool the default. Directories in your secondary file pools could also be accessed in your `PROFILE EXEC`.

Suppose that Sue is enrolled in three file pools: `DEVEL1`, `PUBS`, and `CTEST`. Her file mode A is a minidisk. She wants to make `DEVEL1` her default file pool and have her top directory within `DEVEL1` accessed as file mode B. The other two top directories she wants accessed as file modes O and P. In her `PROFILE EXEC` she would enter:

```
set filepool devel1:  
access . b  
access pubs:. o  
access ctest:. p
```

Because your top directory name is always the same as your user ID, Sue can abbreviate the commands by omitting her user ID. CMS will assume she wants to access the top directories.

After these commands are executed, if Sue enters a command and omits the file pool ID, CMS will use `DEVEL1` as the default.

To change the default file pool during a CMS session, enter the `SET FILEPOOL` command again. For example, to temporarily change her default to `CTEST`, Sue would enter the following command:

```
set filepool ctest:
```

To reset the default, Sue would enter `SET FILEPOOL` again:

```
set filepool devel1:
```

Note: If you wish to reset the default file pool to the one that was specified at the last logon or re-IPL, enter the following command:

```
set filepool primary
```


PRIMARY is a keyword. It means the default file pool in effect at the time of IPL.

Note: Although you can be enrolled in more than one file pool, you cannot write to more than one file pool at the same time.

Sharing Files with Users on Other Systems

SFS lets you share your files with users on other systems. For example, you can share files with users in another building or in another town or state.

To share files owned by another user in a different file pool or on a different system in the same TSAF collection, call the system administrator at the other location and ask to be enrolled in the other user's file pool. Then, the other user could grant you authority on the files you need to share. To access the files you own on your file pool, that user could, in turn, request enrollment in your file pool.

To enable you to share files with users on a different system and not within the same TSAF collection, your system administrator will need to use APPC/VM VTAM Support (AVS). For more information, see the *VM Connectivity Planning, Administration, and Operation* book.

Chapter 5. Storing Your Files on Minidisks

You will be using CMS on a virtual machine. A virtual machine is similar to a real machine in that it has access to disk storage. This disk storage can be either a directory (within a Shared File System file space) or a minidisk. This chapter contains information specific to minidisks. If your files are stored in a Shared File System (SFS) file space, see Chapter 4, "Using the Shared File System" for information on how to manage your files.

Minidisks and How They Are Defined

A minidisk is a location on a real direct access storage device (DASD) which has been allocated for storage of a user's files. Minidisks can also be formatted for use as an OS or DOS disk.

For CMS applications, you never have to be concerned with the location of your data on minidisks; when you use minidisks, they are, for practical purposes, functionally the same as real disks.

You can have two types of minidisks, permanent and temporary.

Permanent minidisks last across terminal sessions (logons); they are defined in the VM/SP directory entry for your virtual machine.

Temporary minidisks are automatically destroyed at logoff. Temporary minidisks are those you define for your own virtual machine using the CP DEFINE command, or those attached to your virtual machine by the system operator.

Both permanent and temporary minidisks can be attached to your machine during a terminal session.

Defining Temporary Minidisks

If you use minidisks to store your files, from time to time you may find it necessary to define a temporary minidisk. Using the CP DEFINE command, you can attach a temporary minidisk to your virtual machine for the duration of a terminal session. The following command allocates a 10-cylinder temporary minidisk from a 3330 device and assigns it a virtual address of 291:

```
define t3330 as 291 cyl 10
```

When you define a minidisk, you can choose any valid address that is not already assigned to a device in your virtual machine. Valid addresses for minidisks are:

- 0001 through FFFF for a 370/XA virtual machine
- 001 through 5FF for a System/370 virtual machine in basic control mode
- 001 through FFF for a System/370 virtual machine in extended control mode.

Formatting Minidisks

Before you can use any new minidisk, you must format it. This applies to new minidisks that have been assigned to you and to temporary minidisks that you have allocated with the CP DEFINE command. When you enter the FORMAT command, you must use the virtual address you have defined for the minidisk and assign a CMS mode letter, for example:

```
format 291 c
```

CMS then prompts you with the following message:

```
DMSFOR603R FORMAT will erase all files on disk C(291).  
Do you wish to continue? Enter 1 (YES) or 0 (NO).
```

You respond:

```
1
```

CMS then asks you to assign a label for the minidisk, which can be anything you choose. Labels can have a maximum of 6 characters. When the message:

```
DMSFOR605R Enter disk label:
```

is displayed, you respond by supplying a minidisk label. For example, if this is a temporary minidisk, you might enter:

```
scrтч
```

CMS then erases all the files on that minidisk, if any existed, formats it for your use, and displays the following messages:

```
DMSFOR733I Formatting disk C  
DMSFOR732I 10 cylinders formatted on C(291)  
Ready; T=0.15/1.60 11:26:03
```

The FORMAT command should only be used to format CMS minidisks, that is, minidisks you are going to use to contain CMS files. In addition, this command gives you a choice of physical disk block size as an option. See the *VM/SP CMS Command Reference* for more information.

Sharing Minidisks: Linking

If your files are stored on minidisks, you will need to link to other users minidisks in order to share files. This section contains the information you will need in order to create temporary or permanent links to other minidisks.

Note: If your files are stored in the Shared File System, you will be able to share files with other users without linking. For more information, see Chapter 4, "Using the Shared File System."

Since only one user can own a minidisk, and there are many occasions that require users to share data or programs, VM/SP lets you share minidisks, on either a permanent or temporary basis, by "linking".

Permanent links can be established for you in your VM/SP directory entry. These minidisks are then a part of your virtual machine configuration every time you log on. You can also have another user's minidisk temporarily added to your configuration by using the CP LINK command. For example, if you have a program that uses data that resides on a minidisk identified in user ID DATA's configuration as a 194, and you know that the password assigned to this minidisk is GO, you could enter the command:

```
link to data 194 as 198 r pass= go
```

DATA's 194 minidisk is then added to your virtual machine configuration at virtual address 198.

Notes:

1. The password cannot be entered on the command line if the password suppression facility was specified when your system was installed.
2. If RACF/VM is installed on your system and is in effect, the procedure for linking may be changed somewhat. For additional information, see your system administrator.

The "R" in the command indicates the access mode; in this case, it tells CP that you only want to read files from this minidisk and you will not write to it. If you try to enter this command when someone already has write access to that minidisk, you will not be able to establish the link. If you want to link to DATA in any event, you can reenter the LINK command using the access mode RR:

```
link data 194 198 rr go
```

The keywords "TO," "AS," and "PASS=" are optional; you do not have to specify them.

Note: As with the previously entered command, the password cannot be entered on the command line if you specified the password suppression facility.

However, note that using the RR access lets one user read a minidisk while another is updating it at the same time. This can produce unpredictable results.

You can also use the CP LINK command to link to your own minidisks. For example, if you log on and discover that another user has access to one of your minidisks, you will be given read-only access, even if it is a read/write minidisk. You can request the other user to detach your minidisk from his virtual machine, and after he has done so, you can establish the link:

```
link * 191 191
```

When you link to your own minidisks, you can specify the user ID as *, and you do not need to specify the access mode or a password.

See "Minidisk File Directories" on page 130 for information on determining if other users have read/write access to your minidisks. You can find more information about the CP LINK command and CP access modes in the *VM/SP CP General User Command Reference*.

Linking to Your Minidisks

Because minidisks are portions of a physical volume that is defined using CP, they must be linked using the CP LINK command. Once minidisks are linked, they can be accessed. See Chapter 3, “The CMS File System,” for information on accessing and releasing your minidisks.

Detaching Minidisks

When you no longer need minidisks in your virtual machine configuration, use the CP command DETACH to disconnect them from your virtual machine:

```
detach 194  
detach 291
```

Minidisk File Directories

Each minidisk has a master file directory that contains entries for each of the CMS files on that minidisk. (The directories discussed in this section are not to be confused with those pertaining to the Shared File System.) When you access a minidisk, information from the master file directory is brought into virtual storage and written into a user file directory. The user file directory has an entry for each file that you can access. If you have accessed a minidisk specifying only particular files, then the user file directory contains entries only for those files.

If you have read/write access to a minidisk, then each time you save the file, the user file directory and master file directory are updated to reflect the current status of the minidisk. If you have read/write access to a minidisk and the FSCLOSE macro is issued, the user file directory is updated. When there are no open files on the minidisk, the master file directory is updated to reflect the current status of the files. If you have read-only access to a minidisk, then you cannot update the master file directory or user file directory. If you access a read-only minidisk while another user is writing files onto it, you may need to periodically reenter the ACCESS command for the minidisk to obtain a fresh copy of the master file directory.

Warning: You should never attempt to write on a minidisk at the same time as another user. CMS does not protect a user from loss of data on a minidisk when multiple users have write access to it.

You can use the CP command QUERY LINKS to determine if other users have links to any minidisks you want to access. See the *VM/SP CP General User Command Reference* for more information. If you are using the Shared File System, files can be locked and unlocked to prevent simultaneous updates. For more information, see Chapter 4, “Using the Shared File System.”

The user file directory remains in virtual storage until you enter the RELEASE command specifying the mode letter or virtual address of the minidisk.

If you detach a minidisk (with the CP DETACH command) without releasing it, CMS does not know that the minidisk is no longer part of your virtual machine. When you attempt to read or write a file on the minidisk, CMS assumes that it is still active (because the user file directory is still in storage) and encounters an error when it tries to read or write the file.

A similar situation occurs if you detach a minidisk and then add a new minidisk to your virtual machine using the same virtual address as the minidisk you detached. For example, if you enter the following sequence of commands:

```
link user1 191 195 rr rpass
access 195 d
detach 195
link user2 193 195 rr rpass
listfile * * d
```

the LISTFILE command produces a list of the files on USER1's 191 minidisk; if you attempt to read one of these files, you receive an error message. You must enter the ACCESS command to obtain a copy of the master file directory for USER2's 193 minidisk.

The entries in the master file directory are sorted alphanumerically by file name and file type, to facilitate the CMS search for particular files. When you are updating minidisks, the entries in the user file directory and master file directory tend to become unsorted as files are created, updated, and erased. When you use the RELEASE command to release a read/write minidisk, the entries are sorted and the master file directory is rewritten.

Minidisks

Chapter 6. What You Can Do with CMS Commands

This chapter provides an overview of the kinds of tasks you can perform in CMS and the appropriate commands necessary to perform each of them.

The chapter is organized by task, covering such topics as beginning and ending your terminal session, requesting information, and moving files. Under each task, there is a short discussion of some of the types of things you can do. This is followed by a table containing a list of specific tasks, commands you would use to perform those tasks, and chapters that contain information on the specific commands mentioned.

The commands are not presented in their entirety, nor is a complete selection of commands represented. The purpose of this chapter is to provide you with an understanding of the kinds of commands available to you, so that when you need to perform a particular task using CMS, you may have an idea of whether it can be done, and know what command to reference for details.

For a complete list of the CP commands available, see Table 23, and for CMS commands, see Table 20 and Table 21.

Beginning and Ending Your Terminal Session

Your terminal session starts when you logon (with CP LOGON) and ends when you logoff (CP LOGOFF). When you know that you are only going to be away from your terminal for a short while, you can disconnect (CP DISCONN). When you reconnect (with CP LOGON) the status of your virtual machine is the same as you left it.

Task	Command(s)	Description
Begin your terminal session	CP LOGON	Chapter 1
End your terminal session	CP LOGOFF CP DISCONN	Chapter 1 Chapter 1

The command formats and usage notes for the commands DISCONN, LOGOFF, and LOGON, are found in the *VM/SP CP General User Command Reference*.

Tailoring Your System

At the start of every terminal session, you can automatically customize your system with the commands invoked by your PROFILE EXEC. Your PROFILE EXEC can include commands to set your program function (PF) keys, access directories and minidisks, and access your synonym table. For example, you may wish to set up a PF key for the RETRIEVE function to provide you with a quick method of correcting errors without rekeying an entire line of input. Once you have assigned the RETRIEVE function to a PF key, when you press the key, the system retrieves the line you previously entered.

You may also tailor your system by using full-screen CMS. If you enter SET FULLSCREEN ON at the beginning of your terminal session, or if the command is in your PROFILE EXEC, CMS is in a window and can take advantage of full-screen capabilities. In full-screen CMS, messages and other information are accessed through windows on your physical screen. When you are using full-screen CMS, you would not need to set a PF key for the RETRIEVE function because the default setting for CMSPF 6 is RETRIEVE.

Your system administrator also tailors your terminal session by making changes to your system profile. These changes would immediately be in effect each time you IPL CMS. For more information about the system profile function, see *VM/SP Application Development Guide for CMS*.

You can also write your own EXEC to execute a series of commands. You could then invoke the EXEC from your PROFILE EXEC.

Another way you could customize your system is by specifying the language you wish to use for entering commands and receiving messages.

When you are communicating with others on your computer, use the NAMES command to assign nicknames. The nicknames can be used with the SENDFILE and TELL commands because both commands reference your NAMES file. By setting up nicknames, you can tailor your system to let you send messages and notes much more quickly.

Action	Command(s)	Description
Set your program function (PF) keys	CP SET PFnn	Chapter 1
Assign a PF key to retrieve previously entered lines	CP SET PFnn RETRIEVE	Chapter 1
Keep information about others with whom you communicate	NAMES	Chapter 10
Specify defaults for the commands: DIRLIST, DISK, FILELIST, NOTE, PEEK, READCARD, RDRLIST, RECEIVE, SENDFILE, and TELL	DEFAULTS	Chapter 10
Assign synonyms for system and your own commands	SYNONYM	Chapter 3
Tailor your System at the start of every session using your PROFILE EXEC	System Product Editor and the System Product Interpreter	Chapter 15
Write your own command that executes several commands or programs	System Product Editor and the System Product Interpreter	Chapter 14

Action	Command(s)	Description
Make execs storage resident	EXECLOAD	Chapter 15
Access execs in a saved segment	SET INSTSEG	Chapter 16
Specify a language for entering CMS commands and receiving system messages	SET LANGUAGE	Chapter 1
Specify your default file pool	SET FILEPOOL	Chapter 4
Change your warning threshold for space usage	SET THRESHOLD	Chapter 4
Specify whether programs should wait for input from locked files or directories	SET FILEWAIT	Chapter 4
Invoke full-screen CMS	SET FULLSCREEN ON	Chapter 11
Display information in windows	POP WINDOW DROP WINDOW	Chapter 11
Customize full-screen CMS	DEFINE WINDOW DEFINE VSCREEN POSITION WINDOW SIZE WINDOW MAXIMIZE WINDOW RESTORE WINDOW	Chapter 17

The CP SET command format and usage notes are found in the *VM/SP CP General User Command Reference*. The command formats and usage notes for the following CMS commands are found in the *VM/SP CMS Command Reference*:

DEFAULTS
 DEFINE VSCREEN
 DEFINE WINDOW
 DROP WINDOW
 EXECLOAD
 MAXIMIZE WINDOW
 NAMES
 POP WINDOW
 POSITION WINDOW
 RESTORE WINDOW
 SET FILEPOOL
 SET FILEWAIT
 SET FULLSCREEN ON
 SET LANGUAGE
 SET THRESHOLD
 SIZE WINDOW
 SYNONYM
 XEDIT

Requesting Information

You can use CP and CMS commands to inquire about your terminal, virtual machine, SFS directories, minidisks, data files, members of a CMS library, and other users.

Inquiring about:	Command(s)	Description
Terminal characteristics	CP QUERY SCREEN CP QUERY SET	Chapter 1 Chapter 1
Languages that are currently active in your virtual machine	CP QUERY CPLANG QUERY LANGUAGE	Chapter 1
Languages that are available for your virtual machine	QUERY LANGLIST	Chapter 1
Files in directories or on minidisks	FILELIST LISTFILE	Chapter 3,4 Chapter 3,4
Files in your reader	RDRLIST CP QUERY RDR ALL	Chapter 10 Chapter 1,10
Directories	DIRLIST LISTDIR	Chapter 4
Shared files	QUERY ALIAS, (ALIALIST when in FILELIST) QUERY AUTHORITY, (AUTHLIST when in DIRLIST or FILELIST)	Chapter 4 Chapter 4
Your spool files	CP QUERY FILES	Chapter 10
Your minidisks	QUERY DISK	Chapter 3
Your directories	QUERY LIMITS	Chapter 4
Your accessed file modes	QUERY ACCESSED	Chapter 4
Your virtual machine	IDENTIFY	Chapter 16
Your print files	CP QUERY PRINTER	Chapter 9
Your reader, printer, and punch	CP QUERY UR	Chapter 9
Locked files or directories	QUERY LOCK QUERY FILEPOOL CONFLICT	Chapter 4 Chapter 4
Other users	CP QUERY user ID	Chapter 10
Storage resident execs	EXECMAP	Chapter 15

The command format and usage notes for the CP QUERY command are found in the *VM/SP CP General User Command Reference*. Command formats and usage notes for the following CMS commands are found in the *VM/SP CMS Command Reference*.

DIRLIST	EXECMAP
FILELIST	IDENTIFY
LISTDIR	LISTFILE
QUERY	RDRLIST

Communicating with Other Computer Users

You can use CP and CMS commands to send files, notes and messages to one or more users on your system or a system attached to yours through the Remote Spooling Communications Subsystem (RSCS) network. The NOTE, SENDFILE, and TELL commands reference your *userid* NAMES file. The names file, created with the NAMES command, contains a collection of information about other computer users with whom you communicate.

Action	Commands	Description
Creating Names file	NAMES	Chapter 10
Sending files	SENDFILE	Chapter 10
Sending messages	TELL	Chapter 10
Sending notes	NOTE and SENDFILE	Chapter 10

Following are CMS commands:

NAMES
NOTE
SENDFILE
TELL

Their command formats and usage notes are documented in the *VM/SP CMS Command Reference*.

Controlling Terminal Output

VM/SP lets you control your terminal output. You can refuse messages with the CP SET MSG command. If you only want to see the short form of the CMS ready message, set this with the CMS SET RDYMSG command.

The CP SCREEN command lets you select colors and extended highlighting on certain 3279 models.

If your program is directing output to your terminal, you can halt the terminal display with the HT Immediate command, and later resume terminal display with the RT Immediate command.

Action	Command(s)	Description
Alter any extended color or highlighting definitions	CP SCREEN	Chapter 1
Control whether or not you receive messages	CP SET	Chapter 1
Indicate the type of CMS ready message that you want	SET RDYMSG	Chapter 1
Suppress terminal output	HT Immediate command	Chapter 2
Resume terminal output that was previously suppressed through HT	RT Immediate command	Chapter 2

The CP command formats are:

SCREEN
SET

These formats are documented in the *VM/SP CP General User Command Reference*.
The CMS command command formats are:

HT (Immediate command)
RT (Immediate command)
SET

They are documented in the *VM/SP CMS Command Reference*.

Sharing Minidisks

VM/SP lets you share minidisks on either a permanent or temporary basis. You can add another user's minidisk to your configuration with the CP LINK command. When you no longer need a minidisk that you have linked to or have temporarily accessed, you can release it with the CMS RELEASE command. When you no longer need a minidisk in your virtual machine configuration, you can disconnect it with the CP DETACH command.

Action	Command(s)	Description
Establish a link to a minidisk	CP LINK	Chapter 1
	ACCESS	Chapter 1
Release a minidisk	RELEASE	Chapter 1
	CP DETACH	Chapter 1

The formats and usage notes for the following CP commands:

DETACH
LINK

are found in the *VM/SP CP General User Command Reference*.

The formats and usage notes for the following CMS commands:

ACCESS
RELEASE

are found in the *VM/SP CMS Command Reference*.

Sharing Your Files and Directories

If your files are stored in a Shared File System (SFS) file pool, you will be able to share individual files or directories (groups of related files) with other users. SFS lets you grant other users read or write authority to any of your files or directories.

Action	Command(s)	Description
Access a directory	ACCESS	Chapter 4
Create a directory	CREATE DIRECTORY	Chapter 4
List your directories	DIRLIST LISTDIR	Chapter 4
List your files or determine the owner of a file	FILELIST LISTFILE	Chapter 4
Create an alias to a file	CREATE ALIAS	Chapter 4
Determine who has aliases to a specific file, or determine the base file for a specific alias	QUERY ALIAS, (ALIALIST when in FILELIST)	Chapter 4
Grant another user read or write authority to a file or directory	GRANT AUTHORITY	Chapter 4
Determine who has authority to read or write to a specific file or directory	QUERY AUTHORITY, (AUTHLIST when in DIRLIST or FILELIST)	Chapter 4
Lock a file or directory against simultaneous updates	CREATE LOCK	Chapter 4
Determine if a file or directory is locked	QUERY LOCK	Chapter 4
Revoke read or write authority to a file or directory from a user	REVOKE AUTHORITY	Chapter 4

The CMS commands

ACCESS
 CREATE ALIAS
 CREATE DIRECTORY
 CREATE LOCK
 DIRLIST
 FILELIST
 GRANT AUTHORITY
 LISTDIR
 LISTFILE
 QUERY ALIAS
 QUERY AUTHORITY
 QUERY LOCK
 REVOKE AUTHORITY

are found in the *VM/SP CMS Command Reference*.

What You Can Do to the Files in Your Virtual Reader

CMS and CP commands allow you to look at, get rid of, keep, load into your directory or onto minidisks and reorder the files in your virtual reader.

Action	Command(s)	Description
Display a list of your reader files	RDRLIST	Chapter 10
Look at a file	PEEK	Chapter 10
Load the file into your directory or onto your minidisk	RECEIVE	Chapter 10
Purge a file	DISCARD (when in PEEK or RDRLIST) CP PURGE	Chapter 10 Chapter 10
Transfer a file to (or from) the reader queue of another user	CP TRANSFER	Chapter 10
Alter the external attributes of a file	CP CHANGE	Chapter 9
Change the order of the files	CP ORDER	Chapter 9

The CMS commands

DISCARD
PEEK
RECEIVE
RDRLIST

are found in the *VM/SP CMS Command Reference*.

The CP commands:

CHANGE
ORDER
PURGE
TRANSFER

are documented in the *VM/SP CP General User Command Reference*.

Receiving or Loading Files into Your Directory or onto Your Minidisk

Files that are in your reader or on a tape can be loaded into your directory or onto your minidisk.

Retrieving Files From ...	Command(s)	Description
Your virtual reader	RECEIVE	Chapter 10
A tape	TAPE LOAD FILEDEF and MOVEFILE	Chapter 9 Chapter 9

Command formats and usage notes for the following CMS Commands are found in the *VM/SP CMS Command Reference*.

FILEDEF
MOVEFILE
RECEIVE
TAPE LOAD

Erasing Files From Your Directory or Minidisk

When you no longer need a file you can erase or discard it from your Shared File System (SFS) directory or minidisk. You can use the ERASE command when you want to erase all the files with a particular file mode letter and number or the files with the same file name or file type. You can also use ERASE to erase an entire directory.

You can use the DISCARD command when in FILELIST to erase files. You can also use DISCARD from PEEK and RDRLIST environments; this is discussed under "What You Can Do to the Files in Your Virtual Reader."

The FORMAT command erases *all* files on a particular minidisk. (The ERASE option of the ACCESS command also erases all the files on the specified minidisk without the need to reformat the entire minidisk.)

Action	Command(s)	Description
Erase files	ERASE	Chapter 1
Erase directories	ERASE	Chapter 4
Erase files or directories from FILELIST	DISCARD	Chapter 3
Erase directories from DIRLIST	DISCARD	Chapter 4
Erase all files on a particular minidisk	FORMAT	Chapter 5

The *VM/SP CMS Command Reference* contains information on the following CMS commands that erase files and directories:

ACCESS
DISCARD
ERASE
FORMAT

Creating and Modifying Files

The System Product Editor, invoked with the XEDIT command, lets you interactively make changes, additions, or deletions to your CMS files. The UPDATE command and the XEDIT command with the UPDATE option let you modify a source program without affecting the original.

Action	Command(s)	Description
Edit a file	XEDIT	Chapter 8

The command formats and usage notes for the UPDATE and XEDIT commands are documented in the *VM/SP CMS Command Reference*.

Moving Files

CMS commands let you move a file or copies of a file from one place to another: from one minidisk to another; from one Shared File System (SFS) directory to another; to or from a tape to a minidisk; or from your directory or minidisk to the virtual reader of another user. Some commands, such as the TAPE command, move a copy of the file to another location. Other commands, such as the CP TRANSFER command, move (not copy) files.

Action	Command(s)	Description
Move files from your virtual reader to the reader of another virtual machine	CP TRANSFER	Chapter 10
Copy a CMS file from one directory or minidisk to another directory or minidisk	COPYFILE	Chapter 3
Dump contents of a minidisk onto tape, restore such files to the minidisk	DDR	Chapter 9
Move files from tapes to minidisk, minidisk to tape	TAPE	Chapter 9
Move files from one SFS directory to another	RELOCATE	Chapter 4
Move files from your directory or minidisk to the reader of another (or your own) virtual machine	SENDFILE	Chapter 10
Move files from your virtual reader into your directory or onto your read/write minidisk	RECEIVE	Chapter 10
Move files from any device supported by VM/SP to another.	MOVEFILE	Chapter 9

The CP TRANSFER command format and usage notes are found in the *VM/SP CP General User Command Reference*.

The formats and usage notes for the following CMS commands

DDR
RECEIVE
RELOCATE
SENDFILE
TAPE

are found in the *VM/SP CMS Command Reference*.

Chapter 7. Using the HELP Facility

The VM/SP HELP facility can assist you in your work by letting you conveniently display information about commands you may wish to use or to display explanatory information on system messages.

This chapter gives you a general understanding of the structure and function of the VM/SP HELP facility. It will get you started using HELP and give you some examples to practice with. If you need specific information on the format and syntax of HELP and related commands, see the *VM/SP CMS Command Reference*.

The HELP facility has the following components:

HELP

Component	Description
AVS	APPC/VM VTAM Support commands
CMS	Conversational Monitor System commands
CMSQUERY	CMS QUERY command options
CMSSET	CMS SET command options
CP	Control Program commands
CPOTHER	CP commands for other than general users (privileged commands)
CPQUERY	CP QUERY command options
CPSET	CP SET command options
EDIT	EDIT subcommands
EXEC	EXEC statements
EXEC2	EXEC 2 statements
IPCS	Interactive Problem Control System commands (includes GCS)
MACRO	CMS assembler language macros
PVM	VM/Pass-Through Facility (5748-RC1) (only if you have this installed on your system.)
QUERY	XEDIT QUERY command options
REXX	System Product Interpreter statements
ROUTINE	CMS routines from the Callable Services Library (CSL)
RSCS	Remote Spooling Communications Subsystem Networking (5664-188) program product (only if you have this installed on your system.)
SET	XEDIT SET command options
SQLDS	SQL/Data System Program Product (5688-004) (only if you have this installed on your system.)
SRPI	Server-Requester Programming Interface subcommands
TSAF	Transparent Services Access Facility
XEDIT	XEDIT subcommands

The HELP facility is used on 3270-type terminals in display mode. It can also be used on line-oriented terminals. All the information about commands and messages is contained in files called HELP files. In display mode, HELP uses the System Product Editor to display these files. In line-mode, the HELP facility types the HELP file to your screen one line at a time.

When you access the HELP facility, HELP will EXECLOAD the HELPXED XEDIT macro if it has not been loaded. Loading this file into storage improves the performance of the HELP command. If you occasionally use HELP, you may want to EXECDROP the HELPXED XEDIT macro to release the storage. See the *VM/SP CMS Command Reference* for more information.

HELP files usually appear on your screen in mixed case. However, in some installations, lowercase characters are reserved for displaying special alphabets. In this case, HELP files are displayed in uppercase. For more details, see the *VM/SP Installation Guide*.

As you work through the examples in this chapter and use HELP, you may need to refer to the "DETAIL HELP" section in Chapter 18, "Tailoring the HELP Facility," to understand the format boxes in the command HELP files.

You may notice when you perform certain tasks within the HELP facility that your PF key listing at the bottom of the screen is covered by a message. If this occurs, simply press ENTER to refresh the screen.

Getting HELP on Commands

To understand the HELP facility, it is best to look first at its structure and second at how it functions.

There are two different types of HELP files, information files and selection files. Information files contain information about commands and messages. Selection files display menus that let you select the appropriate information file. This section explains the use of information files. See "Menus" on page 151 for an explanation of selection files.

The HELP facility provides various ways of getting information for a command, depending on your level of expertise and the amount of detail you require for a particular task. Commands can contain three layers of information: BRIEF, DETAIL, and RELATED. Each layer displays a unique level of HELP.

You can display any of the three available layers by specifying the corresponding layering options: BRIEF, DETAIL, or RELATED. You may specify only one layering option at a time. However, once you have requested one layer of HELP on a specified command, you may toggle (switch) between the other layers available for that command. See "Toggling" on page 154 for more information on how to toggle between layers of HELP.

BRIEF is the default option, meaning that if you do not specify a layering option, the BRIEF layer of HELP is displayed if it exists. If BRIEF HELP is not available for a certain command, DETAIL HELP is displayed. The following sections provide more information on the three layers of command HELP.

Note: The examples used within this chapter include the component name (CP, CMS, REXX, and so forth). However, if you are requesting HELP for a command in CP or CMS, you do not need to specify the component name. For example, to request HELP on the CMS command SENDFILE, you could simply specify HELP SENDFILE. For components other than CP or CMS (REXX, EXEC, TSAF, and so forth), you must specify the component name for HELP to locate the proper command information. For example, to get help on the REXX instruction, TRACE, you would need to specify HELP REXX TRACE.

BRIEF HELP

BRIEF is the first of three layers of HELP and is available for many commands. BRIEF HELP displays a short description of the requested command, its basic syntax (command without options), an example, and, if applicable, a message telling you that either more or related information is available.

If you are in full-screen CMS and request BRIEF HELP, your screen shows the HELP command you entered and just below it, displays the BRIEF HELP information in a window that is displayed on your screen. If you are not in full-screen CMS, your entire screen displays the BRIEF HELP information.

The following example shows how your screen would look if you requested BRIEF HELP for the SENDFILE command in full-screen CMS.

If you are not currently in full-screen CMS, enter the command SET FULLSCREEN ON. Then, enter:

```
help cms sendfile (brief
```

The following screen is displayed:

```

CMS SENDFILE      BRIEF Help Information      line 1 of 10

The SENDFILE command lets you send files to other users. An abbrevi-
ation for SENDFILE is SF.

FORMAT: SENDfile filename filetype userid (options

EXAMPLE: Greg needs a copy of SPEC SCRIPT A, but it is your file. His
user ID is Greg. If you want to send him a copy, then enter:
sf spec script greg

PF1= All      2= Top      3= Quit      4= Return      5= Clocate      6= ?
PF7= Backward 8= Forward 9= PFkeys 10=          11= Related    12= Cursor
Press PF11 to get related information.
====> _
Macro-read 1 File

```

Figure 36. Sample of BRIEF HELP for the SENDFILE Command

Because the remainder of the examples in this chapter will not be done in full-screen CMS, press PF3 to quit the HELP screen. Then enter the command SET FULLSCREEN OFF before proceeding with the next example.

DETAIL HELP

The **DETAIL** layer of **HELP** presents a complete description of the command, the command format, an explanation of its parameters and options, usage notes, and error information. **DETAIL HELP** provides information similar to the information you would find listed in the *VM/SP CMS Command Reference*.

Subsetting Options

The **DETAIL** layer of **HELP** has seven subsetting options: **DESCRIPT**, **FORMAT**, **PARMS**, **OPTIONS**, **NOTES**, **ERRORS**, and **ALL**. By specifying subsetting options, you can display one or more particular sections of the **DETAIL HELP**. **ALL** is the default option, meaning that the entire **DETAIL HELP** is displayed. It is possible to change the default option, but if you do so, you will need to specify **ALL** as the subsetting option to display the entire **DETAIL** layer. See the **DEFAULTS** command in the *VM/SP CMS Command Reference* for more information. The *VM/SP CMS Command Reference* also contains a complete description of the subsetting options.

For example, to display the entire **DETAIL** layer of the **SENDFILE** command in the **CMS** environment, you would enter:

```
help cms sendfile (detail)
```

Let's display just the usage notes in that same file. Enter:

```
help cms sendfile (notes)
```

The following screen is then displayed:

```

CMS SENDFILE      DETAIL Help Information      line 1 of 315
Usage Notes

1. Tailoring the SENDFILE Command Options

   You can use the DEFAULTS command to set up options and/or override
   command defaults for SENDFILE. However, the options you specify
   in the command line when entering the SENDFILE command override
   those specified in the DEFAULTS command. This allows you to cus-
   tomize the defaults of the SENDFILE command, yet override them
   when you desire. Refer to the DEFAULTS command description for
   more information.

2. Using the SENDFILE Menu (Display Terminals Only)

   Enter the SENDFILE command without operands to display a menu, on
   which you "fill in the blanks" with the necessary information. A
   sample SENDFILE menu is shown in the "Examples," below.

PF1= All      2= Top      3= Quit      4= Return      5= Clocate      6= ?
PF7= Backward 8= Forward  9= PFkeys  10= Brief      11= Related     12= Cursor

====> _
Macro-read 1 File

```

Figure 37. Sample of **DETAIL HELP** for the **SENDFILE** Command

RELATED HELP

The RELATED layer of HELP is a multipurpose layer. RELATED HELP makes you aware of commands that are similar to the presently displayed HELP file, making it easier for you to decide which command to use. For example, if you want to remove a file from your readerlist and, after reading HELP ERASE, you realize that ERASE is not the correct command, the RELATED layer of the ERASE command lets you easily access the HELP file for the correct command, DISCARD.

When you request RELATED HELP on the SET or QUERY commands, the screen lists and briefly describes all the SET and QUERY operands available for the system component. You could directly access HELP information on any of the displayed operands from these menu screens by positioning the cursor on a particular operand and pressing ENTER.

For information on how to create your own RELATED HELP files, see Chapter 18, "Tailoring the HELP Facility."

For example, to display the RELATED layer of the ERASE command in the CMS environment, enter:

```
help cms erase (related)
```

The following screen is then displayed:

```

CMS ERASE          RELATED Help Information          line 1 of 20

For RELATED information on removing files or parts of files from your
virtual machine, place the cursor under the topic of your choice and
press ENTER or the PF1 key.

DELETE  - Removes one or more lines from
         a file while using XEDIT.

DISCARD - Removes files from your DIRLIST,
         FILELIST, MACLIST, PEEK, or
         RDRLIST screen.

ERASE   - Removes files from your minidisk
         or SFS directory.

PURGE   - Removes spool files from your
         reader, printer or punch.

PF1= Help   2= Top    3= Quit   4= Return   5= Clocate  6= ?
PF7= Backward 8= Forward 9= PFkeys 10=         11= Brief   12= Cursor

====> _
                                         Macro-read 1 File

```

Figure 38. Sample of RELATED HELP for the ERASE Command

HELP Facility

Other Options

There are five other options that affect the display of HELP: SCREEN/NOSCREEN, TYPE/NOTYPE, and EXTEND. Briefly, these other options control the display of files and error messages and the search order of commands. A complete description of these options can be found in the *VM/SP CMS Command Reference*.

Getting HELP on SET and QUERY

The HELP facility provides a special feature that lets you quickly access HELP on specific SET or QUERY options in CP, CMS, or XEDIT. If you know the component name (CP, CMS, or XEDIT) and the name of the option, you can immediately get to the specific HELP file you need.

Reserved file types in the HELP facility let you access specific HELP files. To access HELP information for any CMS SET option, you would enter HELP, followed by the file type, CMSSET, and the option name. For example, to get HELP on the CMS command SET RDYMSG, you would enter the following:

```
help cmsset rdymsg
```

This command would immediately access the specific HELP file for the CMS SET RDYMSG command.

To request HELP for a CMS QUERY option, you would enter HELP, followed by the file type CMSQUERY, and the option name.

The same is true for CP SET and QUERY options. To get HELP on a CP SET option, you would enter HELP, followed by CPSET, and the option name; for HELP on a CP QUERY option, you would enter HELP, CPQUERY, and the option name.

To request HELP for XEDIT SET and QUERY options, you can simply enter HELP, followed by SET or QUERY, and the name of the option. For example, to get HELP on the XEDIT command SET APL, you would enter the following:

```
help set apl
```

Even though there is also a CMS command SET APL, the HELP facility would correctly provide you with HELP information on the XEDIT SET APL command. You would need to enter:

```
help cmsset apl
```

to request information on the CMS SET APL command.

For more information on HELP for SET and QUERY options, see the *VM/SP CMS Command Reference*.

Facts About Command HELP

The following information helps you to more efficiently use command HELP:

- Command HELP lets you easily move between screens of HELP information. While viewing the command HELP for one command, you can get help for another command by specifying HELP followed by the name of the new command. For instance, if you were viewing the HELP file for the CMS PRINT command, and decided you wanted help on the CMS COPYFILE command, you could enter the following on the command line of your current screen:

```
help cms copyfile
```

This would bring you directly to the HELP information for the COPYFILE command.

- BRIEF, DETAIL, and RELATED are conflicting HELP options. You can specify only one of these options in the command string. If you specify more than one option at a time, the last option entered is effective. For example, if you entered the following on the command line:

```
help cms erase (brief related
```

RELATED HELP for the ERASE command is displayed.

If you entered the following on the command line:

```
help cms erase (descript notes brief
```

BRIEF HELP for the ERASE command is displayed. If you then entered MOREHELP on the command line or pressed PF10, you would see the DESC and NOTES sections for the ERASE command. Use of the PF keys to move between HELP screens is explained in detail in "Using the PA2 and PF Keys" on page 153.

Getting HELP on Messages

Sometimes, when you perform a CMS task, the system responds with a message. To find out why the message was produced and perform any necessary corrective action, see Message HELP.

The HELP files for messages display the message text, an explanation of why the message was entered, the system action, and a user action. For a complete description of all the possible ways to request HELP for messages, see the *VM/SP CMS Command Reference*.

For example, to display the HELP file for the CMS message DMSHLP002E, enter the following on the command line:

```
help DMS002E
```

or

```
help DMSHLP002E
```

Menus

In VM/SP HELP, menus serve as selection files; that is, HELP panels that assist you in reaching the HELP files you need. If you are uncertain of the name of a component or command you might like to use, you can simply choose from a list of likely possibilities.

You can select an entry from the menu by positioning the cursor in front of or under any part of the entry and then pressing ENTER or the PF1 key. After the HELP file is displayed, you can return to the menu by pressing PF3.

To position the cursor at the entry you want, you can do any one of the following:

1. Use the key marked --->|, which functions as a tab key, causing the cursor to move to the first character of the next entry.

2. Use another cursor-movement key.
3. Enter on the command line the desired entry or a string that appears in the task description and press PF5.

When the cursor is positioned at the desired entry, press ENTER or the PF1 key to display the HELP file for that entry.

There are two types of menus, task menus and component menus. An asterisk (*) preceding an entry in a displayed menu indicates that the entry is the name of a component menu file. A colon (:) indicates that the entry is the name of a task menu.

TASK Menus in HELP

TASK menus are especially useful for users new to the VM system. This is because they describe an action that you may want to perform and then guide you to the appropriate HELP file.

You can see a list of tasks and components available to you by typing:

```
help task
```

The following TASK menu is displayed:

```
HELP TASKS      Task Help Information      line 1 of 21

To use VM Help, move the cursor to any topic below,
then press the ENTER key or the PF1 key.

TASKS      - Help if you don't know VM commands.
            Good choice for beginners.
MESSAGE    - Explain how to get help for VM messages.
HELP       - Explain some ways for using HELP.
MENUS      - List the HELP component MENUs.
COMMANDS   - List VM commands that you can use.
CMS        - Show only CMS commands.
MACROS     - Show only CMS macros.
ROUTINES   - Show only CMS routines.
CP         - Show only CP commands.
CPOTHER    - Show other than general user CP commands
XEDIT     - List System Product Editor items.
REXX      - Help you use the REXX language.
PF1= Help   2= Top    3= Quit   4= Return   5= Clocate  6= ?
PF7= Backward 8= Forward 9= PFkeys 10=         11=         12= Cursor

====> _

Macro-read 1 File
```

Figure 39. Sample HELP TASK Menu

Component Menus

Component MENUs list the names of all the command HELP files available for a specific component.

If you followed the example shown in the preceding section, your screen now displays the HELP TASK menu. To display all the command HELP files available for REXX, position your cursor anywhere under the word REXX, and press ENTER.

The following component MENU is then displayed:

```

REXX MENU      Menu Help Information      line 1 of 20
(c) Copyright IBM Corp. 1983, 1988 (adapted from IBM Form SC24-5239)

A file may be selected for viewing by placing the cursor under any
character of the filename wanted and pressing the ENTER key or the PF1
key. For a description of the HELP operands and options, type HELP HELP.

ABBREV  C2D      ERRORTX  LASTPOS  PROCEDUR  SIGNAL  UPPER
ABS     C2X      EXIT      LEAVE    PULL       SOURCELI  USERID
ADDRESS DATATYPE  EXTERNAL  LEFT     PUSH       SPACE    VALUE
ARG     DATE     FIND      LENGTH   QUEUE      STORAGE  VERIFY
BITAND  DELSTR   FORM      LINESIZE  QUEUED    STRIP    WORD
BITOR   DELWORD  FORMAT    MAX       RANDOM    SUBSTR   WORDINDE
BITXOR  DIAG     FUZZ      MIN       RETURN    SUBWORD  WORDLENG
CALL    DIAGRC   IF        NOP       REVERSE   SYMBOL   WORDPOS
CENTER  DIGITS   INDEX     NUMERIC  RIGHT     TIME     WORDS
CENTRE  DO       INSERT   OPTIONS  SAY       TRACE    XRANGE
CMSFLAG DROP     INTERPRE  OVERLAY  SELECT    TRANSLAT X2C
PF1= Help  2= Top    3= Quit   4= Return  5= Clocate  6= ?
PF7= Backward 8= Forward 9= PFkeys 10=       11=         12= Cursor

====> _
Macro-read 1 File

```

Figure 40. Sample Component MENU for REXX

If you enter HELP MENUS, your screen displays a listing of all the available menus.

Using the PA2 and PF Keys

The PA2 key (or its equivalent) and PF keys have special meanings when in the HELP facility. This section provides details on the settings for each key.

As you use the PF keys, you will note that the last PF key you press appears highlighted on your screen.

Toggleing

A toggle key is a key that lets you move back and forth between various HELP displays. Toggle keys for HELP are PF1, PF10, and PF11. These keys let you toggle (switch) between the BRIEF, DETAIL, ALL, and RELATED HELP sections.

The type of information you get when you press each of these keys depends on the type of HELP information available for a particular command and also depends on what information is being displayed on your screen at the time when you press the PF key. The PF key settings shown at the bottom of your HELP screen changes to reflect the kinds of HELP available to you. If a certain type of HELP is not available for a particular command, the corresponding PF key setting at the bottom of your HELP screen will be blank.

The PF1 key toggles between the ALL and BRIEF layers of HELP. PF10 toggles between the DETAIL and BRIEF layers of HELP. PF11 toggles between RELATED and BRIEF. For example, if you are viewing the BRIEF layer of HELP, your PF keys are set as follows:

PF1 = ALL
PF10 = MOREHELP
PF11 = RELATED

This means that if you press PF10, you then receive MOREHELP, which provides you with the DETAIL layer of HELP information. Now, while you were viewing DETAIL HELP, your PF keys would have changed to the following settings:

PF1 = ALL
PF10 = BRIEF
PF11 = RELATED

If you pressed PF10, you would return to BRIEF HELP, and your PF key settings would be as they were in the first part of this example.

The following table lists the settings for PF1, PF10, and PF11 when all HELP layers are available for the displayed file:

Screen Display	PF Key Settings
BRIEF	PF1 = ALL PF10 = MOREHELP PF11 = RELATED
DETAIL	PF1 = ALL PF10 = BRIEF PF11 = RELATED
ALL	PF1 = BRIEF PF10 = blank PF11 = RELATED
RELATED	PF1 = HELP PF10 = MOREHELP PF11 = BRIEF

Following is a listing of the values for PA2 and the PF keys. On a terminal equipped with 24 PF keys, PF keys 13 to 24 are assigned the same values as PF keys 1 to 12.

Table 11 (Page 1 of 2). PA and PF Keys in the HELP Facility		
Key	Meaning	Usage
PA2	Print	Prints a hard copy of currently displayed HELP information. Remember that after quitting HELP, you must enter CP SP PRT CLOSE to print the file.
PF1	Help	Accesses HELP files from a menu or a RELATED section after the cursor is positioned at the desired entry.
	All	Displays the HELP file as if the ALL option was specified.
	Brief	Displays BRIEF information when viewing the ALL option of a specified HELP file. Note: If PF1 appears blank on a HELP screen, this means that either ALL or BRIEF HELP is not available for a particular command.
PF2	Top	Moves the display to the beginning of the HELP file.
PF3	Quit	Exits from the currently displayed HELP file.
PF4	Return	Exits from the HELP facility. PF4 quits all HELP files currently in storage. For example, if you call a menu, then call a HELP file from that menu, PF4 quits both the file and the menu and returns control to the originating environment.
PF5	Clocate	Is the XEDIT subcommand CLOCATE. On the command line, enter the string you are looking for. Then press PF5 to tell HELP to locate the next occurrence of the string. Repeated pressing of the PF5 key locates additional occurrences of the string. HELP highlights the line located.
PF6	?	Displays the last user command entered from the command line.
PF7	Backward	Scrolls the display backward one screen.
PF8	Forward	Scrolls the display forward one screen.
PF9	PF Key	Displays a file containing an explanation of PF key meanings for displayed files.
PF10	Morehelp	Displays the HELP information from a command file as if the DETAIL option was specified.
	Brief	Displays BRIEF information. Note: If PF10 appears blank, this means that either DETAIL HELP or BRIEF HELP is not available for a particular command. In this instance, PF1 would be set to ALL and provide you with all the HELP available for the command.
PF11	Related	Displays the HELP information from a command file as if the RELATED option was specified.
	Brief	Displays BRIEF information. Note: If PF11 appears blank, this means that either RELATED HELP or BRIEF HELP is not available for a particular command.

Table 11 (Page 2 of 2). PA and PF Keys in the HELP Facility

Key	Meaning	Usage
PF12	Cursor	Moves the cursor to the command line or to its previous location on the screen.

Using the MOREHELP Command

If you are viewing HELP facility files on a line-mode terminal, or if you cannot use a PF key to obtain **DETAIL** or **RELATED** information, you may find the **MOREHELP** command useful.

MOREHELP provides you with either additional or related information about the last valid HELP command you entered. For more information on the **MOREHELP** command, see the *VM/SP CMS Command Reference*.

System Product Editor

If you are using HELP in display mode on a 3270-type terminal, the HELP facility uses the System Product Editor to display HELP files. Many of the features of the XEDIT subcommands are used on the displayed files. Two of the available features are:

Clocate Locates a specified character string in the file or uses PF5 to search the file. PF5 positions the cursor under the target string. **CLOCATE** remembers the string that was last used, even if you have performed other functions. You can press PF5 to use that last string again. To change the string being searched, enter a new string on the command line and press PF5 to search for the new string.

Scrolling Moves the display up or down.

See the *VM/SP System Product Editor Command and Macro Reference* for a complete explanation of these features.

Not all XEDIT subcommands are used on the displayed HELP files. The excluded subcommands are listed below:

FILE	REPLACE
INPUT	SET
MACRO	POWERINP
READ	

These subcommands are excluded to prevent unnecessary copying of HELP files or to avoid any inadvertent changes to the HELP files. If you use these subcommands while viewing a HELP file, they will be ignored, and you will receive an error message. While these subcommands will not work on files displayed by the HELP facility, you can use them when you use the XEDIT subcommand to edit the files.

Note: When you enter a command from the command line, the system does not search for XEDIT subcommand synonyms and XEDIT macros such as the **SPLTJOIN** macro. This means that when a macro such as **SPLTJOIN** is invoked from within the HELP facility, the message 'No such subcommand **SPLTJOIN**' is entered.

Printing HELP Screens

When you display HELP files, you can get a printed copy of the screen by pressing PA2 while the screen is displayed. To receive a hard copy:

1. Press the PA2 key.
2. Exit from HELP.
3. Enter the command CP SP PRT CLOSE.

Note: If you want to print the entire HELP file for a certain command, you will need to access the HELP disk and print the file. See your system administrator for information on accessing and printing HELP files.

Working with Your HELP Files

Now that you are familiar with using HELP files, you may wish to learn how to customize your VM/SP HELP files. See Chapter 18, "Tailoring the HELP Facility" for more information.



Part 2: Working with VM/SP

Once you are familiar with the basics of the VM/SP system, you are ready to begin using VM/SP to perform routine tasks such as editing files, using input and output devices, communicating with other users, and using the CMS batch facility. Part 2 provides you with the detailed information you need to perform these tasks.

Chapter 8: Editing Your Files contains some of the basic information you need to create and write a CMS file directly from your terminal, or to correct or modify an existing CMS file.

Chapter 9: Using Real Printers, Punches, Readers, and Tapes discusses how to use tapes and punched cards in CMS, and how to use your virtual printer and punch to get real output.

Chapter 10: Communicating with Other Computer Users discusses the ways in which you send information to other users and receive information from them.

Chapter 11: Looking at VM/SP through Windows describes how to invoke full-screen CMS and use windowing functions to view information in windows on your physical screen.

Chapter 12: Using the CMS Batch Facility describes how to prepare and send job streams to a CMS batch virtual machine. The CMS batch facility is a CMS feature that lets you send jobs to another machine for execution.



Chapter 8. Editing Your Files

To *edit* a file means to make changes, additions, or deletions to a CMS file that is in a SFS directory or on a minidisk. To make these changes interactively: you instruct the editor to make a change, the editor does it, and then you request another change. You can edit a file that does not exist; when you do so, you create the file online, and can modify it as you enter it.

To *file* a file means to write a file you are editing back onto a directory or minidisk, incorporating any changes you made during the editing session. When you enter the FILE subcommand to write a file, you are no longer in edit environment, but are returned to the CMS environment. You can, however, write a file and then continue editing it, by using the SAVE subcommand.

An *editing session* is the time when a file is in your virtual storage area, from the moment you enter the XEDIT command or the EDIT command until you let the editor know that you are finished working on the file, by entering FILE or QUIT.

When you are using XEDIT in full-screen CMS, CMS output, as well as messages and other information, is displayed in windows that automatically appear on your XEDIT screen. If a window appears, move the cursor to any corner of the window border and type a *F* to scroll the window forward and remove it from your screen. You can also type a *C* to clear the window.

This feature of XEDIT in full-screen CMS is particularly useful because it lets you view your messages without being removed from the XEDIT environment. For more information on using full-screen CMS, refer to Chapter 11, "Looking at VM/SP through Windows."

System Product Editor

The editor provides the following capabilities:

- Screen support for IBM 3270 Display Terminals is available including:
 - The ability to display multiple views of the same file or of different files
 - Automatic "wrapping" of lines that are wider than a screen line
 - The ability to enter selected (prefix) subcommands directly on the displayed lines
 - The ability to define the screen format according to individual preferences.
- Extended string search facilities are provided for improved text processing.
- A variety of macros that use the EXEC 2 or System Product Interpreter are offered.
- An enhanced set of functions to handle program development is available, including automatic update generation.
- The ability to import and export data between files is provided.
- The ability to edit and manipulate files that contain Double-Byte Character Set (DBCS) strings.

Many languages have more characters than can be displayed using one-byte codes (Kanji, for example). A Double-Byte Character Set (DBCS) is used to represent those characters. The double-byte characters can appear in a sentence with characters from other languages that are displayed in 1-byte codes. Files containing double-byte characters are handled differently than files that only contain 1-byte characters. Special considerations for editing files that contain double-byte characters are described in the *VM/SP System Product Editor Command and Macro Reference*.

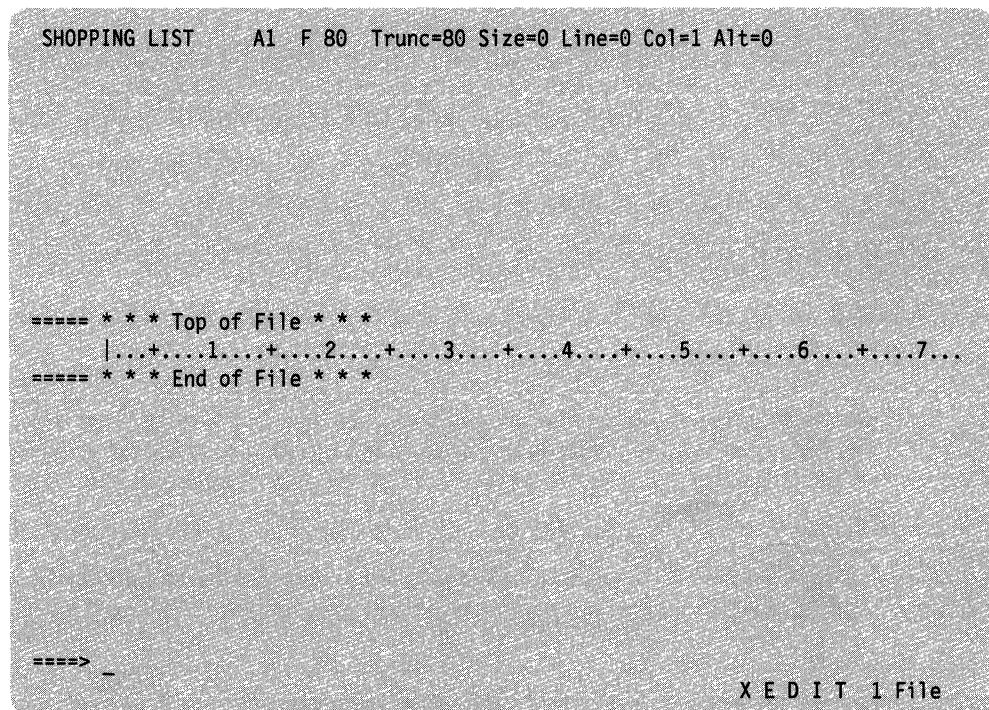
For complete information about the System Product Editor, see the *VM/SP System Product Editor User's Guide* and the *VM/SP System Product Editor Command and Macro Reference*.

Beginning an Editing Session

When you enter the XEDIT command you must specify the file name and file type of the file you want to create or edit. To create a new file called SHOPPING LIST, enter:

```
xedit shopping list
```

The XEDIT command calls the System Product Editor, so you will see a screen similar to that in Figure 41.



```
SHOPPING LIST  A1 F 80  Trunc=80 Size=0 Line=0 Col=1 Alt=0

==== * * * Top of File * * *
|...+...1...+...2...+...3...+...4...+...5...+...6...+...7...
==== * * * End of File * * *

====> _

X E D I T 1 File
```

Figure 41. Sample XEDIT Screen

On the command line (next to the arrow) type INPUT and press the ENTER key. The file is placed in input mode. The cursor is automatically placed on the first line in the input zone, where you can enter your data.

Enter the following data:

apples
lettuce
tomatoes
bread

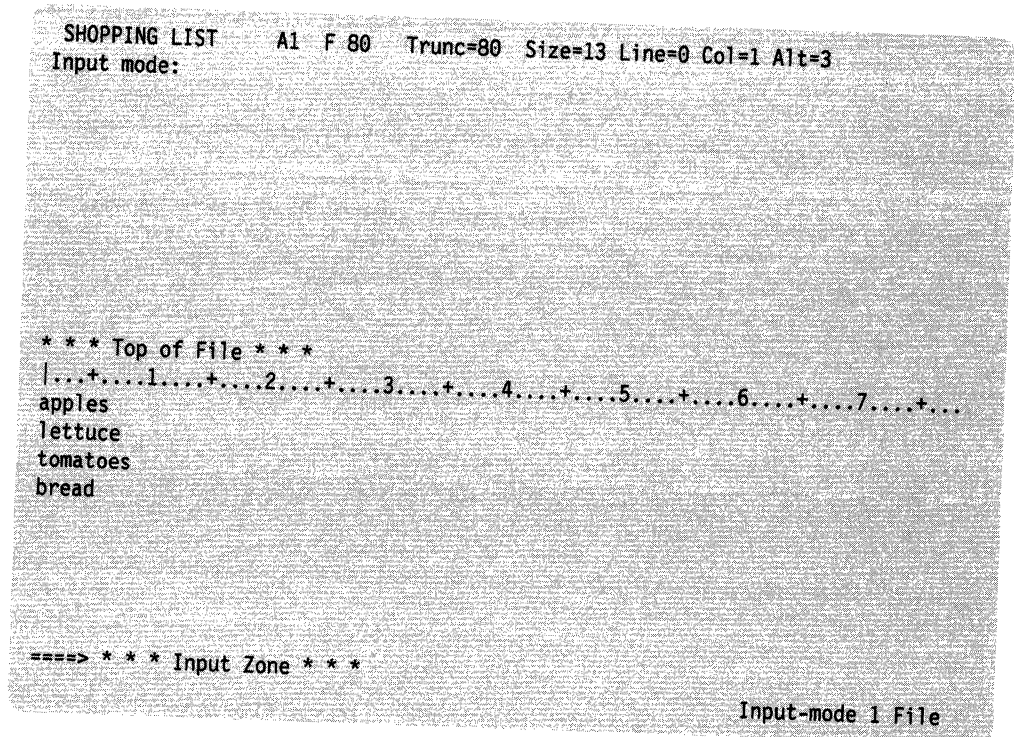


Figure 42. Sample XEDIT Screen in INPUT Mode

Now, press the ENTER key and the screen moves up so that you can enter more data. When you are finished entering data, press the ENTER key again to return to edit mode.

To keep this file in permanent minidisk or directory storage, you enter FILE on the command line. You should see a message that looks something like this:

Ready;

Even though the file has disappeared from your screen, the editor has saved it on your directory or minidisk.

Now check and see if the file was really saved. Use the FILELIST command to list the files on file mode A with a file name of SHOPPING. Enter:

filelist shopping

The display might look like Figure 43.

```
MYLOGON FILELIST  A0 V 108 Trunc=108 Size=1  Line=1 Col=1 Alt=0
Directory = VMSYSU:MYLOGON.
Cmd Filename Filetype Fm Format Lrecl Records Blocks  Date  Time
-  SHOPPING LIST  A1 F          80      4      1  5/16/83 15:07:49

1= Help      2= Refresh  3= Quit      4= Cancel    5= Sort(dir) 6= Sort(size)
7= Backward  8= Forward  9= FL /n    10= Share   11= XED/FILEL 12= Cursor

====>

X E D I T  1 File
```

Figure 43. Sample FILELIST STATS Screen for a Particular File Type

Press the PF3 key to leave the FILELIST screen.

Saving Your Changes

The file you create during an edit session or the modifications you make to an existing file are not automatically written to a minidisk or directory file. There are several ways to save the results:

- Periodically enter the subcommand

save

to write to your directory or minidisk the contents of the file as it exists when you enter the subcommand. Periodically entering this XEDIT subcommand protects your data against a system failure; you can be sure that changes you make are not lost.

- At the beginning of the edit session, enter the SET AUTOSAVE subcommand, with a number:

set autosave 10

Then, for every tenth change or addition to the file, the editor performs an automatic save request, which writes the file.

- To terminate the editing session and write the file, enter the subcommand:

file

The file disappears from your screen, but the editor saved it on your directory or minidisk. You can return to the edit environment by entering the XEDIT command, specifying a different file or the same file.

The editor decides where to write the file according to the following conditions:

- If you specify a file mode on the FILE or SAVE subcommand line, the file is written onto the specified directory or minidisk.
- If you do not specify a file mode, the editor writes the file onto the directory or minidisk specified by the file mode of the file being edited.
- The file mode used may be the file mode of a minidisk or of a directory. The editor can only write a file to minidisks when accessed as read/write or to a directory when you have write authority for the file. If the file does not already exist in the directory, you must have write authority to the directory to create it. If you are unable to file your modifications to an existing file, you will need to file your changes using a different file mode. You can also have the owner of the file grant you write authority to the file.

We Made Changes to the File at the Same Time!

Stan and Anne share a file in an SFS directory. XEDIT, with the default LOCK option, prevents Anne from changing a file while Stan is editing it, and vice versa. But, for this example, let's suppose Stan XEDITs the file with the NOLOCK option, and does not use CREATE LOCK to lock the file beforehand. While Stan is working on the file, Anne XEDITs the file and enters FILE to save her changes. Stan finishes his work and tries to enter SAVE or FILE. He receives a message which informs him someone else has changed the file.

Stan has a decision to make: He can use SSAVE or FFILE to save his updates, but he would overwrite the existing file and the changes Anne made to the file would be lost. Instead of destroying Anne's changes, Stan could file his updates under a different name, and contact Anne about the simultaneous updates.

To avoid this situation altogether, use XEDIT with default LOCK option, or use CREATE LOCK whenever you edit files you share with other users. That way, if someone XEDITs a file you are currently editing, they receive a message stating that the file is already locked. For information on locking shared files, see "Locking Files and Directories" on page 119.

Remember, when you use the default LOCK option with XEDIT, you do not usually need to be concerned with other users changing the file. Using XEDIT with the LOCK option automatically places an UPDATE SESSION lock on your file. It is only when you specify the NOLOCK option of XEDIT that you need to be concerned with creating and deleting SFS locks. The NOLOCK option of XEDIT should only be used when you are not going to make any changes to the file, or if you will save your changes under a different name.

Canceling Your Changes

If you are editing a file and decide that you do not wish to save the changes, you can enter the subcommand:

```
quit
```

No changes that you made since you last used the SAVE subcommand (or the editor last issued an automatic save for you) are retained. If you have just begun an edit session, and have made no changes at all to a file, and for some reason you do not want to edit it at all (for example, you misspelled the file name, or want to change a

CMS setting before editing the file), you can use the QUIT subcommand instead of the FILE subcommand to terminate the edit session and return to CMS.

What To Do When You Run Out of Space

There are two situations that may prevent you from continuing an edit session or from writing a file to a directory or minidisk:

- The editor may run out of virtual storage.
- Your minidisk, file space or storage group may become full.

You should be aware of these situations, know how to avoid them, and how to recover from them, should they occur.

If You Run Out of Virtual Storage

When you enter the XEDIT command to edit a file, the editor copies the file into virtual storage. If it is a large file, or you have made many additions to it, the editor may run out of storage space. If it runs out of storage it displays the message:

```
Not enough virtual storage available
```

When this happens, you cannot make any changes or additions to the file unless you first delete some lines. If you attempt to add a line, the editor displays the message:

```
No storage available to insert lines
```

You should first use the FILE subcommand to write the file. If you want to continue editing, you should provide the editor with more storage space to work with. To do this, you can find out how large your virtual machine is and then increase its size. To find out the size, enter the CP QUERY command:

```
query virtual storage
```

If the response is:

```
STORAGE = 1024K
```

you might want to redefine your storage to 2M. Enter the CP command DEFINE, as follows:

```
define storage 2m
```

This command resets your virtual machine; you must enter the CP IPL command to reload CMS before you can continue editing.

If a file is very large, the editor may not have enough space to let you edit it using the XEDIT command. The message

```
DMSXIN132S File fn ft fm too large
```

indicates that you must obtain more storage space before you can edit the file. If this is the case, or if you are editing large files, you should redefine your storage

before beginning the terminal session with the CP DEFINE command. For example to increase the size of your virtual storage to 4M, enter:

```
define storage 4m
```

This resets your virtual machine and you must IPL CMS again. If you enter the CP DEFINE command and receive the message

```
Storage exceeds allowed maximum
```

you should see your installation support personnel about having the directory entry for your user ID updated so that you have a large storage size to begin with.

Splitting CMS Files into Smaller Files

If the file you are editing is too large, and the data it contains does not have to be in one file, split the file into smaller files, so that it is easier to work with. Two of the methods you can use to do this are described in the text which follows.

Use the COPYFILE Command: The COPYFILE command copies portions of a file into separate files, and then deletes the copied lines from the original file. For example, if you have a file named TEST FILE that has 1000 records, and you want to split it into four files, you could enter:

```
copyfile test file a test1 file a (from 1 for 250
copyfile test file a test2 file a (from 251 for 250
copyfile test file a test3 file a (from 501 for 250
copyfile test file a test4 file a (from 751 for 250
```

When these COPYFILE commands are complete, you have four files containing the information from the original TEST FILE, which you can erase:

```
erase test file
```

Use the XEDIT Command: If you use the editor to create smaller files, you can edit them as you copy them, that is, if you have other changes that you want to make to the data. To copy files with the editor, you use the GET subcommand. Using the file TEST FILE as an example, you might enter:

```
xedit test1 file
get test file a 1 250
:
file
xedit test2 file
get test file a 251 250
:
```

Again, you could erase the original TEST FILE when you are through with your edit session.

If Your Minidisk Is Full

In addition to the virtual storage required for you to XEDIT large files in your virtual machine, you must also have sufficient space to file the updated file on the minidisk. If your files are stored on minidisks, it is possible for you to get in a situation where your read/write minidisks are full. This section explains what kinds of things you can do if you find yourself in this situation.

Note: If your files are stored in an SFS directory, skip this section and instead refer to the following section, "If Your File Space is Full".

When you enter a FILE or SAVE subcommand or when an automatic save request is entered, the editor writes a working copy of the file you are editing onto your minidisk, and names it XEDTEMP CMSUT1. If this causes the minidisk to become full, the editor erases the work file, and you receive the message:

```
Disk or file space is full; set new file mode or clear some space
```

The original file (as last written to your minidisk) remains unchanged. You can use the FILELIST command to list the files on the minidisk, then the DISCARD command to erase any unwanted files.

If you do not want to erase any of the files on the minidisk, there are several alternate recovery paths you can take:

1. If you have another read/write minidisk accessed, you can use the FMODE subcommand to change the file mode of the file, so that when you file the file, it is written to the other minidisk. If you have a directory or a read/write minidisk that is not accessed, you can access it. After filing the file on the second minidisk, erase the original copy, and then use the COPYFILE command to transfer the file back to its original minidisk.
2. If you do not have any other read/write minidisk in your virtual machine, you may be able to transfer some of your files to another user, using either the SENDFILE, PUNCH or DISK command. When the files have been read onto the other user's minidisk, erase them from your minidisk. Then, return to edit mode and enter the FILE subcommand.
3. If you have an original copy of the file, you can erase it to create more space. Move your cursor to the command line and enter the erase command followed by the file name and file type. However, you should not use this method of creating space unless absolutely necessary. If the new version of the file is larger than the original version, you may still be unable to save the file and may lose both versions.
4. You can save the file in SFS. If you have file space in a file pool, access one of your directories and use the FMODE subcommand to change the file mode of the file.

After you use the FILE subcommand to write the file to your minidisk, you should continue erasing any files you no longer need. You should contact your system administrator to obtain more minidisk space.

If Your File Space is Full

After you have completed your changes to a file, when you enter a FILE or SAVE subcommand, or when an automatic save request is issued, the editor writes a copy of the file you are editing to your directory. If this causes the file space allowed for your directories to become full, the file cannot be written. Instead, you receive the following message:

```
Disk or file space is full; set new file mode or clear some space
```

At this point, you can use the FILELIST command to list the files in your file space, then use the DISCARD command to erase any unwanted files.

If you do not want to erase any of the files in your file space, there are several alternate recovery paths you can take:

1. You can contact your system administrator to request that more storage be added to your file space. After your system administrator has given you more storage, reenter the FILE command.
2. If you have a read/write minidisk accessed, you can use the FMODE subcommand to change the file mode of the file, so that when you file the file, it is written to the minidisk. If you have a read/write minidisk that is not accessed, you can access it.
3. If you are able to store any of your files on minidisks, you may be able to use the COPYFILE command to move files from your file space to a read/write minidisk. After copying the files, erase the original copy in your file space.
4. If you do not have any read/write minidisks in your virtual machine, you may be able to transfer some of your files to another user, using either the SENDFILE, PUNCH or DISK commands. When the files have been read into the other user's file space, you can erase them from your file space.

After you use the FILE subcommand to write the file to your file space, you should continue erasing any files you no longer need.

To determine what proportion of your file space you have used and how much more is available, refer to the QUERY LIMITS command in the *CMS Command Reference* or refer to Chapter 4, "Using the Shared File System."

If Your Storage Group is Full

In the Shared File System, users share a large amount of physical disk space called a storage group. It is possible that you could have enough file space defined for your work but still be unable to enter a FILE or SAVE command if the storage group becomes full.

When the maximum number of blocks within a storage group are used, your file will not be written to a directory. Instead, you will receive the following message:

```
Disk or file space is full; set new file mode or clear some space
```

Although the file pool administrator will also receive a message, it may take some time for the problem to be corrected. In the interim, you can erase any unwanted files to free some space. However, space in the storage group is shared by many

users. It is possible that other users may issue a FILE or SAVE command and use the space you have freed before you can do so.

Using the Editor in Line-Mode

The editor display mode is the most common format of operation on a 3270. There are, however, instances when it is not possible or not desirable to use the editor in display mode. For these instances, you should use the line-mode of operation, which is the equivalent to using a typewriter terminal. When you use line-mode, each XEDIT subcommand you enter, and the response (if you have verification on), is displayed, a line at a time, on the screen in the output display area.

You need only be concerned with using line-mode if you are connected to VM/SP by a remote 3270 line, or if you are editing a file from within an exec and you want to control the screen display. Although it is possible to use the editor in line-mode on a local 3270, it is rarely necessary for regular editing purposes.

Editing on a Remote 3270

When you use the editor from a remote 3270, you are placed in line-mode by the editor. The advantage of using the 3270 in line-mode (particularly on a remote editor) is that the editor responds more quickly to display requests. When you use display mode, the editor has to write out the entire output display area when you move the current line pointer; in line-mode, it has only to write a single line.

If you want to use display mode, you enter the XEDIT subcommand:

```
set terminal display
```

The editor begins operating in display mode, and you can use the special editing functions available in display mode.

However, when you are using a remote 3270 in display mode, and you enter the INPUT subcommand to begin entering input lines, the screen is cleared, and your input lines are displayed as if you were in line-mode, beginning at the top of the screen. When you enter a null line to return to edit mode, the editor returns to a full-screen display.

You can resume editing in line-mode by entering the subcommand:

```
set terminal typewriter
```

Editing from an Exec File

If you use the editor from an exec, but you do not want the screen cleared when the editor gets control, you can specify the NOCLEAR option on the XEDIT command line:

```
xedit test file (noclear
```

Entering the command

```
xedit test file (noscreen
```

places the 3270 in line-mode, so that the lines already on the screen are not erased. The 3270 remains in line-mode for the remainder of the edit session, and you cannot use the SET TERMINAL subcommand to place it in display mode.

Chapter 9. Using Real Printers, Punches, Readers, and Tapes

CMS Unit Record Device Support

CMS supports one virtual reader at address 00C, one virtual punch at address 00D, and one virtual printer at address 00E. When you enter a CMS command or run a program that uses one of these unit record devices, the device must be attached at the virtual address indicated.

Using the CP Spooling System

Any output that you direct to your virtual printer or punch, or any input you receive from your reader, is controlled by the spooling facilities of the control program (CP). Each output unit is known to CP as a spool file, and is queued for processing with the spool files of other users on the system. Ultimately, a spooled printer file or a spooled punch file can be released to a real printer or card punch for printing or punching.

The final disposition of a unit record spool file depends on the spooling characteristics of your virtual unit record devices, which you can alter with the CP command SPOOL. To find out the current characteristics of your unit record devices, enter the command:

```
query ur
```

Following is an example of the response you receive from entering this command.

```
RDR 00C CL A NOCONT HOLD EOF READY
PUN 00D CL A NOCONT NOHOLD COPY 001 READY FORM STANDARD
00D TO CMSGDE DIST 2647-706 DEST OFF
PRT 00E CL A NOCONT NOHOLD COPY 001 READY FORM STANDARD
00E TO CMSGDE DIST 2647-706 FLASHC 000 DEST OFF
00E FLASH CHAR MDY FCB
```

Use the SPOOL command to change spool file characteristics. When you use the SPOOL command to control a virtual unit record device, you do not change the status of spool files that already exist, but rather set the characteristics for subsequent output. For information on modifying existing spool files, see "Altering Spool Files" on page 174.

Note: Spool files are managed by CP, not CMS. Therefore they cannot reside in a Shared File System (SFS) file pool.

Spool File Characteristics

Note: When you enter a SPOOL command for a unit record device, you can refer to it by its virtual address, as well as by its generic device type (for example, CP SPOOL E HOLD).

CLASS (CL)

Spool files, in the CP spool file queue, are grouped according to class, and all files of a particular class can be processed together, or directed to the same real output device. The default values for your virtual machine are set in your VM/SP directory entry, and are probably the standard classes for your installation.

You may need, however, to change the class of a device if you want a particular type of output, or some special handling for a spool file. For example, if you are printing an output file that requires special forms, and your installation expects that output to be spooled class Y, enter the command:

```
spool printer class y
```

All subsequent printed output directed to your printer at virtual address 00E (all CMS output) is processed as class Y.

HOLD

If you place a HOLD on your printer or punch, any files that you print or punch are not released to the control program's spooling queue until you specifically alter the hold status. By placing your output spool files in a hold status, you can select which files you print or punch, and you can purge duplicate or unwanted files. To place printer and punch output files in a hold status, enter the commands:

```
spool printer hold  
spool punch hold
```

When you have placed a hold status on printer or punch files and you produce an output file for one of these devices, CP sends you a message to remind you that you have placed the file in a hold:

```
PRT FILE xxxx FOR userid COPY xx HOLD
```

If, however, you entered the command

```
set msg off
```

then, you do not receive the message.

When you place a reader file in a hold status, then the file remains in the reader until you remove the hold status and read it, or you purge it.

COPY

If you want multiple copies of a spool file, use the COPY operand of the SPOOL command:

```
spool printer copy 10
```

If you enter this command, then all subsequent printer files that you produce are each printed 10 times, until you change the COPY attribute of your printer.

FOR

You can spool printed or punched output so that it will be distributed to another user ID by using the FOR operand of the SPOOL command. For example, if you enter

```
spool printer for charlie
```

then, all subsequent printer files that you produce have, on the output separator page, the user ID CHARLIE and the distribution code for that user. The spool file is then under the control of that user, and you cannot alter it further.

CONT, NOCONT

You can print or punch separate spool files with the NOCONT option of the CP SPOOL command. You can also combine them into one continuous spool file if you use the CONT operand of the CP SPOOL command. For example, if you enter the following sequence of commands

```
spool punch cont to brown
punch asm1 assemble
punch asm2 assemble
punch asm3 assemble
spool punch nocont
close punch
```

then, the three files ASM1 ASSEMBLE, ASM2 ASSEMBLE, and ASM3 ASSEMBLE, are punched to user BROWN as a single spool file. When user BROWN reads this file onto a minidisk or Shared File System (SFS) directory, however, CMS creates separate files.

You can send multiple files by continuous spooling (using CP SPOOL PUNCH CONT) or by a series of DISK DUMP commands, but these methods are discouraged. As a sender, you are encouraged to do the following:

1. Always use SENDFILE, which resets any continuous spooling options in effect.
2. Do not spool the punch continuous.

Similarly, if the punch is spooled continuous and PUNCH is used to send multiple files, the file is read in as one file with “:READ” cards imbedded. In this case, although no files are overlaid, the recipient must divide the file into individual files. This problem can also be avoided by using SENDFILE or by not spooling the punch continuous.

TO

When you spool your printer or punch to another user ID, all output from that device is transferred to the virtual reader of the user ID you specify. When you are punching a CMS file, as in the example above, you should use the TO operand of the SPOOL command to specify the destination of the punch file.

This operand places output in your own virtual reader using the * operand:

```
spool printer to *
```

After you enter this command, subsequent printed output is placed in your virtual reader. You might use this technique as an alternative way of preventing a printer file from printing, or, if you choose to read the file onto your minidisk or SFS directory from your reader, of creating a file from printer output.

Similarly, if you are creating punched output in a program and you want to examine the output during testing, you could enter

```
spool punch to *
```

so that you do not punch any real cards or transfer a virtual punch file to another user.

Altering Spool Files

After you have requested that VM/SP print or punch a file, or after you have received a file in your virtual reader and before the file is actually printed, punched, or read, you can alter some of its characteristics, change its destination, or delete it altogether.

Every spool file in the VM/SP system has a unique four-digit number from 1 to 9900 assigned to it, called a spoolid. You can use the spoolid of a file to identify it when you want to do something to it. You can also change a group of files, by specifying that all files of a particular class be altered in some way, or you can manipulate all of your spool files for a certain device at the same time.

The CP commands that let you manipulate spool files are **CHANGE**, **ORDER**, **PURGE**, and **TRANSFER**. In addition, use the CP **QUERY** command to list the status and characteristics of spool files associated with your user ID.

When you use any of these commands to reference spool files of a particular device, you have the choice of referring to the files by class or by spoolid. You can also specify **ALL**. For example, if you enter the command

```
query printer all
```

you might see the display:

ORIGINID	FILE	CLASS	RECORDS	CPY	HOLD	DATE	TIME	NAME	TYPE	DIST
CMSUG	0142	K PRT	000178	002	USER	04/17	07:58:48	SCHED	SCRIPT	BIN706
CMSUG	0180	1 PRT	002021	001	NONE	04/17	08:02:26	TESTFILE	SCRIPT	BIN706

Until any of these files are processed, or in the case of files in the hold status, until they are released, you can change the spool file name and spool file type (this information appears on the first page or first card of output), the distribution code, the number of copies, the class, or the hold status, using the CP **CHANGE** command. For example

```
change printer all nohold
```

changes all printer files that are in a hold status to a nohold status. The CP **CHANGE** command can also change the spooling class, distribution code, and so on.

If you decide that you do not want to print a particular printer file, delete it with the CP **PURGE** command:

```
purge printer 7615
```

After you have punched a file to some other user, you cannot change its characteristics or delete it unless you restore it to your own virtual reader. Do this with the **TRANSFER** command:

```
transfer all from usera
```

This command returns to your virtual reader all punch files that you spooled to **USERA**'s virtual reader.

You can determine, for your reader or printer files, in what order they should be read or printed. If you enter the command:

order printer 8195 6547

Then, the file with spoolid of 8195 is printed before the file with a spoolid of 6547.

The CP spooling system is very flexible, and can be a useful tool, if you understand and use it properly. The *VM/SP CP General User Command Reference* contains complete format and operand descriptions for the CP commands used to modify spool files.

There is an alternate way to send files to other users, using SFS. Use COPYFILE to copy the file into one of the other user's directories (this assumes you have write authority to the directory). This way, the other user would not have to RECEIVE the file; it would already be in that user's directory.

Using Your Card Punch and Card Reader in CMS

The CMS RECEIVE command reads files from your virtual reader and places them on file mode A. Files can be placed in the reader in one of three ways:

- Reading real punched cards into the system card reader. A CP ID card tells the CP spooling system which virtual reader is to receive the file.
- Receiving a file sent to your virtual machine from another user. The SENDFILE or NOTE commands can be used to send a file to the virtual reader of another virtual machine.
- Using the SENDFILE command to send a file to your own reader.

Using Real Cards

If you have a deck of punched cards that you want read into your virtual machine reader, you should punch, preceding the deck, a CP ID card. If your user ID is MCGUIRE, then the ID card would be:

```
ID MCGUIRE
```

Then, to read this file onto your CMS file mode A, you can enter the command:

```
receive = prog6 assemble
```

If a file named PROG6 ASSEMBLE already exists, specify the replace option on the RECEIVE command:

```
receive = prog6 assemble (replace
```

If you are reading many files into the real system card reader, and you want to read them in as separate spool files (or you want to spool them to different user IDs), separate the cards and individually read the decks onto your minidisk or SFS directory. The CP system, after reading an ID card, continues reading until it reaches the end of the deck of cards.

Sending Files to Other Users

You can use the SENDFILE command to send a copy of a file to the virtual card reader of another user. To use the SENDFILE command, you specify the name of the file and the user ID. Enter:

```
sendfile prog6 assemble a henry
```

Once you have sent the file, the other user can use the RECEIVE command to place the file on their file mode A.

Using Your Card Punch

Use the CMS PUNCH command to create a punched copy of a CMS file. Once you use the CMS PUNCH command to punch a file, a READ control card is punched to precede the deck, so that the card deck can be identified. If you do not wish to punch a READ control card (also referred to as a header card), you can use the NOHEADER option on the PUNCH command:

```
punch prog8 assemble * (noheader
```

Use the NOHEADER option whenever you punch a file that is not going to be read by the RECEIVE command.

The PUNCH command can only punch records of up to 80 characters in length. If you need to punch a file that has more than 80 characters, you can use the DISK DUMP command:

```
disk dump prog9 data
```

The RECEIVE command can also be used to read a file that has been punched using the DISK DUMP command:

```
receive = prog6 assemble
```

Using the MOVEFILE Command

Use the MOVEFILE command, in conjunction with the FILEDEF command, to place a file in your virtual reader, or to copy a file from your reader to another device. For example:

```
spool punch to *  
filedef output punch  
filedef input disk coffee exec a1  
movefile input output
```

The file COFFEE EXEC A1 is punched to your virtual card punch (in card-image format) and spooled to your own virtual reader.

Creating Files Using Your Punch

Apart from the preceding procedures that transfer whole files with one or two commands, there are other methods that create files using your virtual punch. From a program or an EXEC file, you can punch one line at a time to your virtual punch. Then enter the CLOSE command to close the spool file:

```
close punch
```

Depending on how the punch was spooled (the TO setting), the virtual punch file is either punched or transferred to a virtual reader.

Punching Cards Using I/O Macros

If you write an OS, DOS, or CMS program that produces punched card output, you should make an appropriate file definition. If you are an OS user, you should use the FILEDEF command to define the punch as an output data device; if you are a DOS user, use the ASSGN command. If you are using the CMS PUNCHC macro, the punch is assigned for you. The spooling characteristics of your virtual punch control the destination of the punched output.

Punching Cards from an Exec

In a System Product Interpreter, EXEC 2, or CMS EXEC, use the CMS commands EXECIO, PUNCH and DISK DUMP to punch CMS files.

Handling Tape Files in CMS

There are a variety of tape functions that you perform in CMS, and a number of commands that you can use to control tape operations or to read or write tape files. One of the advantages of placing files on tapes is portability; it is a convenient method of transferring data from one real computing system to another. In CMS, you can use tapes created under other operating systems. There are also two CMS commands, TAPE and DDR, that create tape files in formats unique to CMS, that you can use to back up minidisks or to archive or transfer CMS files.

Under VM/SP, virtual addresses 180 through 187 and 288 through 28F are usually reserved for tape devices. In most cases, you can refer to these tapes in CMS by using the symbolic names TAP0 through TAPF. In any event, before you can use a tape, have it mounted and attached to your virtual machine by the system operator. When the tape is attached, you receive a message. For example, if the operator attaches a tape to your virtual machine at virtual address 181, you receive the message:

```
TAPE 181 ATTACHED
```

Following are the various types of tape files, and the commands and programs used to read or write them.

TAPE Command

The CMS TAPE command creates tape files from CMS files. They are in a special format, and should only be read by the CMS TAPE LOAD command. For examples of TAPE command operands and options, see "Using the CMS TAPE Command" on page 178.

TAPPDS Command

The TAPPDS command creates CMS files from OS or DOS sequential tape files, or from OS partitioned data sets.

TAPEMAC Command

The TAPEMAC command creates CMS MACLIB files from OS macro libraries that were unloaded onto tape with the IEHMOVE utility program.

MOVEFILE Command

The MOVEFILE command copies a sequential tape file onto a minidisk or SFS directory. MOVEFILE can also be used to copy a file from a minidisk or SFS directory onto tape. It moves files from your reader to tape or from tape to your punch.

User Programs

You can write programs that read or write sequential tape files using OS, DOS, or CMS macros.

Access Method Services

Tapes created by the EXPORT function of access method services are read only by using the access method services IMPORT function. Both the IMPORT and EXPORT functions are run in CMS using the AMSERV command. The access method services REPRO function can also be used to copy sequential tape files.

DDR Program

The DDR program, run with the CMS command DDR, dumps the contents of a minidisk onto tape, and should be used to restore such files to a minidisk. Specifying the COMPACT option on the DDR command will put tape output in a compact format that uses less tape space than standard, noncompact format. Tapes in compact format are then used as input to DDR.

Using the CMS TAPE Command

The CMS TAPE command provides a variety of tape handling functions. It lets you selectively dump or load CMS files to and from tapes, as well as to position, rewind, and scan the contents of tapes. Use the TAPE command to save the contents of CMS files, or to transfer them from one VM/SP system to another. The following example shows how to create a CMS tape with three tape files on it, each containing one or more CMS files, and then shows how you, or another user, might use the tape at a later time.

The example is in the form of a terminal session and shows, in the *Terminal Display* column, the commands and responses you might see. System messages and responses are in black type, and user-entered commands are in blue. The *Comments* column provides explanations of the commands and responses.

Terminal Display	Comments
Tape 181 attached	Message indicates that the tape is attached.
listfile * assemble a (exec Ready; cms tape dump TAPE DUMP PROG1 ASSEMBLE A1	Prepare to dump all ASSEMBLE files by using the LISTFILE command EXEC option; then execute the CMS EXEC using TAPE and DUMP as arguments.
Dumping ... PROG1 ASSEMBLE A1 TAPE DUMP PROG2 ASSEMBLE A1 Dumping ... PROG2 ASSEMBLE A1 TAPE DUMP PROG3 ASSEMBLE A1 : : TAPE DUMP PROG9 ASSEMBLE A1 Dumping ... PROG9 ASSEMBLE A1 Ready;	The TAPE command responds to each TAPE DUMP by printing the file identification of the file being dumped.
tape wtm Ready;	The last file, PROG9 ASSEMBLE, is dumped. TAPE command writes a tape mark to indicate an end of file.

Terminal Display	Comments
tape dump mylib maclib a Dumping ... MYLIB MACLIB A1 Ready; tape dump cmslib maclib * Dumping ... CMSLIB MACLIB S2 Ready;	Two macro libraries are dumped, by specifying the file identifiers.
tape wtm Ready;	Another tape mark is written.
tape dump mylib txtlib a Dumping ... MYLIB TXTLIB A1 Ready;	A TEXT library is dumped.
tape wtm 2 Ready;	Two tape marks are written to indicate the end of the tape.
tape rew Ready;	The tape is rewound.
tape scan (eof 4 Scanning ... PROG1 ASSEMBLE A1 PROG2 ASSEMBLE A1 PROG3 ASSEMBLE A1 PROG4 ASSEMBLE A1 PROG5 ASSEMBLE A1 PROG6 ASSEMBLE A1 PROG7 ASSEMBLE A1 PROG8 ASSEMBLE A1 PROG9 ASSEMBLE A1	The tape is scanned to verify that all of the files are on it.
End-of-file or end-of-tape MYLIB MACLIB A1 CMSLIB MACLIB S2 End-of-file or end-of-tape MYLIB TXTLIB A1	Tape mark indication.
End-of-file or end-of-tape End-of-file or end-of-tape Ready;	Two tape marks indicate the end of the tape.
det 181 Tape 181 Detached	The CP DETACH command rewinds and detaches the tape.
Tape 181 Attached	Message indicating the tape is attached.
tape load prog4 assemble	One file is to be read onto a minidisk.
Loading ... PROG4 ASSEMBLE A1 Ready;	The TAPE command displays the name of the file loaded. Any existing file with the same file name and file type is erased.

tape scan	The remainder of the first tape file
Scanning ...	is scanned.
PROG5 ASSEMBLE A1	
PROG6 ASSEMBLE A1	
PROG7 ASSEMBLE A1	
PROG8 ASSEMBLE A1	

End-of-file or end-of-tape	Indication of end of first tape file.
Ready;	

tape scan	The second tape file is scanned.
Scanning ...	
MYLIB MACLIB A1	
CMSLIB MACLIB S2	
End-of-file or end-of-tape	
Ready;	

Terminal Display	Comments
tape bsf 2 Ready;	The tape is backed up and positioned in front of the last tape file.
tape fsf Ready;	The tape is forward spaced past the tape mark.
tape load (eof 2 Loading ... MYLIB MACLIB A1 CMSLIB MACLIB A2 End-of-file or end-of-tape MYLIB TXTLIB A1 End-of-file or end-of-tape Ready;	The next two tape files are going to be read.
detach 181 Tape 181 detached	The tape is detached.

Tape Labels in CMS

Support in the CMS component of VM/SP to process labeled tapes includes the following features:

- Checks IBM standard labels on input
- Writes IBM standard labels on output
- Lets you specify routines to process standard user labels during DOS and OS macro simulation under CMS
- Lets you specify exits for processing tapes with nonstandard labels during execution of CMS macro simulations and some CMS tape operation commands
CMS processes all tape labels; CP does not process tape labels.

Limitations

CMS tape label processing does not include:

- Label processing for tapes that are read backwards
- Processing of multivolume files on tapes, except under OS simulation for standard labels. With the DISP MOD option, processing is only valid for the tape currently mounted.
- Support for ANSI tapes or ASCII labels
- Label processing for any functions of the CMS TAPE command except the two functions DVOL1 and WVOL1 that process VOL1 labels.

User Responsibilities

You must initiate all your own tape label processing. To specify that you have a labeled tape, use the FILEDEF command for an OS simulation program, or use a DOS DTFMT macro for a CMS/DOS program. The TAPESL macro also processes standard HDR1 and EOF1 labels and the CMS TAPE command writes and displays standard VOL1 labels. You can provide IBM standard label description details with the LABELDEF command for all types of label processing. After label processing has been requested, it automatically occurs and there is no interaction between you and CMS unless an error occurs. See "Error Processing" on page 196 for a discussion of error processing.

Label Processing in OS Simulation

If you are running an OS simulation program and using OPEN and CLOSE macros, specify the type of label processing you want in a FILEDEF command for a given file. Detailed information about the FILEDEF command is found in the *VM/SP CMS Command Reference*.

You can specify that you want standard label processing (with SL) or nonstandard label processing (with NSL). If you choose nonstandard label processing, you must already have written a routine to process nonstandard labels. The name of this routine must be specified by the file name in the NSL parameter on FILEDEF. An example of nonstandard label processing is given in "Nonstandard Label (NSL) Processing" on page 184.

To be sure that the tape you are using contains no IBM labels, you can specify no label processing (NL) in the FILEDEF command. When NL is specified, CMS does not open files on a tape containing a VOL1 label as its first record.

You can also specify bypass tape label processing (BLP) on a FILEDEF command. BLP tells CMS to bypass tape label processing for a file, and instead, to position the tape at a particular file before processing the data records in the file. If you specify LABOFF for a FILEDEF tape file, label processing is turned off and there is no tape positioning or label checking.

LABOFF is the default, so you do not receive any processing or tape positioning for a tape file unless you specifically request it. If you specify BLP, NL, SL, or SUL processing but omit a positional parameter, the position defaults to 1 and the tape is positioned at the first file. Examples of NL, BLP, and LABOFF processing are given in "Nonstandard Label (NSL) Processing" on page 184, "Bypass Label (BLP) Processing" on page 184, and "Label Off (LABOFF) Processing" on page 184.

IBM Standard Tape Label Processing

For IBM standard labels, you specify, SL or SUL, and optional positional and VOLID parameters. On a FILEDEF command, SUL means standard user labels. Everything you do for SL files, you must also do for SUL files. The positional parameter for standard label files works the same way it does in OS/VS. If you specify

```
filedef filex tap1 sl 2
```

the tape is spaced to what is physically the fourth file on the tape before processing begins. The reason for this spacing is that a standard labeled tape has one header file, one data file, and one trailer file for each data file. If you leave off the positional parameter

```
filedef filey tap3 sul
```

you get the first file on the tape.

The optional VOLID parameter on the FILEDEF command lets you specify the volume serial number in the VOL1 label of a tape in case you want only the VOL1 label checked on the tape. If you want to specify other fields in IBM standard labels, you must also provide a LABELDEF statement for the tape file. The LABELDEF statement lets you assign values to all fields in a standard HDR1 or EOF1 label. A complete description of how the LABELDEF command works can be found in the section "LABELDEF Command" on page 192.

The following command defines FILEZ as a standard labeled tape file on a tape with a VOL1 label and a volume serial number of DEPT78:

```
filedef filez tap1 sl volid dept78
```

If you also wish to specify a data set identifier for FILEZ, you must furnish a LABELDEF command for FILEZ as well as the FILEDEF command. Data set name may not be specified on the FILEDEF command. The following LABELDEF statement assigns a data set name of payroll to FILEZ.

```
labeldef filez fid payroll
```

You can also specify file sequence number, volume sequence number, expiration date and other fields on a LABELDEF command. However, if you are using OS simulation macros (OPEN, CLOSE, READ, WRITE, GET, PUT, and so forth) to process your tape file, the only LABELDEF parameter that has meaning for input files is *fid* (data set identifier). This field and the VOLIDs are checked on input by OS simulation. The other LABELDEF fields specify values to be written in output labels. They are also used by other types of tape label processing (CMS/DOS and CMS) to check input labels. If no LABELDEF command has been supplied for output files, default values are used to write out labels (see the section on the LABELDEF command for the default values).

After you have set up your descriptive information for a standard labeled tape file in FILEDEF and LABELDEF statements, you run a regular OS simulation program under CMS. During program execution, HDR1 and HDR2 labels are written or checked at OPEN time. EOF1 and EOF2 labels are written or checked at CLOSE time. To have EOF labels processed, you must enter a CLOSE macro.

The VOL1 label on a tape is checked whenever a file on that tape is opened if you have specified a VOLID parameter on your FILEDEF statement or LABELDEF statement for the file. If the volume ID is specified on both LABELDEF and FILEDEF, the more recent specification is used. If no volume ID is specified, it is not checked.

After checking the volume ID, the tape is positioned and the HDR label is processed. For processing multifile volumes, you may wish to use the LEAVE option on the FILEDEF command. This option prevents a tape from being rewound and positioned before each tape file is processed. The LEAVE option does not exist on an OS DD statement. To process multivolume files, specify the 'VOLID ?' operand on the LABELDEF command. You will be prompted for the list of serial numbers needed to process the file.

For input files, the EOF2 label is skipped. The information from the HDR2 label is merged with the FCB information and then with the DCB information. This merged information does not overlay existing data. Only the empty fields are filled with the information from the HDR2 label and the FCB. Output HDR2/EOF2 records are written from information in the DCB and the CMSCB (FCBSECT). Note that the tape density and TRTCH fields in HDR2/EOF2 records are taken from what the user specifies in his FILEDEF command for the tape file. They may not correspond to the actual density and TRTCH fields used to write the tape.

When processing standard labeled tapes in OS simulation, the following applies:

1. Multivolume tape processing does not occur if you enter the TEOVEXIT macro.
2. Multivolume tape processing only applies to CMS OS QSAM simulation.
3. During end-of-volume processing, the NOEOV operand of the FILEDEF command is ignored.
4. The VOLSEQ operand of the LABELDEF command is ignored.
5. Existing VOL1 labels are automatically rewritten for density incompatibility. However, VOL2 - VOLn, and user header labels are not rewritten.

To process standard user labels in OS simulation, you must do the following:

1. Specify the file as SUL in a FILEDEF command.
2. Provide a routine to process the user standard labels in your program.
3. Put the address of the user label routine in the DCB EXIT list of the DCB for the file. See *OS/VS1 Data Management Services Guide* or *OS/VS2 MVS Data Management Services Guide*, for instructions on how to establish a DCB EXIT list, and the exact linkage for communication between user label routines and the operating system. This exact linkage should be used under CMS with the following exceptions:
 - a. There is no support for code X'06' EOVS EXIT routine.
 - b. For input labels, return codes 8 and 12 from the user routine are not supported. If an input return code is not 0, it is treated as if it were 4.
4. Note that your standard user label routines do not perform any I/O. They set up an output label for writing, but the CMS tape label processing routines actually write out the label. For input, the CMS label processing routines read in your user standard label but then give control to your routine to check the label.

No Label (NL) Processing

You should specify NL in the FILEDEF command when you expect a tape does not contain any IBM standard tape labels. CMS reads your tape at the time a file is opened and does not open the file if the tape contains a VOL1 label as its first record. If the tape does not contain a VOL1 label, a file is opened and the tape is positioned by using the position parameter (n). For example, if you specify

```
filedef fileq tap1 nl 2
```

FILEQ is not opened if the tape on tap1 (181) has a VOL1 label. If the tape does not have a VOL1 label, FILEQ is opened and the tape is positioned at the second file. If you do not specify a position parameter, the tape is positioned at the first file, (that is, the load point).

Bypass Label (BLP) Processing

You should specify BLP in the FILEDEF command to bypass tape label processing. CMS does not check your tape for an IBM standard tape label. It uses the position parameter you specified to position the tape during open processing. If you do not specify a position parameter, the default is 1. For example

```
filedef fileabc tap1 blp 4
```

positions the tape at the fourth file when it opens FILEABC. Because CMS does not know whether files on the tape are label files or data files, the tape is positioned at what is physically the fourth file, regardless of file content. Any label files on the tape are included in counting files.

Label Off (LABOFF) Processing

You should specify LABOFF in the FILEDEF command if you want no positioning or label processing to occur during open processing. The position parameter is not valid for LABOFF. If you specify LABOFF, and your tape is positioned at record 6 in the third file before you issue an OPEN macro, the tape is positioned at exactly the same record after open processing (record 6 in the third file). The following FILEDEF command does not move tape2 (182) before processing the data in FILEB:

```
filedef fileb tap2 laboff
```

Nonstandard Label (NSL) Processing

To process nonstandard labels, you must write your own routine to read, write, and check the labels. If you have such a routine as a CMS TEXT or MODULE file, you put the file name of the routine after the NSL keyword parameter in the FILEDEF command for the file. The file name must be the name of the first CSECT in the program. It is to this point that control is transferred when the NSL routine gets control. If you do not have a TEXT or MODULE file with the NSL file name you specify, you get an error message. The OPEN and CLOSE routines load your module if it is not already in storage and pass control to it at the time they are opening or closing the file. Your routines are then responsible for processing the tape labels. Nonstandard label routines must do the actual reading and writing of tape labels as well as checking and setting up the label. This is one of several ways nonstandard label processing is different from standard user label processing. Because the CMS label processing routines do not know the size or format of your nonstandard labels, they cannot read or write the labels.

If you use a MODULE file for an NSL routine, it is important that you create the MODULE file so that it starts at an address that will not let it overlay the program or command you are executing at the time the NSL routine is run. The reason for this restriction is that the NSL routine is dynamically loaded while your program is

running. For the TAPEMAC and TAPPDS commands, starting the NSL routine at an address above X'21000' prevents such an overlay. If the NSL routine is run from your own program that is running in the user area, you must determine how big your program is and where the NSL MODULE file should be located to prevent overlay. Note that you do not have to specify a starting address for NSL routines that are TEXT files. The CMS loader loads such files for you at an address that does not cause an overlay.

Although any user can write his own NSL routine, it is expected that a system programmer usually writes such routines and then other programmers in the installation use them. Before writing an NSL routine, see *VM/SP Application Development Guide for CMS* for details on interrupt handling, storage, supervisor calls, and related information. To ensure proper communication with the CMS system routines, you must use the linkage described in the following chart when you write nonstandard label routines.

When an NSL tape label processing routine gets control, register 1 points to a 16-byte parameter list with the following format:

byte 0	Type call	Caller ID	Tape Mode-Set Byte	Reserved	
byte 4	TAPID				ID parameter for TAPEMAC and TAPPDS
byte 8	FCBSECT address				
byte 12	DCB address				

The Type call field is a code telling the type of label processing being done:

- x'00' is OPEN input
- x'04' is OPEN output
- x'08' is CLOSE input
- x'0C' is CLOSE output
- x'10' is End Of Tape output

The Caller ID is a 1-byte code which is one of the following:

- x'80' Call by OS simulation
- x'20' Call by CMS TAPEMAC or TAPPDS commands

Tape modeset byte communicates with the CMS tape I/O routines. It is a 1-byte hexadecimal code that depends on the type of tape (7, 9, or 18 track), tape density, and so forth. For more information on the Mode Set, see the *VM/SP CMS Command Reference* (You probably will pass this byte to the CMS tape controlling module to read and write your tape labels and will never need to know what its codes mean.)

FCBSECT address is the address of the CMSCB (FCBSECT) for the tape file you are processing.

DCB address is the address of the DCB for the tape file you are processing.

Note: For the TAPEMAC and TAPPDS commands, the same interface is used, except that instead of the FCBSECT and DCB address fields, the eight character identifier specified in the ID=identifier field in the command is passed. This identifier lets you identify which file you are processing because the TAPEMAC and TAPPDS commands do not work with CMSCBs or DCBs.

Control is passed to your NSL routine by a BALR 14,15 instruction so register 15 contains the address of your routine when you receive control. Register 14 contains the address you should return to when you are finished processing the nonstandard labels. You can return with a BR 14 instruction. When you receive control, register 13 points to a save area in which to store the callers register. The save area linkage is standard OS/VS linkage. You receive control with a PSW key of X'E' that lets you modify only user storage. When you are finished processing, place a code in register 15 to the CMS label processing routine that called your routine. Place the value 0 (zero) in register 15 if there have been no errors and you want processing to continue normally and the data set to be opened. If you return a nonzero value in register 15, a message is issued to your terminal and the data set is not opened.

If you write the following FILEDEF statement

```
filedef tapf1 tap1 nsl readlab
```

and have a program called READLAB as a MODULE or TEXT file, your program receives control when the data set called *tapf1* is opened. When your program gets control, register 1 contains the address of the parameter list described in the preceding example. Using the data in this parameter list, you can read or write your own tape header labels. When the same data set is closed, your program again receives control and you can read or write your own trailer labels. Your program can test whether it is getting control for OPEN or CLOSE by examining the type call byte in the parameter list passed to you. If the type call byte is X'10', your NSL routine is run while you are writing an output data set and you have reached the reflective mark that indicates end of tape. You may wish to do special processing in this case. See "End-of-Volume and End-of-Tape Processing" on page 193 for more information on end of tape processing.

Differences Between Tape Label Processing Under OS/VS and OS Simulation in CMS

There are a some differences in the way CMS OS simulation processes tapes and the way OS/VS processes them:

- If you are using OS/VS and you do not specify any label parameter on your JCL statement, the default is SL or standard labels. When you use OS simulation under CMS and do not specify any label information on a FILEDEF statement, the default is LABOFF. LABOFF turns off label processing and nothing is done to position the tape or process labels. Thus, if you specify no label information on FILEDEF, the system processes your tape files exactly the same way they are processed on a CMS system that has no tape label processing facilities.
- You must specify CLOSE to process all trailer labels. No automatic CLOSE occurs at end of data or after reading a tape mark. There is no EOVS monitor to process labels before a data set is closed. If an input tape is positioned at an EOF1 or EOVS1 record when CLOSE is entered, the label is processed. If a tape file is closed before all data records are read, the trailer label is not processed. Output tapes have EOF records written only at CLOSE time.
- There is no deferred label processing under OS simulation in CMS.
- When you have not specified a block count routine in your DCB EXIT list under OS/VS, the program abends when a block count error occurs. Under CMS, this condition produces a message that asks whether or not to abend the operation.
- Certain fields in HDR1 and EOF1 labels default to values different from those under OS/VS. These values can always be specified in a LABELDEF command if you do not like the default values. For example, the default for data set name

in an output label under OS simulation is DDNAME and not DSNAME. The default data set sequence number is always one even when the data set is not the first data set on the tape. The default volume sequence number is always one. Read "LABELDEF Command" on page 192 to learn what the default values are under CMS. You can find what default values are in OS/VS by reading *OS/VS Tape Labels*.

Note: You can always get exactly what you want written on a tape label by explicitly specifying the field on a LABELDEF command. For example, you can specify DSNAME as FID on such a command and have it written in the label instead of DDNAME.

- Default volume IDs (when you do not specify a volume ID in a LABELDEF or FILEDEF statement) in output HDR1 and EOF1 records under CMS will be CMS001 and will not be the actual volume serial in the VOL1 record already on the tape, unless you are processing SL tapes, in which case it will be the actual volume serial already on the tape. You should always specify the volume ID in FILEDEF or LABELDEF to be sure the information written is correct.
- Expiration date specification is always done in absolute form rather than by retention period. You must always use the form yyddd where yy is the year (0-99) and ddd the day (0-366). CMS does not handle expiration dates specified by retention periods.
- When CMS reads a HDR1 label and finds an unexpired file, it always issues a message letting you enter ERROR or IGNORE. ERROR prevents opening the file in OS simulation. When the DISP MOD option of the FILEDEF command is specified for SL tapes, IGNORE lets you have the tape positioned at the end of the file, ready to add new records. Otherwise, IGNORE causes the existing record to be overwritten.
- The NSL routine linkage is quite different under CMS than in OS/VS. (See "Nonstandard Label (NSL) Processing" on page 184 for more information.)
- Volume serial number verification occurs every time a file on a tape is opened under OS simulation unless the FILEDEF LEAVE option is used for multifile tapes.
- Existing VOL1 labels are automatically rewritten for density incompatibility in CMS as they are in OS/VS. However, VOL2 - VOLn and user header labels are not rewritten.
- The information from the HDR2 label is checked and merged with the FCB information and then with the DCB information. The merged information does not overlay existing data. Only the empty fields are filled with the information from the HDR2 label or the FCB.
- To maintain OS compatibility in the EOV2/EOF2 label, you must specify LRECL in the output FILEDEF.
- Multivolume tape processing only applies to CMS OS QSAM simulation.
- Blank tapes used for output in CMS cause the tape to run off the reel if you define the tape file as SL or NL. The tape label processing routines try to read an existing VOL1 or HDR1 label before writing on the tape. Therefore, you should always use the CMS TAPE command to write at least one tape mark (for NL tapes) or a VOL1 label (for SL or SUL tapes) before using the tape to write an output data set.
- If you specify a position parameter that is too big (that is, there are not that many files on the tape), the tape runs off the reel in CMS.

- There are no user exits for user standard labels for EOVL label processing in CMS.
- CMS does not support user return codes of 8 and 12 for input standard user labels. If the return code from a user routine is not zero after input label processing, CMS treats it as if the return code was 4. (See *OS/VS1 Data Management Services Guide* or *OS/VS2 MVS Data Management Services Guide* for more information).
- No count is kept of user standard labels read or bypassed in CMS. If more than eight such labels exist, the fact is not detected.
- User label processing routines do not receive control under CMS when an abend or a permanent I/O error occurs.
- If a CMS output tape is not positioned at a HDR1 label or a tape mark when label processing begins, error message 422 is issued. Under OS/VS such conditions cause an abend.
- TCLOSE with the REREAD option causes a tape to be rewound under CMS and then forward spaced one file if the tape has standard labels. Under OS/VS, the tape is backspaced four files and forward spaced one file. REREAD for unlabeled tapes in CMS always causes a rewind.

For more information on OS/VS tape label processing, see *OS/VS1 Data Management Services Guide*, *OS/VS2 MVS Data Management Services Guide*, and *OS/VS Tape Labels*.

For details on end-of-tape/end-of-volume processing under CMS, see "End-of-Volume and End-of-Tape Processing" on page 193.

Label Processing in CMS/DOS

You specify the type of label processing you want in CMS/DOS on a DTFMT macro in exactly the same way you specify it when you want to run your program under VSE. See *VM/SP Application Development Guide for CMS* for details on CMS support for the DTFMT macro.

Labeled tapes are only supported if you use the DTFMT macro. There is no support for labeled tapes in CMS/DOS for any other type. If you try to read labeled tapes with a DTFCP or DTFDI macro, input standard IBM header labels are skipped, but no other input labels are processed. Output tapes with standard labels have these labels overwritten with a tape mark. All tape work files are treated as output unlabeled files in CMS/DOS although they are defined by a DTFMT. Tapes used for such files have a tape mark written as the first record when the file is opened.

Unlabeled and Nonstandard Labeled Tapes

You define an unlabeled tape with the DTFMT parameter FILABL=NO. The tape file is processed as having no labels.

You define a nonstandard labeled tape with the DTFMT parameter FILABL=NSTD. You must also provide a routine to process your nonstandard labels in the LABADDR= parameter of the DTFMT. Tape processing in CMS for these files is the same as it is under VSE.

Standard Labeled Tapes

You define a standard label tape with the DTFMT parameter `FILABL=STD`. You also must supply a LABELDEF command to specify label description information. This command replaces the VSE TLBL card and is required for standard label processing under CMS/DOS. See "LABELDEF Command" on page 192.

To connect the LABELDEF command for a file with the DTFMT for the same file, you must use the same name to label your DTFMT as you use for a file name in your LABELDEF command. If you code a DTFMT macro in your program as

```
MT1 DTFMT    ...FILABL=STD
```

you must then supply the following type of LABELDEF command:

```
labeldef mt1 fid yourfile fseq...
```

You can put any description parameters you want on your LABELDEF command but the file name for it must be *mt1* if you coded *MT1* as the label on the DTFMT.

After you have set up your DTFMT and LABELDEF, you execute your CMS/DOS program. HDR1 labels are checked or written when an OPEN macro is run. EOF1 labels are checked or written when a CLOSE macro is run. A VOL1 label volume serial number is checked only if the tape is positioned at load point when the label processing begins and if you have specified a VOLID parameter on a LABELDEF statement for the file. Note, if NOREWIND is not specified in the DTFMT macro for the file, the tape is rewound so it is positioned at load point for label processing.

If you want to process user standard labels as well as standard labels in CMS/DOS, you specify `FILABL=STD` and also supply a LABADDR parameter in the DTFMT for the file. Control is then transferred to your label processing routines after standard labels are processed. The linkage to user standard label routines is exactly the same as in VSE.

Differences Between Tape Label Processing Under VSE and CMS/DOS

There are some differences in the way tapes are processed by CMS/DOS and the way they are processed by VSE:

- The tape error messages are CMS error messages and not VSE error messages. In some cases VSE lets the system operator reply NEWTAP to an error message. The system then waits for the operator to mount a new tape and continues processing with this new tape. Such a reply is never possible under CMS/DOS. In CMS/DOS, you usually can reply IGNORE to ignore a tape label error condition or CANCEL to cancel a job. NEWTAP is never allowed. In a few cases, CMS/DOS allows an IGNORE reply where VSE does not.
- You must specify CLOSE to process all trailer labels. No automatic CLOSE occurs at end of data or after reading a tape mark. If an input tape is positioned at an EOF1 or EOVI record when CLOSE is run, the label is processed. If a tape file is closed before all data records are read, the trailer label is not processed. Output tapes have EOF records written only at CLOSE time. For nonstandard labeled tapes, your own routines do not receive control on input when a tape mark is read. You must enter a CLOSE macro in your EOFADDR routine to have the trailer labels processed.
- Certain fields in HDR1 and EOF1 labels default to values different from those in VSE. For example, the default volume serial number written in a HDR1 label is CMS001 and not the actual volume ID (volid) in the VOL1 label already on the tape. The default file sequence and volume sequence numbers are always one even when the file is not the first file on the tape. You should read

“LABELDEF Command” on page 192 to learn what the default values are in CMS/DOS. You also can read *VSE/AF Tape Labels* to find what the default values are for VSE. If you do not like the default values, you can always specify the exact values you want in label fields in a LABELDEF command.

- Expiration date specification is always done in absolute form rather than by retention period. You must always use the form yyddd where yy is the year (0-99) and the ddd the day (0-366). CMS does not handle expiration dates specified by retention periods.
- VOL1 labels written in the wrong density are not automatically rewritten by CMS/DOS as they are by VSE.
- Blank tapes should not be used for tape files specified as FILABL = STD in CMS/DOS; they will run off the reel. Use the CMS TAPE command to write a VOL1 label or a tape mark on a blank tape before using it for a STD file.
- Not all tape movement and label checking that occurs in VSE occurs under CMS. For example, when opening an output file, a VSE system expects the tape to be positioned at a HDR1 label or a tape mark. It then backspaces the tape to read the last EOF1 label on the tape. If it does not find the label it expects, it displays an error message. This check is not performed by CMS/DOS. If the tape is not positioned at a HDR1 label or a tape mark when output open processing begins, error message 422 is issued.
- After an EOVI label is written (see “End-of-Volume and End-of-Tape Processing” on page 193), the tape is always rewound and unloaded under CMS/DOS. VSE lets a DTFMT parameter control whether or not the tape is rewound.
- User label processing routines do not receive control when an I/O error occurs under CMS/DOS.
- Control is not passed to user standard label routines in CMS/DOS when EOT has been sensed on output and an EOVI label has been written by the system routines.
- Work tapes are not checked for an expiration date when they contain standard labels under CMS/DOS. If a tape is to be opened as a work tape, CMS/DOS tests to see if it contains a VOL1 label. If it does, a dummy HDR1 label and a tape mark are immediately written on the tape after the VOL1 label. If the tape does not contain a VOL1 label, a tape mark is written at the beginning of the tape. VSE checks expiration dates on previously labeled tapes used as work tapes and gives the operator a chance to reject the tapes if the expiration date has not expired.

For more information on VSE and CMS/DOS tape label processing, see the *VSE/AF Tape Labels* and *VSE/AF Macro User's Guide*.

CMS TAPESL Macro

Use the TAPESL macro in CMS programs that do not use OS and DOS simulation features. You can also use the CMS TAPESL macro to process IBM standard HDR1 and EOF1 labels without using DOS or OS OPEN and CLOSE macros. You will probably use TAPESL with the RDTAPE, WRTAPE, and TAPECTL macros.

TAPESL processes only HDR1 and EOF1 labels. It does not perform any functions of opening a tape file other than label checking or writing. The TAPESL macro generates linkage to the CMS tape label processing routine that actually processes

the label. The macro generates a block of data (32 bytes long) to communicate with the tape label processing routines. TAPESL is used both to check and to write tape labels. A LABELDEF command must be entered before running the program that contains this macro. The LABID parameter of the TAPESL macro specifies the name of the LABELDEF to be used. For example, if you use the macro

```
TAPESL HOUT,181,LABID=GOODLAB
```

in your assembler language program, you must supply a LABELDEF command for GOODLAB:

```
labeldef goodlab fid file10 fseq 4 exdte 78235
```

The tape must be correctly positioned (at the label to be checked or at the place where the label is to be written), before you run the macro. TAPECTL can be used to position the tape. TAPESL reads or writes only one tape record unless you specify SPACE= YES for input. Then it spaces the tape to beyond the tape mark that ends the label file. TAPESL reads and checks a tape VOL1 label provided the tape is positioned at load point and you have specified a volume ID in your LABELDEF command.

Tape Label Processing by CMS Commands

There are three types of CMS commands that do some type of tape label processing. They are:

- TAPEMAC and TAPPDS
- TAPE
- MOVEFILE.

TAPEMAC and TAPPDS Commands

TAPEMAC and TAPPDS have operands where you can indicate the type of label processing you want. The tape must be properly positioned (at the data file or label file you want) before you enter the command. The TAPE command can be used for positioning. A separate LABELDEF command is required for these commands if IBM standard label checking is desired. If SL label type is specified without a labeldefid, standard header labels are displayed on the terminal but not checked by the CMS label processing routines. The command

```
tapemac macfile SL (tap2
```

displays any standard labels that exist on your terminal while the series of commands:

```
labeldef maclab fid macro volseq 2 crdte 77102
tapemac macfile sl maclab (tap2
```

runs the CMS tape label processing routines. These routines check to see that your tape has a HDR1 label that has a file identifier of macro, a volume sequence number 2, and a creation date of 77102. VOL1 labels are not checked during label processing by TAPEMAC and TAPPDS unless the tape is positioned at load point and you have specified a volume ID on your LABELDEF command. The DVOL1 function of the TAPE command can be used for volume verification before positioning the tape if the user does not want to start at the first file. These commands process only HDR1 labels; they skip HDR2, UHL, and all trailer labels without processing them.

To process nonstandard tape labels with TAPEMAC and TAPPDS, you use the same interface described in "Nonstandard Label (NSL) Processing" on page 184. The only difference is that instead of putting the CMSCB and DCB addresses in the

parameter list, the ID parameter you placed in the command line is passed to your NSL routine.

```
tappds pdsfile cmsut1 * nsl superck id XYZ12345
```

passes the EBCDIC identifier XYZ12345 to your nonstandard label checking routine called SUPERCK. This identifier may be up to eight characters long and is left justified in bytes 8 to 15 of the parameter list. You can use the identifier to inform your NSL routine of what file you are processing.

Tape Command DVOL1 and WVOL1 Functions

Use the DVOL1 function of the TAPE command to display the VOL1 label of a tape on your terminal. You can use this command to ensure the system operator has mounted the correct tape before you begin processing the tape. If the tape does not have a VOL1 label and you enter the TAPE command, you are informed that the VOL1 label is missing. Do not use TAPE DVOL1 if you have a blank tape. If TAPE DVOL1 is entered and a blank tape is used, CMS will search the entire tape to find the label record; because the tape is void of any records, the tape will run off the end of the reel.

Use the WVOL1 function on the TAPE command to write a VOL1 label on a tape. You can specify a one-to-six character volume ID (volid) through this command and also a one-to-eight character owner field.

MOVEFILE Command

You can use the MOVEFILE command to move labeled tape files if these files are defined as labeled by the FILEDEF command. The MOVEFILE command supports only SL, NSL, BLP, NL, and LABOFF processing. SUL files are processed as SL files and no user exits are taken.

You can also use the MOVEFILE command to display tape labels on your terminal if you want to see what these labels look like. The following sequence displays the VOL1 and first HDR1 labels on *tap4* if the tape has standard labels:

```
filedef in tap4
filedef out term
tape rew (tap4
move in out
```

LABELDEF Command

The LABELDEF command specifies the exact data you want written in certain fields of a HDR1 or EOF1 tape label for output. It can also be used to specify fields in the same labels that you want checked on input. If you do not explicitly specify a field for output, a default value is used. If you do not explicitly specify a field for input, the field is not checked. For example:

```
labeldef abc fid master volseq 1 exdte 77364
```

used for input tells CMS to check the file identifier volume sequence number and expiration date in an input HDR1 label. No other fields in the label are checked. The same specification used for output causes the HDR1 label to have MASTER written in the file identifier field, 1 written in the volume sequence number field and 77364 written in the expiration date field. Default values are written in the HDR1 fields that are not specified.

Default values for HDR1 labels are as follows:

FID	For OS simulation, the DDNAME (Specified on FILEDEF) For CMS/DOS, the DTFMT symbolic name For CMS TAPESL macro, the LABELDEF ID (LABID = labeldefid) parameter
VOLID	CMS001 For OS simulation, the actual VOLID from the tape mounted is used if processing an SL tape file.
VOLSEQ	0001
FSEQ	0001
GENN	Blanks
GENV	Blanks
CRDTE	Current date that label is written
EXDTE	Current date that label is written
SEC	0

The file name on the LABELDEF command connects your label definition to a file defined elsewhere. This is why you specify different data for file name depending on the type of tape label processing you are doing. File name is DDNAME for OS simulation, DTFMT symbolic name for CMS/DOS and labeldefid for TAPESL.

The LABELDEF command takes the place of the VSE TLBL statement for CMS/DOS.

End-of-Volume and End-of-Tape Processing

There is no true end-of-volume support available with CMS tape label processing, except under OS simulation for standard labeled (SL) tapes. Under OS simulation, automatic volume switching and multivolume files are supported, but FEOV instructions are not. The multivolume support has an option to switch to a different tape drive when EOV is encountered. The alternate drive is specified by the ALT option of the FILEDEF command. The following features exist to aid the IBM standard labeled tape user when an end-of-tape on output or an EOV label on input is detected. These are the only ways in which CMS supports EOV processing.

- **Input** - For other than OS simulation processing for standard labeled tapes, when a CLOSE macro is run or when a TAPESL macro processes an input trailer label, a message is issued if the label read is an EOV1 label instead of an EOF1 label. The EOV1 label is then processed exactly as if it were an EOF1 label. You must request that the operator mount a new tape and reopen a file if you want to continue processing the data.

Under OS simulation, if the record after the tape mark is an EOF1 label, the exit specified in the DCB gets control. If the record is an EOV1 label, tape volume switching occurs. If the next tape has already been mounted on an alternate drive, processing continues. Otherwise, the system notifies the operator to mount the next tape. For example:

```
filedef in tap2 sl
filedef out term
labeldef in volid ?
```

When you enter the LABELDEF command, you receive the following response:

DMSLBD441R Enter VALID information:

Now enter the volume IDs that you need followed by the MOVEFILE command. When you have entered all the volume IDs that you need, then enter a null line. If you initially respond with a null line, it is treated as a volume ID of scratch. For example, if you have two volume IDs, hal001 and hal002, you would enter them as follows:

```
hal001 hal002
(null line)
movefile in out
```

If the last label read on tap2 is an EOVI rather than EOF1, you get the following message:

Attempting to change tape volume for DDNAME xxxxxxx
To cancel the tape volume switch, type CANCEL

You can stop the tape volume switch at anytime by typing CANCEL. This notifies the tape operator that you do not want the tape mounted, and terminates the execution of the MOVEFILE command. Otherwise, the system waits for the tape operator to mount the requested tape volume and issues the following messages at five minute intervals until the volume is mounted.

```
Message sent to userid OPERATOR:
Mount tape volume HAL002 on virtual 182 without a write ring;
Request number 1
(five minute interval)
Message sent to userid OPERATOR:
Mount tape volume HAL002 on virtual 182 without a write ring;
Request number 2
(five minute interval)
Message sent to userid OPERATOR:
Mount tape volume HAL002 on virtual 182 without a write ring;
Request number 3
(five minute interval)
Wait time for tape volume switch has almost expired; to
continue waiting, type EXTEND
```

Note: Your system programmer can extend the wait time to longer than five minutes by using the TVSPARMS macro. Refer to *VM/SP Application Development Guide for CMS* for more information.

At this point you can give the tape operator more time to mount the tape by typing 'EXTEND'. The operator receives the tape mount prompts once again, beginning with request number one. Otherwise, the following message is displayed at your terminal:

```

Message sent to userid OPERATOR:
Mount tape volume HAL002 on virtual 182 without a write ring;
Request number 4
(five minute interval; type EXTEND if you need more time)
Wait time for tape volume switch has expired; tape volume
switch for volume HAL002 on virtual 182 canceled

```

The allotted time for the tape volume switch is over and the MOVEFILE command is terminated. If you still want to process the data, you must begin again by reentering the initial FILEDEF and LABELDEF commands.

Notes:

1. If you enter the TEOVEXIT macro, multivolume tape processing will not occur. For a description of the TEOVEXIT macro, see the *VM/SP CMS Diagnosis Reference*.
 2. Your system support personnel can replace the OS simulation multivolume support described previously through user exits. If the multivolume support appears to be different on your system, see your system administrator.
- **Output** - Under CMS/DOS and OS simulation processing only (that is, the processing does not occur for TAPESL or CMS commands), the following limited EOVS processing occurs:
 1. Except under OS simulation, if you specify that you have an IBM standard labeled tape file, a single tape mark is written to end your data. This occurs when end-of-tape is sensed on output while you are using regular access method macros to write the file. The tape mark is written immediately after the record that caused the EOT to be sensed. Following this tape mark, CMS writes an EOVS1 label and a single tape mark. It then rewinds and unloads your tape. A message is issued indicating that an EOVS1 label was written.

Under OS simulation processing for standard labeled tapes, if you are at the end of the file, an EOF1 label, an EOF2 label, and a tape mark are written. If you are at the end of a tape and you need another tape to finish writing, an EOVS1 label, an EOVS2 label, and a tape mark are written at the end of the first tape. Then, the tape is rewound and unloaded, a message is issued indicating that an EOVS label was written, and the operator is notified to mount the next tape needed to continue writing if a tape is not already mounted on an alternate drive specified in the FILEDEF command. (See the preceding example for the messages that are issued.)

If you specified nonstandard labels, instead of writing the EOVS label, an exit to the nonstandard label routine you specified for the file is taken after the end-of-data tape mark is written. For BLP or NL files, only the ending tape mark is written.

2. CMS/DOS jobs are always canceled after an EOT condition is detected on output. To continue processing the tape, you must have a new tape mounted, run the same job over again or run a new job and reopen the file.
3. OS simulation programs that contain a BSAM CHECK macro cause an abend when EOT is detected, with code 001 after an error message. A BSAM program that does not use a CHECK macro has no way of detecting the EOT condition. Such a program continues to try to write on the tape after it is rewound and unloaded. The program enters a wait state rather

than continue running to a normal or abnormal completion. Therefore, you should always include a BSAM CHECK macro after the WRITE macro if you expect your program to reach end-of-tape. OS simulation BSAM users are also responsible for completing processing on a new tape with the same or a new job after an EOT is detected.

4. If you are a CMS/DOS user you always get the automatic output end-of-tape processing described above. However, if you are an OS simulation user and the NOEOV option was specified on your FILEDEF command, it is ignored at the end-of-volume processing. However, the program causes an abend if you use QSAM or include a BSAM CHECK macro after your WRITE macro. Without a CHECK macro, a BSAM program runs the tape off the reel when EOT is sensed and NOEOV is specified.

Notes:

1. If the DISP MOD option of the FILEDEF command is used to indicate tape positioning, positioning will only be valid for the tape currently mounted (in other words, no volume switching will be done).
2. If you indicate positioning via the DISP MOD option of the FILEDEF command and you indicate the file position number by the SL *n* option, *n* cannot exceed the number of files on the tape or the results will be unpredictable.

Error Processing

When the standard label processing routines find errors or discrepancies on tape labels, they send a message to the CMS terminal user who is processing the tape. After an error message is issued, you can ask the system operator to mount a new tape, use the CMS TAPE command to position the tape at a different file, or respecify your label description information. If you are a terminal user and want another tape mounted, you send the system operator a message telling him or her what tape to mount.

Some errors cause program termination and others do not. The effect of tape label processing errors depends on both the type of error and the type of program (that is, CMS/DOS, OS simulation, CMS command, and so forth) that runs the label processing. The following are general guidelines on error handling:

- Messages identifying the error are always issued
- Under OS simulation, tape label errors result in open errors. These errors prevent a tape file from being opened. They do not necessarily end a job. Errors in trailer labels (except block count errors) have no effect on processing.
- In CMS/DOS, the terminal user is generally given two choices: ignore the error or cancel the job. The new-tape option is not allowed.
- The CMS commands TAPEMAC and TAPPDS terminates with a nonzero return code after a tape label error
- Certain error situations such as unexpired files and block count errors for OS simulation let you ignore the error and do not cause open errors. In these cases, you enter your decision at the terminal after you are notified of the error.
- Errors that occur during the loading of an NSL routine cause an abend (code 155 or 15A). A block count abend gives an error code of 500.

In all cases, after an error has been detected and diagnosed, you must decide what to do. You may wish to have a new tape mounted and then reenter the command or you may want to respecify your LABELDEF description if it was incorrect. You can also use the TAPE command to space the tape to a new file if it was incorrectly positioned.

MOVEFILE Command

The MOVEFILE command can copy sequential tape files into CMS files (minidisk files or files that reside in an SFS directory), or sequential CMS files onto tape. It can be particularly useful when you need to copy a file from a tape and you do not know the format of the tape.

To use the MOVEFILE command, you must first define the input and output files using the FILEDEF command. For example, to copy a file from a tape attached to your virtual machine at virtual address 181 to a CMS minidisk, you would enter:

```
filedef input tap1
filedef output disk tape file a
movefile input output
```

This sequence of commands creates a file named TAPE FILE A1. Then use CMS commands to manipulate and examine the contents of the file.

MOVEFILE displays tape labels and/or moves labeled tape files. See "Tape Labels in CMS" on page 180 for more information. This command also copies data from a shared (for example, 3420) to a nonshared tape subsystem. For more information see "Moving Data Between Shared and Nonshared Tape Subsystems" on page 199.

OS Utility Programs

The CMS command TAPPDS can read OS partitioned and sequential data sets from tapes created by, or for, the IEBTPCH, IEBUPDTE, and IEHMOVE utility programs. When you use the TAPPDS command, the OS data set is copied into a CMS file, which can reside on a minidisk or in a Shared File System (SFS) directory, or in the case of partitioned data sets, into multiple CMS files.

IEBTPCH

Sequential or partitioned data sets created by IEBTPCH must be unblocked for CMS to read them. If you have a tape created by this utility, each member (if the data set is partitioned) is preceded with a card that contains "MEMBER = membername." If you read this tape with the command

```
tappds *
```

then, CMS creates a minidisk file from each member, using the membername for the file name and assigning a file type of CMSUT1. If you want to assign a particular file type, for example TEST, you could enter the command as follows:

```
tappds * test
```

If the file you are reading is a sequential data set, you should use the NOPDS option of the TAPPDS command:

```
tappds test file (nopds
```

The preceding command reads a sequential data set and assigns it a file identifier of TEST FILE. If you do not specify a file name or file type, the default file identifier is TAPPDS CMSUT1.

IEBUPDTE

Tapes in control file format created for the IEBUPDTE utility program can be read by CMS. Data sets can be blocked or unblocked, and can be either sequential or partitioned. Because files created for IEBUPDTE contain `./ADD` control cards to signal the addition of members to partitioned data sets, you must use the `COL1` option of the `TAPPDS` command. Also, you must indicate to CMS that the tape is in IEBUPDTE format. For example, to read a partitioned data set, you would enter the command:

```
tappds * test (update coll
```

The CMS minidisk files created are always in fixed, 80-character format.

IEHMOVE

OS unloaded partitioned data sets on tapes created by the IEHMOVE utility program can be read either by the `TAPPDS` command or by the `TAPEMAC` command. The `TAPPDS` command creates an individual CMS file from each member of the PDS.

If the PDS is a macro library, you can use the `TAPEMAC` command to copy it into a CMS `MACLIB`. A `MACLIB`, a CMS macro library, has a special format and is usually only created by using the `CMS MACLIB` command. If you use the `TAPPDS` command, you have to use the `MACLIB` command to create the macro library from individual files containing macro definitions.

Specifying Special Tape Handling Options

For most of the tape handling that you do in CMS, you do not have to be concerned with the density or recording format of the magnetic tapes that you use. There are, however, some instances when it may be important and there are command options that you can use with the `TAPE` command `MODESET` operand and with `ASSGN` and `FILEDEF` command options.

The specific situations and the command options you should use are listed as follows.

- If you are reading or writing a 7-track tape and the density of the tape is either 200 or 556 bpi, specify `DEN 200` or `DEN 556`.
- If you are reading or writing a 7-track tape with a density of 800 bpi, specify `7TRACK`.
- If you are reading or writing a 7-track tape without using the data convert feature, use the `TRTCH` option.
- When using the `ASSGN`, `FILEDEF`, and `TAPE` commands, if you are writing a tape using a 9-track dual density tape drive with the `9TRACK` option specified, and you want the density to be 800 (on an 800/1600 drive) or 1600 (on a 1600/6250 drive), then specify `DEN 800` or `DEN 1600` because the highest available density is the default.
- If you are reading or writing a tape cartridge of the 3480 Magnetic Tape Subsystem, specify the following:

TAPE command 18TRACK and/or DEN 38K

FILEDEF command 18TRACK and/or DEN 38K

RDTAPE macro MODE=(18,38K)

TAPECTL macro MODE=(18,38K)

TAPESL macro MODE=(18,38K)

WRTAPE macro MODE=(18,38K)

Note: When using the DUMP option of the TAPE command with a 3480 device, you can also specify the tape write transfer mode, either buffered or immediate.

TRANSfer BUFFered
 TRANSfer IMMEDIATE

- If you are writing a tape, the default tape block size is 4096 bytes plus a 5-byte header. This format is not compatible with previous VM/370 systems. Therefore, if you want to write a tape compatible with previous VM/370 systems, use the BLKSIZE 800 option of the TAPE command. The TAPE command is described in detail in *VM/SP CMS Command Reference*.

Moving Data Between Shared and Nonshared Tape Subsystems

Virtual tape addresses 180 through 187 are at one virtual control unit and 288 through 28F are at a second. This lets you copy data between a shared (for example, 3420) and nonshared (for example, 3480) tape subsystems. Using the FILEDEF command, assign the first tape drive to one virtual control unit (for example, TAP1) and the other tape drive to the second virtual control unit (for example, TAP8).

For example, these commands copy data from a nonshared device to a shared device.

```
FILEDEF IN TAP0 (18TRACK            *TAP0 is assigned to 180
FILEDEF OUT TAP8 (9TRACK           *TAP8 is assigned to 288
MOVEFILE IN OUT
```

These commands copy data from a shared device to a nonshared device.

```
FILEDEF IN TAPA (18TRACK           *TAPA is assigned to 28A
FILEDEF OUT TAP1 (9TRACK           *TAP1 is assigned to 181
MOVEFILE IN OUT
```



Chapter 10. Communicating with Other Computer Users

CMS commands let you send information (files, messages, and notes) to other computer users and receive information from them. You can collect the necessary information about other computer users with whom you communicate to keep in your *userid NAMES* file. The following CMS commands reference the NAMES file created using the CMS NAMES command:

NAMEFIND	Display/Stack information from a NAMES file.
NOTE	Prepare a <i>note</i> for one or more computer users, to be sent using the SENDFILE command.
RECEIVE	Read into a directory or onto a minidisk, a file or note that is in your virtual reader.
SENDFILE	Send files or notes to one or more users on your system or a system that is attached to yours via Remote Spooling Communications Subsystem (RSCS) by entering the command or by using a menu (display terminal only).
TELL	Send a message to one or more computer users who are logged on to your computer or to one attached to yours through RSCS.

What Is a Names File?

A names file is a collection of information about other users with whom you communicate. Having a names file makes it easier for you to communicate with others because you can assign nicknames to them. An *entry* in a names file contains all of the information associated with a particular nickname that you enter on one menu. You can also create an entry for a list of names, where the nickname would refer to the whole list.

When you create nicknames, do not use any of the special characters defined for your virtual machine. Otherwise, unpredictable results can occur for commands using the nickname. For example, your system can be set to use # for the logical line end symbol (LINEND). Therefore, it is recommended that you do not include the '#' symbol in a nickname.

Following is a list of these special symbols and their default values:

Character	Symbol
Logical character delete (CHARDEL)	@
Logical line end (LINEND)	#
Logical line delete (LINEDEL)	¢
Logical escape character (ESCAPE)	"
Logical tab character (TABCHAR)	

For more information on these special characters, consult the *VM/SP CP General User Command Reference*.

Creating a Names File

The first entry in your names file should be for yourself. The information will be used for note headings, which are discussed later. To display the names screen, enter:

names

When you enter NAMES, your user ID automatically appears in the first line. The following is an entry in the file *ZOOKEEP NAMES*:

```

====> ZOOKEEP NAMES <=====> N A M E S   F I L E   E D I T I N G <====
Fill in the fields and press a PFkey to display and/or change your NAMES file
Nickname: ZOO      Userid: ZOOKEEP  Node: CITYZOO  Notebook:
                Name: Zoo Keeper
                Phone: 555-2229
                Address: City Zoo
                :
                :
                :
                List of Names:
                :
                :
                :

You can enter optional information below. Describe it by giving it a "tag."

Tag:              Value:
Tag:              Value:

1= Help   2= Add   3= Quit   4= Clear   5= Find   6= Change
7= Previous 8= Next  9=       10= Delete 11=      12= Cursor

====> _
Macro-read 1 File

```

Figure 44. Sample NAMES Screen

Note: When you create your NAMES file, the file will, by default, be in the first read/write minidisk or directory found in the search order.

Adding Entries for Remote Users

If you need to communicate with someone on another processor, you can add to your names file a nickname for that user. In addition, you will need to find out the user's *local ID*, the logon ID for your local system. You will need to add this to the section of optional information at the bottom of the names screen. (If the optional information tag section is full, you must XEDIT your names file and manually enter the information.) When you wish to issue a command, send information, or grant authority to this user, the optional information will enable your system to locate him, even though he is not at your location.

For example, suppose ZOOKEEP wanted to add to the names file an entry for an animal that resides at the San Diego Zoo. The user ID at the remote location is PENGUIN, and his node is SANDIEGO. However, locally, the system has identified him as TUXEDO. The names file entry might look like this:

```

====> ZOOKEEP NAMES <=====> N A M E S   F I L E   E D I T I N G <====
Fill in the fields and press a PFkey to display and/or change your NAMES file
Nickname: TOM      Userid: PENGUIN  Node: SANDIEGO  Notebook:
                Name: Tom Tuxedo
                Phone: 555-7310
                Address: San Diego Zoo
                :
                :
                :
                List of Names:
                :
                :
                :

You can enter optional information below. Describe it by giving it a "tag."

Tag: Localid      Value: Tuxedo
Tag:              Value:

1= Help    2= Add    3= Quit    4= Clear    5= Find    6= Change
7= Previous 8= Next   9=         10= Delete  11=        12= Cursor

====> _
Macro-read 1 File

```

Figure 45. NAMES Screen for a Remote User

Once this entry is part of the ZOOKEEP NAMES file, any command that will accept a nickname will correctly locate the user at the remote location.

If, for example, you did not specify a local ID for PENGUIN at SANDIEGO, and you tried to use the GRANT AUTHORITY command with his nickname, you would receive the following message:

```
DMSJNL647E Localid not specified for PENGUIN at SANDIEGO in ZOOKEEP NAMES file
```

Entering a List of Names

The list of names is something like a distribution list. If you send notes, files, or messages to groups of people, you can create an entry in your names file for each group. In this case, the nickname represents the name that you want to call this list. You can specify the names of the people in the list in the following ways:

- As a nickname of an entry in the names file
- As a user ID of a user who shares your computer
- In the form *userid AT node*.

Each time you send a note, a file, or a message to the nickname specified, it goes to everyone on this list.

You might find such a list particularly useful if your files are stored in the Shared File System and you choose to grant or revoke authority on your files or directories. For example, it is much easier to grant authority to all the members of Department 63G with one command, rather than issuing a separate command to grant authority to each of the ten department members.

The following menu shows an entry for a list of names. Each name in the list is the nickname of an entry in the names file.

```
====> ZOOKEEP NAMES <=====> N A M E S FILE EDITING <====  
Fill in the fields and press a PFkey to display and/or change your NAMES file  
Nickname: ANIMALS Userid: Node: Notebook:  
Name:  
Phone:  
Address:  
:  
:  
:  
List of Names: BEAR LION MONKEY  
:  
:  
:  
  
You can enter optional information below. Describe it by giving it a "tag."  
  
Tag: Value:  
Tag: Value:  
  
1= Help 2= Add 3= Quit 4= Clear 5= Find 6= Change  
7= Previous 8= Next 9= 10= Delete 11= 12= Cursor  
  
====> _  
  
Macro-read 1 File
```

Figure 46. Sample Entry for a List of Names

Entering Chained Lists of Names

You can use chained Lists of Names to allow many users to be included in the List of Names tag. For example, there is an entry in the ZOOKEEP NAMES file called BIRDS containing a List of Names as shown in Figure 47 on page 205.

```

====> ZOOKEEP NAMES <=====> N A M E S  FILE EDITING <====
Fill in the fields and press a PFkey to display and/or change your NAMES file
Nickname: BIRDS   Userid:           Node:           Notebook:
      Name:
      Phone:
      Address:
      :
      :
      :
      List of Names:  OWL SWAN TURKEY
      :
      :
      :

You can enter optional information below. Describe it by giving it a "tag."

Tag:           Value:
Tag:           Value:

1= Help   2= Add   3= Quit   4= Clear   5= Find   6= Change
7= Previous 8= Next   9=       10= Delete 11=       12= Cursor

====> _

```

Macro-read 1 File

Figure 47. Another Sample Entry for a List of Names

Figure 48 shows how BIRDS and ANIMALS can be represented by two nicknames. Each name in the list is the nickname of an entry in the names file.


```

====> ZOOKEEP NAMES <=====> N A M E S   F I L E   E D I T I N G <====
Fill in the fields and press a PFkey to display and/or change your NAMES file
Nickname: BOARDERS  Userid:      Node:      Notebook:
      Name:
      Phone:
      Address:
      :
      :
      :
      List of Names: ANIMALS BIRDS
      :
      :
      :
You can enter optional information below. Describe it by giving it a "tag."
Tag:      Value:
Tag:      Value:
1= Help   2= Add   3= Quit   4= Clear   5= Find   6= Change
7= Previous 8= Next  9=       10= Delete 11=      12= Cursor
====> _
Macro-read 1 File
    
```

Figure 48. Sample Entry for a Chained List of Names

If a note is sent to BOARDERS, the following receive the note:

To Nickname	Chained List of Names	Actual Recipients
BOARDERS	ANIMALS	BEAR LION MONKEY OWL SWAN TURKEY
	BIRDS	

The following represents the ZOOKEEP NAMES file:

```

:nick.TOM      :userid.PENGUIN :node.SANDIEGO
                :name.Tom Tuxedo           :phone.555-7310
:nick.ZOO      :userid.ZOOKEEP :node.CITYZOO
                :name.Zoo Keeper           :phone.555-2229
                :addr.City Zoo
:nick.BEAR     :userid.GRIZZLY :node.DEN
                :name.I. M. Grizzley       :phone.555-4567
                :addr.Den;City Zoo
:nick.LION     :userid.COWARD  :node.DEN
                :name.I.M.A. COWARD        :phone.555-4567
                :addr.Lion Den;City Zoo
:nick.MONKEY   :userid.MONKEY  :node.TREE
                :name.T.O.P. Banana       :phone.555-4567
                :addr.Banana Tree;City Zoo
:nick.ANIMALS  :list.BEAR LION MONKEY
:nick.OWL      :userid.OWL    :node.TREE
                :name.I. M. Wise           :phone.555-4567
                :addr.Big Tree;City Zoo
:nick.SWAN     :userid.SWAN   :node.SWANLAKE
                :name.Grace Full           :phone.555-4567
                :addr.Swan Lake;City Zoo
:nick.TURKEY   :userid.TURKEY :node.COTTAGE
                :name.T.TURKEY            :phone.555-4567
                :addr.Turkey Coop;City Zoo
:nick.BIRDS    :list.OWL SWAN TURKEY
:nick.BOARDERS :list.ANIMALS BIRDS

```

Figure 49. Sample 'userid NAMES' File

Sending Messages

You can send messages to one or more users on your computer or on other computers that are connected to yours through the Remote Spooling Communications Subsystem (RSCS) network. The users must be logged on to receive your message. Because the TELL command references your names file, you can use nicknames. For example:

```
tell bear There is honey for dessert!
```

If bear is logged on, he sees the following message on his screen:

```
MSG FROM ZOOKEEP : There is honey for dessert!
```

You can send a message to a list of people when you have a nickname for the list. For example

```
tell animals Good Morning!
```

sends the message to the list of names for the nickname ANIMAL (BEAR, LION, and MONKEY). They must be logged on to receive your message.

You can also use the CP MESSAGE command to send a message to a user or to the primary system operator. The recipient must be logged on to receive your message. Use the CP QUERY user ID command to see if another user is logged on. The CP MESSAGE command does not use your names file. An example of the CP MESSAGE command using the abbreviation (MSG) is

```
msg jonescj we are leaving now.
```

Receiving Messages

During a terminal session, you can receive many kinds of messages from VM/SP, from the system operator, from other users, or from your own programs. You can decide whether or not you want to receive these messages. For example, if you enter the command

```
set msg off
```

you will not receive any messages sent by the TELL command or the CP MESSAGE command; if another virtual machine user tries to send you a message, that user receives the message:

```
userid not receiving; MSG OFF
```

If your virtual machine handles special messages and you do not want to receive special messages at this time, enter:

```
set smsg off
```

You will not receive any special messages sent by the CP SMSG command; if another virtual machine user attempts to do so, that user receives a message:

```
userid not receiving; SMSG OFF
```

Similarly, to prevent warning messages (which usually come from the system operator) from coming to you, enter:

```
set wng off
```

However, you would only do this in cases where you were typing some output at your terminal and did not want the copy ruined.

VM/SP enters error messages whenever you incorrectly enter a command or if a command or program fails. These messages have a long form, consisting of the error message code and number, followed by text describing the error. If you wish to receive only the text portion of messages with severity codes I, E, and W (for informational, error, and warning, respectively), you can enter the command:

```
set emsg text
```

If you want to receive only the message code and number (from which you can locate an explanation of the error in the *VM/SP System Messages and Codes*) book, you enter:

set emsg code

You can also cancel error messages completely:

set emsg off

To restore the EMSG setting to its default, which is the message text, enter:

set emsg text

Some CP commands issue informational messages telling you that CP has performed a particular function. You can prevent the reception of these messages by entering:

set imsg off

or restore the default by entering:

set imsg on

The setting of EMSG applies to CMS commands as well as to CP commands.

You can also control the format of the CMS ready message. If you enter:

set rdymsg msg

you receive only the *Ready;* or shortened form of the ready message after the completion of CMS commands. If you are not receiving error messages (as described earlier) and an error occurs, the return code from the command still appears in parentheses following the *Ready;*.

Full-screen CMS also provides several commands to control whether or not you receive messages from the system, other users, or your programs during your terminal session. In full-screen CMS, the MESSAGE window is set to pop when you receive a message. In addition, the terminal alarm will sound, and the message class indicator will indicate that you have received a message. If you do not want the window to automatically pop, enter the command SET WINDOW MESSAGE NOPOP. After you have entered this command, when you receive a message, the terminal alarm will still sound, and the message class indicator in the lower left corner of your screen will indicate that you have received a message, but the window will not automatically pop. When you are ready to view the message, you can press CMSPF 2 to pop the MESSAGE window.

Note: If you try to pop the MESSAGE window before any messages have been received, the window will not be shown. Because the window is variable in size, it will only be shown when it contains at least one message.

The default settings for the terminal alarm and message class indicator are for the alarm to sound and the indicator to be updated. However, you can change these defaults by using the ALARM/NOALARM and NOTIFY/NONOTIFY options of the ROUTE command. For example, you would use the NOALARM option if you did not want the terminal alarm to sound when you receive a message.

If you set full-screen CMS on, you can take advantage of several choices for receiving and logging messages. By default, messages and warnings are logged for you into the files MESSAGE LOGFILE and WARNING LOGFILE, respectively. If you wish, you can use the SET LOGFILE command to log your CMS output and other information into separate files that you can later XEDIT or print.

You can also use the TYPE/NOTYPE options of the SET VSCREEN command to control how your virtual screens are updated. The TYPE option, which is the default, means that data is moved to the virtual screen when the virtual screen is

updated. If you simply wanted to log messages into the file MESSAGE LOGFILE, without updating the MESSAGE virtual screen or viewing the MESSAGE window, you could specify SET VSCREEN MESSAGE NOTYPE.

For more information on these commands, see the *VM/SP CMS Command Reference*.

Sending Notes and Files

When you have short communication, like a letter, use the NOTE command to prepare a *note* to send to one or more users on your computer or on other computers that are connected to yours through the Remote Spooling Communications Subsystem (RSCS) network. The person to whom you are sending the note need not be logged on. The NOTE command references your *userid NAMES* file, so you can use nicknames when you create your notes.

Composing Notes

Entering NOTE name puts you in XEDIT mode. Enter the INPUT subcommand on the command line and type the body of the note in the space provided. Press the PF2 key to add lines if you need more space. For example

```
note bear
```

results in a screen as in Figure 50 where the body of the note was entered in the space provided.

```
ZOOKEEP NOTE  A0 F 80 Trunc=80 Size=24 Line=12 Col=1 Alt=0
OPTIONS: NOACK LOG SHORT NOTEBOOK ALL

Date: 11 February 1981, 11:04:52 EDT
From: Zoo Keeper          555-2229          ZOOKEEP at CITYZOO
To:  BEAR at DEN

Dear Bear,
  I have some good news.  Someone ordered 500 jars of honey.
  You can have honey for dessert all month if you like.

                                Sincerely,
                                Zoo Keeper

1= Help    2= Add line 3= Quit    4= Tab      5= Send    6= ?
7= Backward 8= Forward 9= =    10= Rgtleft 11= Splitjoin 12= Power input

====> _

                                X E D I T 1 File
```

Figure 50. Sample Note with Short Headings

Sending a Note

To send the note, you can do *one* of the following:

1. Press the PF5 key.
2. Enter on the command line one of the following:
 - `sendfile (note)`
 - `sendfile (note old)`

Note: Use the OLD option when recipients do not have the RECEIVE command available to them so that the file can be DISK LOADED.

3. Enter:

`send`

which is a synonym for SENDFILE (NOTE).

Continuing a Note

If you want to save a note and finish it later, enter the FILE command on the input line. The note will *not* be sent. It is saved as *userid NOTE A0*. To continue the note later on, enter the NOTE command *with no parameters*.

Keeping a Copy of Notes

Each time that you send a note, a copy is kept in a file called ALL NOTEBOOK. Your ALL NOTEBOOK is, by default, kept in the directory or minidisk you have accessed with a file mode of A. A note is saved by appending it to the NOTEBOOK file. Notes are separated by a line of 73 equal signs (=). If you want to keep separate notebooks for certain correspondence, you can:

1. Specify a notebook file name with the NOTE command. For example, to save a note in ANIMALS NOTEBOOK, enter:
`note bear (notebook animals)`
2. Specify a notebook file name on an entry in your *userid NAMES* file
3. Set up a default notebook file name with the DEFAULTS command. Notes will go into this notebook unless you have specified a notebook file name with the NOTE command or if you have specified a notebook file name on the recipient's entry in your names file. For example to set up the default to save notes in ANIMALS NOTEBOOK, enter
`defaults note notebook animals`

See the *VM/SP CMS Command Reference* for more information about the DEFAULTS command.

Sending Files

You can use the SENDFILE command to send files and notes to one or more computer users on your computer or on other computers that are connected to yours through the Remote Spooling Communications Subsystem (RSCS) network. Because SENDFILE is one of the commands that references your names file, you can use nicknames to identify the recipients. The nickname is automatically converted into node and user ID.

If you know the name of the file that you want to send, you can just enter the file identification and nickname following the SENDFILE command. For example

sendfile banana split a to monkey

Otherwise, if you cannot remember the name of the file or if you have many files to send, enter SENDFILE without operands. When you enter the SENDFILE command (or the abbreviation SF), a special screen is displayed.

The following is a sample SENDFILE menu:

```
----- SENDFILE -----  
File(s) to be sent (use * for Filename, Filetype and/or Filemode  
to select from a list of files)  
Enter filename : *  
filetype : data  
filemode : a  
  
Send files to : monkey  
  
Type over 1 for YES or 0 for NO to change the options:  
  
0 Request acknowledgement when the file has been received?  
  
1 Make a log entry when the file has been sent?  
  
1 Display the file name when the file has been sent?  
  
0 This file is actually a list of files to be sent?  
  
1= Help      3= Quit      5= Send      12= Cursor  
  
====> _  
  
Macro-read 1 File
```

Figure 51. Sample SENDFILE Menu

In Figure 51, the sender entered an asterisk for file name, 'data' for file type, and 'a' for file mode.

Note: If you do not specify a file mode, it will default to A. Be sure to specify the file mode of the directory or minidisk where the file is stored, if other than A.

The name of the recipient (MONKEY) is also entered on the screen. When PF5 (or the ENTER key) is pressed, a special FILELIST screen is displayed. The files to be sent can be selected from this screen (see Figure 52 on page 213).

```

ZOOKEEP FILELIST  A0 V 108 Trunc=108 Size=8  Line=1 Col=1 Alt=0
Directory = VMSYSU:yourid.
Cmd Filename Filetype Fm Format Lrec1 Records Blocks  Date      Time
ANIMAL  DATA    A1 V      95    34     2 10/04/88 21:12:04
s BEAR    DATA    A1 V      95    29     2 10/04/88 20:58:07
CAMEL   DATA    A1 V     107   281    10 10/04/88 17:59:00
s LION   DATA    A1 V      92   101     4 10/02/88 15:33:05
MYSTERY DATA    A2 V      75    28     1  9/25/88 12:10:03
s MONKEY DATA    A2 V     120   277    10  9/24/88  9:14:02
SWAN    DATA    A1 V      26     7     1  9/23/88 16:50:06
ZOO     DATA    A1 V      80   489    30  8/26/88 16:05:08

1= Help    2= Refresh 3= Quit    4= Cancel    5= Sort(dir) 6= Sort(size)
7= Backward 8= Forward 9= FL /n 10= Share    11= XED/FILEL 12= Cursor
Type 'S' in front of each file to be sent and press ENTER.
====> -
X E D I T 1 File

```

Figure 52. Sample FILELIST Screen Generated from SENDFILE

To send one or more of these files, you can type a letter 's' in front of the file name of each file you want sent and then press the ENTER key. You can also position the cursor on the line describing the file you want to send, and then press the PF5 key.

Sending One File

To send only one file:

1. Enter:

```
sendfile
```

(or its abbreviation, SF).

2. On the SENDFILE menu, enter the file name, file type and file mode in the spaces provided. If the file mode is A, you can leave file mode blank.
3. Enter the names of the recipient(s). Remember that you can use nicknames.
4. Select the 0 or 1 options.
5. Press either PF5 or the ENTER key to send the file. Pressing:
 - PF5 sends the file and exits from the menu.
 - The ENTER key sends the file and keeps the menu.

Note: If your files are stored in a Shared File System (SFS) file pool and you want another user to be able to see a copy of one of your files, you can use SFS commands to accomplish this, rather than SENDFILE. With SFS commands, you can create an alias to your file for the user within one of their directories, or you can grant them authority to read from or write to your file. (For details on how to do this, refer to Chapter 4, "Using the Shared File System").

If you and the other user want to work on separate copies of the file, simply use the COPYFILE command. But there are some reasons why you would want to use SENDFILE rather than COPYFILE or CREATE ALIAS:

- You are not authorized to put files in the other user's directory.
- The other user is not enrolled in your file pool.
- You want the user to RECEIVE the file and make the choice where the file should be put.

Receiving Notes and Files

After you logon, you might see a message notifying you that you have files in your reader. For example:

```
FILES: 004 RDR, NO PRT, NO PUN
```

During your terminal session if you want to find out if you have files in your virtual reader, enter any of the following commands. The following table shows the command, provides a description, and the system response if there are no files in your reader.

Command	Description	System Response
RDRLIST	Displays information about the files in your virtual reader and lets you issue commands from the list.	No files in your reader. Ready(00028);
CP QUERY RDR ALL	Lists your reader files (if any) and their characteristics.	NO RDR FILES Ready;
CP QUERY FILES	Displays the number of spool files in your virtual machine.	FILES: NO RDR, NO PRT, NO PUN Ready;

Using the RDRLIST Command

The command RDRLIST displays information about the files in your reader. See Figure 53 on page 215.

```

OHARA  RDRLIST  A1 V 108 Trunc=108 Size=8 Line=1 Col=1 Alt=1
Cmd  Filename Filetype Class User At Node Hold Records Date Time
PIZZA TOPPINGS PUN A KEN  NODE04 NONE 10 10/06 10:39:38
COOKIE ASSEMBLE PUN A KEN  NODE04 NONE 10 10/06 10:25:11
$JELLY NOTE PRT A KEN  NODE04 NONE 7 10/06 10:15:50
DIETING TIPS PUN A KEN  NODE04 NONE 11 10/06 09:40:28
KEN NOTE PUN A KEN  NODE04 NONE 10 10/06 08:43:07
SEND EXEC PUN A BOB  NODE02 NONE 2 10/06 07:12:35
GOOD DAY PUN A GEOFF  NODE02 NONE 29 10/05 11:44:34
Acknowl edgment PUN A BOB  NODE02 NONE 2 10/05 11:42:21

1=Help 2=Refresh 3=Quit 4=Sort(type) 5=Sort(date) 6=Sort(user)
7=Backward 8=Forward 9=Receive 10= 11=Peek 12=Cursor

====>
X E D I T 1 File

```

Figure 53. Sample RDRLIST Screen

Some of the commands that you can enter from the list are:

- PEEK** Displays a file in your virtual reader without reading it to your directory or minidisk.
- RECEIVE** Reads to your directory or minidisk a file or note that is in your virtual reader.
- DISCARD** Purges a file displayed in the reader list.

Receiving a File

If you have entered the RDRLIST command and you want to receive a file:

1. Move the cursor to the line describing the file that you want to receive.
2. Press PF9. A notice appears on that line, telling you that the file has been received.

For an example of the RDRLIST screen once a file has been received, see Figure 54 on page 216.

```

OHARA   RDRLIST   A1 V 108 Trunc=108 Size=4 Line=1 Col=1 Alt=1
Cmd  Filename Filetype Class User At Node Hold Records Date Time
      PIZZA   TOPPINGS PUN A KEN   NODE04 NONE      10 10/06 10:39:38
*     COOKIE  ASSEMBLE received from Ken at NODE04
      $JELLY  NOTE     PRT A KEN   NODE04 NONE      7 10/06 10:15:50
      DIETING TIPS     PUN A KEN   NODE04 NONE     11 10/06 09:40:28

1=Help      2=Refresh  3=Quit      4=Sort(type) 5=Sort(date) 6=Sort(user)
7=Backward  8=Forward  9=Receive  10=          11=Peek     12=Cursor

====> _
                                         X E D I T 1 File

```

Figure 54. Sample RDRLIST Screen after Receiving a File

Note: When you receive files from another user, the files will, by default, be placed on file mode A (the directory or minidisk you have accessed with a file mode of A). If your files are stored in the Shared File System, you may later wish to copy or relocate the new file to another directory. Refer to Chapter 4, “Using the Shared File System” for details.

If the file in your reader has the same name as a file that is already in your directory or on your minidisk, after you press PF9, you receive the following message on your RDRLIST screen:

```
File fn ft fm already exists; specify REPLACE option
```

If you want to replace the file on your directory or minidisk, enter the following in the *Cmd* space next to the file that you want to receive:

```
receive / (replace
```

Clear the remainder of the line, and press the ENTER key. The file will be replaced.

Note: If you have any authorities granted on the file you are replacing, these authorities are not lost when you use RECEIVE with the REPLACE option. However, if you were to erase the original file, and RECEIVE the new version, all authorities granted on the file would be lost, even though the file name is the same.

If you want to keep the file on your directory or minidisk, you can either:

- Rename the original file (use the CMS RENAME command), or
- Give the file to be read in a new name. For example, enter the following in the *Cmd* space:

```
receive / banana split
```

Receiving a Note

You can receive notes in the same way that you receive other types of files. A note has a file type of NOTE. Just move the cursor to the line describing the note and press PF9.

The note is appended to your ALL NOTEBOOK file, unless you have a notebook specified in your names file entry for that person. For example, after receiving \$JELLY NOTE, the sample screen would look like the following:

```
OHARA  RDRLIST  A1  V 108  Trunc=108  Size=4  Line=1  Col=1  Alt=1
Cmd  Filename  Filetype  Class  User  At  Node  Hold  Records  Date  Time
      PIZZA    TOPPINGS  PUN  A  KEN   NODE04  NONE    10  10/06  10:39:38
*     COOKIE   ASSEMBLE  received from Ken at NODE04
*     $JELLY   NOTE     added to ALL NOTEBOOK A0
      DIETING  TIPS     PUN  A  KEN   NODE04  NONE    11  10/06  09:40:28

1=Help      2=Refresh   3=Quit      4=Sort(type) 5=Sort(date) 6=Sort(user)
7=Backward  8=Forward   9=Receive  10=          11=Peek     12=Cursor

====> _
X E D I T  1 File
```

Figure 55. Sample RDRLIST Screen after Receiving a Note

Discarding a File

Use the DISCARD command to purge a file displayed in your RDRLIST file. Unlike the CP PURGE command, DISCARD lets an acknowledgment be sent to the sender (if one was requested). The acknowledgment indicates that the file was discarded. DISCARD also makes an entry in your *userid NETLOG* file, (if the log option was in effect in the RECEIVE command) indicating that the file was discarded. To discard a file, you can enter the command DISCARD on the *Cmd* space next to the file that you want to discard.

Chapter 11. Looking at VM/SP Through Windows

Windowing lets you manage several pieces of information on the physical screen at the same time. Through windows, you can manipulate information as you might rearrange pieces of paper on your desk top.

Windowing support primarily consists of the following:

- Windows
- Virtual Screens.

Working with windows has several advantages. When you set full-screen CMS on, special windows are automatically defined for you, such as the MESSAGE window. If, for example, you receive a message while you are editing a file, you can look at it through the MESSAGE window without leaving the file you are editing. Or, you can hold the message until you have time to look at it later.

This chapter explains windowing support. It is divided into two sections: “What are Windows and Virtual Screens?” and “Using Full-Screen CMS.”

“What are Windows and Virtual Screens?” defines windows and virtual screens. “Using Full-Screen CMS” discusses how CMS can have functions similar to XEDIT such as special PF keys and full-screen input for CMS commands.

What Are Windows and Virtual Screens?

A window is an area on the physical screen where virtual screen data can be displayed and manipulated. A window lets you see what is in a virtual screen.

A virtual screen can be thought of as a *presentation space* where data can be stored. A virtual screen (vscreen for short) simulates a physical screen, but is not confined to the size of the physical screen.

When you are looking at a window, you are actually viewing a virtual screen. Depending on the size of the window and the size of the virtual screen, you may be seeing a portion of the virtual screen or the entire virtual screen. For more information on virtual screens, see the DEFINE VSCREEN command in the *VM/SP CMS Command Reference*.

Because a window shows you a portion of a virtual screen, you can perform several operations against the data in a virtual screen and view the results in the window connected to the virtual screen. The characteristics of virtual screens that you can manipulate include:

- Reserved areas for information such as titles and PF key descriptions
- Color and highlighting
- Options to log data into a file.

The following diagram illustrates the relationship between the physical screen, a window, and a virtual screen.

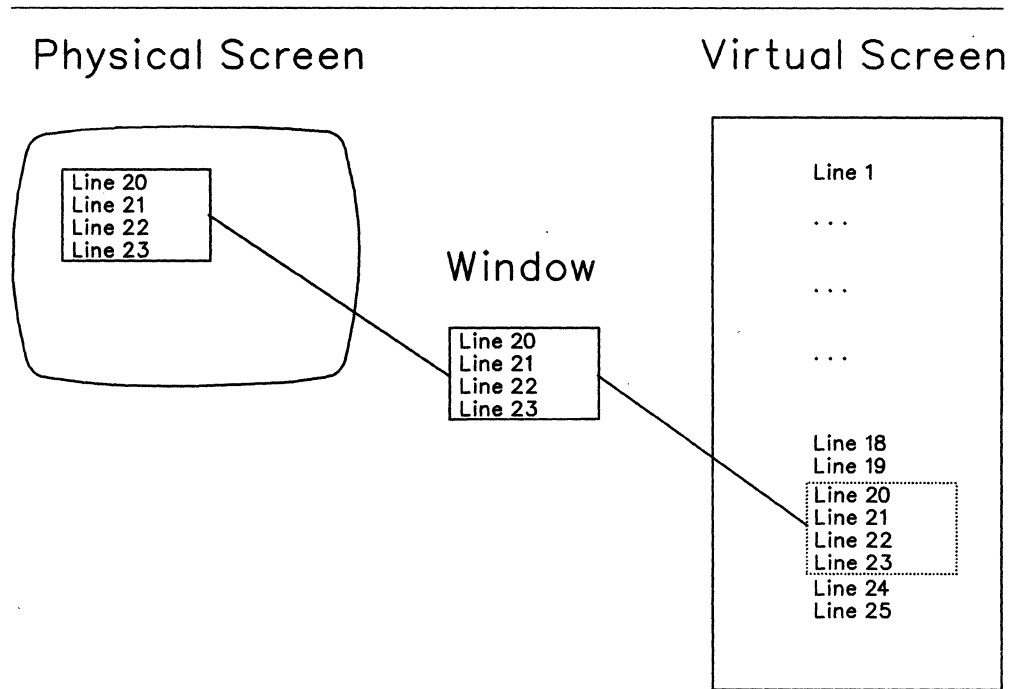


Figure 56. A Window into a Virtual Screen

When working with windows, you do not have to be concerned with the internal interactions between windows and virtual screens. However, as you become more familiar with how they work, you may find it useful to manipulate information by using the CMS commands for windows and virtual screens. Some of these commands will be discussed in this section. For more information on each command, see the *VM/SP CMS Command Reference*.

What Can You Do With a Window?

Windows can be positioned anywhere on the screen as long as the entire window fits on the screen. The maximum size of any window is the size of the screen. You can have many windows on the screen at once. The windows can be displayed on top of each other and can overlap.

When you manipulate data in a window, you are actually manipulating the data in a virtual screen. Virtual screen data can be viewed by scrolling the window over the virtual screen.

Windows are maintained in an ordered list. You can shuffle the order by “popping” and “dropping” windows. The CMS commands to do this are POP WINDOW and DROP WINDOW.

Default windows are those that the system defines for you when you enter SET FULLSCREEN ON. Default windows can have reserved areas at the top and bottom, where system information is shown, and a data area between the reserved areas. In addition, location information may be displayed in the upper right corner of the windows.

The following section shows a sample full-screen CMS window and provides detailed information on the parts of the window.

Using Full-Screen CMS

There are many advantages to using full-screen CMS. When you set full-screen on, you can enter commands from almost anywhere on the physical screen. You can scroll forward and backward through your CMS session to see commands you previously entered and CMS responses to these commands. To reenter any command, you do not need to retype the entire command. Instead, scroll back until the command is displayed on your screen, position your cursor on the command, type over any letter, and press ENTER. The command will be re-executed.

During your full-screen CMS session, messages and other output appear in windows on your physical screen and can be viewed without leaving your current work environment.

After you have logged on to the system, you can enter SET FULLSCREEN ON. Or, you can put this command in your PROFILE EXEC. Once you have entered this command, CMS is in a window and can take advantage of windowing support.

If you wish to leave full-screen CMS, just enter SET FULLSCREEN OFF. You can also enter SET FULLSCREEN SUSPEND or press the CMSPF 3 key to suspend full-screen CMS. The advantage to suspending full-screen CMS is that you can later enter SET FULLSCREEN RESUME and reenter your full-screen session where you left off. None of the default or user-defined settings for windows, virtual screens, or PF keys would be lost.

Now try some examples in full-screen CMS. Note that the examples and screens in this chapter are based on a physical screen size of 24 lines by 80 columns.

Enter SET FULLSCREEN ON from the command line. Your screen now looks like this:

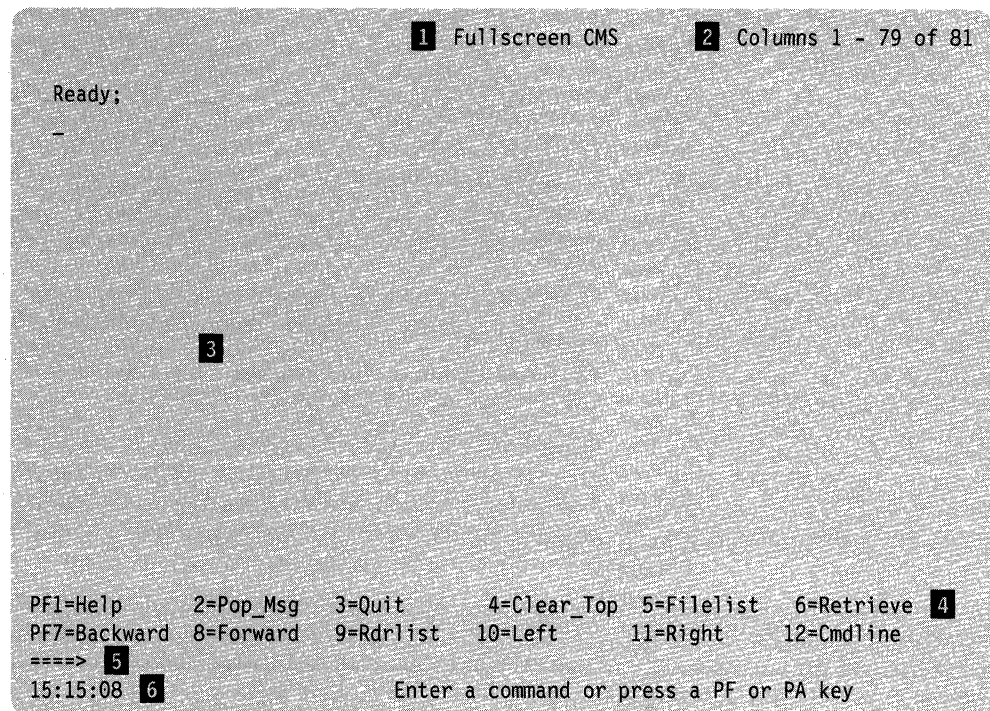


Figure 57. Full-Screen CMS

Before you enter data, look at how your physical screen is organized.

- | | |
|---------------------------------|--|
| 1 Title Line | Identifies full-screen CMS. This is the CMS window. Other windows that have a title line are the MESSAGE, WARNING, and NETWORK windows. |
| 2 Location Information | Shows the location of the window on the virtual screen. It appears in the upper right corner of the window when there may be additional virtual screen data to be displayed. For more information, see "Location Information" on page 224. |
| 3 Data Area | Is the area of the window where CMS output is displayed. You can enter commands anywhere in this area, and you can scroll backwards and forwards through the displayed data. |
| 4 PF Key Definition Area | Displays the CMSPF keys and their functions. Each function is described with a nine-character pseudonym that represents a command. You can change the function and pseudonym assigned to a key. For now, we will use the preassigned or <i>default</i> settings. |
| 5 Command Line | Is the area of the window, in addition to the data area, where commands can be entered. This area is indicated by the large arrow. |
| 6 Status Area | Reflects conditions or states that exist in your virtual machine.

The status area contains the following indicators: |

Clock

Indicates the current time in hours, minutes, and seconds.

Message Class

Indicates that a message was received by the virtual machine. Each time a message is received, its message class appears in this area. One of the following default message classes can be displayed in this area:

- Message
- Warning
- Network.

Execution State

Reflects the status of the session. The next section gives a description of each response that appears in the status area.

Status Information

During a full-screen CMS session, one of the following execution state responses will appear in the status area, which is the lower right corner of the physical screen.

Executing a command

The system is processing your command.

Enter your response in vscreen 'vname'

The system is waiting for your reply to a request.

Note: In this message, 'vname' will be replaced by the name of the virtual screen where you are to enter your response.

Scroll forward for more information in vscreen 'vname'

The system is waiting for you to scroll forward a window that is connected to the specified virtual screen.

Note: In this message, 'vname' will be replaced by the name of a virtual screen.

When you receive this message, it means that new information is waiting to be added to the virtual screen. If you scroll a window connected to the virtual screen (or, in the instance when multiple windows on your screen are showing the virtual screen, if you scroll the window showing the virtual screen that is closest to the top of the ordered list of windows), the virtual screen will be updated with the new information. The virtual screen will also be updated if you enter one of the following commands: CLEAR WINDOW, CLEAR VSCREEN, SHOW WINDOW, or HIDE WINDOW. When a virtual screen is updated, this means that the oldest information in the virtual screen will be deleted from the top and the new information will be added to the bottom. This updating process will occur even if the window connected to the virtual screen is hidden or overlaid by other windows.

Enter a command or press a PF or PA key

The system is waiting to process your next input.

Note: If you SET AUTOREAD ON, "Enter a command or press a PF or PA key" will be replaced by the status notice, "Enter your response in vscreen CMS" following each command.

Whenever there is a conflict between status notices, the highest priority status notice will be displayed. In the previous list, the status notices are listed in order of priority, from highest to lowest.

Location Information

This information is specified in the following ways. You will see this data in the upper right corner of your window when there may be additional virtual screen data to be displayed.

Lines x - y of z

indicates the position of the window in relation to lines of virtual screen data. In other words, it shows which lines you are seeing in relation to how many more you can view.

Note: The CMS virtual screen is filled starting at the top. Lines are sequentially added until the virtual screen is full. Once the virtual screen is full, and you continue to work, older data that you have already scrolled begins to be deleted off the top of the virtual screen; new data is added to the bottom. Lines that you have not yet scrolled are not deleted.

Because old lines are deleted when new ones are added, the lines are renumbered. For this reason, the location information may appear to remain constant as you scroll forward. However, the data will change.

Columns x - y of z

indicates the position of the window in relation to columns of virtual screen data. Depending on the placement of the window, there can be more data to the right or left of your window.

Your Default Windows and Virtual Screens

When you are in full-screen CMS, several windows and virtual screens are automatically available to you. The windows and the virtual screens to which the windows are connected are listed in the following table:

Table 13. Default Windows and Virtual Screens		
Window	Virtual Screen	Description
CMS	CMS	Displays CMS and CP output
CMSOUT	CMS	Displays CMS and CP output while in XEDIT or the productivity aids that use XEDIT (FILELIST, RDRLIST, and so forth)
MESSAGE	MESSAGE	Displays user messages
NETWORK	NETWORK	Displays network messages
STATUS	STATUS	Displays status messages
WARNING	WARNING	Displays warnings
WM	WM	Provides the capability to enter windowing commands

The examples throughout this chapter show only the default windows and virtual screens. You can find more information on the characteristics of these windows and

virtual screens by referring to the tables at the end of Chapter 17, “Customizing Full-Screen CMS.”

The WM Window

The WM window is a special window for window manipulation; that is, for dropping, moving, or changing the size of other windows. You can choose to display the window at any time during your CMS session. There are also certain situations when the WM window is automatically displayed on your screen. Depending on how the WM window was obtained, one of the following three messages will appear in the window:

- Active window overlaid; enter a windowing command or press a PF key
- Output displayed; enter a windowing command or press a PF key
- Enter a windowing command or press a PF key.

The WM window is automatically displayed on your screen in the following situations:

- When the entire screen is protected and the active window is overlaid. For example, this situation could occur if you maximize a window so that it fills the entire screen and covers all other windows. If the window is protected, you would not be able to enter any commands in it. In this case, when the WM window is displayed, you could return the window to the way it was before entering the maximize command by pressing WMPF 12 (which defaults to the command RESTORE WINDOW =) and then drop the WM window by pressing WMPF 3.

When the WM window appears because the active window was overlaid, the window displays the message, “Active window overlaid; enter a windowing command or press a PF key.”

- When you run an application that uses the CONSOLE macro to perform I/O and
 - The CMS virtual screen is updated, or
 - Any virtual screen (other than CMS) is updated and a pop-type window is showing it.

In this instance, you can simply drop the WM window to return to the application.

When the WM window appears while you are running an application using the CONSOLE macro and the previous two conditions are true, the window will contain the message, “Output displayed; enter a windowing command or press a PF key.”

If you wish, you can also pop the WM window at any time during your CMS session. To do this, you would simply enter the command pop window wm or press the PA1 key (which in full-screen CMS defaults to the command pop window wm).

When you pop the WM window, the window will display the message, “Enter a windowing command or press a PF key.”

The WM window provides you with a command line and a set of PF keys so that you can enter commands to manipulate the windows that are covering up your screen. You can enter commands with the WMPF keys, or you can enter windowing commands from the command line in the WM window.

Following is a list of all the commands you can enter from the WM window:

CLEAR WINDOW	PUT SCREEN	SCROLL
CP	QUERY BORDER	SET BORDER
DROP WINDOW	QUERY HIDE	SET LOCATION
HELP	QUERY LOCATION	SET RESERVED
HIDE WINDOW	QUERY RESERVED	SET WINDOW
MAXIMIZE WINDOW	QUERY SHOW	SET WMPF
MINIMIZE WINDOW	QUERY WINDOW	SHOW WINDOW
POP WINDOW	QUERY WMPF	SIZE WINDOW
POSITION WINDOW	RESTORE WINDOW	

As you can see, the WM window is very useful because it provides you with a way to enter commands when you may not have access to the CMS command line or the CMS window.

To drop the WM window, you can press WMPF 3. This returns you to the CMS window.

As an alternative to pressing WMPF 3 to drop the WM window, you can also use the CLEAR key. The CLEAR key scrolls the topmost window forward. When there is no more data to scroll, you will exit from the WM environment.

For more information on the WM window and how to use it, see "Using the WM Window" on page 242.

Special Keys

When you are in full-screen CMS, you can access the CMSPF keys. As we discussed earlier, when the WM window is displayed on your screen, you can use the WMPF keys. The PA1 and PA2 keys also have special settings in full-screen CMS. The PA1 key pops the WM window, and the PA2 key scrolls the top window forward. In addition, the CLEAR key serves the same purpose as the PA2 key.

This section explains how to use the PA and PF keys and the CLEAR key to simplify your work and provides information on the default PF key settings.

The CMSPF Keys

Looking again at your physical screen with full-screen CMS on, let's work with the CMSPF keys.

Each key is given a pseudonym, which represents a command, as shown in the following table:

Table 14. CMSPF Key Settings		
CMSPF Key	Pseudonym	Command
CMSPF 1	Help	ECHO HELP
CMSPF 2	Pop_Msg	NOECHO #WM POP WINDOW MESSAGE *
CMSPF 3	Quit	NOECHO SET FULLSCREEN SUSPEND
CMSPF 4	Clear_Top	NOECHO #WM CLEAR WINDOW =
CMSPF 5	Filelist	ECHO EXEC FILELIST
CMSPF 6	Retrieve	RETRIEVE
CMSPF 7	Backward	NOECHO #WM SCROLL BACKWARD CMS 1
CMSPF 8	Forward	NOECHO #WM SCROLL FORWARD CMS 1
CMSPF 9	Rdrlst	ECHO EXEC RDRLIST
CMSPF 10	Left	NOECHO #WM SCROLL LEFT CMS 10
CMSPF 11	Right	NOECHO #WM SCROLL RIGHT CMS 10
CMSPF 12	Cmdline	NOECHO CURSOR VSCREEN CMS -2 8 (RESERVED

For terminals equipped with 24 PF keys, PF keys 13 through 24 have the same values as PF keys 1 through 12, respectively. If you want to see the complete list of commands assigned to all the CMSPF keys, you can enter:

```
query cmspf *
```

You may have to scroll forward to see all the settings.

The CMS commands assigned to the PF keys appears like this:

```

                                     Fullscreen CMS
                                     Lines 1 - 17 of 27
                                     Columns 1 - 79 of 81

Ready;
query cmspf *
CMSPF 01 Help      ECHO      HELP
CMSPF 02 Pop_Msg  NOECHO    #WM POP WINDOW MESSAGE *
CMSPF 03 Quit      NOECHO    SET FULLSCREEN SUSPEND
CMSPF 04 Clear_Top NOECHO    #WM CLEAR WINDOW =
CMSPF 05 Filelist  ECHO      EXEC FILELIST
CMSPF 06 Retrieve  ECHO      RETRIEVE
CMSPF 07 Backward NOECHO    #WM SCROLL BACKWARD CMS 1
CMSPF 08 Forward  NOECHO    #WM SCROLL FORWARD CMS 1
CMSPF 09 Rdrlist  ECHO      EXEC RDRLIST
CMSPF 10 Left     NOECHO    #WM SCROLL LEFT CMS 10
CMSPF 11 Right    NOECHO    #WM SCROLL RIGHT CMS 10
CMSPF 12 Cmdline  NOECHO    CURSOR VSCREEN CMS -2 8 (RES)
CMSPF 13 Help     ECHO      HELP
CMSPF 14 Pop_Msg  NOECHO    #WM POP WINDOW MESSAGE *
CMSPF 15 Quit     NOECHO    SET FULLSCREEN SUSPEND

PF1=Help      2=Pop_Msg  3=Quit      4=Clear_Top 5=Filelist  6=Retrieve
PF7=Backward  8=Forward  9=Rdrlist   10=Left     11=Right   12=Cmndline
====>
15:32:11
                                     Enter a command or press a PF or PA key

```

Figure 58. Displaying the CMSPF Key Settings

To scroll through your CMSPF key settings in full-screen CMS, you can use your CMSPF keys. CMSPF 7 scrolls backward one window display. CMSPF 8 scrolls forward one window display.

#WM Command

When you displayed the CMSPF key settings in the example above, you may have noticed that several of the CMSPF key settings contain the Immediate command *#WM*. By using *#WM* in the PF key definition, you can set CMSPF keys to perform windowing commands that will be immediately executed in the CMS window. You can set your CMSPF keys without using *#WM*; however, your PF key commands can be executed after other commands that are pending at the time you press the PF key. You can use *#WM* to set PF keys to perform any of the windowing commands listed:

CLEAR WINDOW	PUT SCREEN	RESTORE WINDOW
CP	QUERY BORDER	SCROLL
DROP WINDOW	QUERY HIDE	SET BORDER
HIDE WINDOW	QUERY LOCATION	SET LOCATION
MAXIMIZE WINDOW	QUERY RESERVED	SET RESERVED
MINIMIZE WINDOW	QUERY SHOW	SET WINDOW
POP WINDOW	QUERY WINDOW	SET WMPF
POSITION WINDOW	QUERY WMPF	

You can also enter *#WM* commands from the command line or anywhere else in the CMS window.

Setting a CMSPF Key

You can change the setting of a CMSPF key by entering SET CMSPF followed by the PF key number (represented as nn) and the pseudonym, optional keyword, and associated command.

Let's reset a PF key, PF9, to the TELL command, so whenever you want to send a message, you can press PF9, and then type a user ID or nickname and message. Enter the command:

```
set cmspf 9 Tell DELAYED TELL
```

Your CMSPF 9 key pseudonym at the bottom of the physical screen should show that it is assigned the pseudonym *Tell*. Now press PF9. TELL appears on the command line like this:

```

                                     Fullscreen CMS          Lines 17 - 29 of 29
                                     Columns 1 - 79 of 81

CMSPF 15 Quit      NOECHO  SET FULLSCREEN SUSPEND
CMSPF 16 Clear_Top NOECHO  #WM CLEAR WINDOW =
CMSPF 17 Filelist  ECHO    EXEC FILELIST
CMSPF 18 Retrieve  NOECHO  RETRIEVE
CMSPF 19 Backward NOECHO  #WM SCROLL BACKWARD CMS 1
CMSPF 20 Forward  NOECHO  #WM SCROLL FORWARD CMS 1
CMSPF 21 Rdrlist  ECHO    EXEC RDRLIST
CMSPF 22 Left     NOECHO  #WM SCROLL LEFT CMS 10
CMSPF 23 Right    NOECHO  #WM SCROLL RIGHT CMS 10
CMSPF 24 Cmdline  NOECHO  CURSOR VSCREEN CMS -2 8 (RES
Ready;
set cmspf 9 Tell DELAYED TELL
Ready;

PF1=Help    2=Pop_Msg  3=Quit    4=Clear_Top  5=Filelist  6=Retrieve
PF7=Backward 8=Forward  9=Tell    10=Left     11=Right   12=Cmdline
====> TELL_
15:34:05                                     Enter a command or press a PF or PA key

```

Figure 59. Setting CMSPF 9 to TELL

For complete descriptions of the CMSPF keys, see the SET CMSPF command in the *VM/SP CMS Command Reference*.

PA1 Key

In addition to PF keys, you have a PA1 key on your keyboard that is assigned to POP WINDOW WM. The key can have different labels, depending on your terminal. If you do not have a key labeled PA1, ask your system administrator to show you the equivalent key.

PA1 pops the WM window. As discussed in "The WM Window" on page 225, the WM window provides you with a command line and a set of PF keys for manipulating other windows.

Windows

PA2 Key

The PA2 key works in the CMS window to scroll the top window displayed on your screen forward. PA2 is very useful for controlling windows that are automatically displayed on your screen such as the MESSAGE or WARNING window. When a MESSAGE or WARNING window appears on your screen, simply press PA2 to scroll the data forward. When there is no more data, the window disappears from your screen.

In the WM window, the PA2 key scrolls the data in the topmost window. Once you have scrolled all the data, pressing PA2 will cause you to exit from the WM environment.

Depending on your terminal type, you may not have a key labeled PA2. Again, your system administrator should be able to show you the equivalent key.

CLEAR Key

The CLEAR key performs the same function as the PA2 key. In the CMS window, the CLEAR key scrolls forward the topmost window displayed on your screen. In the WM window, the CLEAR key scrolls the topmost window and exits the WM environment when you have scrolled all the data. In both cases, pressing the CLEAR key causes the entire screen to be rewritten.

Once again, depending on your terminal, you may not have a key labeled CLEAR. Your system administrator should be able to show you the equivalent key.

WMPF Keys

The PF keys in the WM environment are different from the CMS PF key settings. The WM settings are:

WMPF Key	Pseudonym	Command
WMPF 1	Help	NOECHO HELP
WMPF 2	Top	NOECHO SCROLL TOP =
WMPF 3	Quit	NOECHO DROP WINDOW WM
WMPF 4	Clear	NOECHO CLEAR WINDOW =
WMPF 5	Copy	NOECHO PUT SCREEN COPY SCREEN
WMPF 6	Retrieve	RETRIEVE
WMPF 7	Backward	NOECHO SCROLL BACKWARD = 1
WMPF 8	Forward	NOECHO SCROLL FORWARD = 1
WMPF 9	Maximize	NOECHO MAXIMIZE WINDOW =
WMPF 10	Left	NOECHO SCROLL LEFT = 10
WMPF 11	Right	NOECHO SCROLL RIGHT = 10
WMPF 12	Restore	NOECHO RESTORE WINDOW =

Note that for terminals equipped with 24 PF keys, PF keys 13 through 24 have the same values as PF keys 1 through 12, respectively.

Messages in Full-Screen CMS

If you are already familiar with sending and receiving messages on the system, you will find the MESSAGE window helpful in full-screen CMS. Through this window, you can view messages without clearing the physical screen and your work will not be interrupted. (If you are not in full-screen CMS, the screen is cleared when you press ENTER to see a message).

If you have not already created entries in your names file, see Chapter 10, "Communicating with Other Computer Users" and create an entry with a nickname for someone you work with.

Now, send a message to your friend. If you completed the previous exercise, you can press PF9 to display "TELL" at the command line. Complete a message to your friend:

```
TELL debbie Send me a message.
```

If your friend does not respond with a message, try again with someone else in your names file. When you receive a message, you will be notified as follows:

- The terminal alarm sounds
- The status area message class indicator is updated to show that you have received a message.

Now press the ENTER key to see the message in the MESSAGE window. The MESSAGE window will pop.

Note: The MESSAGE window must contain at least one message before the window is displayed.

Dropping and Popping a Window

There are many ways to manage your MESSAGE window. The easiest way is to use the PA2 key. When the MESSAGE window pops, you can press PA2 to scroll the window. When you have seen all the messages in the window, pressing PA2 again causes the window to disappear from your screen.

You can also use the POP WINDOW and DROP WINDOW commands to view the MESSAGE window or remove it from the physical screen.

The MESSAGE window is variable in size; that is, it expands as more messages are received. If you have not received any messages, the window is not displayed. Once you receive a message, the window displays and expands as you receive more messages.

Remove the previous message from your screen by entering:

```
drop window message
```

If you want to redisplay the MESSAGE window, press CMSPF 2 that is assigned to pop the MESSAGE window or enter POP WINDOW MESSAGE.

Working with the Names File

The MESSAGE window is very useful when you are editing a file and need to ask someone for information. To show you how this works, follow this example to add a new entry to your names file. First enter the command:

```
names
```

When the names panel appears on your screen, begin filling in the following information as shown:

```
====> VMUSER NAMES <=====> NAMES FILE EDITING <====>
Fill in the fields and press a PFkey to display and/or change your NAMES file
Nickname: Rori   Userid: _   Node: Sky   Notebook:
                Name: Aurora Borealis
                Phone:
                Address:
                :
                :
                :
                List of Names:
                :
                :
                :

You can enter optional information below. Describe it by giving it a "tag".

Tag:           Value:
Tag:           Value:

1= Help      2= Add      3= Quit      4= Clear    5= Find     6= Change
7= Previous  8= Next      9=          10= Delete  11=         12= Cursor

====>

Macro-read 1 File
```

Figure 60. Adding an Entry to the Names File

Suppose that you suddenly realized you do not know the user ID. Move the cursor to the command line, and send a message to a friend:

```
tell babs What's Rori's user ID?
```

When your friend sends a reply to your message, the terminal alarm sounds. Press ENTER to display the MESSAGE window. The window appears on top of the names file like this:

```

====> VMUSER NAMES <=====> NAMES FILE EDITING <====
Fill in the fields and press a PFkey to display and/or change your NAMES file
Nickname: RORI      Userid:          Node: SKY      Notebook:
                    Name: Aurora Borealis
                    Phone:
                    Address:
                    :
                    :
                    :
+-----+-----+
|                         Messages                         |
| 15:35:16 MSG FROM VMUSER1 : Hi there!                   |
| 15:48:19 MSG FROM VMUSER2 : Rori's userid is BOREAL.    |
+-----+-----+

Tag:          Value:
Tag:          Value:

1= Help      2= Add      3= Quit      4= Clear      5= Find      6= Change
7= Previous  8= Next      9=          10= Delete   11=         12= Cursor

====> _
Macro-read 1 File

```

Figure 61. MESSAGE Window in the Names File

You may notice that other messages you received since you set full-screen on are shown in the MESSAGE window. This will occur if you have not cleared the window by scrolling it forward or by entering the command CLEAR WINDOW MESSAGE. If you have received several messages, you may need to enter the command SCROLL FORWARD to view your most recent message.

Now you can fill in the user ID. When you are finished, drop the MESSAGE window by entering the DROP WINDOW MESSAGE command. You can also use the CLEAR WINDOW MESSAGE command that scrolls the window forward past the current messages and removes the window from your screen. The CLEAR WINDOW MESSAGE command also positions the window so that when a new message is received, the message appears at the top of the window.

Once you have dropped the MESSAGE window, you can then add the entry to your names directory (with PF2) and exit the names file (with PF3).

Reentering Commands

Full-screen CMS provides you with several ways to easily reenter commands you previously entered. You press the C MSPF 6 key, which is set to RETRIEVE. You can also scroll back through your CMS session and edit and reenter commands you entered previously.

Using the RETRIEVE Key

When you first press CMSPF 6, the latest command you entered redisplay on the command line. If you press it again, the previous line displays. If you continue to press CMSPF 6, the commands you previously entered display one at a time.

When the command you wish to reenter is displayed, simply press ENTER to execute the command again.

Entering Commands from the Screen

You have probably noticed that while working in full-screen CMS, each time you enter a command on the command line, the command you entered remains on the physical screen. For example, if you completed the previous exercises, you have the following commands on your screen:

```
Ready;  
set cmspf 9 Tell DELAYED TELL  
Ready;  
tell debbie Send me a message.  
Ready;  
drop window message  
Ready;  
names  
Ready;
```

With full-screen CMS, you can reenter any of these commands by moving your cursor to the place on the screen where the command is written, typing over at least one character, then pressing the ENTER key. You can also change a word or words of a command that you previously entered, then reenter the new command.

For example, move your cursor under the command **DROP WINDOW MESSAGE** that currently appears on your screen. Type **POP** over **DROP** and enter the command:

```
pop window message
```

from the same line on your physical screen.

The MESSAGE window reappears on your screen as follows:

```

                                Fullscreen CMS                Lines 33 - 38 of 38
                                                                Columns 1 - 79 of 81

drop window message
Ready;
names
Ready;
pop window message
Ready;

+-----+
| 15:35:16 MSG FROM VMUSER1 : Hi there!
| 15:48:19 MSG FROM VMUSER2 : Rori's userid is BOREAL.
+-----+

PF1=Help      2=Pop_Msg   3=Quit      4=Clear_Top   5=Filelist   6=Retrieve
PF7=Backward  8=Forward   9=Tell     10=Left      11=Right     12=Cmndline
====>
15:45:02                                Enter a command or press a PF or PA key

```

Figure 62. Popping the MESSAGE Window

To drop the MESSAGE, move your cursor under the command DROP WINDOW MESSAGE. Re-type any letter and press ENTER.

Logging Messages and Other Information

When you enter the command SET FULLSCREEN ON, by default, messages and warnings are logged for you. Messages are logged into a file with the file name and file type of MESSAGE LOGFILE; warnings are logged into WARNING LOGFILE.

To view all the messages you sent or received during your terminal session, you would simply need to XEDIT or print the file MESSAGE LOGFILE. To view warnings you have received, you would XEDIT or print the file WARNING LOGFILE.

If you wish, you can use the SET LOGFILE command to log your CMS output and other information into separate files that you can later XEDIT or print. For details on how to log information, see the SET LOGFILE command in the *VM/SP CMS Command Reference*.

Working with Border Commands

You have already learned enough about full-screen CMS to work with windows. But there is another feature that makes working with windows even easier. Single character commands are typed in the corner of a window border. These commands are called Border commands. You can scroll, move, drop or clear a window by entering a letter in the border corner.

Windows

Borders are optional and are set on or off (see the `DEFINE WINDOW` and `SET BORDER` commands in the *VM/SP CMS Command Reference* for more information). Because borders frame a window, if the window is the same size as the physical screen, or if it is positioned in such a way that the borders do not fit on the physical screen, the borders are not shown. For the following examples, we will use windows with predefined borders that fit within the physical screen.

To try some Border commands, look at the MESSAGE window again. Because the MESSAGE window is variable in size, you will notice that the window size changes as we go through the examples.

First, clear the message virtual screen of all old messages by entering:

```
clear vscreen message
```

Next, send the following two messages to yourself. Type a # (the default line end character) between sentences.

```
tell * Let's see how border commands work.#tell * We'll try some!
```

Now, press the ENTER key (once). Your messages now appear in the MESSAGE window. The corners of the window border are represented by a plus (+) sign.

```

Fullcreen CMS                               Lines 33 - 45 of 45
                                           Columns 1 - 79 of 81

drop window message
Ready:
names
Ready:
pop window message
Ready:
drop window message
+-----+
|               Messages                    |
| 15:51:34 MSG FROM VMUSER : Let's see how border commands work. |
| 15:51:35 MSG FROM VMUSER : We'll try some!                       |
+-----+

PF1=Help      2=Pop_Msg   3=Quit      4=Clear_Top  5=Filelist   6=Retrieve
PF7=Backward  8=Forward   9=Tell     10=Left     11=Right     12=Cmndline
====>
15:51:35 Message                               Enter a command or press a PF or PA key

```

Figure 63. Looking at the Corners of a Window Border

Scrolling Forward and Backward

Now, to scroll the window forward, type the letter *F* in any corner and press the ENTER key:

```
Fullsreen CMS                               Lines 33 - 45 of 45
                                           Columns 1 - 79 of 81

drop window message
Ready;
names
Ready;
pop window message
Ready;
drop window message
+-----+
|                                     |
|                                     |
|          Messages                   |
|                                     |
| 15:51:34 MSG FROM VMUSER : Let's see how border commands work. |
| 15:51:35 MSG FROM VMUSER : We'll try some! |
|                                     |
+-----+
f

PF1=Help      2=Pop_Msg   3=Quit      4=Clear_Top  5=Filelist  6=Retrieve
PF7=Backward  8=Forward   9=Tell     10=Left     11=Right    12=Cmdline
====>
15:52:20 Message                               Enter a command or press a PF or PA key
```

Figure 64. Using a Border Command to Scroll Forward

Notice that the last line displayed becomes the first and only line displayed. The window size also changes because the MESSAGE window is variable in size. The window grows or shrinks depending on how much data there is to display.

The following example shows the result of scrolling forward:

```
FullScreen CMS                               Lines 33 - 45 of 45
                                              Columns 1 - 79 of 81

drop window message
Ready;
names
Ready;
pop window message
Ready;
drop window message
+-----+
|                                     Messages                                     | Lines 2 - 2 of 2 |
|-----+-----+
| 15:51:35 MSG FROM VMUSER : We'll try some.                                |
+-----+-----+
Ready;

PF1=Help      2=Pop_Msg   3=Quit      4=Clear_Top  5=Filelist  6=Retrieve
PF7=Backward  8=Forward   9=Tell     10=Left     11=Right   12=Cmdline
====>
15:53:11

Enter a command or press a PF or PA key
```

Figure 65. Result of Scrolling Forward

Now that you have viewed all the data in the MESSAGE window, if you scrolled forward again, the window would disappear from your screen.

Windows

To scroll the same window backward, enter a *B* in any corner. Assuming you did not receive any new messages, the window now looks like this:

```
Fullscreen CMS                               Lines 33 - 45 of 45
                                           Columns 1 - 79 of 81

drop window message
Ready;
names
Ready;
pop window message
Ready;
drop window message
+-----+
|                                     |
|                               Messages |
|                                     |
| 15:51:34 MSG FROM VMUSER : Let's see how border commands work. |
| 15:51:35 MSG FROM VMUSER : We'll try some! |
|                                     |
+-----+

PF1=Help    2=Pop_Msg  3=Quit    4=Clear_Top  5=Filelist  6=Retrieve
PF7=Backward 8=Forward  9=Tell   10=Left    11=Right   12=Cmdline
====>
15:53:54                               Enter a command or press a PF or PA key
```

Figure 66. Scrolling Backward through a Window Border

Scrolling Right and Left

Next, try scrolling the window to the right and left. Enter an *R* in a corner to move the window to the right. Notice the location information, *Columns 48 - 70 of 70* that appears within the window. This indicates that the data you are viewing in the window represents the right-most portion of the data available for viewing.

The window looks like this:

```

                                Fullscreen CMS
                                Lines 33 - 45 of 45
                                Columns 1 - 79 of 81

drop window message
Ready;
names
Ready;
pop window wm
Ready;
drop window message
+-----+
|                                         Columns 48 - 70 of 70 |
| der commands work.                    |
+-----+

PF1=Help    2=Pop_Msg  3=Quit    4=Clear_Top  5=Filelist  6=Retrieve
PF7=Backward 8=Forward  9=Tell   10=Left     11=Right   12=Cmndline
====>
15:54:28                                     Enter a command or press a PF or PA key
    
```

Figure 67. Scrolling to the Right through a Window Border

Windows

Now, return the window to its previous position. Enter an *L* in a corner. The resulting window now looks like this:

```

                                     Fullscreen CMS
                                     Lines 33 - 45 of 45
                                     Columns 1 - 79 of 81

drop window message
Ready;
names
Ready;
pop window message
Ready;
drop window message
+-----+
|                                     Messages                                     |
|                                                                              |
| 15:51:34 MSG FROM VMUSER : Let's see how border commands work.             |
| 15:51:35 MSG FROM VMUSER : We'll try some!                                |
|                                                                              |
+-----+

PF1=Help      2=Pop_Msg   3=Quit      4=Clear_Top  5=Filelist   6=Retrieve
PF7=Backward  8=Forward   9=Tell     10=Left     11=Right    12=Cmdline
=====>
15:55:35
                                     Enter a command or press a PF or PA key
```

Figure 68. Scrolling to the Left through a Window Border

For more information on using Border commands, see the *VM/SP CMS Command Reference*.

Using the WM Window

If you did not wish to use border commands to manipulate windows, you could, instead, press PA1 to pop the WM window and use the WMPF keys to perform these same functions. The WM window is useful for manipulating the topmost window showing on your screen.

If you have followed the previous exercises, the MESSAGE window is currently showing on your screen. Press PA1 to pop the WM window.

Note: If you receive a message regarding the SET FULLREAD command, disregard the message.

Your window now looks like this:

```

                                Fullscreen CMS                Lines 33 - 45 of 45
                                                                Columns 1 - 79 of 81

drop window message
Ready;
names
Ready;
pop window message
Ready;
drop window message
+-----+
|                                     Messages                                     |
|                                                                              |
| 15:51:34 MSG FROM VMUSER : Let's see how border commands work.           |
| 09:50:53 MSG FROM VMUSER : We'll try some!                               |
+-----+

-----
PF1=Help   2=Top     3=Quit   4=Clear   5=Copy   6=Retrieve
PF7=Backward 8=Forward 9=Maximize 10=Left 11=Right 12=Restore
Enter a windowing command or press a PF key
====> _

```

Figure 69. Popping the WM Window

If you wanted to scroll the MESSAGE window forward, one way to do so would be to use the *F* border command, as we did in the previous exercises. However, you could also use WMPF 8 to scroll the window forward.

Press WMPF 8 to scroll the MESSAGE window forward. Press it again to scroll to the bottom of the window and remove the window from your screen.

Your window now looks like this:

```
Fullscren CMS                               Lines 33 - 45 of 45
                                           Columns 1 - 79 of 81

drop window message
Ready;
names
Ready;
pop window message
Ready;
drop window message
Ready;
clear vscreen message
Ready;
tell * Let's see how border commands work.#tell * We'll try some.
Ready;
Ready;

-----
PF1=Help    2=Top    3=Quit    4=Clear    5=Copy    6=Retrieve
PF7=Backward 8=Forward 9=Maximize 10=Left   11=Right  12=Restore

====> _
```

Figure 70. Dropping the MESSAGE Window

Now, only the WM window remains on your screen. Press WMPF 3 to drop the WM window.

You could also use other WMPF keys to manipulate windows. For example, WMPF 10 performs the same function as the *L* border command we previously used; WMPF 11 performs the same function as *R*.

Now, to show you another unique feature of the WM window, we will purposely create a situation where the window will automatically pop. First, press CMSPF 2 to pop the MESSAGE window.

Your window now looks like this:

```
Fullcreen CMS                               Lines 33 - 46 of 46
                                           Columns 1 - 79 of 81

drop window message
Ready;
names
Ready;
pop window message
Ready;
drop window message
+-----+
|                                     Messages                                     Lines 2 - 2 of 2 |
| 09:50:53 MSG FROM VMUSER : We'll try some! |
+-----+
Ready;
Ready;
-

PF1=Help    2=Pop_Msg  3=Quit    4=Clear_Top  5=Filelist  6=Retrieve
PF7=Backward 8=Forward  9=Tell   10=Left    11=Left    12=Cmdline
====>
16:05:32                               Enter a command or press a PF or PA key
```

Figure 71. Displaying the MESSAGE Window

You will see only the second message you received because in the previous exercise, WMPF 8 scrolled the window. Enter the following command on the command line:

```
set window message fixed
```


Your screen now looks like this:

```

Fullscreen CMS                               Lines 33 - 48 of 48
                                              Columns 1 - 79 of 81

drop window message
Ready;
names
Ready;
pop window message
Ready;
drop window message
+-----+
|                                     Messages                               Lines 2 - 2 of 2 |
|                                                                                               |
| 15:51:35 MSG FROM VMUSER : We'll try some! |
|                                                                                               |
|                                                                                               |
|                                                                                               |
+-----+

PF1=Help    2=Pop_Msg  3=Quit    4=Clear_Top  5=Filelist  6=Retrieve
PF7=Backward 8=Forward 9=Tell    10=Left     11=Right   12=Cmdline
====>
16:07:17                                     Enter a command or press a PF or PA key

```

Figure 72. Changing the MESSAGE Window

Now, enter the border command *X* from any corner of the MESSAGE window. The *X* command maximizes the window; that is, it enlarges the window to let you view more data.

Your screen now looks like this:

```

Messages                                     Lines 2 - 2 of 2
15:51:35 MSG FROM VMUSER : We'll try some!

-----
PF1=Help   2=Top   3=Quit   4=Clear   5=Copy   6=Retrieve
PF7=Backward 8=Forward 9=Maximize 10=Left 11=Right 12=Restore
Active window overlaid; enter a windowing command or press a PF key
====> _

```

Figure 73. WM Window

You will notice that the MESSAGE window is maximized, and the WM window automatically popped. The WM window pops because the maximized MESSAGE window is so large that it is covering up the screen, command line, and PF key settings. The WM window provides you with an area to enter commands to manipulate the window that is covering up your screen.

At this point, you could use the WMPF keys or enter windowing commands from the command line in the WM window to manipulate the maximized MESSAGE window. We will use one of the WMPF keys. Press WMPF 12 to restore the MESSAGE window.

The MESSAGE window now looks like this:

```

                                Fullscreen CMS
                                Lines 33 - 48 of 48
                                Columns 1 - 79 of 81

drop window message
Ready;
names
Ready;
pop window message
Ready;
drop window message
+-----+
|                                     Messages                                     |
|                                     Lines 2 - 2 of 2                             |
|                                     |
| 15:51:35 MSG FROM VMUSER : We'll try some! |
|                                     |
|-----|
PF1=Help   2=Top     3=Quit   4=Clear   5=Copy   6=Retrieve
PF7=Backward 8=Forward 9=Maximize 10=Left  11=Right  12=Restore

====> _

```

Figure 74. Restoring the MESSAGE Window

One way to exit from the WM environment is to press WMPF 3. This removes both the WM and MESSAGE windows from your screen.

Even though you cannot see the MESSAGE window, you can reset it. Enter the command

```
set window message variable
```

to reset the MESSAGE window to its default status.

If you want to leave full-screen CMS, enter the command SET FULLSCREEN OFF. You can also suspend full-screen CMS by entering SET FULLSCREEN SUSPEND or pressing CMSPF 3.

Now that you are familiar with using full-screen CMS and windows to display information, you may wish to refer to Chapter 17, "Customizing Full-Screen CMS" for more information on ways to tailor full-screen CMS and windowing support for your particular needs.

You may also wish to refer to Appendix C, "Considerations for Full-Screen CMS and Windowing." This section contains information you may find useful when you begin using windowing and full-screen CMS.

Chapter 12. Using the CMS Batch Facility

The CMS batch facility provides a way of submitting jobs for batch processing in CMS. You can use the CMS batch facility when:

- You have a job (like an assembly or execution) that takes a lot of time, and you want to be able to use your terminal for other work while the time-consuming job is being run
- You do not have access to a terminal.

The CMS batch facility is really a virtual machine, generated and controlled by the system operator. The operator logs on VM/SP using the batch user ID and initiating the CMSBATCH facility by either:

- Entering the BATCH parameter in the PARM field of the IPL command, or
- Specifying the NOSPROF parameter and entering CMSBATCH at the VM READ.

All jobs submitted for batch processing are spooled to the user ID of this virtual machine, which sequentially executes the jobs. To use the CMS batch facility at your location, you must ask the system operator what the user ID of the batch virtual machine is.

Submitting Jobs to the CMS Batch Facility

Under a real OS or DOS system, jobs submitted in batch mode are controlled by JCL specifications. Batch jobs submitted to the CMS batch facility are controlled by the control cards /JOB, /SET, and /*, and by CMS commands.

Any application or development program written in a language supported by VM/SP can be executed on the batch facility virtual machine. However, there are restrictions on programs using certain CP and CMS commands, as described later in this section.

Input to the Batch Machine

Input records must be in card-image format, and may be punched on real cards, placed in a CMS file with fixed-length, 80-character records, or punched to your virtual punch. These jobs are sent to the batch virtual machine in one of two ways:

- By reading the real punched card input into the system card reader, or
- By spooling your virtual punch to the virtual reader of the batch virtual machine.

When you submit a real card deck to the batch machine, the first card in the deck must be a CP ID card. The ID card takes the form:

ID userid

where:

Batch Facility

ID must begin in card column one and be separated from *userid* (the batch facility virtual machine user ID) by one or more blanks.

For example, if the batch virtual machine for your installation has a user ID of BATCH1, you punch the card

```
ID BATCH1
```

and place it in front of your deck.

When you are going to submit a job using your virtual punch, you must first be sure that your punch is spooled to the virtual reader of the batch virtual machine:

```
spool punch to batch1
```

Batch Considerations for Shared File System (SFS) Files

It is not recommended that you use the CMS batch facility for files that contain Shared File System (SFS) commands or SFS Callable Services Library (CSL) routines. However, if you choose to do so, you should use the following guidelines to ensure that your jobs are properly executed.

When you submit a job to the batch machine, it connects to the SFS server by using the user ID of the batch machine. With the batch machine user ID, the program or application checks the authorization of the files and directories it must access to run your job. Therefore, if you want to submit jobs to refer to SFS files or directories, you must issue commands to authorize the batch machine to access them. The batch machine must also be enrolled in your file pool (or if an ENROLL PUBLIC command was issued, the batch machine can connect).

Even if you grant write authority on a directory to the batch machine, when the directory is accessed, it will be accessed as read-only. The batch machine will not be able to write files to the directory unless it uses COPYFILE or the SFS program function interface.

Warning: You should be aware that by authorizing the batch machine to access files and directories, you may be creating a security problem. Other batch machine users can connect to your file pool and use your files or directories.

Because batch jobs use the user ID of the batch machine, it is important that you explicitly state your user ID in ACCESS commands. If you use a period (.) to refer to your user ID in an SFS directory, your batch job may fail. For example, if you (YOURID) submit a batch job that accesses the PUBS:YOURID.SALES directory with a file mode of B, you must state the command as follows:

```
access pubs:yourid. sales b
```

If, instead, you specify:

```
access pubs: .sales b
```

the batch machine substitutes its own user ID, (for example, batch1) for the user ID. Because the directory will be identified incorrectly, the batch job will fail, or if batch1 .sales b exists, you will get unexpected results.

Therefore, if you are writing an application or exec that may eventually be run on a batch machine, be sure to specify the user ID.

The same problem may occur if you write an application or exec that allows the file pool ID to default. The default will not be the default file pool ID of the submitting

machine. Instead, it will be the default (if one is defined) for the batch machine. Therefore, if you want to allow the file pool ID to default, you should also submit a SET FILEPOOL command with your batch job.

Finally, if you submit a batch job with FILEWAIT on, and the resource is being used, your batch job and all other batch jobs in queue will wait until the required resource is freed. This could conceivably tie up the batch facility for the maximum job limit of up to 32,767 seconds. Therefore, it is suggested that you do not set FILEWAIT on when submitting jobs to the CMS batch facility.

Note: Some batch facilities may be set up to run batch jobs under the user ID of the virtual machine submitting the job. Your system administrator can tell you if the batch facility you are using is set up that way. If it is, the batch machine runs your job under your user ID. For example, if you place an explicit lock on a file or directory, the job running on the batch machine can still use that file. However, if your batch job has a file open for update, any job you are running (from your user ID) will be barred from updating the file until the batch job is finished. Note that the batch facility provided with VM/SP does not run under the user ID of the virtual machine submitting the job, but rather under its own user ID.

Submitting Virtual Card Input to the CMS Batch Facility

Virtual card input can be spooled to the batch machine in several ways. You may create a CMS file that contains the input control cards and use the CMS PUNCH command to punch the virtual cards:

```
punch batch jcl (noheader
```

When you punch a file this way, you must use the NOHEADER option of the PUNCH command, since the CMS batch facility cannot interpret the header card that is usually produced by the PUNCH command. As it does with cards in an invalid format, the batch virtual machine would flush the header card.

You can use an EXEC procedure to submit input to the batch machine. From an EXEC, you can punch one line at a time into your virtual punch, using the EXECIO command. When you do this, you must remember to enter the CP CLOSE command to release the spool punch file when you are finished:

```
close punch
```

If you are using the exec to punch individual lines and entire CMS files to be read by the batch virtual machine as one continuous job stream, you must remember to spool your punch accordingly:

```
/* EXEC to submit a batch job to CMS BATCH */
spool punch cont
execio 1 punch ('string '/JOB MCGUIRE 999888'
punch batch jcl '* ('noheader
spool punch nocont
close punch
```

The /JOB and /* Cards

A /JOB card must precede each job to be executed under the batch facility. It identifies your user ID to the batch virtual machine and provides accounting information for the system. It takes the form:

```
/JOB userid acctnum [jobname] [comments]
```

where:

userid is your user identification, or the user ID under which you want the job submitted. This parameter controls:

- The user ID charged by the CP accounting routines for the system resources used during a job
- The name and distribution code that appear on any spooled printer or punch output. Any spooled output will be spooled under the user ID specified.
- The user ID to whom status messages are sent while the batch machine is executing the job.

Note: Items 1 and 2 are correct only if the directory for the user ID involved contains the accounting option.

acctnum is your account number. This account number appears in the accounting data generated at the end of your job. It overrides the account number in the CP directory entry for the user ID specified for this job.

jobname is an optional parameter that specifies the name of the job being run. If you specify a jobname, it appears as the CP spool file identification in the file type field. The file name field always contains CMSBATCH. See "Batch Facility Output" on page 257 for more information.

comments may be any additional information you want to provide.

The /* card indicates the end of a job to the batch facility. It takes the form:

```
/*
```

The batch facility treats all /* cards after the first as null cards. Therefore, if you want to ensure against the previous job not having a /* end-of-job indicator, you should precede your /JOB card with a /* card.

The /* card is also treated as an end-of-file indicator when a file is being read from the input stream. This is a special technique used in submitting source or data files through the card reader and is discussed under "A Batch Exec for a Non-CMS User" on page 262.

Notes:

1. Both “/JOB” and “/*” must begin in column 1.
2. The /* card can contain only the characters “/*.” No other characters can appear on this input card.

/SET Card

The /SET card sets limits on a system's time, printing, and punching resources during the execution of a job. It takes the form:

/SET [TIME seconds] [PRINT lines] [PUNCH cards]

where:

- seconds** is a decimal value that specifies the maximum number of seconds of elapsed clock time a job can use. This value is added to the current time-of-day clock value. This causes the batch facility to terminate the job, if still executing, when the resultant time-of-day clock value is reached. Enforcement of this time limit requires that the batch machine be running with ECMODE ON. If the batch machine is running with ECMODE OFF, the time limit is ignored and the job continues to run until it normally ends or is terminated by the operator.
- lines** is a decimal value that specifies the maximum number of lines a job can print.
- cards** is a decimal number that specifies the maximum number of cards a job can punch.

The default values for the batch facility are set at 32,767 seconds, printed lines, and punched cards per job. Any new limits defined using the /SET card must be less than these maximum settings. The system resources can be set at lesser values than the default values by an installation's system programmer; be sure you know the maximum installation values for batch resource limits before you use the /SET card.

A /SET card appears anywhere in the job following the /JOB card. The new limits defined by the /SET card apply only to the part of the job that follows the /SET card.

A job can contain up to three /SET cards (one for each operand); a /SET card cannot be entered more than once for the same operand.

Only use /SET cards for the operands whose values you want to change from the default; the default values are reset between jobs. A /SET card for an operand overrides its default but does not reset the other operands.

Notes:

1. If ECMODE is ON, using the STIMER and TTIMER macros will not affect the batch machine time limit. Setting ECMODE to OFF causes an internal timer to be used. In either case, BLIP processing is non-operational; no BLIP function is performed.
2. “/SET” must begin in column 1.

Other Input Records

The remainder of input records in the batch job consist of CP and CMS commands that are entered. (For a description of command restrictions, refer to “Restrictions on CP and CMS Commands in Batch Jobs” on page 255). EXEC, EXEC 2, or System Product Interpreter statements cannot be imbedded in the input stream.

How the Batch Facility Works

The CMS batch facility, once initialized, runs continuously. When it begins executing a job, it sends a message to the user ID of the user submitting the job. If you are logged on when the batch machine begins executing a job that you sent it, you receive the message

```
MSG FROM BATCHID: JOB 'yourjob' STARTED
```

When the batch machine finishes processing a job, it sends the message:

```
MSG FROM BATCHID: JOB 'yourjob' ENDED
```

where *yourjob* is the jobname you specified on the /JOB card. Before it reads the next job from its card reader, the batch virtual machine:

- Closes all spooling devices and releases spool files
- Resets any spooling devices identified by the CP TAG command
- Detaches any minidisk devices that were accessed
- Punches accounting information to the system
- Reloads CMS.

All of this *housekeeping* is done by the CMS batch facility so that each job that is executed is unaffected by any previous jobs.

If a job that you sent to the batch virtual machine abnormally terminates (abends), the batch machine sends you a message

```
MSG FROM BATCHID: JOB 'yourjob' ABEND
```

and spools a CP storage dump of your virtual machine to the printer. The batch virtual machine stops processing your job and leaves the job in its reader in hold status.

Whenever the batch virtual machine has read and executed all of the jobs in its reader, it waits for more input.

Preparing Jobs for Batch Execution

When you want to submit a job to the CMS batch facility for execution, you should provide the same CMS and CP commands you would use to prepare to execute the same job in your own virtual machine.

You must provide the batch virtual machine with read access to any input files that are required for the job. You do this by supplying the LINK and ACCESS commands necessary. The batch virtual machine has a file mode A (the minidisk at

virtual address 195), so you can enter commands to access your minidisks or SFS directories as read-only extensions. For example, if you wanted the batch machine to execute a program module named LONDON on your 291 minidisk, your input file might contain the following:

```
/JOB FISH 012345
CP LINK MCGUIRE 291 291 RR SECRET
ACCESS 291 B/A
LONDON
```

Similarly, if you are using the batch virtual machine to execute a program using input and output files, you must supply the file definitions:

```
CP LINK ARDEN 391 391 RR FOREST
ACCESS 391 B/A
FILEDEF INFILE DISK VITAL STAT B
FILEDEF OUTFILE PUNCH
CP SPOOL PUNCH TO MCGUIRE
LONDON
```

If you expect printed or punched output from your job, you may need to include the spooling commands necessary to control the output. In the previous example, the punch of the batch machine is spooled to user ID MCGUIRE's virtual reader.

Any output printer files produced by your job are spooled by the batch virtual machine to the printer. *These files are spooled under your user ID and with the distribution code associated with your user ID, provided the user ID's directory has the accounting option set.* You can change the characteristics of these output files with the CP SPOOL command:

```
cp spool e class t
```

If you want output to appear under a name other than your user ID, use the FOR operand of the SPOOL command:

```
cp spool e for jonson
```

Output punch files are spooled, by default, to the real system card punch (under your user ID), unless you enter a SPOOL command in the batch job to control the virtual card punch of the batch virtual machine.

Note: If you are using the batch machine for files stored in SFS directories, you may not want to set FILEWAIT on. If your batch job is accessing several files in different directories, the job could wait too long and hold up other jobs.

Restrictions on CP and CMS Commands in Batch Jobs

The batch facility permits many CP and most CMS commands. The following CP commands are used to control the batch virtual machine:

CHANGE	MSG
CLOSE	QUERY
DETACH	REWIND
DUMP	SMSG
DISPLAY	SPOOL
LINK	STORE
LOADVFCB	TAG

Following are some restrictions on the use of CP and CMS commands in batch jobs:

- The **CHANGE**, **CLOSE**, and **SPOOL** commands cannot be used to affect the virtual reader.
- You cannot use the detach command to detach any spooling devices or the system or IPL disks.
- The **LINK** command must be entered on one line in the format:
CP LINK userid vaddr vaddr mode password
None of the **LINK** command keywords (**AS**, **PASS**, **TO**) are accepted. If the minidisk has no password associated with it, you must enter the password as **ALL**. A maximum of 26 links can be in effect at any one time.
- If a **DIAGNOSE** code X'08' is issued, the **CHANGE** and **SPOOL** commands will have an effect on the virtual card reader.
- All CP commands in a batch job must be prefaced with the **CP** command.
- If you are running a **CMS**, **EXEC 2**, or **System Product Interpreter EXEC** in the batch environment, the return code may be different than if the same exec is run standalone. **BATCH** intercepts some commands and checks their validity for the **CMSBATCH** environment. The return code may be from **BATCH** and not from **CP**.
- Because the batch virtual machine reads input from its reader, you cannot use the following commands or operands that affect the reader:
ASSGN SYSxxx READER (CMS/DOS only)
DISK LOAD
FILEDEF READER
READCARD
RECEIVE
- The **RDCARD** macro cannot be used by jobs that run under the **CMS** batch machine.
- Invalid **SET** command operands are:
BLIP PROTECT
IMPCP REDTYPE
INPUT RELPAGE
OUTPUT
- All of the other **CMS SET** command operands can be used in a job executing in the batch virtual machine.
- All forms of the **CP SET** command are invalid.
- The **SET TIMER REAL** command should not be used to get the equivalency of real time (elapsed time) timing. The **STIMER** macro with the **REAL** function should be used. Use of this function requires **ECMODE** to be **ON**. With **ECMODE OFF**, the timer support is ignored.

- Concatenation of commands by the use of the new line character (usually an X'15') will result in an error message.

Batch Facility Output

Any files that you request to have printed during the execution of your job are spooled to the real system printer under your user ID, unless you have spooled it otherwise. Once released for processing, these output files are under the control of the CP spooling facilities; if you are logged on, you can control the disposition of these files before they are printed with the CLOSE, PURGE, ORDER, and CHANGE commands. See "Purging and Reordering Batch Jobs."

Output files produced by the batch virtual machine are identifiable by the file name CMSBATCH in the CP spool file name field. The spool file type field contains the file type JOB, unless you specified a jobname on the /JOB card. This applies to both printer and punch output files.

In addition to your regular printed output, the CMS batch facility spools a console sheet that contains a record of all the lines read in, and the responses, error messages, and return codes that resulted from command or program execution. This file is identified by a spool file name of BATCH and a spool file type of CONSOLE.

Purging and Reordering Batch Jobs

If you are logged on to the batch user ID, you can control the execution of batch virtual machine jobs when required by purging, reordering, and restarting them; by the same token, because all the closed printer files are queued for system output under the submitting user ID, you can change, purge, or reorder these files before processing on the system printer.

To purge a job executing under the batch monitor, use the following procedure:

1. Signal attention and enter the virtual machine environment.
2. Enter the HX (halt execution) Immediate command.
3. Disconnect the virtual machine using the CP DISCONN command.

The HX command causes the batch facility to abnormally terminate. This provides the user with an error message and a CP dump of the batch facility virtual machine. The batch monitor then loads itself again and starts the next job (if any).

To purge an individual input spool file that is not yet executing, enter the CP PURGE command:

```
PURGE READER spoolid
```

In this format, spoolid is the spool file number of the job to be purged from the job queue of the batch virtual machine. For example, the statement

```
purge reader 123
```

purges 123 from the job queue of the batch virtual machine.

To reorder individual spool files in the job queue of the batch facility, enter the CP ORDER command:

```
ORDER READER spoolid1 spoolid2...
```

In this format, spoolid1 and spoolid2 are the assigned spool file identifications of the jobs to be reordered.

Batch Facility

You can determine which jobs are in the queue by entering the CP QUERY command:

```
query reader all
```

This QUERY command lists the file names and file types of all the jobs in the batch virtual machine's job queue. You can then reorder them using the ORDER command.

Using Exec Files for Input to the Batch Facility

There are a variety of ways that exec procedures can help facilitate the submission of jobs to the CMS batch facility. You can prepare an exec file that contains all of the CMS commands you want to execute, and then pass the name of the exec to the batch virtual machine. For example, consider the files COPY JCL and COPYF EXEC:

```
COPY JCL:  /JOB CARBON 999999
           EXEC COPYF
           /*

COPYF EXEC: COPYFILE FIRST FILE A SECOND = =
            COPYFILE THIRD FILE A FOURTH = =
```

Then, if you enter the commands

```
spool punch to cmsbatch
punch copy jcl * (noheader
```

the commands in the exec file are executed by the batch virtual machine.

You could also use an exec to punch input to the batch virtual machine. Using the same commands as in the previous example, you might have an exec named BATCOPY:

```
/* exec to submit a batch job */
'CP SPOOL PUNCH TO BATCH3'
punch = execio 1 punch '('string
punch '/JOB CARBON 999999'
punch 'COPYFILE FIRST FILE A SECOND = ='
punch 'COPYFILE THIRD FILE A FOURTH = ='
punch '/*'
'CP CLOSE PUNCH'
```

Then, when you enter the exec name

```
batcopy
```

the input lines are punched to the batch virtual machine.

The previous examples are very simple; you probably would not go to the trouble of sending such a job to the batch virtual machine for processing. The examples do, however, illustrate the two basic ways that you can use exec procedures with the batch facility:

- Initiating an exec procedure from a batch virtual machine
- Using an exec procedure to create a job stream for the batch virtual machine.

In either case, the execs that you use may be very simple or very complicated. In the first instance, an exec might contain many steps, with control statements to conditionally control execution, error routines, and so on.

In the second instance, you might have an exec that is versatile so that it can be used with different arguments so as to satisfy more than one situation. For example, if you want to create a simple exec to send jobs to the batch virtual machine to be assembled, it might contain:

```

/* An exec for batch assemblies */
'CP SPOOL PUNCH TO BATCH3 CONT'
arg filename .
punch = execio 1 punch '('string
punch '/JOB MCGUIRE 888888'
punch 'CP LINK MCGUIRE 191 391 RR LINKPASS'
punch 'ACCESS 391 B/A'
punch 'ASSEMBLE' filename '(PRINT'
punch 'CP SPOOL PUNCH TO MCGUIRE'
punch 'PUNCH' filename 'TEXT A (NOHEADER'
punch '/*'
'CP SPOOL PUNCH NOCONT CLOSE'

```

If this file were named BATCHASM EXEC, then whenever you wanted the CMS batch facility to assemble a source file for you, you would enter:

```
batchasm filename
```

and the batch virtual machine would assemble the source file, print the listing, and send you a copy of the resulting TEXT file.

Sample System Procedures for Batch Execution

To extend the above example a little further, suppose you wanted to process source files in languages other than the assembler language. You want, also, for any user to be able to use this exec. You might have a separate exec file for each language, and an exec to control the submission of the job. This example shows the controlling exec file BATCH and the ASSEMBLE EXEC.

BATCH EXEC

```
/*
 * This exec submits assemblies/compilations to CMS Batch
 *
 * - Punch batch job card;
 * - Call appropriate language exec to punch executable commands
 */
arg userid filename language
if language = ''
  then do
    say 'Correct form is: BATCH userid fname ftype (language)'
    exit 100
  end
punch = 'EXECIO 1 PUNCH (STRING'
trace errors
signal on error
cp spool d cont to batchcms
punch '/JOB' userid '1111' filename
punch 'CP LINK' userid '191 291 RR SECRET'
punch 'ACCESS 291 B/A'
exec language filename userid
punch '/'
cp spool d nocont
cp close d
cp spool d off
exit
Error: exit 100
```

ASSEMBLE EXEC

```

/*
 *   Correct form is : ASSEMBLE fname userid
 *
 *   Punch commands to:
 *       - Invoke CMS assembler
 *       - Return text deck to caller
 */
arg fname userid
signal on error
trace errors
punch = 'EXECIO 1 PUNCH (STRING'
punch 'GLOBAL MACLIB UPLIB CMSLIB OSMACRO'
punch 'CP MSG' userid 'Asmbling' fname
punch 'ASSEMBLE' fname '(PRINT NOTERM)'
punch 'CP MSG' userid 'Assembly done'
punch 'CP SPOOL D TO' userid 'NOCONT'
punch 'PUNCH' fname 'TEXT A1 (NOHEADER)'
punch 'CP CLOSE D'
punch 'CP SPOOL D OFF'
punch 'RELEASE 291'
punch 'CP DETACH 291'
exit
Error: exit 102

```

Executing the Sample Exec Procedure

If the previous exec procedure is invoked with the line:

```
batch fay payroll assemble
```

the BATCHCMS virtual machine's reader should contain the following statements (in the same general form as a FIFO console stack):

```

/JOB FAY 1111 PAYROLL
CP LINK FAY 191 291 RR SECRET
ACCESS 291 B/B
GLOBAL MACLIB UPLIB CMSLIB OSMACRO
CP MSG FAY Asmbling PAYROLL
ASSEMBLE PAYROLL (PRINT NOTERM)
CP MSG FAY Assembly done
CP SPOOL D TO FAY NOCONT
PUNCH PAYROLL TEXT A1 (NOHEADER)
CP CLOSE D
CP SPOOL D OFF
RELEASE 291
CP DETACH 291
/*

```

When the batch facility executes this job, the commands are executed as you see them: if you are logged on, you receive, in addition to the usual messages that the batch facility issues, those messages that are included in the exec.

A Batch Exec for a Non-CMS User

Many installations run the CMS batch facility for non-CMS users to submit particular types of jobs. Usually, a series of exec files are stored on the system disk so that each user only needs include a card to use the exec, which executes the correct CMS commands to process data included with the job stream.

For example, if a non-CMS user wanted to compile FORTRAN source files, the following BATFORT EXEC file could be stored on the system disk:

```
/* EXEC for batch FORTRAN Compiles */
arg filename
'FILEDEF INMOVE TERM (RECFM F BLOCK 80 LRECL 80'
'FILEDEF OUTMOVE DISK' filename 'FORTRAN A1 (RECFM F LRECL 80 BLOCK 80'
'MOVEFILE'
'GLOBAL TXTLIB FORTRAN'
'FORTGI' filename '(PRINT)'
fortret = rc
if rc = 0 then
  'PUNCH' filename 'TEXT A1 (NOHEADER)'
exit fortret
```

To use this exec, a non-CMS user might place the following real card deck in the system card reader:

```
ID CMSBATCH
/JOB JOEUSER 1234 JOB10
BATFORT JOEFORT
:
source file
:
/* (end-of-file indicator)
/* (end-of-job indicator)
```

When the batch virtual machine executes this job, it begins reading the exec procedure, and executes one line at a time. When it encounters the MOVEFILE command, it begins reading the source file from its card reader (the batch facility interprets a terminal read as a request to read from the card reader). It continues reading until it reaches the end-of-file indicator (the /* card), and then resumes processing the exec on the next line following the MOVEFILE command.

Additional functions may be added to this exec procedure, or others may be written and stored on the system disk to provide, for example, a compile, load, and execute facility. These exec procedures would let an installation accommodate the non-CMS users and maintain common user procedures.

Part 3: Learning to Use Execs

The CMS facilities known as the System Product Interpreter, EXEC 2, and CMS EXEC processors or interpreters let you create exec files. Using exec files, you can execute many commands and programs by entering a single command from your terminal; in effect, this is like writing your own CMS commands.

In this part, the exec facilities are described in general terms to acquaint you with the expressions used in exec files and the basic way that execs function.

Chapter 13: Introduction to the Exec Processors presents a survey of the basic characteristics and functions of exec facilities available to you.

Chapter 14: Creating System Product Interpreter Execs describes how to create and use System Product Interpreter execs. Sample execs are provided for you to try.

Chapter 15: Creating a PROFILE EXEC describes how you can create your own PROFILE EXEC.

Chapter 16: Commands Used with System Product Interpreter Execs provides examples of using some CMS commands with System Product Interpreter execs.



Chapter 13. Introduction to the Exec Processors

Three exec processors are available:

- System Product Interpreter
- EXEC 2
- CMS EXEC.

The System Product Interpreter handles System Product Interpreter programs, which are written in the Restructured Extended Executor (REXX) language. The EXEC 2 processor handles EXEC 2 programs. The CMS EXEC processor handles CMS EXEC programs. EXEC 2 programs and processing are similar to those of the CMS EXEC. The System Product Interpreter programs are *not* similar to those of EXEC 2 or CMS EXEC.

The System Product Interpreter

The System Product Interpreter is an interpretive command and macro processor. It coexists with the CMS EXEC and EXEC 2 processors. The System Product Interpreter is functionally a superset of CMS EXEC and EXEC 2, but it uses a completely different language and syntax. There is no compatibility between System Product Interpreter programs and those of CMS EXEC or EXEC 2.

VM/SP differentiates System Product Interpreter programs from CMS EXEC or EXEC 2 programs by their first statement. The first statement of every System Product Interpreter program must be a comment. A comment begins with a `/*`, and ends with an `*/`, with anything you want in between. For example:

```
/* This is a comment. */
```

The System Product Interpreter functions are easy to learn and use. They use a general-purpose programming language called REXX, much like that used by PL/I and other programming languages. REXX instructions use structured programming concepts like IF/THEN/ELSE, SELECT, DO WHILE, and so forth, which let you write programs while using words much like those you use to think and communicate.

Other features of the System Product Interpreter and the REXX language are:

- It has a number of useful built-in functions you can use in your programs.
- Programs can be written in mixed case with free form layout (which makes them easier to read and follow).
- It has extensive mathematical capabilities (you can even use it as a desk calculator if you wish).
- There is no limit (except the user's virtual storage size) to the length of manipulated data.
- It is easy to find syntax errors in a program. The System Product Interpreter executes programs line-by-line and word-by-word, without translating them to another form (no compiling). Thus, when there is a syntax error, the place where it occurred is clearly indicated.
- You can use the TRACE instruction to see how the System Product Interpreter is interpreting a particular instruction. This should help you in debugging.

The following books tell you how to use the System Product Interpreter and the REXX language:

- The *VM/SP System Product Interpreter User's Guide* is a step-by-step, tutorial-like, guide to using the System Product Interpreter. It is intended for new users. There is also an introductory chapter in the *VM/SP CMS Primer* about the System Product Interpreter.
- The *VM/SP System Product Interpreter Reference* is a complete compilation of reference information for using the System Product Interpreter. It is intended for all users.

As a CMS user, you should become familiar with the System Product Interpreter and use it often to tailor CMS commands to your own needs, as well as to create your own commands. The System Product Interpreter and the EXEC 2 interpreter can be used by the System Product Editor for XEDIT macro processing support.

Complete details about using the System Product Interpreter are found in the books listed previously.

EXEC 2 Processor

The EXEC 2 processor handles EXEC 2 programs. These EXEC 2 programs and processing are similar to CMS EXEC programs and processing.

EXEC 2 differs from CMS EXEC in the following ways:

- EXEC 2 has extended string manipulation functions.
- EXEC 2 has arithmetic functions for multiplication and division.
- EXEC 2 has extended debugging facilities.
- EXEC 2 supports user defined functions and subroutines.
- EXEC 2 lets CMS user programs manipulate EXEC 2 variables.
- There is no 8-byte token restriction. Statements are composed of *words* of up to 255 characters each.
- Commands can be entered from EXEC 2 either to CMS or to specified *subcommand* environments (for example, the System Product Editor).

The CMS EXEC Processor

A CMS EXEC procedure is a CMS file that contains executable statements. The statements can be CMS or CP commands or EXEC control statements. The execution can be conditionally controlled with additional EXEC statements, or it can contain no EXEC statements at all. In its simplest form, an exec file can contain only one record, have no variables, and expect no arguments to be passed to it. In its most complex form, it can contain thousands of records and can resemble a program written in a high-level programming language.

Two CMS commands create EXEC files. One is LISTFILE, which can be run with the EXEC option; it creates a file named CMS EXEC. The CMS/DOS command, LISTIO, creates an EXEC file named \$LISTIO EXEC, which creates records for each of the system and programmer logical unit assignments. For information on

the LISTIO command and the \$LISTIO EXEC, see the *VM/SP Application Development Guide for CMS*.

The EXEC control statements for the CMS EXEC facility can be found by using the HELP command.

Relationship of the Exec Interpreters

The three interpreters described previously have their own distinct keywords and syntax. So for example, you can not place EXEC 2 statements within a System Product Interpreter program.

The three interpreters coexist, so EXEC programs will continue to correctly execute with no user modifications, regardless of the language. To run CMS EXEC programs as EXEC 2 programs, you must convert the EXEC programs to EXEC 2 programs.

While you may not use, for example, EXEC 2 language statements in an EXEC to be interpreted by the System Product Interpreter nor REXX language statements in an EXEC to be interpreted by the EXEC 2 interpreter, any EXEC can call another EXEC, regardless of the language. Thus an EXEC 2 procedure may be run from within a CMS EXEC procedure, and vice versa.

To allow greater user flexibility with EXEC 2 and the System Product Interpreter, automatic cleanup of an active OS, VSAM, or Vector environment is not obtained at command completion as it is in the CMS EXEC processor. It is your responsibility to ensure that OS, VSAM, and Vector cleanup functions are obtained when needed. VSAM cleanup can be explicitly invoked by entering DMSVSR as a command. The CMS EXEC processor invokes OS, VSAM, and Vector cleanup after the execution of any CMS command. Consequently, any CMS EXEC used resets the OS, VSAM, and Vector environments if it contains a CMS command that is executed.

Running Execs

Exec programs may reside in exec files (with a file type of exec), on a minidisk or SFS directory, or in storage as storage resident execs, and can be used via the EXEC command.

When an exec file is used, CMS examines the first statement of the exec file to determine which exec processor must handle it. If the first line contains /* a comment *//, then the System Product Interpreter is called. If the first statement of the exec is &TRACE, CMS calls the EXEC 2 processor to handle it. If the first statement is not &TRACE or /* a comment *//, CMS calls the exec processor to handle it.

Note: The &TRACE statement does not have to be the first statement in an EXEC 2 file if the file does not have a file type of EXEC (if the EXEC is invoked by an SVC 202).

Attributes of EXEC Files

EXEC files can have any file name that is valid for a CMS file name. EXEC 2 and System Product Interpreter files have file type EXEC for files that are run from the CMS environment, and the file type XEDIT for files used as System Product editor macros.

Exec Processors

System Product Interpreter or EXEC 2 files can be either F or V format. The maximum line length for lines read from the console is 130; for lines read from the stack it is 255.

Chapter 14. Creating System Product Interpreter Execs

A System Product Interpreter file, like a CMS EXEC or EXEC 2 file, has a file type of EXEC. To determine which exec interpreter will be run by an exec file, look at the first line in the file.

First line	Interpreter
/* a comment */	System Product Interpreter
&TRACE	EXEC 2 Processor
Anything else.	CMS EXEC Facility

You can create exec files with the CMS editors by using CMS commands or programs, or by punching cards. When you create a file (file type of EXEC) using XEDIT, records are, by default, variable-length with a logical record length (lrecl) of 130 characters and case is upper. The CMS EXEC facility can process variable-length files of up to 130 characters. EXEC 2 can process variable-length files of up to 255 characters. The System Product Interpreter processes files of any logical record length (lrecl). For example, to create an exec file, enter:

```
xedit new exec
```

If you have a fixed-length file that you want converted to a variable-length file, then you can edit the exec file and enter the XEDIT subcommand:

```
recfm v
```

Or, you can enter the COPYFILE command:

```
copyfile new exec a (recfm v
```

Whenever possible, you should use variable-length exec files.

If you use XEDIT to create a CMS EXEC or an EXEC 2 exec, you cannot enter the exec statements in mixed case. Enter the XEDIT subcommand:

```
set case uppercase
```

Running Your Exec Files

Exec procedures are run when you enter the file name of the exec file. You can precede the file name on the command line with the CMS command, EXEC. For example:

```
exec test
```

where TEST is the file name of the exec file. For example, an exec named THANKYOU would be executed when you entered either:

```
exec thankyou
-- or --
thankyou
```


System Product Interpreter Execs

You must precede the exec file name with the exec command when you:

- Run an exec from CMS EXECs and EXEC 2 execs.
- Run an exec from REXX with *address command*. (The default is *address CMS*, which means exec need not be specified.)
- Run an exec from a program.
- Call a System Product Interpreter exec recursively.
- Have the implied exec (IMPEX) function set OFF for your virtual machine.

The implied exec (IMPEX) function is controlled by the SET command. It lets you treat exec files as commands so that you only must enter the file name of the exec program. The default setting for IMPEX is ON; you almost never need to change it. To find out what the IMPEX setting is, enter:

```
q impex
```

If the response is:

```
IMPEX = OFF
```

this means that the exec command must precede the exec file name to run an exec procedure. To set IMPEX to ON, so that you only need to enter the exec file name, enter:

```
set impex on
```

An exec procedure having a synonym defined for it can be run by its synonym if the implied exec (IMPEX) function is on. You may use the synonym for an exec program within a System Product Interpreter program.

One exec file that you never have to specifically execute is a PROFILE EXEC. It automatically executes after you IPL CMS, when your directory or minidisk with a mode of A is accessed. PROFILE EXECs are discussed in Chapter 15, "Creating a PROFILE EXEC."

Sample System Product Interpreter Execs

Here are two sample System Product Interpreter Execs to give you some flavor of the language.

The first sample is an exec to copy a file from any directory or minidisk to the directory or minidisk accessed as file mode A. Note that the exec uses the required first comment statement as a description of its function.

```
/* Copies a file from any accessed directory or minidisk to file mode A */
arg fn ft fm extra
if fn = '?' then signal tell
if extra = '' | ft = ''
then do
  Parse source . . me .
  say 'Invalid command for 'me' exec.'
  exit
end
if fm = '' then fm = '*'
'COPYFILE' fn ft fm '= = A'
exit rc
tell:
parse source . . me .
say 'This exec,' me', copies the given file to'
say 'file mode A and passes back the return'
say 'code from copyfile'.
exit 100
```

System Product Interpreter Execs

The second sample sends the file to the user ID that you specify. Note that in System Product Interpreter execs you do not need to preface a CP command with CP.

```
/* This exec sends a file to a user */
parse source . . me .
arg user fn ft fm extra
if fn = ''
  then
  do
    say 'Command is:' me 'user fn ft <fm>'
    exit 100
  end
if ft = '' | extra ^= ''
  then
  do
    say 'Invalid' me 'message'
    exit 101
  end
'SPOOL PUNCH TO' user 'CLASS A'
if rc ^= 0
  then
  do
    say user 'is not a valid user ID'
    exit 102
  end
if fm = '' then fm = 'A'
'PUNCH' fn ft fm
retsave = rc
'SPOOL PUNCH TO' * 'CLASS A'
if retsave ^= 0
  then
  do
    say 'Error' retsave 'from punch (while in' me)''
    exit 103
  end
'MSG' user 'I have punched you my file' fn ft fm
exit
```

More information about the System Product Interpreter is found in the *VM/SP System Product Interpreter User's Guide* and in the *VM/SP System Product Interpreter Reference*. Also, details on CMS functions that you can use from execs can be found in the *VM/SP Application Development Guide for CMS*.

Chapter 15. Creating a PROFILE EXEC

A PROFILE EXEC is different from other execs. It has the special file name PROFILE and it is automatically executed whenever you enter *IPL CMS* (or if you have automatic IPL). Your PROFILE EXEC contains the CP and CMS commands that you enter at the start of every terminal session. You can write your PROFILE EXEC for any of the exec interpreters. It usually contains commands that:

- Access SFS directories or minidisks.
- Describe your terminal and printer
- Set up your PF keys
- Describe macro and text libraries that you commonly use
- Set your screen colors (color terminals only)
- Run your synonym table
- Make frequently used execs storage resident.

A PROFILE EXEC written with System Product Interpreter statements might look like this:

```

/* sample profile */
'ACCESS VMSYSU:JONES.TOOLS B'      /* Access a local tools directory */
'SET RDYMSG SMSG'                  /* Short form of ready msg */
'SET BLIP' *                        /* set blip character to * */
'SYNONYM MYSYN'                    /* Run my synonym table */
'GLOBAL MACLIB OSMACRO PRIVMAC'    /* MACRO libraries */
'GLOBAL TXTLIB PRIVLIB'           /* TEXT libraries */
'SET PF1 IMMED RDRLIST'            /* PF1 key set to RDRLIST */
'SET PF6 RETRIEVE'                /* PF6 key RETRIEVE function */
'SET PF11 IMMED FILELIST'         /* PF11 key set to FILELIST */
'EXECLOAD FILELIST EXEC (SYSTEM'   /* Make FILELIST, RECEIVE, */
'EXECLOAD RECEIVE EXEC (STSTEM'    /* and EXECUTE storage */
'EXECLOAD EXECUTE XEDIT (SYSTEM'   /* resident execs. */

```

Do not use the CP DEFINE STORAGE command in your PROFILE EXEC. It resets your virtual machine and you would have to IPL CMS again.

You can enter *profile* at any time to execute your PROFILE EXEC. If you make changes to your PROFILE EXEC during your terminal session, the changes will not be in effect until you execute your profile again.

Should you want to suppress the execution of your PROFILE EXEC, the first command you enter after you enter the IPL command is the CMS ACCESS command with the NOPROF option specified. For example, if you enter:

```
ipl cms
```

The system response may be:

```
VM/SP Release 6 4/30/88 11:22:33
```

PROFILE EXEC

To suppress the execution of your PROFILE EXEC, you enter:

```
access (noprof
```

When the system responds with

```
Ready;
```

you have loaded CMS and accessed file mode A without running your PROFILE EXEC.

You can find more information about the CMS ACCESS command in the *VM/SP CMS Command Reference*.

The EXECLOAD command makes a particular exec storage resident. The exec remains storage resident for the entire session and consequently, does not need to be reloaded each time it is invoked. For example, if you are a frequent user of the DIRLIST, FILELIST, MACLIST, NOTE, NAMES, RDRLIST, SENDFILE, and TELL commands, you might consider making the following execs storage resident:

```
ALIALIST EXEC
AUTHLIST EXEC
DIRLIST EXEC
DISCARD EXEC
EXECUTE XEDIT
FILELIST EXEC
MACLIST EXEC
NAMES EXEC
NOTE EXEC
PEEK EXEC
PROFALIA XEDIT
PROFAUTH XEDIT
PROFDLST XEDIT
PROFFLST XEDIT
PROFFSEA XEDIT
PROFFSHR XEDIT
PROFMLST XEDIT
PROFNOTE XEDIT
PROFPEEK XEDIT
PROFRLST XEDIT
PROFSEND XEDIT
RDRLIST EXEC
RECEIVE EXEC
RECEIVE XEDIT
SENDERFILE EXEC
TELL EXEC
X$DLST$X XEDIT
X$FLST$X XEDIT
X$MLST$X XEDIT
X$NAME$X XEDIT
X$NDIR$X XEDIT
X$PEEK$X XEDIT
X$SEND$X XEDIT
```

If your installation has a CMS installation saved segment containing any of these execs, you would not want to load them into storage. Frequently used execs are loaded into a saved segment so that users can share the same executing copy of the

exec. To use the execs in a CMS installation saved segment, you can load the saved segment when you IPL CMS, or you can issue SET INSTSEG ON.

To find more information about using the CMS installation saved segment, see the IPL command in the *VM/SP CP General User Command Reference* and the SET, QUERY, and EXECMAP commands in the *VM/SP CMS Command Reference*.



Chapter 16. Commands Used with System Product Interpreter Execs

Following are some commands used along with System Product Interpreter execs. Command formats, descriptions and usage notes for these commands are found in the *VM/SP CMS Command Reference*.

EXECDROP	Removes execs and macros from storage or discontinues execs and macros in a saved segment.
EXECIO	Manages movement of lines between files or virtual devices and the program stack or a variable. Also causes execution of CP commands and recovers resulting output.
EXECLOAD	Loads an exec into storage.
EXECMAP	Lists execs and macros in storage and in active saved segments.
EXECOS	Resets the OS, VSAM, and Vector environments under CMS without returning to the interactive environment.
EXECSTAT	Provides the status of a specified exec.
GLOBALV	Sets, maintains, and retrieves a collection of named variables.
IDENTIFY	Displays or stacks user ID, node ID, rscs ID, date, time, time zone, and day of the week.
IMMCMD	Establishes or cancels Immediate commands from an exec.
LISTFILE	Lists information about CMS files in accessed SFS directories or on minidisks.
NAMEFIND	Displays or stacks information from a NAMES file (default <i>userid NAMES</i>).
QUERY	Requests information about a CMS virtual machine.
RDR	Generates a return code and either displays or stacks a message that identifies the characteristics of the next file in your virtual reader.
SET EXECTRAC	Sets tracing ON or OFF for your System Product Interpreter or EXEC 2 exec.
SET INSTSEG	Sets the access to the CMS installation saved segment to ON or OFF and sets the file mode where it is searched.
WAITREAD VSCREEN	Updates the virtual screen with data in the virtual screen queue, refreshes the physical screen, and waits for the next attention interrupt. This is entered from an exec.
XEDIT	Initiates the System Product Editor to create or modify a file.

Commands for System Product Interpreter Execs

The following Immediate commands can be used along with System Product Interpreter execs:

HI halt interpretation

TS trace start

TE trace end

The following exec samples show you how you can use some of the CMS commands with your System Product Interpreter execs. You can use CMS commands in your execs to operate on your Shared File System (SFS) files. Callable Services Library (CSL) routines are also available for use with System Product Interpreter execs to operate on SFS files. For more information, see the *VM/SP Application Development Guide for CMS*.

Using EXECIO

The EXECIO command manages movement of lines between files or virtual devices and the program stack or a variable. It also causes execution of CP commands and recovers resulting output.

Note: For manipulating SFS files, consider using Callable Services Library (CSL) SFS routines instead of EXECIO. SFS routines provide more functions than EXECIO; however, they cannot be used to manipulate minidisk files. For more information on Callable Services Library SFS routines, see the *VM/SP Application Development Reference for CMS*.

This example is not intended to teach you all you need to know to write System Product Interpreter programs. If you are not already familiar with the System Product Interpreter, see the *VM/SP System Product Interpreter User's Guide*.

The example illustrates how you might use EXECIO commands in a System Product Interpreter program to read a CMS file from the program stack, then print that file, 60 lines per page, with the output indented 15 spaces.

If the CMS file in this example resides in a Shared File System (SFS) file pool, it must either be not shared or shared read-only. Otherwise, this example may not work properly.

This is not the only, nor necessarily the best, way to accomplish the results. However, it does show some uses of the EXECIO command within a System Product Interpreter program. The statement numbers in the left margin of the sample reference the explanations following the sample, and are not a part of the program.

Because the program reads, prints, and indents, you can name it RDPRIND EXEC (the file type must be EXEC).

RDPRIND EXEC

```

1. /* This program reads, prints, and indents */
2. trace n
3. 'EXECIO 1 PRINT (CC 1 STRING'
4. arg filename filetype filemode .
5. do until execiorc ^=0
6. 'EXECIO 100 DISKR' filename filetype filemode '(STEM' line.
7.   execiorc=rc
8.   do I=1 to line.0
9.     'EXECIO 1 PRINT (STRING           'line.i
10.    if i//60=0
11.      then 'EXECIO 1 PRINT (CC 1 STRING'
12.    end
13.  end
14. 'CLOSE PRT NAME' filename filetype
15. exit

```

Following are numbered explanations of each statement in the RDPRIND program:

1. The first statement in a System Product Interpreter program must always be a comment (*/* comment */*). Note how we use the comment to tell what the program does.
2. Trace all host commands that return a negative return code.
3. This is a CMS command to write a line to the printer (space to top of new page).
4. Read in the passed parameters, assigning values to file name, file type, and file mode. The "." at the end of this line is a placeholder, used here to ignore any passed data after the third parameter.
5. Starts a DO loop. This statement says that the instructions following the DO, up to the END statement that is paired with DO (line 13), should be repeated until the return code from EXECIO (saved in EXECIORC) is not 0.
6. This is a CMS command to read 100 lines from the file called *filename filetype filemode*. Those values are set by the ARG command in line 4. The number of lines actually read is assigned to variable LINE.0 and the actual file lines are assigned to the variables LINE.1, LINE.2, and so on.
7. The return code from the previous host command (in this case from EXECIO on line 6) is saved in the special variable named RC. This statement saves the return code in a variable called EXECIORC so it can be checked later.
8. Another DO loop starts here, similar to the one started in line 5. In this loop, the set of instructions between the DO and its END (on line 12) will be repeated while I is incremented from 1 until it is equal to the number of lines returned from EXECIO.
9. This is a CMS command to write a line to the printer. The blanks will be preserved and the value of LINE.I will be placed on the end of the command before it is passed to CMS.
10. This is a conditional check. It asks if the remainder of I divided by 60 is equal to 0. This will be true when I=60, 120, etc.

Commands for System Product Interpreter Execs

11. If the previous condition checked (in line 10) is true, then this line is executed. If it's executed, it spaces the printer to the top of a new page (the same command was used in line 3).
12. This END ends the DO loop started in line 8.
13. This END ends the DO loop started in line 5.
14. This is a CP command to close the printer and name the file. Its file name, and file type will be set based on the values set in line 4.
15. This statement ends normal processing.

Now, to cause the exec to read and print a CMS file named TESTFILE DATA A, enter:

```
rdprind testfile data a
```

TESTFILE, DATA, and A are substituted into the program for file name, file type, and file mode respectively.

Using EXECDROP, EXECLOAD, EXECMAP, and EXECSTAT

The EXECLOAD command loads an exec into storage and prepares the exec for execution. The following command

```
execload tphone exec a
```

loads TPHONE EXEC into user free storage. When the exec is subsequently run, the storage resident exec is executed. This eliminates the need for CMS to reload the exec into storage each time the exec is run. The exec remains storage resident during the entire session or until specifically dropped by the EXECDROP command. Be aware that if you make any changes to the exec file on your minidisk or SFS directory, the storage resident copy of the exec remains unchanged. To have the new version execute, you will have to do one of the following:

- Drop the exec from storage (using EXECDROP).
- Drop the exec from storage (using EXECDROP) and reload it (using EXECLOAD).
- Load the new version (EXECLOAD with the PUSH option).

Note: Before loading an exec into storage, you should determine whether or not you are using the CMS installation saved segment and if there is another exec with the same name already in storage. If so, you should do one of the following:

- Specify the PUSH option on the EXECLOAD command, or
- Use the EXECDROP command to drop your access to that exec and then enter the EXECLOAD command, or
 - If EXECMAP shows that the exec has the shared attribute, then use the SET INSTSEG OFF command to discontinue use of the CMS installation saved segment and then enter the EXECLOAD command, or
 - If the exec resides in a logical segment (that is, the SEGNAME field returned from EXECMAP for that exec name is non-blank), then enter a SEGMENT PURGE for the logical segment containing the exec(s).

To verify the existence of the TPHONE EXEC in storage and on your minidisk or SFS directory, you can use the EXECSTAT command. For example,

```
execstat tphone exec
```

gives a return code of 0 in register 15, verifying that the exec is storage-resident. The EXECMAP command lists the storage-resident execs.

execmap

returns the following if TPHONE EXEC is the only storage-resident exec:

Name	Type	Usage	Records	Bytes	Attribute
TPHONE	EXEC	0	15	512	USER

Should you decide that you no longer require an exec to be storage resident, you can delete it from storage with the EXECDROP command. For example, entering:

execdrop tphone exec

deletes the TPHONE EXEC from storage.

Using IPL, SET INSTSEG, EXECMAP, and EXECDROP

The IPL command links the CMS installation saved segment for your CMS session. The command

ipl cms parm instseg yes

links the default CMS installation saved segment, CMSINST. The saved segment contains the executing copy of frequently used execs, eliminating the need to load your own copy of the execs into storage. The system accesses the saved segment at file mode S, meaning it is searched before the S-disk (system disk).

To change the file mode, use the SET INSTSEG command. For example, enter:

set instseg on b

to access the saved segment at file mode B. If you already have a file mode b, then the saved segment is immediately searched before it.

The EXECMAP command lists the execs in storage and the ones in saved segments. EXECMAP returns a list with the attribute specifying the location of the exec. For example, assume that you had loaded TPHONE EXEC into user storage and NAMES EXEC into system storage, that FILELIST EXEC and RECEIVE EXEC were in the CMS installation saved segment, and that NOTE EXEC was in a logical saved segment. In this case, the list would be:

Name	Type	Usage	Records	Bytes	Attribute	Segname
TPHONE	EXEC	0	15	515	USER	
NAMES	EXEC	0	60	4616	SYSTEM	
FILELIST	EXEC	0	163	7488	SHARED	
RECEIVE	EXEC	0	625	27568	SHARED	SEGMENT1
NOTE	EXEC	0	554	24952		SEGMENT1

If you no longer want to use an exec in the saved segment, you can discontinue your access to it with the EXECDROP command. For example, entering:

execdrop receive exec (shared)

deletes your access to the RECEIVE EXEC in the saved segment. If you decide you want to drop your access to the saved segment entirely, use the SET INSTSEG or SEGMENT PURGE command. SET INSTSEG OFF discontinues use of the CMS installation saved segment until you set it ON or until you re-IPL CMS.

Commands for System Product Interpreter Execs

Using EXECOS

The EXECOS command resets the OS, VSAM, and Vector environments under CMS without returning to the interactive environment. If you request a reset of the OS, VSAM, or Vector environment, after the execution of a CMS EXEC, the EXECOS command should *precede* the CMS EXEC command. For example:

```
/* example of using EXECOS within an EXEC */
'EXECOS EXEC VMFASM DMSSEB DMSSP'
exit
```

Using GLOBALV

The GLOBALV command sets, maintains, and retrieves a collection of named variables. You can pass these global variables between execs.

For example, we have two exec files named FIRST EXEC and SECOND EXEC, where the FIRST EXEC calls the SECOND EXEC. The variables are established as global variables in the SECOND EXEC by the statement "globalv put RUMORS." The statement "globalv get RUMORS" in the FIRST EXEC assigns the global variables to the FIRST EXEC.

<pre>/* first exec */ . . . second . globalv get 'RUMORS' . . . exit</pre>	<pre>/* second exec */ globalv put 'RUMORS' /* assign variables */ . . . exit</pre>
--	---

Using IDENTIFY

You can use the information returned by the IDENTIFY command within your exec.

For example:

```
/* example of using identify within your exec */
:
:
'IDENTIFY (LIFO' /* get some useful information */
pull userid . node . rscsid .
:
:
exit
```

Using IMMCMD

The IMMCMD command establishes or cancels Immediate commands from an exec.

For the following example, we will assume that you have an exec that performs a repetitive process. Each time this exec is processed, one record is logged to a CMS file. Suppose you wanted to suppress logging of the records without terminating the exec. Because HX terminates the exec, you would not want to use it. Using Pull is not a good alternative because you want to decide at what point to terminate logging. You can create your own Immediate command to stop logging using the CMS IMMCMD command within your exec. For example:

```

/* Sample exec using the CMS IMMCMD command      */
/* Set up stoplog Immediate command              */
'IMMCMD SET STOPLOG'
/* Set default logging                            */
arg log .
if log='' then log='YES'
if log='YES' & log='NO' then do
  say 'Invalid parameter :' log
  exit 24
end
do forever
  /* Check for STOPLOG */
  'IMMCMD STATUS STOPLOG'
  if rc=0 then log='NO'
  /* Perform process ... */
  .
  .
  if log='YES' then 'EXECIO 1 DISKW LOG FILE A'
  .
  .
end
/* Clear STOPLOG Immediate command */
'IMMCMD CLEAR STOPLOG'
exit

```

Commands for System Product Interpreter Execs

Using LISTFILE

The LISTFILE command lists information about your CMS files on accessed SFS directories or minidisks. You can use this information within your exec.

```
/* Example using LISTFILE to find file ID of the first file */
/* that matches a given file name. */
address command
  'MAKEBUF'
  'LISTFILE' filename '* * (FIFO'
  if rc=0 then filetype='EXEC'
  else pull filename filetype filemode .
  address command 'DROPBUF'
```

Using NAMEFIND

The NAMEFIND command displays/stacks information from a NAMES file (default 'userid NAMES'). Following is an example of how you can use the CMS NAMEFIND command in an exec:

```
/* Program to retrieve phone numbers */
arg nick .
'NAMEFIND :NICK' nick ':PHONE :NAME (LIFO'
if rc=0 then do
  say 'Sorry, no phone listing for' nick
  exit
end
parse pull name
parse pull phone
if phone='' then do
  say 'Sorry, no phone listing for' name
  exit
end
say name"'s phone number is" phone'.'
exit
```

Using QUERY and RDR

The following example illustrates one way that you can use the information returned from the QUERY and RDR commands in an exec:

```

/* Sample exec to show QUERY and RDR command uses */

/* This section uses the CMS QUERY command to stack information on
the contents of the user's file mode A. Then, it reads in
the information (throwing away the header line stacked by
the QUERY command) and prints out a formatted message. Unused
variables set by the PULL command can be displayed if you desire */

'QUERY DISK A (LIFO)' /* get disk information */
pull label cuu m stat cyl type, /* read from the stack, */
      blksize files used '-' percent, /* separate into all */
      left total . /* variables */
pull . /* read header line */
used = strip(used,1) /* strip leading blanks */
say 'Filemode A is' percent'% full ('used' used blocks out of' total,
    'available)'

/* This section invokes the CMS RDR command, which sets a
return code depending on the status of the reader and also on
the type of file in the reader, should one exist. The System
Product Interpreter sets the variable RC to this returned
value. Next, depending on the returned value, this exec
selectively executes one of several commands. */

'RDR (NOTYPE)' /* get info on rdr file */
/* RC set to return code from RDR command */

select
  when rc=0 then say 'Reader is empty'
  when rc=22 then 'disk load'
  when rc=13 then say 'Reader is not ready'
  otherwise
    say 'Return code other than expected'
end

```

Using SET EXECTRAC

You can trace your System Product Interpreter or EXEC 2 exec by specifying SET EXECTRAC ON prior to exec invocation. To turn tracing off, specify SET EXECTRAC OFF.

Using Windows and Virtual Screens

WAITREAD VSCREEN is an important command for execs that read from and write to windows. A typical sequence for such an exec would be:

- Define a window and virtual screen (DEFINE WINDOW and DEFINE VSCREEN commands)
- Connect the window to the virtual screen (SHOW WINDOW command)
- Write data to the virtual screen (WRITE VSCREEN command)
- Enter the WAITREAD VSCREEN command

When an attention interrupt is received, the exec can process the WAITREAD. *n* variables and update the virtual screen using the WRITE VSCREEN command.

For more information on these commands, refer to the *VM/SP CMS Command Reference*.

Using XEDIT

You can use the XEDIT command within an exec and stack XEDIT subcommands to manipulate a file.

Writing XEDIT macros

Writing an XEDIT macro is like creating a new XEDIT subcommand. An XEDIT macro is a System Product Interpreter or EXEC 2 file invoked from the XEDIT environment.

Refer to the *VM/SP System Product Editor User's Guide* for information on writing XEDIT macros using the System Product Interpreter. For information about XEDIT macros written in EXEC 2 language, refer to the *VM/SP EXEC 2 Reference*.

Exchanging Data Between Programs Through the Stack

Refer to the *VM/SP System Product Interpreter User's Guide* for information on those REXX instructions that put data into the program stack.

Part 4: Tailoring Your System

VM/SP provides several ways for you to tailor your virtual machine. You can use full-screen CMS and windowing commands to customize your CMS session, or you can tailor the HELP facility to conform to your particular needs.

Chapter 17: Customizing Full-Screen CMS describes ways in which you can change the attributes of your windows and virtual screens.

Chapter 18: Tailoring the HELP Facility describes ways in which you can tailor the HELP Facility to your needs and describes techniques provided by the HELP Facility for creating user HELP description files.

Chapter 17. Customizing Full-Screen CMS

As you become more familiar with full-screen CMS, you may want to tailor windows and virtual screens to your special needs. First, it is necessary to understand what happens when you enter the command, `SET FULLSCREEN ON`.

During full-screen CMS initialization:

- All default virtual screens that you have not defined are defined, such as a virtual screen for CMS and CP output and virtual screens for messages, network messages, warnings from the operator, and status information.
- All default windows that you have not defined are defined, with the exception of the WM window. The WM window is defined when you enter the command `POP WINDOW WM`, when you press the PA1 key, or when the window is automatically displayed on your screen. (See Chapter 11, “Looking at VM/SP Through Windows” on page 219 for detailed information on when the WM window is displayed).
- All the reserved areas for the default virtual screens are written.
- Default windows are connected to the appropriate virtual screens.
- CMSPF key definitions are established.
- A connection to the IUCV Message All System Service is established and various message classes are routed to appropriate virtual screens.
- Logging is started for the MESSAGE and WARNING virtual screens. Messages are logged into the file MESSAGE LOGFILE, and warnings are logged into WARNING LOGFILE.
- The cursor is set in the CMS virtual screen.
- The CP `TERMINAL BRKKEY NONE` command is entered.

For more information, see the `SET FULLSCREEN` and `SET CMSPF` commands in the *VM/SP CMS Command Reference*.

You may want to enter certain commands before entering full-screen CMS. Other commands may be entered after setting full-screen CMS ON. For example, if you want to change the size of a default virtual screen or change the definition of a default window, you must enter the appropriate commands before setting full-screen CMS on. The following chapter teaches you how to easily change a default setting.

You should read this section at the terminal so you can try the exercises as you read the text. If you want to find out more about a command, see the *VM/SP CMS Command Reference*.

Tailoring System Defaults

By defining windows and virtual screens before entering full-screen CMS, you can override full-screen CMS default definitions (See the tables at the end of this chapter for default values.) Once in full-screen CMS, you can change features such as virtual screen reserved lines, border characters, and the CMSPF keys.

Now change the MESSAGE window and MESSAGE virtual screen to see how tailoring works. If you are in full-screen CMS, set full-screen off by entering

```
set fullscreen off
```

Suppose you define the MESSAGE virtual screen to be 35 lines by 70 columns, which means it will be larger than the default of 20 lines by 70 columns. We will put two reserved lines at the top (for a title) and no reserved lines at the bottom. We will also specify the default options for the MESSAGE virtual screen: SYSTEM, PROTECT, and WHITE. See the tables at the end of this chapter for a description of these options. Enter the command as follows:

```
define vscreen message 35 70 2 0 (system protect white
```

Next, define the MESSAGE window to be 10 lines by 71 columns, which means it will be larger than the default of 8 lines by 71 columns. It will be located at line 10 and column 5 on your physical screen. (The default location is line 11 and column 3). Again, we will specify the default options: SYSTEM, VARIABLE, and POP. Enter the command as follows:

```
define window message 10 71 10 5 (system variable pop
```

Now, change the border character on the MESSAGE window to a \$ for all sides of the window border. By default, the top and bottom characters are - (dash) and the right and left sides are | (vertical bar). To do this, enter

```
set border message on (all $
```

Here is where you set full-screen CMS on. Enter:

```
set fullscreen on
```

Next, define a field at reserved line 1 and column 1 of the MESSAGE virtual screen with a length of 70 (the number of columns in the virtual screen). This prepares the MESSAGE virtual screen for a new title by replacing the default title with blanks. Thus, you do not have to worry about trying to overlay existing data.

Enter the command

```
write vscreen message 1 1 70 (reserved blank
```

Finally, change the title of the MESSAGE virtual screen. With the WRITE VSCREEN command, we will change the title from the default title *MESSAGES* to *My Personal Messages*. The new title begins in column 26 and is 20 characters long.

```
write vscreen message 1 26 20 (reserved data My Personal Messages
```

Now you have tailored the MESSAGE window and virtual screen to your own specifications. Another way to accomplish this would be to write an exec. Here is what the exec would look like:

```

/* EXEC to tailor the MESSAGE virtual screen and window */
'define vscreen message 35 70 2 0 (system protect white'
'define window message 10 71 10 5 (system variable pop'
'set border message on (all $'
'set fullscreen on'
'write vscreen message 1 1 70 (reserved blank'
'write vscreen message 1 26 20 (reserved data My Personal Messages'

```

To see your new MESSAGE window, send this message to yourself:

```
tell * Let's see what happened.
```

When the MESSAGE window is popped, you see the following:

```

Fullscreen CMS                Columns 1 - 79 of 81

Ready;
write vscreen message 1 1 70 (reserved blank
Ready;
write vscreen message 1 26 20 (reserved data My Personal Messages
Ready;
tell * Let's see what happened.
+ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ +
$                               My Personal Messages                $
$                               $                                    $
$   15:10:45 MSG FROM VMUSER   : Let's see what happened.          $
+ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ +

PF1=Help    2=Pop_Msg  3=Quit    4=Clear_Top  5=Filelist  6=Retrieve
PF7=Backward 8=Forward 9=Rdrlist 10=Left     11=Right   12=Cmdline
====>
15:10:45 Message                     Enter a command or press a PF or PA key

```

Figure 75. Your Newly-Defined MESSAGE Window

Remember, because you defined the window as variable in size, the size of the window varies depending on how many messages you receive.

Now, enter a *D* in any corner of the window to drop it from your screen.

You may wish to tailor your CMS session by writing other simple execs to perform windowing functions. For example, you may find it useful to set a PF key to toggle between popping and dropping the MESSAGE window. To try this example, create a file with a file name of POPDROP and file type of exec. Then, XEDIT the file and enter the following:

```

/* EXEC to set a PF key to POP and DROP the MESSAGE window*/
PARSE UPPER ARG whichway
ADDRESS COMMAND
IF whichway = "DROP" THEN
  DO
    'POP WINDOW MESSAGE'
    'SET CMSPF 02 Drop_Msg NOECHO POPDROP DROP'
  END
ELSE
  DO
    'DROP WINDOW MESSAGE'
    'SET CMSPF 02 Pop_Msg NOECHO POPDROP POP'
  END
END

```

File the exec. Now, when you enter the command POPDROP, CMSPF 2 will initially be set to drop the MESSAGE window. Therefore, when you receive a message, and the MESSAGE window pops, you can simply press CMSPF 2 to drop it.

Try it! Enter:

popdrop

to issue the POPDROP EXEC. The exec will pop the MESSAGE window. The window appears again as shown in the following screen sample:

```

                                     Fullscreen CMS          Columns 1 - 79 of 81
Ready;
write vscreen message 1 1 70 (reserved blank
Ready;
write vscreen message 1 26 20 (reserved data My Personal Messages
Ready;
tell * Let's see what happened.
+ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ +
$                               My Personal Messages                               $
$                                     $                                               $
$    15:10:45 MSG FROM VMUSER  : Let's see what happened.                            $
+ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ +
-----
PF1=Help      2=Drop_Msg  3=Quit      4=Clear_Top  5=Filelist  6=Retrieve
PF7=Backward  8=Forward  9=Rdrlist  10=Left     11=Right   12=Cmdline
====>
15:20:03                          Enter a command or press a PF or PA key

```

Figure 76. Using the POPDROP EXEC

Press CMSPF 2 to drop the window. From now on, CMSPF 2 will toggle between popping and dropping the MESSAGE window. When the MESSAGE window is

showing on your screen, you can press CMSPF 2 to drop it. When the MESSAGE window is not showing on your screen, you can press CMSPF 2 to pop the window.

Note: For the MESSAGE window to appear on your screen, it must contain at least one message. If you try to pop the MESSAGE window before any messages have been received, the window is not displayed.

Press CMSPF 2 to pop the MESSAGE window and prepare for the next exercise.

POSITION WINDOW

With the POSITION WINDOW command, you can move a window anywhere on the physical screen. Move our new MESSAGE window.

If you use an “=” in the command syntax instead of the window name, the command moves the topmost window. Because the MESSAGE window is currently the topmost window, you can enter the following command to move it:

```
position window = 11 3
```

The 11 and 3 are the line and column, respectively, where the window's UPPER left corner will be repositioned relative to the TOP of the screen. When you enter this command, the MESSAGE window is repositioned on the screen as shown:

```

                                Fullscreen CMS           Columns 1 - 79 of 81

Ready;
write vscreen message 1 1 70 (reserved blank
Ready;
write vscreen message 1 26 20 (reserved data My Personal Messages
Ready;
tell * Let's see what happened.
Ready;
+ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ +
$                               My Personal Messages                                $
$                                                                                                $
$  15:06:35 MSG FROM VMUSER : Let's see what happened.                                $
+ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ +
Ready;
position window = 11 3
Ready;
-

PF1=Help      2=Drop_Msg  3=Quit      4=Clear_Top  5=Filelist  6=Retrieve
PF7=Backward  8=Forward  9=Rdrlist  10=Left    11=Right   12=Cmdline
=====
15:23:39                                Enter a command or press a PF or PA key
```

Figure 77. Using the POSITION WINDOW Command

SIZE WINDOW

If you wish, you can change the maximum number of lines in your MESSAGE window with the SIZE WINDOW command. To do this, enter:

```
size window = 8 70
```

Because the MESSAGE window is variable in size, you may not see the result of the change in size until you receive more messages that fill and expand the window.

MAXIMIZE WINDOW and RESTORE WINDOW

Suppose you wanted to enlarge a window to look at more of the data contained in the virtual screen. With the MAXIMIZE WINDOW command, you can expand the size of the window.

If you are following the exercises, the MESSAGE window is still on your screen. It has the title: "My Personal Messages." It also has one message in it: "Let's see what happened."

To see what happens with the MAXIMIZE WINDOW command, send yourself the following 10 messages (press ENTER after each message):

```
tell * This is message 1.  
tell * This is message 2.  
tell * This is message 3.  
tell * This is message 4.  
tell * This is message 5.  
tell * This is message 6.  
tell * This is message 7.  
tell * This is message 8.  
tell * This is message 9.  
tell * This is message 10.
```

After you scroll forward, your screen appears like this:

```

                                Fullscreen CMS                               Lines 33 - 37 of 37
                                                                 Columns 1 - 79 of 81

Ready;
tell * This is message 9.
Ready;
tell * This is message 10.
Ready;

+ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ +
$                               My Personal Messages                      Lines 1 - 6 of 11 $
$                               Columns 1 - 69 of 70                       $
$ 15:10:45 MSG FROM VMUSER : Let's see what happened.                    $
$ 15:32:43 MSG FROM VMUSER : This is message 1.                          $
$ 15:32:50 MSG FROM VMUSER : This is message 2.                          $
$ 15:32:56 MSG FROM VMUSER : This is message 3.                          $
$ 15:33:01 MSG FROM VMUSER : This is message 4.                          $
$ 15:33:08 MSG FROM VMUSER : This is message 5.                          $
+ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ +

PF1=Help      2=Drop_Msg 3=Quit      4=Clear_Top 5=Filelist 6=Retrieve
PF7=Backward  8=Forward  9=Rdrlist 10=Left    11=Right  12=Cmdline
====>
15:34:07 Message                                Enter a command or press a PF or PA key

```

Figure 78. Popping the MESSAGE Window

Here is how your screen looks (the same as it was before using the MAXIMIZE WINDOW):

```

Fullcreen CMS                               Lines 33 - 41 of 41
                                           Columns 1 - 79 of 81

Ready;
tell * This is message 9.
Ready;
tell * This is message 10.
Ready;
maximize window message
Ready;
+ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ +
$                               My Personal Messages                Lines 1 - 6 of 11 $
$                               Columns 1 - 69 of 70                $
$  15:10:45 MSG FROM VMUSER   : Let's see what happened.           $
$  15:32:43 MSG FROM VMUSER   : This is message 1.                  $
$  15:32:50 MSG FROM VMUSER   : This is message 2.                  $
$  15:32:56 MSG FROM VMUSER   : This is message 3.                  $
$  15:33:01 MSG FROM VMUSER   : This is message 4.                  $
$  15:33:08 MSG FROM VMUSER   : This is message 5.                  $
+ $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ +

PF1=Help      2=Drop_Msg  3=Quit      4=Clear_Top  5=Filelist  6=Retrieve
PF7=Backward  8=Forward   9=Rdrlist   10=Left     11=Right   12=Cmdline
====>
15:47:07
                                           Enter a command or press a PF or PA key

```

Figure 80. The Window after RESTORE WINDOW

Using the SET Command

The following sections provide information and examples on using SET commands.

SET BORDER

With the SET BORDER command, you can tailor the characters of a border or change how a border is displayed. These features help you to visually separate information displayed in different windows. With the earlier example, you changed all the borders to \$. Now change just the top border to %.

When you enter the command for only the top border, the top edge changes to %, but you will not see the other sides of the border. For example, enter this command:

```
set border message on (top %
```

The border looks like this:

```

                                Fullscreen CMS
                                Lines 33 - 43 of 43
                                Columns 1 - 79 of 81

Ready;
tell * This is message 9.
Ready;
tell * This is message 10.
Ready;
maximize window message
Ready;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                                My Personal Messages
                                Lines 1 - 6 of 11
                                Columns 1 - 69 of 70

15:10:45 MSG FROM VMUSER : Let's see what happened.
15:32:43 MSG FROM VMUSER : This is message 1.
15:32:50 MSG FROM VMUSER : This is message 2.
15:32:56 MSG FROM VMUSER : This is message 3.
15:33:01 MSG FROM VMUSER : This is message 4.
15:33:08 MSG FROM VMUSER : This is message 5.

PF1=Help      2=Drop_Msg  3=Quit      4=Clear_Top  5=Filelist  6=Retrieve
PF7=Backward  8=Forward   9=Rdrlist 10=Left     11=Right    12=Cmdline
====>
16:05:47
                                Enter a command or press a PF or PA key
```

Figure 81. Changing Only the Top Border

To see just the bottom and left borders, enter
set border message on (bottom = left *

Here is what happens to the window:

```

Fullscreen CMS                               Lines 33 - 45 of 45
                                           Columns 1 - 79 of 81

Ready;
tell * This is message 9.
Ready;
tell * This is message 10.
Ready;
maximize window message
Ready;
restore window message
*                               My Personal Messages           Lines 1 - 6 of 11
*                               Columns 1 - 69 of 70
* 15:10:45 MSG FROM VMUSER : Let's see what happened.
* 15:32:43 MSG FROM VMUSER : This is message 1.
* 15:32:50 MSG FROM VMUSER : This is message 2.
* 15:32:56 MSG FROM VMUSER : This is message 3.
* 15:33:01 MSG FROM VMUSER : This is message 4.
* 15:33:08 MSG FROM VMUSER : This is message 5.
+ =====
PF1=Help      2=Drop_Msg  3=Quit      4=Clear_Top  5=Filelist  6=Retrieve
PF7=Backward  8=Forward  9=Rdrlist  10=Left     11=Right   12=Cmdline
====>
16:09:49
Enter a command or press a PF or PA key

```

Figure 82. Changing the Bottom and Left Window Borders

SET RESERVED

Suppose you do not want to see a title in a window. With the SET RESERVED command, you can delete the title *My Personal Messages* from the MESSAGE window and reuse the area previously reserved for the title. Enter:

```
set reserved message 0 0
```

Here is what happens:

```

Fullcreen CMS                               Lines 33 - 47 of 47
                                           Columns 1 - 79 of 81

Ready;
tell * This is message 9.
Ready;
tell * This is message 10.
Ready;
maximize window message
Ready;
restore window message
* 15:10:45 MSG FROM VMUSER : Let's see what happ Lines 1 - 8 of 11
* 15:32:43 MSG FROM VMUSER : This is message 1 Columns 1 - 69 of 70
* 15:32:50 MSG FROM VMUSER : This is message 2.
* 15:32:56 MSG FROM VMUSER : This is message 3.
* 15:33:01 MSG FROM VMUSER : This is message 4.
* 15:33:08 MSG FROM VMUSER : This is message 5.
* 15:33:14 MSG FROM VMUSER : This is message 6.
* 15:33:20 MSG FROM VMUSER : This is message 7.
+ =====
PF1=Help      2=Drop_Msg  3=Quit      4=Clear_Top  5=Filelist  6=Retrieve
PF7=Backward  8=Forward  9=Rdrlist  10=Left     11=Right   12=Cmdline
====>
16:11:16
Enter a command or press a PF or PA key
    
```

Figure 83. Deleting a Window Title

Now, set the reserved line to 1 so there will not be a blank line after the window title. Enter:

```
set reserved message 1 0
```

Your screen now looks like this:

```

                                     Fullscreen CMS
                                     Lines 33 - 49 of 49
                                     Columns 1 - 79 of 81
Ready;
tell * This is message 9.
Ready;
tell * This is message 10.
Ready;
maximize window message
Ready;
restore window message
*
*           My Personal Messages           Lines 1 - 7 of 11
* 15:10:45 MSG FROM VMUSER : Let's see what h Columns 1 - 69 of 70
* 15:32:43 MSG FROM VMUSER : This is message 1.
* 15:32:50 MSG FROM VMUSER : This is message 2.
* 15:32:56 MSG FROM VMUSER : This is message 3.
* 15:33:01 MSG FROM VMUSER : This is message 4.
* 15:33:08 MSG FROM VMUSER : This is message 5.
* 15:33:14 MSG FROM VMUSER : This is message 6.
+ =====
PF1=Help      2=Drop_Msg 3=Quit      4=Clear_Top 5=Filelist 6=Retrieve
PF7=Backward 8=Forward  9=Rdrlst 10=Left   11=Right  12=Cmdline
====>
16:13:25
                                     Enter a command or press a PF or PA key

```

Figure 84. Deleting a Blank Reserved Line

Now, drop the MESSAGE by typing a *D* in any corner. Then, enter the command

```
clear vscreen message
```

to clear the MESSAGE virtual screen of all old messages.

SET WINDOW

By using the SET WINDOW command, you can choose whether or not you want the MESSAGE window to pop when you receive a message. The default setting is for the window to pop. In addition, the message class indicator will be updated, and the terminal alarm will sound. You can change this by setting the window to NOPOP.

To try it, enter

```
set window message nopop
```

Now send yourself the following message:

```
tell * This window won't pop automatically.
```

After you press ENTER, you will notice that your terminal alarm sounds and the message class indicator will be updated as shown in the following example.

However, the window will not be automatically displayed on your screen. At this point, you would need to enter the command POP WINDOW MESSAGE to pop the window.


```
Fullscreen CMS                               Lines 49 - 55 of 55
                                              Columns 1 - 79 of 81

Ready;
clear vscreen message
Ready;
set window message nopop
Ready;
tell * This window won't pop automatically.
Ready;

PF1=Help   2=Drop_Msg  3=Quit    4=Clear_Top  5=Filelist  6=Retrieve
PF7=Backward 8=Forward  9=Rdrlist 10=Left     11=Right    12=Cmndline
====>
16:15:51 Message                               Enter a command or press a PF or PA key
```

Figure 85. Message Class Indicator

Now set the window to pop when you receive a message. Enter

```
set window message pop
```

Send yourself the following message:

```
tell * I'll bet it pops this time!
```

Your screen displays the MESSAGE window as shown in the next example:

```

                                Fullscreen CMS                Lines 49 - 59 of 59
                                                                Columns 1 - 79 of 81

Ready;
clear vscreen message
Ready;
set window message nopop
Ready;
tell * This window won't pop automatically.
Ready;
set window message pop
*
                                My Personal Messages    Columns 1 - 69 of 70
*  16:15:51 MSG FROM VMUSER : This window won't pop automatically.
*  16:18:15 MSG FROM VMUSER : I'll bet it pops this time!
+ =====

PF1=Help      2=Drop_Msg  3=Quit      4=Clear_Top  5=Filelist  6=Retrieve
PF7=Backward  8=Forward  9=Rdrlist  10=Left     11=Right   12=Cmdline
====>
16:18:15 Message                                Enter a command or press a PF or PA key

```

Figure 86. MESSAGE Window

Now, clear the MESSAGE window with the *C* border command, or press PA2 to scroll and drop the MESSAGE window. This brings you back to the CMS window. If you want to leave full-screen CMS, enter the command: SET FULLSCREEN OFF. You can also enter SET FULLSCREEN SUSPEND or press CMSPF 3 to suspend full-screen CMS. If you suspend full-screen CMS, you can later resume your CMS session where you left off. None of the default or user-defined settings for windows, virtual screens, or PF keys is lost.

Exploring Other Commands

These are just a few ways you can tailor your windows and virtual screens. By looking at the *VM/SP CMS Command Reference* and trying some commands, you can make windows and virtual screens work for you.

For more information that can be useful when you begin using windowing and full-screen CMS, see Appendix C, "Considerations for Full-Screen CMS and Windowing" on page 349

Window and Virtual Screen Tables

If you are interested in tailoring windows or virtual screens, you may find it useful to refer to the following tables for the full-screen CMS default characteristics for windows and virtual screens.

Table 16. Default Windows					
Window	Lines	Cols	Psline	Pscol	Options
STATUS	1	Pscr	-1	1	FIXED NOBORDER NOPOP NOTOP
CMS	Pscr	Pscr	1	1	FIXED BORDER NOPOP TOP
NETWORK	8 (max.)	71	-12	7	VARIABLE BORDER NOPOP TOP
WARNING	6 (max.)	71	3	3	VARIABLE BORDER POP TOP
MESSAGE	8 (max.)	71	11	3	VARIABLE BORDER POP TOP
WM	5	Pscr	-1	1	FIXED BORDER NOPOP NOTOP
CMSOUT	8	75	9	3	VARIABLE BORDER POP TOP

NOTES:

Pscr

Size of the physical screen.

Psline

The line on the physical screen where the upper (when psline is positive) or lower (when psline is negative) corner of the window will be placed.

Pscol

The column on the physical screen where the upper left corner of the window is placed.

FIXED

The number of lines in the window is always constant.

VARIABLE

The number of lines in the window may vary from zero to the maximum, depending on the amount of scrollable data to be displayed.

BORDER

The window borders are displayed when possible.

Note: In the case of the CMS window, even though the borders are on, you cannot see them because the window is the size of the physical screen.

NOBORDER

The window borders are not displayed.

POP

The window is displayed on top of all other windows when the virtual screen that the window is showing is updated.

NOPOP

There is no effect on the window's position in the ordered list of windows when the virtual screen that the window is showing is updated.

TOP

The window may qualify as the topmost window.

NOTOP

The window cannot qualify as the topmost window.

Although the WM window is a default window, it is not defined when you enter full-screen CMS. The WM window is defined when you enter the command POP WINDOW WM, when you press the PA1 key, or when the window is automatically displayed on your screen.

All default windows are SYSTEM windows; they are not lost when the system abnormally terminates (abends) or when the HX (halt execution) command is entered.

Virtual Screen	Lines	Cols	Rtop	Rbot	Dcolor	Options
WM	1	Pscr	0	5	White	NOPROTECT
STATUS	1	Pscr	0	0	White	PROTECT
NETWORK	16	70	2	0	Blue	PROTECT
WARNING	4	70	2	0	Red	PROTECT
MESSAGE	20	70	2	0	White	PROTECT
CMS	120	Pscr	2	5	Green	NOPROTECT

NOTES:**Pscr**

Physical screen size. For terminals with a screen size of 80 columns or less, the CMS virtual screen contains 81 columns. For terminals with a screen size greater than 80 columns, the CMS virtual screen contains the same number of columns as the physical screen. The status and WM virtual screens always contain the same number of columns as the physical screen.

Rtop

Top reserved lines

Rbot

Bottom reserved lines

Dcolor

Data color (if your terminal is equipped for color)

PROTECT

You cannot type into the window(s) connected to the virtual screen because the data is protected.

NOPROTECT

You can type into the window(s) connected to virtual screen; the data is not protected.

Notes:

1. Although the WM virtual screen is a default virtual screen, it is not defined when you enter full-screen CMS. The WM virtual screen is defined when you enter the command POP WINDOW WM, when you press the PA1 key, or when the window is automatically displayed on your screen.
2. All default virtual screens are TYPE virtual screens. TYPE means data is moved to the virtual screen when the virtual screen is updated. In addition, all default virtual screens are SYSTEM virtual screens. SYSTEM means the virtual screen is retained when the system abnormally terminates (abends) or when the HX (halt execution) command is entered.
3. The following CMS commands are invalid for the STATUS default virtual screen:

CLEAR VSCREEN
CURSOR VSCREEN
GET VSCREEN
PUT VSCREEN
ROUTE
SET LOGFILE
SET VSCREEN
WAITT VSCREEN
WAITREAD VSCREEN
WRITE VSCREEN

However, the CMS commands DELETE VSCREEN and DEFINE VSCREEN may be used to replace the STATUS default virtual screen with a user version that will allow execution of these CMS commands.

Considerations When Disconnecting and Reconnecting

If you disconnect from your terminal and later reconnect at a terminal that has a different physical screen size, you may notice certain changes.

Note: You must reconnect to a terminal with a physical screen size of at least 24 lines by 80 columns. If you reconnect to a terminal with a physical screen smaller than 24 by 80, full-screen CMS is suspended. You need to reconnect to a terminal with a larger physical screen to continue your full-screen CMS session.

Once you reconnect and resume your full-screen CMS session, certain windows and virtual screens that are the size of the physical screen are resized, relocated, or redefined to fit the dimensions of the new physical screen.

The windows and virtual screens that may be affected are those listed in the previous tables with "Pscr" under the headings 'Lines' or 'Columns'.

The following list provides details on how these windows and virtual screens change:

- Default windows are resized to fit the new physical screen.
- Default windows you have moved are resized and relocated to the default size and location.
- User-defined windows with a size and/or location that cannot fit on the new physical screen are adjusted.
- Default virtual screens are redefined to fit the size of the new physical screen only if the width of the physical screen changes. Default reserved line data are rewritten in the new virtual screens. In the instance where the virtual screens must be redefined, you lose the data contained in those virtual screens.
- User-defined virtual screens are untouched.

You may also notice after you reconnect at another terminal that certain commands that depend upon the physical device characteristics temporarily reflect the characteristics of the previous terminal. For example, such commands as `QUERY DISPLAY` and `QUERY WINDOW` (in the case where the window is the size of the physical screen) does not automatically reflect the physical screen size of the new terminal.

Once you set full-screen on or enter `XEDIT` (or a productivity aid which uses `XEDIT`), the settings relating to terminal size are adjusted. Any subsequent commands you enter reflect the physical screen size of your current terminal.

Message Routing Tables

You may also find it useful to refer to the following tables that contain information regarding the default settings for message routing.

Table 18. Default Settings for Message Routing		
Message Class	Virtual Screen	Options
CMS	CMS	NOALARM NONOTIFY
CP	CMS	NOALARM NONOTIFY
MESSAGE	MESSAGE	ALARM NOTIFY
WARNING	WARNING	ALARM NOTIFY
SCIF	MESSAGE	NOALARM NONOTIFY
NETWORK	NETWORK	NOALARM NOTIFY

ALARM

The alarm sounds when a message is received.

NOALARM

The alarm does not sound when a message is received.

NOTIFY

The message class indicator is shown in the status area when you receive a message.

NONOTIFY

The virtual screen name is not displayed in the status area when you receive a message.

Chapter 18. Tailoring the HELP Facility

One of the most useful features of the HELP facility is its versatility. You can tailor the displayed HELP to suit your needs by simply creating or changing CMS files.

You might want to generate new HELP files for any new commands, execs, or error messages you create. You may also want to make updates to the VM/SP HELP files to fulfill your individual needs.

If you have your own set of HELP files, you can do as you wish with them. However, if you share a set of HELP files with other system users, you will have to get authority from the system administrator to alter the HELP facility.

Creating HELP Files

Each existing HELP file has a distinct file name and file type. When you wish to write your own HELP files, there are naming conventions you must follow to ensure that your files are recognized by the VM/SP HELP facility.

Note: If your installation uses the Shared File System (SFS), you can create your own HELP files on minidisks or in any accessed directory. However, the system HELP disk cannot be replaced by a directory in a file pool.

Therefore, you can create your own HELP files, store them in an SFS directory, and HELP will display the files if the directory is accessed. For HELP files sent with VM/SP, installations will continue to use a minidisk. The HELP facility will access the minidisk if it cannot find the requested file on a directory or minidisk already in the user's search order.

File Names for HELP Files

When you change the existing HELP files or write your own HELP files, you must follow these file name conventions:

- The file name for components, commands, subcommands, or execs must be the exact full name of the component, command, subcommand, or exec.
- The file name for messages has the form xxxnnn(n)t, where:

xxx is the component code prefix (for example, DMS for CMS messages). See the *VM/SP System Messages and Codes* book for a list of the component code prefixes.

nnn(n) is the message number.

t is the message type code (for example, E for error messages in CMS).

For example, the HELP file name for the CMS message

No filename specified

would be DMS001E.

- For commands or statement names containing special characters, HELP creates the file name by translating that special character.

Tailoring the HELP Facility

If the name is a single special character, then the file name is the name of the special character. For example, & and ? have the file names of AMPRSAND and QUESMARK respectively. To display the HELP file XEDIT subcommand ?, you would enter HELP XEDIT ?. However, the actual file ID for that file would be QUESMARK HELPXEDI. The special characters are translated as follows:

Character	Translated To
?	QUESMARK
=	EQUAL
/	SLASH
"	DBLQUOTE
&	AMPRSAND
*	ASTERISK
.	PERIOD
<	LESSTHAN
>	MORETHAN

In command names or file names with more than one character, replace the special character with the first letter of its name. For example, "&" (AMPRSAND) would be replaced by an "A." An exception is "*", which is replaced by an "R."

Thus, the EXEC 2 statements &STACK and &EXIT would have the file names ASTACK and AEXIT. Remember that these changes only apply to the file names of the statements; they do not affect the way you call for a HELP file display. To display the HELP files for &STACK and &EXIT, you would enter HELP EXEC2 &STACK and HELP EXEC2 &EXIT.

File Types for HELP Files

You must also follow certain conventions for file types when you create your own HELP files:

- The file type of the HELP file is HELPxxxx, where xxxx is the name of the component to which the file belongs. For example, the file type for a CMS command would be HELPCMS.
- The file type for subcommands is HELPxxxx, where xxxx identifies the command name associated with this subcommand.
- The file type for messages is HELPMMSG.
- The file type for a list of supported commands for a given function is HELPMENU.
- The file type for a list of tasks supported for a given function is HELPTASK.
- The file type for a list of abbreviations and synonyms for a given function is HELPABBR.

If the component name is shorter than four characters, the file type is shortened. For example, HELPCP is the file type for CP commands. If the component name is

longer than four characters, only the first four characters are used. For example, HELPQUER is the file type for XEDIT QUERY subcommands.

The only exception to the above rule is for EXEC 2 HELP files. Because EXEC and EXEC 2 have the same first four characters, CMS examines the fifth character to determine if the request is for EXEC or EXEC 2. Because file types are limited to eight characters, CMS assigns the file type HELPEXEC to EXEC files and the file type HELPEXC2 to EXEC 2 files.

If you create a file with a file ID of HELP HELPMENU, the HELP facility displays your HELP HELPMENU file instead of the HELP HELPTASK file when HELP is generated without any other parameters.

There are also certain file types that are reserved for the various components of the HELP facility:

File Type	Reserved for
HELPABBR	Lists of command or subcommand abbreviations or synonyms for a component
HELPAVS	AVS commands
HELPCMS	CMS commands
HELPCMSQ	CMS QUERY operands
HELPCMSS	CMS SET operands
HELPCP	CP commands
HELPCPOT	CP commands for other than general users (privileged commands)
HELPCPQU	CP QUERY operands
HELPCPSE	CP SET operands
HELPEEDIT	EDIT subcommands
HELPEXEC	EXEC statements
HELPEXC2	EXEC 2 statements
HELPGROU	HELP files for the GROUP command
HELPHelp	HELP files for HELP
HELPIPCS	IPCS commands
HELPMACR	CMS assembler language macros
HELPMENU	Menus of HELP components
HELPMMSG	CP, CMS, GCS, IPCS, and TSAF messages
HELPPREF	XEDIT PREFIX subcommands
HELPQUER	XEDIT QUERY subcommands
HELPREXX	System Product Interpreter Statements
HELPROUT	CMS routines from the Callable Services Library (CSL)
HELPRSCS	Remote Spooling Communications Subsystem (RSCS) Networking (5664-188 Version 2.3) Program Product (only if you have this installed on your system.)

Tailoring the HELP Facility

HELPSSET	XEDIT SET subcommands
HELPSQLD	SQL/Data System (5688-004) Program Product (only if you have this installed on your system.)
HELPSRPI	Server-Requester Programming Interface subcommands
HELPTASK	HELP task menus
HELPTSAF	Transparent Services Access Facility commands
HELXPEDI	XEDIT subcommands

Examples of Naming Conventions

The following examples illustrate the naming conventions you would use to create specific files to work with the HELP command:

File Name	File Type	Description
ACCESS	HELPCMS	A CMS command description
BEGIN	HELPCP	A CP command description
ADD	HELXPEDI	An XEDIT subcommand description
DMS186W	HELPMMSG	A CMS message description
CMS	HELPMENU	A list of the CMS command and/or exec names supported by the HELP facility
HELP	HELPTASK	A list of HELP tasks supported by the HELP facility.
CMS	HELPABBR	A list of CMS abbreviations and synonyms supported by the HELP facility.

Creating Menus for HELP Files

The HELP facility has two types of menus:

- Menus that are command component menus, having a file type of HELPMENU
- Menus that are task menus, having a file type of HELPTASK.

The file name of a HELPMENU file is the name of the component to which the commands in the menu belong. For example, EXEC2 HELPMENU is the file name and file type for the menu containing EXEC 2 statements. Menus with a file type of HELPMENU contain a list of the HELP files for that component.

The file name of a HELPTASK file can be any file name up to eight characters and should describe the list of tasks contained in the file. For example, DISK HELPTASK is the file name and file type for the task menu containing tasks that involve using minidisks. Menus with a file type of HELPTASK contain a list of tasks.

Creating HELPMENU Files

There are a few rules you must follow when creating HELPMENU files:

1. Precede the list of names with any amount of information for the user. This information should include instructions for selecting an entry from the menu.
2. Include two blank lines between the information for the user and the list of names. If two consecutive blank lines are not found, then the file is considered preformatted and is not sorted or formatted by HELP. Any two or more consecutive blank lines indicate the end of the user information section and the beginning of the list of names for the HELP files for that component. Therefore, you are limited to one blank line between lines in the user information area.
3. Following these two blank lines, enter the file names in any order. File names must have the following characteristics:

- They must begin in column 1.
- The file names must be one to a line and limited to eight characters.

The list of names is sorted in ascending alphabetical order (in columns 1 thru 9) and is formatted by the HELP facility for display on the screen.

- Names of submenus and subtasks should be prefixed with an extra character, "*" (asterisk) or ":" (colon), allowing for a length of nine characters. If an entry on a menu is the name of another menu, the entry must be preceded by an "*" (asterisk). For example, on the CMS MENU, the REXX entry is preceded by an "*." This means that if you select the *REXX entry, HELP displays a menu of REXX commands.

If an entry on a menu is the name of a task menu, the entry must be preceded by a ":" (colon). For example, on the COMMANDS MENU, the TASK entry is preceded by a ":". If you select the :TASK entry, HELP displays a TASK MENU of tasks you may wish to perform.

The file name of the menu file determines the component ID for items selected from a menu. Thus, the menu file REXX HELPMENU is a menu of REXX keywords and functions. For example, when the ABBREV entry is selected from the REXX HELPMENU display, HELP looks for the ABBREV HELPREXX file and displays it.

It is possible to break down a set of commands by function and to create a menu that is a subset of a command set. A menu that is a list of the REXX built-in functions might be called FUNCTION HELPMENU. When you select an entry such as ABBREV from the FUNCTION HELPMENU, the ABBREV HELPFUNC file is displayed if it exists. This means that to have the ABBREV entry on both the REXX menu and the FUNCTION menu, you must have an ABBREV HELPFUNC file and an ABBREV HELPREXX file.

To simplify the creation of menus that are a subset of another component, an additional control word, .MT (Menu Type), is allowed in menu files. The .MT control word defines a component to override the component derived from the file name. Therefore, you could include the following in your FUNCTION HELPMENU file:

```
.MT REXX
```

This control word tells HELP to look for a HELPREXX file instead of a HELPFUNC file when you select an entry from the FUNCTION HELPMENU.

Tailoring the HELP Facility

The .MT control word is treated as a comment if it appears in other types of HELP files.

Example of HELPMENU File Creation

Assume you want to add HELP files concerning your internal system (System 5). procedures to the HELP facility. You would follow the procedure outlined below.

1. Choose the component name of SYS5 (System 5)
2. Create the HELP files for these procedures.
3. Following the rules given in "File Names for HELP Files" on page 309, give each procedure file a file name and file type. Thus, the file containing the information about CLASS8 (a class identifying the type of printing desired) would be named CLASS8 HELPSYS5.
4. Create a new file and give it a file name of SYS5 and a file type of HELPMENU. This file will be your menu.
5. In the first few lines of the menu, type in any descriptive information you wish to appear when the menu is displayed. Skip two lines after this information and list the names of all the files you want to access from the menu.

Your menu should look similar to the following example:

```
A file may be selected for viewing by placing the cursor under any
character of the file you want to display and pressing the ENTER key.

CLASS8
CLASS7
CLASS0
CLASSC
MOUNT
DEMOUNT
:
```

Figure 87. Sample of HELPMENU File Creation

When you specify HELP SYS5 MENU, the HELP facility will alphabetize and stack in columns the file names and display this file. You can then work with this menu as you would with any other HELP menu.

Creating HELPTASK Files

There are a few rules you must follow when creating HELPTASK files:

1. The file must start with a section of information for the user. This information should include instructions for selecting an entry from the task menu.
2. The file must include two blank lines between the information for the user and the list of names. Any two or more consecutive blank lines indicate the end of the user information section and the beginning of a list of tasks. Therefore, you are limited to one blank line between lines in the user information area.
3. Following the two blank lines is the task selection section. The lines in the selection section are vertically divided into two parts:

- a. The first part, which starts in column 1 and ends in column 24, contains the operands and options of the HELP command to be entered when the item is selected.
- b. The second part, starting in column 25, is the task description.

For example,

```
CMS PRINT (ALL          Print a file
|_____||_____
1                25
```

When you select the task "Print a file," the HELP file for the CMS PRINT command is displayed just as if you had entered HELP CMS PRINT (ALL on the command line.

Example of HELPTASK File Creation

Assume you want to add HELP files for your internal system tasks to the HELP facility. You would follow the procedure outlined in the following steps.

1. Choose the task name of INTERNAL (Internal Procedures).
2. Create the HELP files for these tasks.
3. Following the rules given in "File Names for HELP Files" on page 309, give each task file a file name and file type. Thus, the file containing the information about PRINT (printing a file on a local printer) would be named PRINT HELPINTE.
4. Create a new file and give it a file name of INTERNAL and a file type of HELPTASK. This file will be your task menu.
5. In the first few lines of the task menu, type in any descriptive information you wish to appear when the task menu is displayed. Skip two lines after this information and list in columns 1 through 24 the file names of the files created for Step 2. In the columns after 25, add a description of each task.

Your task menu should look similar to the following example.

A task may be selected for viewing by placing the cursor under any character of the task you want to display and pressing the ENTER key.


```
PRINT          Print a file on a local printer.
PHONE         Display a phone list for my department.
:
```

Figure 88. Sample of HELPTASK File Creation

When you specify HELP INTERNAL TASK, the HELP facility will alphabetize and stack in columns the file names and display this file. You may then work with this task menu as you would with any other HELP menu.

Creating Command HELP Files

When you request HELP for a specific command, there are several options available to display only a single subset of the information available for that command. These options are BRIEF, RELATED, DESCRIPT, FORMAT, PARMS, OPTIONS, NOTES, and ERRORS. In addition, the DETAIL option is used to display one or more of the DESCRIPT, FORMAT, PARMS, OPTIONS, NOTES, or ERRORS sections (See the *VM/SP CMS Command Reference* for information on these options.)

In creating a HELP file, you must identify the parts of the information that correspond to each option. You would do this by using the .CS control word to identify the beginning and end of each section of HELP text. You would note the beginning of a section by specifying “.CS xxx ON,” where ‘xxx’ is the corresponding keyword or number for that section. After the text for the section, you would specify “.CS xxx OFF.”

For example, before the text for a BRIEF section, you would specify “.CS BRIEF ON” or “.CS 0 ON.” At the end of the text, you would specify “.CS BRIEF OFF” or “.CS 0 OFF.” The text included in this section is displayed when you request BRIEF HELP for a command.

The following tables show the format of a HELP file. Remember, you can use the numbers shown in the table on the left or the keywords shown in the table on the right.

.CS 0 on	.CS BRIEF on
Text for BRIEF option	Text for BRIEF option
.CS 0 off	.CS BRIEF off
.CS 1 on	.CS DESCRIPT on
Text for DESCRIPT option	Text for DESCRIPT option
.CS 1 off	.CS DESCRIPT off
.CS 2 on	.CS FORMAT on
Text for FORMAT option	Text for FORMAT option
.CS 2 off	.CS FORMAT off
.CS 3 on	.CS PARMS on
Text for PARMS option	Text for PARMS option
.CS 3 off	.CS PARMS off
.CS 4 on	.CS OPTIONS on
Text for OPTIONS option	Text for OPTIONS option
.CS 4 off	.CS OPTIONS off
.CS 5 on	.CS NOTES on
Text for NOTES option	Text for NOTES option
.CS 5 off	.CS NOTES off
.CS 6 on	.CS ERRORS on
Text for ERRORS option	Text for ERRORS option
.CS 6 off	.CS ERRORS off
.CS 7 on	.CS RELATED on
Text for RELATED option	Text for RELATED option
.CS 7 off	.CS RELATED off

BRIEF HELP

BRIEF HELP provides a short description of a command, its basic syntax (without options), and an example. If you wish to follow the current VM/SP conventions, any BRIEF HELP sections you create would contain this information in a maximum of 10 lines and 75 columns of text.

DETAIL HELP

DETAIL HELP presents a complete description of a command, the command format, an explanation of its parameters and options, usage notes, and error information. It consists of the DESCRIPT, FORMAT, PARMS, OPTIONS, NOTES, and ERRORS sections.

The DESCRIPT Section: The DESCRIPT section should contain a description of the command.

The FORMAT Section: The FORMAT section should include a diagram of the command syntax, including any operands or options. You can use the same notational conventions used in the VM/SP HELP file. Here are the conventions:

- Use the less than (<) and greater than (>) symbols to denote choices where the user *must select one*. For example,

<A>

means that the user *must* specify A.

<A>

<C>

means that the user *must* specify either A, B, or C.

- The use of parentheses, bars, and plus signs means that the user does not have to select any of the items. For example,

(A)

means that the user *can* specify A or may omit the field.

+ +

|A|

|B|

|C|

+ +

means that the user *can* specify A, B, or C, or can omit the field.

The PARMS Section: The PARMS section should include a description of the parameters of the command.

The OPTIONS Section: The OPTIONS section should include a description of the options available for the command.

The NOTES Section: The NOTES section should include any usage notes and examples you wish to include in the HELP file.

The ERRORS Section: The ERRORS section contains any responses related to the command, and a sentence on how to use HELP to get information on a specific message. If you wish to add additional information to your own HELP files, simply add the errors text that you want displayed.

RELATED HELP

RELATED HELP provides information on commands that are similar to a command presently being displayed. It is especially useful if you are looking for similar functions or tasks but are unsure of the specific command name.

The information in a RELATED section must be in task menu format. See "Creating HELPTASK Files" on page 314 for details. The following screen sample shows the format of the RELATED section for the ERASE command:

```
.cs 6 off
.cs 7 on
.cm (c) Copyright IBM Corporation 1988

For RELATED information on removing files or parts of files from your
virtual machine, place the cursor under the topic of your choice and
press ENTER or the PF1 key.

XEDIT DELETE (BRIEF      DELETE% - Removes one or more lines from
XEDIT DELETE (BRIEF      %      a file while using XEDIT.
XEDIT DELETE (BRIEF
DISCARD TASK             DISCARD% - Removes files from your DIRLIST,
DISCARD TASK             %      FILELIST, MACLIST, PEEK, or
DISCARD TASK             %      RDRLIST screen.
DISCARD TASK
CMS ERASE (BRIEF        ERASE% - Removes files from your minidisk
CMS ERASE (BRIEF        %      or SFS directory.
CMS ERASE (BRIEF
CP PURGE (BRIEF         PURGE% - Removes spool files from your
CP PURGE (BRIEF         %      reader, printer, or punch.
CP PURGE (BRIEF

.cs 7 off
```

Figure 89. Format of the RELATED Section of the ERASE Command

Creating HELP Files for Messages

You may want to create your own HELP files for messages. The file names for your HELP files must conform to the file name conventions described in "File Names for HELP Files" on page 309. The file type for HELP files for messages is HELPMMSG.

The HELPMMSG files should display the message code and message, an explanation of why you received the message, the system action, and the user response. There is no need to use the .CS format word for HELP message files. The format for HELP message files is not as structured as that of command HELP files.

Following is the suggested format for the HELP message files:

Message Code and Message: This section lists the message code number and the format and possible variations of the message.

Explanation: The explanation should give a description of why the message was received. Depending on its complexity, you may want to list reasons separately, with a plan of corrective action for each one.

System Action: Again, depending on the complexity of the results of the message, you should give the return code, system status, and status of the command that was to be executed when the message appeared.

User Response: This section should describe what action(s) must be taken to correct the problem that caused the message.

You can use an existing HELP message file and the *VM/SP System Messages and Codes* book as guides when writing your HELP message files.

Highlighting Words within a File

When you create and format your own HELP files, you can also wish to highlight certain words for emphasis. The HELP facility lets you highlight selected portions of a file when those portions are viewed in display mode. To highlight a portion of a line, enclose that part of the line with the pairs of characters “`␣|`” and “`␣%`.” The “`␣|`” control character turns the highlighting on (white on a color terminal); the “`␣%`” control character turns the highlighting off. Everything on the line appears highlighted until the “`␣%`” control characters are encountered. Care must be taken when inserting the highlight control characters within a HELP file because the control characters appear as a single blank when displayed on the screen (or typed on a line-oriented terminal). This can affect the formatting of a file.

Note: With TASK menus or RELATED sections, the selection entries are automatically displayed by HELP as highlighted lines. If you desire to highlight only a portion of these lines, the control characters “`␣%`” should be used to turn highlighting off at the desired place in the file. You can use these characters as previously explained to turn highlighting on or off elsewhere in the line. The previous example of a RELATED section shows how “`␣%`” is used to turn highlighting off after selection entries.

Using Command Abbreviations

To use the shortened version or a synonym of a command name in your HELP files, you must create an abbreviation file with the proper entry. This file has the name of the component to which the command belongs as a file name and the file type of **HELPABBR**. The abbreviation file contains the command, its synonym, or the proper abbreviation for the command name, and the number that represents the shortest possible character string that is acceptable. If the number is omitted, the default is the length of the second word. The command should be in uppercase. Here is an example of an entry in an abbreviation file:

```
ACCESS  Access  2
```

Any of the following will generate the CMS ACCESS command HELP file:

```
HELP AC
HELP ACC
HELP ACCE
HELP ACCES
HELP ACCESS
```

An abbreviation file can also be used to provide help for command synonyms and for command or subcommand names that contain special characters other than those supported by the HELP facility.

Tailoring the HELP Facility

You should create an abbreviation file whenever a component is added to the HELP facility. The file should be given a file mode number of 2 (for example, CMS HELPABBR A2).

If no abbreviations are supported for the component, create an abbreviation file containing a comment line, indicated by an asterisk (*) in column 1. This lets HELP search more quickly for your HELP files.

Adding Your Own HELP Components

You can create your own HELP components for commands or applications you have added to your system. First, you must decide on the name for the component and create the HELP files you wish to be displayed for that component.

To display these HELP files, simply enter HELP *component name*, followed by the name of one of your commands. This will invoke the HELP facility. The file with the file name 'command name' and file type of HELPxxxx, where xxxx is the first four characters of the component name, is displayed.

You can also create a menu of your commands. See "Creating HELP Files" on page 309 for details.

If you have used the parsing facility and defined your command syntax using the Definition Language for Command Syntax (DLCS), you must also update the APPLID HELPABBR file. HELP uses this file to resolve synonyms, translations, and abbreviations for commands with syntax defined by using DLCS. When a user requests HELP for a command in your component, HELP uses the APPLID HELPABBREV file to locate the correct DLCS file application identifier.

For example, if you have created an application for generating monthly activity reports for your department called the Monthly Activity System, with a DLCS application identifier of MAS and a HELP component of MONTHLY, you should add the following line to the APPLID HELPABBR file:

```
MAS      MONTHLY
```

This tells HELP that when a user enters the command HELP MONTHLY PRI, HELP should search the MAS DLCS file and recognize that PRI is an abbreviation for the PRINT command in your application. HELP then displays your PRINT HELPMONT HELP file.

Note: The APPLID HELPABBR file follows the same HELPABBR file format. This means that you could specify an abbreviation for your component. The length of the abbreviation for a component in the APPLID file must always be at least four characters for HELP to find the component HELP files.

See *VM/SP Application Development Guide for CMS* for more information on using the parsing facility and defining command syntax using DLCS.

Using HELPCONV to Create HELP Files

HELPCONV is an additional text processing tool that helps you format your HELP files. HELPCONV control words do the following:

- Draw boxes to enclose tables, illustrations, or text
- Place comments within a file
- Separate the sections of a HELP file

- Cause concatenation of input lines and left- and right-justification of output
- Indent only the next input line the specified number of spaces
- Indent a series of input lines the specified number of spaces
- Indent the specified number of spaces for all but the first line in a series of input lines
- Override the derived component for a menu file
- Insert blank lines between output lines
- Change the final output representation of any input character.

The `HELPCONV` command converts an unformatted `HELP` file containing `SCRIPT` control words into a formatted `HELP` file. These control words are summarized in Table 19 on page 322. The following format words remain in the file:

```
.CS  
.CM  
.MT
```

The output file has the file type `$HLPxxxx`, where 'xxxx' represents the last four characters of the file type of the input file.

Tailoring the HELP Facility

Table 19. HELP and HELPCONV Control Word Summary				
Control Word	Operand Format	Function	Break	Default Value
.BX (Box)	V1 V2...Vn OFF	Draws horizontal and vertical lines in the blank columns around subsequent output text.	Yes	Draws a horizontal line.
.CM (COMMENT)	Comments	Places comments in a file for future reference.	Yes	
.CS (CONDITIONAL SECTION)	n ON/OFF keyword ON/OFF	Separates sections of HELP files.	Yes	
.FO (FORMAT MODE)	ON/OFF	Causes concatenation of input lines, and left and right justification of output.	Yes	On
.IL (INDENT LINE)	n +n -n	Indents only the next line the specified number of spaces.	Yes	0
.IN (INDENT)	n +n -n	Specifies the number of spaces subsequent text is to be indented.	Yes	0
.MT (MENU TYPE)	component	Correlates a component to a menu file when the component is not to be derived from the file name. For files other than menu files, .MT is seen as a comment.	Yes	
.OF (OFFSET)	n +n -n	Provides a technique for indenting all but the first line of a section.	Yes	0
.SP (SPACE)	n	Specifies the number of blank lines to be inserted before the next output line.	Yes	1
.TR (TRANSLATE)	s t	Specifies the final output representation of any input character.	No	

.BX (BOX)

The HELPCONV command inserts vertical and horizontal lines in the formatted output to enclose text, illustrations, or tables. The BOX control word defines and initializes a horizontal rule for output and defines vertical rules for subsequent output lines.

Now for some examples. The first time you enter the .BX control word, specify the columns in which you want the vertical lines to appear. The highest value that can be specified for a column is 239. Entering

```
.bx 1 10 20 30
```

results in the following output:

```
+-----+-----+-----+
```

Subsequently, entering the .BX control word with no operands causes HELPCONV to create a horizontal line with vertical bars at the columns indicated.

As HELPCONV formats each line, vertical bars are placed in the columns that you specified on .BX, unless a column is already occupied by a data character. In this case, HELPCONV does not place a vertical bar in the column.

The next example shows how you can change the vertical structure several times in succession. Entering

```
.bx 10 20
.sp
.bx 5 25
.sp
.bx 10 20
.sp
.bx 5 25
.sp
.bx 10 20
.sp
.bx off
```

results in:

```

      +-----+
      |         |
+----+-----+----+
|         |         |
+----+-----+----+
      |         |
+----+-----+----+
|         |         |
+----+-----+----+
      |         |
      +-----+
```

You can specify a .BX control word with different columns while a box is being drawn. When this happens, HELPCONV puts in vertical ascenders for all the old columns and vertical descenders for all the new columns. The vertical rules then appear in all subsequent output lines in the designated new columns.

The .BX control word causes a break in the text.

The column specification for the .BX control word uses a different rule than is than others used in HELPCONV. In some control words, the numbers in the control word do not represent columns, but displacements.

For example, the HELPCONV control word .IN 5 means that a blank character should be expanded to enough blanks to fill up *through* column 5; the next word starts in column 6. In the .BX control word, .BX 5 means to put vertical rules *in* column 5. Thus, you can use the same numbers for a .IN control word as for a .BX control word, and the vertical bar appears in the column immediately preceding the first word on that line.

Tailoring the HELP Facility

For example, consider the file called MARYHADA that looks like this:

```
.fo off
.bx 5 43
.in 5
Mary had a little lamb,
Whose fleece was white as snow,
And everywhere that Mary went,
The lamb was sure to go.
.bx off
```

This file, when processed by HELPCONV, creates the following output:

```
Mary had a little lamb,
Whose fleece was white as snow,
And everywhere that Mary went,
The lamb was sure to go.
```

.CM (Comment)

The .CM (Comment) control word places comments within a HELP file. Comments are useful for:

- Tracking files

This is a useful way for you to keep track of your changes to HELP files. You can include a comment that gives your name, the date and reason you created a file, and a future date when the file can be erased. HELP does not display any lines in a HELP file beginning with .CM. Therefore, you can include information about any alterations you have made to your HELP files in the files themselves.

- Documenting formats

If you use a special format in a HELP file that can be accessed by other people, you may want to place notes within the file explaining how to update the file.

- Place-holders

If a file is incomplete, you may want to put comments in the file where information should be added later.

For example, you could add the following to your HELP file:

```
.CM Remember to change the date.
```

You would only see the line above when you were editing the HELP file.

You can also use comments to store unique identifications for locating a specific region of the file during editing.

Note that comments cause a format break. Comment lines are retained in the formatted file and do not appear when the HELP file is displayed. They are not removed by the HELPCONV command.

.CS (Conditional Section)

The .CS (Conditional Section) control word was explained in “Creating Command HELP Files” on page 316. As we discussed earlier, the .CS control word lets you identify the specific sections of the input file that are directly associated with the HELP facility command *options*. These options are BRIEF, RELATED, DESCRIPT, FORMAT, PARMS, OPTIONS, NOTES, and ERRORS. (See the *VM/SP CMS Command Reference* for more information on these options).

For HELP command processing to display the appropriate information, the control word `.CS n ON` or `.CS keyword ON` is required before each section of the HELP text. The control word `.CS n OFF` or `.CS keyword OFF` should follow each HELP section. This statement tells HELP that the end of the section has been reached.

The keyword designating each section exactly corresponds to the HELP option that specifies the section to be displayed. These keywords cannot be abbreviated in the HELP file.

The .CS (Conditional Section) control word acts as a break. If blank lines or portions of a file are between the conditional sections (.CS), these lines are displayed with the DETAIL information. These conditional section lines are not removed from formatted output by the HELPCONV command.

.FO (Format Mode)

The .FO (Format Mode) control word cancels or restores concatenation of input lines and right justification of output lines.

Use `.FO ON` to restore default HELP formatting, including both justification and concatenation of lines. If you use the .FO control word with no operands, ON is assumed. Use `.FO OFF` to cancel concatenation of input lines and justification of output lines. Subsequent text is printed *as is*.

When format mode is in effect, lines are formed by shifting words to or from the next line (concatenation) and by padding with extra blanks to produce an aligned right margin (justification).

The .FO control word acts as a break. When format mode is in effect, a line without any blanks that exceeds the current line length is extended into the right margin. If a line is processed so that only one word fits on the line, the word is left justified.

If no formatting is done by HELPCONV, HELP description files *must* contain a `.FO OFF` control word before the section of text you wish to leave unformatted. `.FO OFF` should be placed as the first line of MENUS and TASK menus and before the RELATED section of a file.

For example, when `.FO OFF` is found, justification and concatenation are completed for the preceding line or lines, but the following lines are typed exactly as they appear in the file.

When `.FO ON` is found, justification and formatting are resumed with the next input line. Output from this point on in the file is padded to produce an aligned right margin on the output page.

Tailoring the HELP Facility

Note: You may not want this type of formatting in all cases; you may want certain output to exactly appear as it appears in your file or when presented on the prior releases of VM/SP CMS HELP. In this case, place .FO OFF in the file.

.IL (Indent Line)

Use the .IL (Indent Line) control word to set off paragraphs or portions of text by indenting them. This often improves the readability by emphasizing certain text. The .IL (Indent Line) control word indents *only the next line* a specified number of characters. The line is shifted to the right or left of the current margin (which includes any indent or offset values in effect).

When successive .IL control words are encountered without intervening text, or when you specify positive or negative increments for .IL control words entered without intervening text, the indent amount is modified to reflect the last .IL encountered; that is, the increments are added together. Thus the lines

```
.il 4  
.il +6
```

result in the next line being indented 10 spaces.

A negative value following .IL shifts the text to the left. If the resulting amount causes a shift to the left of character position one, an error message is generated.

The .IL control word acts as a break. Therefore, text accumulated before the .IL control word is processed and displayed before the next piece of text is processed.

Because the .IL control word causes a break in text, you may find it useful to indicate the beginning of a new paragraph. For example:

```
.il 3  
This line begins a new paragraph.  
.il 3  
This line begins another.
```

These lines result in:

```
    This line begins  
a new paragraph.  
    This line begins  
another.
```

.IN (Indent)

The .IN (Indent) control word changes the left margin displacement of HELP output.

For example,

```
This line is not indented.  
.in 5  
This line is indented.
```

results in:

```
This line is not indented.  
    This line is indented.
```

The .IN control word resets the current left margin. For example, .IN 5 sets the left margin at 6, leaving five blank spaces at the left. This indentation remains in effect

for all following lines until another .IN control word is encountered. .IN 0 cancels the indentation, and output continues at the original left margin setting.

.IN cancels any .OF (OFFSET) setting. The .OF 0 request cancels the current offset, but leaves the left margin specified by the .IN control word unchanged.

Because .IN causes a break, text accumulated before the .IN control word is processed and displayed, then the next text is processed.

The .IN control word effectively sets a new left margin for output text so that when you want text indented you do not have to enter blanks in front of the input lines (as you would for regular typing). HELPCONV continues to concatenate and justify input text lines that begin in column 1, but displays the output indented the number of spaces you specify.

Here is another example:

```
These few lines of text
are formatted
with enough words
.in 5
so that you can
see how HELP's formatting
process
.in +3
continues and may
.in -6
even be reversed, by using a
negative value.
```

These lines result in:

```
These few lines of
text are formatted
with enough words
    so that you can
    see how HELP's
    formatting
    process
        continues and
        may
    even be reversed,
    by using a negative
    value.
```

In this example, the first .IN control word (.in 5) shifts output to the right five spaces so that text begins in column 6. The second .IN control word (.in +3) requests that the current indentation increase by three spaces so the left margin is now in column 9. When you supply a negative value with the .IN control word (.in -6), the margin is shifted to the left.

.MT (Menu Type)

Use the .MT (Menu Type) control word to specify the component of a menu. The .MT control word overrides the default component of a menu file. When a menu file is used, the file name of the menu generates the name of the component. This component locates the appropriate HELP file when a selection is made. For example, if you select a command from the XEDIT menu, it is equivalent to entering the HELP XEDIT command. If the line .MT xxxxx was included in the file,

selecting a command from the menu would be equivalent to entering `HELP XXXXX` command.

The `.MT` control word assists in the creation of menus that are a subset of another menu. For example, a menu that contains a list of REXX functions might be called `FUNCTION HELPMENU`. In this case, the `HELP` files for the individual functions are only located if they are duplicated under the file type `HELPFUNC`. The `.MT` control word defines a component ID to override that derived from the file name. The `FUNCTION` menu could include:

```
.mt REXX
```

This specifies that the menu contains a list of REXX commands and thus are found under the file type `HELPREXX`.

The `MENU TYPE` control word acts as a break.

.OF (Offset)

The `.OF` (Offset) control word indents all but the first line of a block of text. An offset differs from an indentation. Offsets do not affect the first line immediately following the control word; the second and subsequent input lines are indented the specified number of characters. This is useful when formatting numbered lists where text is blocked to the right of the number.

The `.OF` control word does not take effect until after the next line is formatted. The indentation remains in effect until an `.IN` (Indent) control word or another `.OF` (Offset) control word is encountered.

You can use the `.OF` control word within a section that is also indented with the `.IN` control word. Note that `.IN` settings take precedence over `.OF`; however, any `.IN` request causes a previous offset to be cleared.

If you want to start a new section with the same offset as the previous section, you need only repeat the `.OF n` request.

This control word acts as a break; subsequent text is printed at the current left margin, that is, whatever the indentation is (0, if no `.IN` control word is in effect).

`.OF` shifts all but the first line of text. You can use the `.IL` (Indent Line) control word to shift only the next line to the left or right of the current margin.

Following is an example to try:

1. Starting an offset:

```
.of 10
```

The line immediately following the `.OF` control word is printed at the current left margin. All lines thereafter (until the next indent or offset request) are indented 10 spaces from the current margin setting. These two examples were processed with offset control words in the positions shown.

2. Ending an offset:

```
.of
```

The effect of any previous `.OF` request is canceled, and all output after the next line continues at the current left margin setting.

.SP (Space Lines)

The .SP (Space Lines) control word leaves blank lines between text lines of output.

For example

```
.sp 5
```

indicates that you want to leave five lines of space in the text output. You can use multiple spaces when you want a heading or a title to stand out, for example the lines

```
A Love Story
.sp 5
The quick brown fox
was eager
to meet the pretty poodle.
```

results in:

```
A Love Story
```

```
The quick brown fox
was eager to meet the
pretty poodle.
```

.TR (Translate Character)

The .TR (Translate Character control) word lets you specify the output representation of each character in the source text. For example, you could specify that all blanks in the file appear as question marks in the output.

After formatting of an input source line has been completed and immediately before actual output, each character of the output line may be translated to a different output code.

Translate character specifications remain in effect until explicitly respecified. Because control words are only internally processed, they are never translated in the file.

A .TR control word with no operands causes the translation table to be reinitialized and all previously specified translations to be reset.

The .TR control word does not cause a break. If you have a section of text that has translation characters in effect, followed by a .TR to reset the translations, the last line of the text may not yet have been printed. In this case, that last line is not translated.

For example

```
.tr 40 ?
```

This causes all blanks in the file to be typed as question marks (?) on output.

Changing Existing HELP Files

Because all HELP files are CMS files, you can add or delete files or menus or change any existing file or menu. However, there are a few restrictions you must follow when tailoring HELP files; they are discussed in the following sections.

Note: If you tailor your HELP files, you should retain documentation of the changes you have made by using the .CM control word to indicate that what follows is a comment. You can use this documentation later to help you update your files when IBM issues updates to the HELP facility files.

Adding HELP Files

The HELP facility lets you:

- Add HELP files to existing components or create a new component with its own HELP files
- Modify the command and message description files IBM provides with additional description files of your choice
- Produce a formatted HELP file by entering the HELPCONV command and the HELP control words when creating the HELP description file.

To create your own HELP file, follow the instructions in “Creating HELP Files” on page 309.

If you add HELP files to an existing component, you should follow the naming conventions for HELP files given in this chapter. If you update a component, you should also update its menu. For information on menus see “Creating Menus for HELP Files” on page 312.

A file that contains control words other than .CM, .CS, .FO, or .MT and has not been processed by the HELPCONV command is identified by HELP as being unformatted or containing extra control words.

Note: Remember that if your installation uses the Shared File System (SFS), you can create your own HELP files on minidisks or in any accessed directory. However, the system HELP disk cannot be replaced by a directory in a file pool.

Deleting HELP Files

You delete HELP files just as you delete any CMS file, by specifying ERASE file name file type. If you delete a file, you should delete the file name from the menu for that component and also from any task file.

Altering Existing HELP Files

To alter a HELP file:

1. Edit the file with a text processing editor
2. Add or delete as you wish, making sure that you follow the instructions given in “Creating HELP Files” on page 309.
3. When using HELPCONV on a file that contains a RELATED section, formatting should be turned off for the entire RELATED section.

If your installation maintains several releases of VM/SP CMS, and runs the HELP facility on them, you can modify the VM/SP CMS Release 4.0 (and following releases) formatted informational files so that they appear the same on all the

releases. You would do this by adding the .FO OFF control word as the first statement of the HELP file and changing any unsupported control words to comments. For example, to let a preformatted file be used on VM/SP CMS Release 3.0, place .FO OFF before the informational HELP text lines. The .FO OFF control word tells HELP not to format the files but to display the information in its present format.

You must also change any .CS keyword ON/OFF lines to use the section number instead of the keyword. For example, .CS DESCRIPT ON would become .CS 1 ON. The .CS lines for the BRIEF and RELATED sections must also be changed to .CM, or the text for those sections must be included in a different section.

Note: When using HELPCONV on a HELPTASK or HELPMENU file, formatting should be turned off for the entire file.

Changing Menus

If you add, delete or change files, you must change the associated menu. Edit the menu file (the file name is component name; the file type is HELPMENU) and make the necessary changes. Remember, there is an eight-character limit on file names (a nine-character limit for submenus and subtasks). Only one file name goes on a line, and you can insert file names anywhere in the list. If you delete a HELP file, you should delete from all HELPMENU files any line on which the file name occurs.



Appendix A. Summary of CMS Commands

Table 20 on page 335 contains an alphabetical list of CMS commands and the functions performed by each. These commands are described in detail in the *VM/SP CMS Command Reference*. Table 21 on page 341 describes the special CMS commands that can be used while in full-screen environments (such as FILELIST).

There are 10 commands called Immediate commands that are handled in a different manner from the other commands listed in Table 20 and Table 21. They may be entered while another command is executing by pressing the Attention key (or its equivalent) and are executed immediately.

The Immediate commands are:

HB	Halt batch execution
HI	Halt Interpretation
HO	Halt tracing
HT	Halt typing
HX	Halt execution
RO	Resume tracing
RT	Resume typing
SO	Suspend tracing
TE	Trace end
TS	Trace start

You can define your own Immediate commands by using any of the following:

- IMMCMD macro in an assembler language program
- IMMCMD command within an exec (CMS EXEC, EXEC 2, System Product Interpreter)
- NUCXLOAD command with the IMMCMD option specified.

Border commands are single-character windowing commands that you may enter in the corners of window borders. Following is a list of Border commands:

B	Scrolls the window backward
C	Clears the window of data
D	Drops the window
F	Scrolls the window forward
H	Hides the window
L	Scrolls the window to the left
M	Changes the location of the window
N	Minimizes the window
O	Restores the window

- P** Pops the window
- R** Scrolls the window to the right
- S** Changes the size of the window
- X** Maximizes the window

Table 20 (Page 1 of 7). CMS Commands Described in CMS Command Reference	
Command	Usage
ACCESS	Allows you to access a minidisk or an SFS directory with a file mode letter.
ALARM VSCREEN	Sounds the terminal alarm the next time the display is refreshed.
AMSERV	Uses access method services utility functions to create, alter, list, copy, delete, import, or export VSAM catalogs and data sets.
ASSEMBLE	Assembles assembler language source code.
ASSGN	Assigns or unassigns a CMS/DOS system or programmer logical unit for a virtual I/O device.
CATCHECK	Allows a CMS VSAM user (with or without DOS set ON) to use the VSE/VSAM Catalog Check Service Aid to verify a complete catalog structure.
CLEAR VSCREEN	Erases data in the virtual screen by overwriting the data buffer with nulls.
CLEAR WINDOW	Scrolls past all data in the virtual screen to which the window is connected so that no data is displayed in the data area of the window.
CMDCALL	Converts EXEC 2 extended plist function calls to CMS extended plist command calls.
CMSBATCH	Calls the CMS batch facility.
CMSSERV	Starts IBM Enhanced Connectivity Facilities communications between your VM/SP host system and your workstation (IBM Personal Computer).
COMPARE	Compares records in CMS files.
CONVERT COMMANDS	Converts a CMS file containing Definition Language for Command Syntax (DLCS) statements into an internal form for the parsing facility.
CONWAIT	Causes a program to wait until all pending terminal I/O is complete.
COPYFILE	Copies CMS files from one minidisk to another, one SFS directory to another, or between minidisks and directories.
CP	Enters CP commands from the CMS environment.
CREATE ALIAS	Places an additional name for a file in a specified directory.
CREATE DIRECTORY	Creates an SFS directory.
CREATE LOCK	Creates an explicit lock on an SFS directory or a file in an SFS directory.
CREATE NAMEDEF	Assigns a temporary name which can be used by a program, instead of a file name and file type or a fully-qualified directory name.
CSLLIST	Lists information about all members of a specified callable services library.
CURSOR VSCREEN	Positions the cursor on specified line and column in a virtual screen.
DEBUG	Displays state of virtual machine at time ofabend.
DEFAULTS	Sets or displays default options for various commands.

Table 20 (Page 2 of 7). CMS Commands Described in CMS Command Reference

Command	Usage
DEFINE VSCREEN	Creates a virtual screen.
DEFINE WINDOW	Creates a window.
DELETE LOCK	Releases the explicit lock placed on a directory or a file in a directory by the CREATE LOCK command.
DELETE NAMEDEF	Deletes the temporary name given to a file or directory by the CREATE NAMEDEF command, and makes it no longer usable by a program.
DELETE VSCREEN	Removes a virtual screen definition.
DELETE WINDOW	Removes a window definition.
DESBUF	Clears the program stack and the terminal input buffers.
DIRLIST	Lists directories of a specified directory structure in a full screen environment.
DISK	Performs disk-to-card and card-to-disk operations for CMS files. Can be used for files residing on minidisk or in directories.
DLBL	Defines a VSE file name or VSAM <i>ddname</i> and relates that name to a file.
DOSLIB	Deletes, compacts, or lists information about the phases of a CMS/DOS phase library.
DOSLKED	Link-edits CMS text decks or object modules from a VSE relocatable library and places them in executable form in a CMS/DOS phase library.
DROPBUF	Eliminates a program stack buffer.
DROP WINDOW	Moves a window down in the order of displayed windows.
DSERV	Displays information contained in the VSE core image, relocatable, source, procedure, and transient directories.
EDIT	Uses the VM/SP System Product Editor in CMS editor (EDIT) compatibility mode to create or modify a file residing on a minidisk or in an SFS directory.
ERASE	Deletes CMS files from a minidisk, or deletes both files and subdirectories from an SFS directory.
ESERV	Displays, punches or prints an edited (compressed) macro from a VSE source statement library (E sublibrary).
EXEC	Executes special procedures made up of frequently used sequences of commands.
EXECDROP	Purges storage-resident execs.
EXECIO	Does I/O operations between a device and the program stack or a variable.
EXECLOAD	Loads execs into storage.
EXECMAP	Lists storage-resident execs and displays execs in saved segments.

Table 20 (Page 3 of 7). CMS Commands Described in CMS Command Reference

Command	Usage
EXECOS	Resets the OS and VSAM environments under CMS without returning to the interactive environment.
EXECSTAT	Obtains status of a specific exec.
EXECUPDT	Produces an updated executable version of a System Product Interpreter source program.
FETCH	Fetches a CMS/DOS or VSE executable phase.
FILEDEF	Defines an OS <i>ddname</i> and relates that <i>ddname</i> to any device supported by CMS or to a file residing on a minidisk or in an SFS directory.
FILELIST	Lists information about CMS files on a minidisk or in an SFS directory, with the ability to edit and issue commands from the list.
FINIS	Closes an open file on a minidisk or in a directory.
FORMAT	Prepares minidisks in CMS fixed block format.
GENDIRT	Fills in auxiliary module directories.
GENMOD	Generates nonrelocatable CMS files (MODULE files).
GENMSG	Converts a message repository file into an internal form.
GET VSCREEN	Writes data from a CMS file to the specified virtual screen.
GLOBAL	Identifies specific CMS libraries to be searched for macros, copy files, missing subroutines, LOADLIB modules, or DOS executable phases.
GLOBALV	Sets, maintains and retrieves a collection of named variables.
GRANT AUTHORITY	Authorizes other users to read and modify your SFS directories or the files in the directories.
HELP	Displays information about CP, CMS, or user commands, EDIT or XEDIT subcommands, EXEC, EXEC 2 and System Product Interpreter control statements, and descriptions of CMS and CP messages.
HELPCONV	Converts a CMS file into an acceptable form to be used by the HELP facility.
HIDE WINDOW	Prevents the specified window from being displayed, and connects the window to a virtual screen.
IDENTIFY	Displays or stacks user ID, node ID, RSCS ID, date, time, time zone, and day of the week.
IMMCMD	Establishes or cancels Immediate commands from within an exec.
INCLUDE	Brings additional TEXT files into storage and establish linkages.
LABELDEF	Specifies standard HDR1 and EOF1 tape label description information for CMS, CMS/DOS, and OS simulation.
LISTDIR	Lists directories in a specified directory structure.
LISTDS	Lists information about data sets and space allocation on OS, DOS, and VSAM minidisks.

Table 20 (Page 4 of 7). CMS Commands Described in CMS Command Reference

Command	Usage
LISTFILE	Lists information about CMS files stored on a minidisk or in an SFS directory.
LISTIO	Displays information concerning CMS/DOS system and programmer logical units.
LKED	Link-edits a CMS TEXT file or OS object module into a CMS LOADLIB.
LOAD	Brings TEXT files into storage for execution.
LOADLIB	Maintains CMS LOADLIB libraries.
LOADMOD	Brings a single MODULE file into storage.
MACLIB	Creates or modifies CMS macro libraries.
MACLIST	Lists information about all members in a specified maclib, with the ability to edit and issue commands from the list.
MAKEBUF	Creates a new program stack buffer.
MAXIMIZE WINDOW	Expands a window to the physical screen size.
MINIMIZE WINDOW	Reduces the size of the window to one line.
MODMAP	Displays the load map of a MODULE file.
MOREHELP	Obtains either additional or related information about the latest valid HELP command you issued.
MOVEFILE	Moves data from one device to another device of the same or a different type.
NAMEFIND	Displays/stacks information from a NAMES file. (NAMES file = file name of <i>userid</i> and file type of NAMES).
NAMES	Displays a menu to create, display or modify entries in a 'userid NAMES' file. (The menu is available only on display terminals.)
NETDATA	Used from an exec to query, receive or send files to users at a network node or on your system.
NOTE	Prepares a 'note' for one or more computer users, to be sent via the SENDFILE command.
NUCXDROP	Deletes specified nucleus extensions.
NUCXLOAD	Loads a nucleus extension.
NUCXMAP	Identifies existing nucleus extensions, including those residing in saved segments.
OPTION	Changes the DOS/VS COBOL compiler (FCOBOL) options that are in effect for the current terminal session.
OSRUN	Loads, relocates, and executes a load module from a CMS LOADLIB or OS module library.
PARSECMD	Calls the parsing facility from within an exec.
PEEK	Displays a file that is in your virtual reader without reading it onto a disk or directory.

Table 20 (Page 5 of 7). CMS Commands Described in CMS Command Reference

Command	Usage
POP WINDOW	Moves a window up in the order of displayed windows.
POSITION WINDOW	Changes the location of a window on the physical screen.
PRINT	Spools a specified CMS file to the virtual printer.
PROGMAP	Displays or places on the program stack information on programs currently loaded in storage or in a saved segment.
PSERV	Copies a procedure from the VSE procedure library onto a CMS minidisk or an SFS directory, displays the procedure at the terminal, or spools the procedure to the virtual punch or printer.
PUNCH	Spools a copy of a CMS file to the virtual punch.
PUT SCREEN	Makes a copy of the physical screen and writes the image to a CMS file.
PUT VSCREEN	Writes the data from the data area of a virtual screen to a CMS file.
QUERY	Requests information about CMS files, minidisks or Shared File System (SFS) directories or other virtual machine characteristics.
RDR	Generates a return code and either displays or stacks a message that identifies the characteristics of the next file in your virtual reader.
RDRLIST	Displays information about files in your virtual reader with the ability to issue commands from the list.
READCARD	Reads data from spooled card input device.
RECEIVE	Reads to your SFS directory or minidisk a file or note that is in your virtual reader.
REFRESH	Updates virtual screens and their associated windows.
RELEASE	Releases an SFS directory or minidisk previously accessed.
RELOCATE	Moves a file or directory structure from one directory to another that you own within the same file pool.
RENAME	Changes the name of a CMS file or directory.
RESERVE	Allocates all available blocks of a 512-, 1K-, 2K-, or 4K-byte block formatted minidisk to a unique CMS file.
RESTORE WINDOW	Returns a maximized or minimized window to its size and location prior to the maximize or minimize.
REVOKE AUTHORITY	Cancels authorities that you granted for a directory or files in a directory.
ROUTE	Directs data of a particular message class to a virtual screen.
RSERV	Copies a VSE relocatable module onto a CMS minidisk or SFS directory, displays it at the terminal, or spools a copy to the virtual punch or printer.
RTNDROP	Cancels the binding of a callable services library routine.
RTNLOAD	Searches for, loads, and binds a callable services library routine to a fixed location in storage, and makes it available for invocation.

Table 20 (Page 6 of 7). CMS Commands Described in CMS Command Reference

Command	Usage
RTNMAP	Displays information about the callable services library routines that are currently loaded and bound to an address.
RTNSTATE	Obtains the status of one or more specific callable services library routines.
RUN	Initiates series of functions to be performed on a source, MODULE, TEXT, or EXEC file.
SCROLL	Moves a window to a new location on the virtual screen.
SEGMENT ASSIGN	Indicates the logical segment to be associated with the physical segment.
SEGMENT LOAD	Loads a saved segment.
SEGMENT PURGE	Purges a saved segment.
SEGMENT RELEASE	Releases the storage held by a segment space.
SEGMENT RESERVE	Creates a segment space for subsequent loading.
SENDFILE	Sends files or notes to one or more computer users, locally or remotely attached, by issuing the command or by using a menu (display terminal only.)
SENTRIES	Determines the number of lines currently in the program stack.
SET	Establishes, sets, or resets CMS virtual machine characteristics.
SETKEY	Changes settings for CMS storage keys.
SETPRT	Loads a virtual 3800 printer.
SHOW WINDOW	Places a window on top of all other displayed windows and connects the window to a virtual screen.
SIZE WINDOW	Changes the number of lines and columns for a specified window.
SORT	Arranges a specified file in ascending order according to sort fields in the data records.
SSERV	Copies a VSE source statement book onto a CMS minidisk or SFS directory, displays it at the terminal, or spools a copy to the virtual punch or printer.
START	Begins execution of programs previously loaded (OS and CMS) or fetched (CMS/DOS).
STATE	Verifies the existence of a CMS file on an accessed minidisk or in an SFS directory.
STATEW	Verifies a file on a read/write file mode.
SVCTRACE	Records information about supervisor calls.
SYNONYM	Uses a table containing synonyms you have created for CMS and user-written commands.
TAPE	Transfers files between tapes and minidisks (or SFS directories), positions tapes, and displays or writes VOL1 labels.
TAPEMAC	Creates CMS MACLIB libraries directly from an IEHMOVE-created partitioned data set on tape.

Table 20 (Page 7 of 7). CMS Commands Described in CMS Command Reference	
Command	Usage
TAPPDS	Loads OS partitioned data set (PDS) files or card image files from tape to minidisk or SFS directory.
TELL	Sends a message to one or more computer users who are logged on to your computer or to one attached to yours via RSCS.
TXTLIB	Generates and modifies text libraries.
TYPE	Displays all or part of a CMS file at the terminal.
UPDATE	Makes changes in a program source file as defined by control cards in a control file.
VALIDATE	Verifies the syntax of a file identifier and verifies whether or not a file mode is accessed.
WAITREAD VSCREEN	Used from an exec to update the virtual screen with data, refresh the physical screen, and wait for the next attention interrupt.
WAITT VSCREEN	Updates the virtual screen with data.
WRITE VSCREEN	Enters information in a virtual screen.
XEDIT	Uses the VM/SP System Product Editor to create or modify a CMS file.
XMITMSG	Retrieves a message from a CMS message repository file or your own message repository file.

Special CMS Commands Used from Full-Screen Environments

Table 21. Special CMS Commands Used from Full-Screen Environments	
Command	Usage
AUTHLIST	Displays authority information. May be issued only from the DIRLIST or FILELIST environments.
ALIALIST	Displays alias information. May be issued only from the FILELIST environment.
DISCARD	Erases a file (or SFS directory, or MACLIB member) that is displayed from the DIRLIST, FILELIST, MACLIST, RDRLIST, or PEEK command environments.
EXECUTE	Use to issue commands (or execs) from the CSLLIST, DIRLIST, FILELIST, MACLIST, and RDRLIST environments.

Appendix B. Summary of CP Commands

The CP commands are interactive console functions that control the VM/SP system and your virtual machine. The commands you can enter depend on your assigned privilege class(es), as described in this section. If appropriate, CP shows its processing results with responses (for example: COMMAND COMPLETE, MISSING ARGUMENT, or INVALID OPTION).

Privilege Classes for CP Commands

Each CP command has one or more of the following function types:

- Operations
- Resource
- Programmer
- Spooling
- Analyst
- CE (Customer Engineer - Service)
- General.

The IBM-defined class structure is based on these seven function types. A command keeps its function type even if your installation establishes its own class structure. In most cases, each command class (A-G) has a corresponding function type (O, R, P, S, A, C or G). Some commands fall into more than one class. Table 22 on page 344 shows the function of each privilege class and function type. Then, each CP command is listed in a table, along with its IBM-defined privilege class, function type, and a description of what each command does. If your installation changes the class for a command, record the change in the blank column in the summary table.

Your installation assigns each user, as part of the user's entry in the CP directory, one or more privilege classes. The exceptions are users with a password of NOLOG. These users have no privilege class and can only:

- Send messages
- Receive spooled output as punched cards or printed forms.

The NOLOG password identifies them to receive spooled output when a virtual machine user spools output for them.

The CP Class *Any* commands are available to all user classes. These commands perform basic functions required by all virtual machines, such as:

- Logging on
- Logging off
- Sending messages.

If you try to enter a command that does not have your command class, CP does not execute the command and issues an error message.

Note: If your installation adds or removes any commands from the general user class (IBM-defined class G), your installation should update the HELP files to show these changes. See Chapter 18, "Tailoring the HELP Facility" for more information.

This table shows the different privilege classes, the function codes, and the major tasks that can be performed for each privilege class.

Table 22. CP Privilege Classes		
IBM-Defined Class	Function	Function, Primary User, and Use
A	O	<p>Operations - Primary system operator</p> <p>The system assigns class A to the user at the VM/SP console during IPL. The class A user is responsible for the availability, communication lines, and resources for VM/SP. These commands control system accounting, broadcast messages, run virtual machine performance options and affect VM/SP performance.</p> <p>Note: The Class A system operator, who is automatically logged on during CP initialization, is designated as the primary system operator.</p>
B	R	<p>Resource - System Resource Operator</p> <p>These commands control allocation and deallocation of real resources of the VM/SP system, except those that the primary system operator and the spooling operator control.</p>
C	P	<p>Programming - System Programmer</p> <p>These commands update functions of the VM/SP system and change real storage in the real machine.</p>
D	S	<p>Spooling - Spooling Operator</p> <p>These commands control spool data files and specific functions of the unit record equipment of the system.</p>
E	A	<p>Analyzing - System Analyst</p> <p>These commands examine and save certain data in the VM/SP storage area.</p>
F	C	<p>CE -Service Representative (Customer Engineer)</p> <p>These commands get and examine data about input and output devices connected to the VM/SP system.</p>
G	G	<p>General - General User</p> <p>These commands control functions to run users' virtual machines.</p>
Any	None	<p>These commands, available to any user, help you gain and take away access to the VM/SP system.</p>

CP Commands

This section lists the following:

- The CP class G and class Any commands in alphabetical order
- The corresponding function type
- The IBM-defined privilege classes that can execute the command
- A partially blank column in which you can record changes your installation makes to the class of a command.
- A brief description of each command.

Note: Brackets indicate type is optional in the OVERRIDE statement.

Command	Function Type	IBM-Defined Privilege Class	User-Defined Class	Usage
*	N/A	Any	None	Annotate the console sheet.
#CP	N/A	Any	None	Execute a CP command while remaining in the virtual machine environment.
ADSTOP	<G>	G		Halt execution at a specific virtual machine instruction address.
ATTN	<G>	G		Make an attention interruption pending for the virtual machine console.
BEGIN	<G>	G		Continue or resume execution of the virtual machine at either a specific storage location or at the address in the current PSW.
CHANGE	G	G		Alter one or more attributes of a closed spool file.
CLOSE	<G>	G		Terminate spooling operations on a virtual card reader, punch, printer, or console.
COMMANDS	N/A	Any	None	Display the commands and diagnose codes you are authorized to use.
COUPLE	<G>	G		Connect channel-to-channel devices.
CP	N/A	Any	None	Execute a CP command while remaining in the CMS virtual machine environment.
DEFINE	G	G		Reconfigure your virtual machine.
DETACH	G	G		Detach a virtual device from a virtual machine. Detach a channel from your virtual machine.

Table 23 (Page 2 of 4). CP Command Summary

Command	Function Type	IBM-Defined Privilege Class	User-Defined Class	Usage
DIAL	N/A	Any	None	Connect a terminal or display device to the virtual machine's virtual communication line.
DISCONN	N/A	Any	None	Disconnect your terminal from your virtual machine.
DISPLAY	<G>	G		Display virtual storage on your terminal.
DUMP	<G>	G		Print the following on the virtual printer: Virtual PSW, general registers, floating-point registers, storage keys, and contents of specified virtual storage locations.
ECHO	<G>	G		Test terminal hardware by redisplaying data entered at the terminal.
EXTERNAL	<G>	G		Simulate an external interruption for a virtual machine and return control to that machine.
INDICATE	G	G		Indicate resource utilization and contention.
IPL	<G>	G		Simulate IPL for a virtual machine.
LINK	<G>	G		Provide access to a specific DASD by a virtual machine.
LOADVFCB	<G>	G		Load virtual forms control buffer for a virtual 3203, 3262, 3289E, 3211, 4245, or 4248 printer.
LOGOFF	N/A	Any	None	Disable access to CP.
LOGON	N/A	Any	None	Provide access to CP.
MESSAGE	N/A	Any	None	Transmit messages to other users.
NOTREADY	<G>	G		Simulate NOT READY for a device to a virtual machine.
ORDER	G	G		Rearrange closed spool files in a specific order.
PER	<G>	A, B, C, D, E, F, G		Monitor certain events in the user's virtual machine as they occur during program execution.
PURGE	G	G		Remove closed spool file from the system.
QUERY	G	G		Request information about machine configuration and system status.

Table 23 (Page 3 of 4). CP Command Summary

Command	Function Type	IBM-Defined Privilege Class	User-Defined Class	Usage
READY	<G>	G		Simulate device end interruption for a virtual device.
REQUEST	<G>	G		Make an attention interruption pending for the virtual machine console.
RESET	<G>	G		Clear and reset all pending interruptions for a specified virtual device and reset all error conditions.
REWIND	<G>	G		Rewind (to load point) a tape and ready a tape unit.
SCREEN	<G>	G		Change the color and extended highlighting values of the virtual machine.
SEND	<G>	G		Pass commands and message replies to disconnected virtual machine for processing. Pass message text to logical printer.
SET	G	G		Control various functions within the virtual machine.
SLEEP	N/A	Any		Place the virtual machine in a dormant state but allow messages to be displayed.
SMSG	<G>	G		Send special messages to specified virtual machine.
SPOOL	<G>	G		Alter spooling control options; direct a file to another virtual machine or to a remote location through the RSCS virtual machine.
STORE	<G>	G		Alter specified virtual storage locations and registers.
SYSTEM	<G>	G		Simulate RESET, CLEAR STORAGE, and RESTART buttons on a real system console.
TAG	<G>	G		Specify variable information to be associated with a spool file or output unit record device. Interrogate the current TAG text setting of a given spool file or output unit record device.
TERMINAL	<G>	G		Define or redefine the input and attention handling characteristics of your virtual console.

Table 23 (Page 4 of 4). CP Command Summary

Command	Function Type	IBM-Defined Privilege Class	User-Defined Class	Usage
TRACE	<G>	G		Trace specified virtual machine activity at your terminal, spooled printer, or both.
TRANSFER	G	G		Transfer input files or reclaim input files from a specified user's virtual card reader.
VMDUMP	<G>	G		Dump virtual machine (use VM/SP IPCS to view dump.)

Appendix C. Considerations for Full-Screen CMS and Windowing

The information in this section consists of a basic overview of full-screen CMS and its functions and default settings. This section also includes specific information regarding CMS, CP, System Product Editor, and application interactions with full-screen CMS. The information will assist users working in these areas in making a smooth transition from line-mode to full-screen CMS.

Windowing Commands

There are several commands in VM/SP that support windowing and full-screen CMS. For the most part, these commands are two-word commands, consisting of a verb followed by a noun. The verb is the action requested, and the noun is the object of the action.

For example, `DEFINE WINDOW` is the command used to create a window. The actual command name is the verb `DEFINE`.

Table 24 on page 350 contains an alphabetical listing of the full-screen CMS commands.

For more information on any CMS command, see the *VM/SP CMS Command Reference*.

Table 24 (Page 1 of 2). CMS Command Summary

Command	Usage
ALARM VSCREEN	Sounds the terminal alarm the next time the display is refreshed.
CLEAR VSCREEN	Erases data in the virtual screen by overwriting the data buffer with nulls.
CLEAR WINDOW	Scrolls past all data in the virtual screen to which the window is connected so that no data is displayed in the data area of the window.
CURSOR VSCREEN	Positions the cursor on specified line and column in a virtual screen.
DEFINE VSCREEN	Creates a virtual screen.
DEFINE WINDOW	Creates a window.
DELETE VSCREEN	Removes a virtual screen definition.
DELETE WINDOW	Removes a window definition.
DROP WINDOW	Moves a window down in the order of displayed windows.
GET VSCREEN	Writes data from a CMS file to the specified virtual screen.
HIDE WINDOW	Prevents the specified window from being displayed, and connects the window to a virtual screen.
MAXIMIZE WINDOW	Expands a window to the physical screen size.
MINIMIZE WINDOW	Reduces the size of the window to one line.
POP WINDOW	Moves a window up in the order of displayed windows.
POSITION WINDOW	Changes the location of a window on the physical screen.
PUT SCREEN	Makes a copy of the physical screen and writes the image to a CMS file.
PUT VSCREEN	Writes the data from the data area of a virtual screen to a CMS file.
QUERY	The following QUERY command options are used for windowing: APL, BORDER, CHARMODE, CMSPF, CURSOR, DISPLAY, FULLREAD, FULLSCREEN, HIDE, KEY, LINEND, LOCATION, LOGFILE, NONDISP, REMOTE, RESERVED, ROUTE, SHOW, TEXT, VSCREEN, WINDOW, and WMPF. For specific information about the usage of each command, see the <i>VM/SP CMS Command Reference</i> .
REFRESH	Updates virtual screens and their associated windows and refreshes the screen.
RESTORE WINDOW	Returns a maximized or minimized window to its size and location before the maximize or minimize.
ROUTE	Directs data of a particular message class to a virtual screen.
SCROLL	Moves a window to a new location on the virtual screen.

Table 24 (Page 2 of 2). CMS Command Summary	
Command	Usage
SET	The following SET command options are used for windowing: APL, BORDER, CHARMODE, CMSPF, FULLREAD, FULLSCREEN, LINEND, LOCATION, LOGFILE, NONDISP, REMOTE, RESERVED, TEXT, VSCREEN, WINDOW, and WMPF. For specific information about the usage of each command, see the <i>VM/SP CMS Command Reference</i> .
SHOW WINDOW	Places a window on top of all other displayed windows and connect the window to a virtual screen.
SIZE WINDOW	Changes the number of lines and columns for a specified window.
WAITREAD VSCREEN	Uses from an exec to update the virtual screen with data, refreshes the physical screen, and waits for the next attention interrupt.
WAITT VSCREEN	Updates the virtual screen with data.
WRITE VSCREEN	Enters information in a virtual screen.

Several single-character Border commands are used for windowing and full-screen CMS. Border commands are windowing commands that you can enter in the corners of a window border. Following are the command summaries:

- B Scrolls the window backward
- C Clears the window of data
- D Drops the window
- F Scrolls the window forward
- H Hides the window
- L Scrolls the window to the left
- M Changes the location of the window
- N Minimizes the window
- O Restores the window
- P Pops the window
- R Scrolls the window to the right
- S Changes the size of the window
- X Maximizes the window

For more information on these Border commands, see the *VM/SP CMS Command Reference*.

General Information on Full-Screen CMS

There are many advantages to using full-screen CMS. When you set full-screen CMS on, you can type commands from almost anywhere on the physical screen. You can scroll forward and backward through your CMS session to see commands you previously entered and CMS responses to these commands. To reenter any command, you do not need to retype the entire command. Instead, you can scroll back to where the command was previously entered, retype any letter, and press ENTER.

During your full-screen CMS session, messages and other output appear in windows on your physical screen and are viewed without leaving your current work environment.

Virtual Screens and Windows

In full-screen CMS, a virtual screen (vscreen for short) is a presentation space where data is maintained. A window is an area on the physical screen that is used to display and manipulate virtual screen data. For more information on windows and vscreens, see Chapter 11, "Looking at VM/SP Through Windows."

Following is a listing of the default virtual screens and windows which are available to you in full-screen CMS.

Virtual Screen	Lines	Cols	Rtop	Rbot	Dcolor	Options
STATUS	1	Pscr	0	0	White	PROTECT
NETWORK	16	70	2	0	Blue	PROTECT
WARNING	4	70	2	0	Red	PROTECT
MESSAGE	20	70	2	0	White	PROTECT
CMS	120	Pscr	2	5	Green	NOPROTECT

Table 26. Default Windows					
Window	Lines	Cols	Psline	Pscol	Options
STATUS	1	Pscr	-1	1	FIXED NOBORDER NOPOP NOTOP
CMS	Pscr	Pscr	1	1	FIXED BORDER NOPOP TOP
NETWORK	8 (max.)	71	-12	7	VARIABLE BORDER NOPOP TOP
WARNING	6 (max.)	71	3	3	VARIABLE BORDER POP TOP
MESSAGE	8 (max.)	71	11	3	VARIABLE BORDER POP TOP
CMSOUT	8	75	9	3	VARIABLE BORDER POP TOP

Definitions of the terminology and abbreviations used in these tables are located at the end of Chapter 17, "Customizing Full-Screen CMS."

Screen Organization

Once you enter the command `SET FULLSCREEN ON`, your screen is organized differently from the way it appears in line-mode CMS. Following is an example of the screen layout for full-screen CMS.

```

Fullscreen CMS          Columns 1 - 79 of 81

Ready;
-

PF1=Help      2=Pop_Msg   3=Quit      4=Clear_Top  5=Filelist   6=Retrieve
PF7=Backward  8=Forward    9=Rdrlist   10=Left      11=Right     12=Cmdline
====>
15:15:08                      Enter a command or press a PF or PA key

```

Figure 90. Screen Organization

Chapter 11, “Looking at VM/SP Through Windows” contains a detailed description of each of the areas of the screen.

You will also notice that the line-mode VM status notices are replaced by the following full-screen status notices:

- Executing a command
- Enter your response in vscreen '*vname*'
- Scroll forward for more information in vscreen '*vname*'
- Enter a command or press a PF or PA key

These status notices are also fully explained in Chapter 11, “Looking at VM/SP Through Windows.”

Default Settings

In full-screen CMS, you are given a set of PF keys called CMSPF keys. These keys are set by default to perform certain windowing functions as listed in the following table.

Table 27. CMSPF Key Settings		
CMSPF Key	Pseudonym	Command
CMSPF 1	Help	ECHO HELP
CMSPF 2	Pop_Msg	NOECHO #WM POP WINDOW MESSAGE *
CMSPF 3	Quit	NOECHO SET FULLSCREEN SUSPEND
CMSPF 4	Clear_Top	NOECHO #WM CLEAR WINDOW =
CMSPF 5	Filelist	ECHO EXEC FILELIST
CMSPF 6	Retrieve	RETRIEVE
CMSPF 7	Backward	NOECHO #WM SCROLL BACKWARD CMS 1
CMSPF 8	Forward	NOECHO #WM SCROLL FORWARD CMS 1
CMSPF 9	Rdrlist	ECHO EXEC RDRLIST
CMSPF 10	Left	NOECHO #WM SCROLL LEFT CMS 10
CMSPF 11	Right	NOECHO #WM SCROLL RIGHT CMS 10
CMSPF 12	Cmdline	NOECHO CURSOR VSCREEN CMS -2 8 (RESERVED)

You can change any of these settings by entering the command SET CMSPF, followed by the PF key number (represented as nn).

You can set your CMSPF keys to perform any command, including the windowing commands listed under “WM Environment” on page 356, with the exception of HELP. (HELP is entered from the WM window but cannot be entered as a #WM command).

If you wish to set your CMSPF keys to perform any of these windowing commands, you can begin the command with “#WM” to indicate that the windowing command is immediately executed in the CMS window. While you can set your CMSPF keys to these windowing commands without using #WM, if you do so, the PF key commands can be executed after other commands which are pending at the time you press the PF key.

For more information on the default settings and how to change them, see Chapter 11, “Looking at VM/SP Through Windows.”

The PA1, PA2, and CLEAR keys also have special meanings in full-screen CMS. The PA1 key pops the WM window. The PA2 key and the CLEAR key serve the same purpose and are used to scroll the top window forward. See Chapter 11, “Looking at VM/SP Through Windows” for details.

Full-screen CMS also provides you with several commands to control how you receive messages from the system, other users, and your programs. These commands are discussed in Chapter 10, “Communicating with Other Computer Users” and are fully described in the *VM/SP CMS Command Reference*.

You may find it useful to refer to the following table which contains information regarding the default settings for message routing. The terms used in the table are discussed at the end of Chapter 17, "Customizing Full-Screen CMS."

Table 28. Default Settings for Message Routing		
Message Class	Virtual Screen	Options
CMS	CMS	NOALARM NONOTIFY
CP	CMS	NOALARM NONOTIFY
MESSAGE	MESSAGE	ALARM NOTIFY
WARNING	WARNING	ALARM NOTIFY
SCIF	MESSAGE	NOALARM NONOTIFY
NETWORK	NETWORK	NOALARM NOTIFY

WM Environment

The WM window is a special window available to you for window manipulation. That is, it is used to drop, move, or change the size of other windows. Depending on how the WM window was invoked, one of the following messages will appear in the window:

- Active window overlaid; enter a windowing command or press a PF key
- Output displayed; enter a windowing command or press a PF key
- Enter a windowing command or press a PF key.

Chapter 11, "Looking at VM/SP Through Windows" provides more information on when each message is displayed.

The WM window is automatically displayed on your screen in the following special situations:

- When your entire screen is protected and the active window is overlaid (for example, this situation could occur if you maximize a window that is protected so that it fills the entire screen and covers all other windows.) Chapter 11, "Looking at VM/SP Through Windows" gives a detailed explanation of the WM window and provides an example of this type of scenario.
- When you run an application that uses the CONSOLE macro to perform I/O and
 - The CMS virtual screen is updated, or
 - Any virtual screen (other than CMS) is updated and a pop window is showing it.

In this instance, you simply drop the WM window to return to the application.

If you wish, you can also pop the WM window at any time during your full-screen CMS session. To do this, you would simply enter the command POP WINDOW WM or press the PA1 key (which in full-screen CMS defaults to the command POP WINDOW WM).

If you pop the WM window, or when it appears on your screen, you can access a special set of PF keys called the WMPF keys. These keys are used to manipulate other windows on your screen. The default settings for these keys are listed in the figure that follows.

Table 29. WMPF Key Settings		
WMPF Key	Pseudonym	Command
WMPF 1	Help	NOECHO HELP
WMPF 2	Top	NOECHO SCROLL TOP =
WMPF 3	Quit	NOECHO DROP WINDOW WM
WMPF 4	Clear	NOECHO CLEAR WINDOW =
WMPF 5	Copy	NOECHO PUT SCREEN COPY SCREEN
WMPF 6	Retrieve	RETRIEVE
WMPF 7	Backward	NOECHO SCROLL BACKWARD = 1
WMPF 8	Forward	NOECHO SCROLL FORWARD = 1
WMPF 9	Maximize	NOECHO MAXIMIZE WINDOW =
WMPF 10	Left	NOECHO SCROLL LEFT = 10
WMPF 11	Right	NOECHO SCROLL RIGHT = 10
WMPF 12	Restore	NOECHO RESTORE WINDOW =

In addition, you can manipulate other windows on your screen by entering windowing commands in the WM window. Any of the windowing commands in the following table can be entered from the WM window:

CLEAR WINDOW	PUT SCREEN	SCROLL
CP	QUERY BORDER	SET BORDER
DROP WINDOW	QUERY HIDE	SET LOCATION
HELP	QUERY LOCATION	SET RESERVED
HIDE WINDOW	QUERY RESERVED	SET WINDOW
MAXIMIZE WINDOW	QUERY SHOW	SET WMPF
MINIMIZE WINDOW	QUERY WINDOW	SHOW WINDOW
POP WINDOW	QUERY WMPF	SIZE WINDOW
POSITION WINDOW	RESTORE WINDOW	

Migration Considerations

The remainder of this section provides specific information for users of CP, CMS, XEDIT, and applications. You should carefully review these items so that you will understand the benefits of full-screen CMS as well as any adjustments you may need to make to utilize full-screen CMS capabilities fully.

CMS Considerations

- CMS Immediate commands (including #WM and #CP in full-screen CMS), Border commands, and windowing commands entered in the WM window do not support synonyms and cannot be translated to another language.
- When full-screen CMS is on, your PF keys are called CMSPF keys and are set by default to perform windowing functions such as scrolling the CMS window, popping the MESSAGE window, and clearing the window on top. To override these settings or to make your CMSPF keys equivalent to your line-mode CP PF keys, set full-screen CMS on and then use the SET CMSPF command. Changing your CMSPF keys does not affect your CP PF keys.
- Fullscreen CMS sets CMSPF 6 to RETRIEVE. You do not have to define your own key to perform this function unless you want to override the default. Another way to retrieve commands that you previously entered is to scroll your CMS window back so that the command you wish to reenter is visible. Then, position the cursor over the command, retype any character, and press ENTER. The command is reentered. (The command echo and output are appended to the bottom of the CMS virtual screen by default).
- In full-screen CMS, virtual screen output may be logged to CMS files. Messages and warnings are logged by default. For more information, see the SET LOGFILE and SET FULLSCREEN ON commands in the *VM/SP CMS Command Reference*. The CP SPOOL CONSOLE START command does not log full-screen CMS interactions.
- To copy a full-screen CMS screen image, use the PUT SCREEN command. This command copies the image of your physical screen to a CMS file which you can then XEDIT or print.
- To tailor window and virtual screen attributes and extended attributes, use the DEFINE VSCREEN, SET VSCREEN, WRITE VSCREEN, and SET BORDER commands. The CP SCREEN command has no effect on the attributes of full-screen CMS output.
- To interrupt an exec that is reading data from your terminal, enter an Immediate command prefixed by “#” (or the current LINEND character). For example, you would enter “#HT” to halt entering or “#HI” to halt interpretation.
- In full-screen CMS, the CMS output (CMS messages, BLIP character, or application messages) is not displayed immediately. When execution of the command is completed, the screen is refreshed and the CMS output that has been queued is displayed. When full-screen CMS is suspended, the CMS output displayed prior to the full-screen suspend remains in the CMS output virtual screen and is displayed when full-screen CMS is resumed.
- If you wish to enter ahead the next command (or commands) while a command is executing, do so on the full-screen CMS command line. Entering ahead in the I/O area in the CMS window is not recommended. When the currently-executing command completes, the screen is refreshed with new output written to the CMS window. This new output can overlay the command that you were in the process of entering.
- Enter the command SET VSCREEN CMS NOTYPE to suppress output to the CMS virtual screen. The CP SPOOL CONSOLE NOTERM command does not suppress full-screen CMS output.

- To enter long commands, type in the full-screen CMS I/O area. If you wish to enter long commands from the command line, drop or hide the STATUS window. (See the DROP WINDOW and HIDE WINDOW commands in the *VM/SP CMS Command Reference* for more information). CMS command input is limited to 255 characters.
- When you press a PF or PA key, any input on the screen that has not been processed is discarded except input that is entered on the command line. If the key that was pressed does not update the command line, then input on the command line is rewritten.
- PF keys set to execute Immediate commands (the key definition is preceded by #WM or #CP) are immediately executed. All other key definitions are stacked.
- On terminals with 80 columns, command output that is 80 characters wide cannot be completely viewed without scrolling the CMS window to the right. This is because each line of CMS output is typically preceded by two Start Fields, one in the CMS vscreen and one that is placed at the start of each line when the physical screen is displayed. (See the WAITREAD VSCREEN command in the *VM/SP CMS Command Reference* for more information). Therefore, the CMS window, in its default position at column one of the CMS vscreen, displays columns 1 through 79 of the output. The remaining output is viewed by scrolling to the right.
- If the virtual screen is wider than the window displaying it, then a field that wraps onto multiple lines in the virtual screen cannot be displayed as a single field on the screen. Certain keys on the terminal (ERASE EOF, DELETE, and INSERT MODE) will only work on fields on the screen and not on the entire field in the virtual screen. In this case, the field is reconstructed using the changes on the screen and the parts of the field that are in the virtual screen (including parts that are not displayed on the screen). Note that the function of these keys will always work on fields on the screen; whenever a window vertically splits another window, these keys will not work on the entire field in the virtual screen.
- If you disconnect from full-screen CMS, when you reconnect, you may get a "MORE..." status. Press the CLEAR key to return to full-screen CMS.
- The screen is refreshed when there is a large amount of output waiting to be displayed. When a program or command issues a number of lines of output which is equal to the number of lines on the physical screen less one, the screen is refreshed.
- In full-screen CMS, there is no terminal escape character (ESCAPE), line delete character (LINEDEL), or character delete character (CHARDEL).
- Because CMS is a single-tasking system, CMS windowing only supports one active virtual screen on the physical screen. Other interactive virtual screens can be displayed on the physical screen but are protected.
- The IUCV Message All System Service can stack up to 255 messages at any one time. If this limit is exceeded, any additional incoming messages are directly sent to the terminal.
- If the window, virtual screen, and physical screen do not have the same number of columns, it is recommended that you define the window with one column greater than the number of columns in the vscreen that it is displaying. This provides for the additional field definition character (Start Field) which is necessary for the proper display of the window on the screen and ensures that a maximum number of columns of virtual screen data are displayed.

- The detection of loaded programmed symbol sets occurs when full-screen CMS is initialized (SET FULLSCREEN ON), resumed (SET FULLSCREEN RESUME), or when XEDIT is invoked. Therefore, programmed symbol sets should be loaded prior to invoking these commands. They will then be available for use by full-screen CMS or XEDIT in displaying the screen. In line mode CMS, programmed symbol sets are detected the first time a window is displayed or when XEDIT is invoked. If XEDIT has not been invoked and Session Services commands are used to display a window, the check to determine if programmed symbol sets are loaded is only done the first time a window is displayed. If programmed symbol sets are loaded after the initial display of a window or after XEDIT has been invoked, you must invoke XEDIT to cause detection of the new programmed symbol sets.

CP Considerations

- To enter CP commands while in full-screen CMS, preface the commands with CP or #CP. The output from the CP commands will be displayed by full-screen CMS. If you wish to drop into CP during your full-screen CMS session, enter CP. This results in a cleared screen with a CP READ status. While you can then enter CP commands and receive output to your terminal, any CP commands you enter while in the CP READ status will not be displayed in full-screen CMS.
- #CP (the default LINEND character, followed by CP) is processed by full-screen CMS before it is processed by CP. In full-screen CMS, #CP is treated as a CMS Immediate command.
- If the CP SLEEP command is entered while full-screen CMS is on, it may appear that your terminal is hung. Any write to the terminal unlocks the keyboard. See the CP SLEEP command in the *VM/SP CP General User Command Reference* for more information.
- The CP SCREEN command has no effect on the attributes of full-screen CMS output. To tailor window and virtual screen attributes and extended attributes, use the DEFINE VSCREEN, SET VSCREEN, WRITE VSCREEN, and SET BORDER commands.
- The CP SPOOL CONSOLE NOTERM command does not suppress full-screen CMS output. Use the command SET VSCREEN CMS NOTYPE to suppress output to the CMS virtual screen.
- All messages classified as message class MESSAGE are displayed with headers. (See the ROUTE command in the *VM/SP CMS Command Reference* for further information). This includes messages sent via the CP MESSAGE and the CP MSGNOH commands.
- CP does not edit lines entered in a full-screen environment. In full-screen CMS, there is no terminal escape character (ESCAPE), line delete character (LINEDEL), or character delete character (CHARDEL).

System Product Editor Considerations

- XEDIT no longer carries out its own I/O. Windowing functions are responsible for XEDIT I/O. As a result, certain CMS settings affect the XEDIT environment, especially the following:
 - SET LANGUAGE (affects Double-Byte Character Set (DBCS) display and the nondisplayable character set)
 - SET APL/TEXT

- SET FULLREAD
- SET NONDISP
- SET REMOTE
- The XEDIT SET BRKKEY works differently. XEDIT no longer restores the break key to whatever it was when SET BRKKEY ON was entered. Instead, if BRKKEY was set in XEDIT, the CP setting remains when you are no longer in XEDIT.

In addition, the initial SET BRKKEY setting now reflects the CP TERMINAL BRKKEY setting. It is no longer always ON by default.

- The default PA1 key for XEDIT (and the NAMES and SENDFILE commands) is now COMMAND CMS POP WINDOW WM if BRKKEY is not assigned to the PA1 key.
- QUERY and EXTRACT return virtual screen information rather than physical screen information.
- COPYKEY copies the content of the virtual screen, rather than the content of the physical screen, into the printer spool.
- The initial SET ETMODE setting is no longer OFF by default. This setting is now based on whether the terminal in use can handle double-byte characters.
- If you are using a 3277 terminal and you enter QUERY PF, you now get the settings for 24 PF keys instead of just 12.
- Nullkey is an existing option you can specify on any XEDIT PF or PA key or on the ENTER key. Now, the nullkey function replaces trailing blanks with nulls on the field of the screen that contains the cursor. If the cursor is on a prefix area, the nulls are written to the field of the file line associated with that prefix area. Before, the nullkey function wrote the nulls on the line containing the cursor.
- Although you can define your own XEDIT virtual screen and assign to it any valid default attributes and extended attributes, XEDIT overrides these when it writes fields. You should continue to use the XEDIT SET COLOR subcommand to change the XEDIT screen attributes.

Considerations for Writing Applications

- If the CP SLEEP command is entered while full-screen CMS is on, it will appear that your terminal is hung. Any write to the terminal will unlock the keyboard. See the CP SLEEP command in the *VM/SP CP General User Command Reference* for more information.
- Some programs imbed the hexadecimal code X'1D' to affect the highlighting and color attributes of output. In full-screen CMS, X'1D' is a nondisplayable character and does not affect the attributes of data following it. The DEFINE VSCREEN, SET VSCREEN, and WRITE VSCREEN commands (as well as the LINEWRT macro) let programs specify attributes for data in full-screen CMS.
- Error messages generated from windowing commands are displayed based on how the command was executed:
 - When called from a program or from an exec procedure with &CONTROL NOMSG, then no message is displayed and only the return code is set.

- When called from an exec procedure with ADDRESS COMMAND, then the variable *message.1* is set to the message text and *message.0* is set to 1 to indicate the number of message variables set.
- In other cases, such as a command entered from the command line or from an EXEC procedure with ADDRESS CMS, the message is displayed at the terminal. These messages are edited based on the CP EMSG setting.
- When returning to full-screen CMS from an application that performs its own full-screen management, your screen can contain mixed data. Press the CLEAR key to scroll forward and refresh the screen. Alternatively, you can enter the command SET FULLSCREEN SUSPEND before executing the application.
- The following messages are not trapped by the IUCV Message All System Service and are directly sent to the terminal:
 - Asynchronous CPCONIO (including PER/TRACE events)
 - EMSGs not generated as part of a DIAG X'08' instruction
 - Accounting messages.
- Certain applications must be changed to correctly work in the full-screen CMS environment:
 - Programs which issue a 3215 SIO to do a line-mode read
 - Programs which issue DIAG X'58' to do full-screen I/O.

Line-mode output and prompts written by these applications will not be immediately displayed in the full-screen CMS environment. Also, messages and warnings from other computer users are not displayed until you exit these applications. These applications should be changed to use CMS macros:

- Change 3215 SIO to use LINERD macro
- Change DIAG X'58' to use CONSOLE macro

Alternatively, to avoid problems viewing output during the execution of such programs, you can temporarily suspend full-screen CMS (SET FULLSCREEN SUSPEND) before running the application and then resume your session (SET FULLSCREEN RESUME) upon exiting the application.

- When full-screen CMS is on, most CMS console output is not passed to CP. In addition, applications that use the IUCV Message System Service (*MSG) and SET VMCONIO IUCV will not trap all CMS output. Before running such applications, it is recommended to suspend full-screen CMS.
- If an application runs disconnected using the Single Console Image Facility (SCIF) to communicate with the disconnected user, the primary user must have FULLSCREEN set OFF or SUSPEND. If FULLSCREEN is ON, message DMKSND068E will be received instead of the normal response following the second or third attempt to SEND a command to the primary machine.
- Full-screen CMS initialization issues the CP TERMINAL BRKKEY NONE command. Application developers may want to reset the CP TERMINAL BRKKEY to PA1 when debugging.
- You can specify the EXIT parameter for the OPEN function of the CONSOLE macro instruction to handle unrequested device interrupts.
- If EXIT is specified, **do not** define an interruption routine using the HNDINT macro for the same device. Use of the CONSOLE and HNDINT macros is mutually exclusive. CONSOLE OPEN with EXIT supersedes an HNDINT

routine when the interrupt is requested. Therefore, if you want to do I/O to a 3270 device, use the `CONSOLE` macro instead of the `HNDINT` macro.

- When using full-screen CMS, an application which uses the `STAX` macro or the `HNDINT` macro may work differently than when not using full-screen CMS. In line mode CMS, only the attention interrupts cause by the `ENTER` key are passed to the `STAX` or `HNDINT` exit. However, in full-screen CMS, attention interrupts caused by any key may be passed to the exit. If your exit is not prepared to handle interrupts other than those caused by the `ENTER` key, then full-screen CMS should be suspended.

Summary of Changes

Summary of Changes for SC19-6210-5 for VM/SP Release 6

Major changes to this publication for Release 6 include the addition of one new chapter and the relocation of information throughout the book. Chapter 4, "Using the Shared File System," was added for this release. The minidisk information previously located in Chapter 1 has been made into a separate chapter (now Chapter 5) and the chapters of the book have been resequenced. Appendix A, "Considerations for Line Mode Terminals," can now be found in the *VM/SP CP General User Command Reference*. The remaining appendices have been resequenced.

Shared File System (SFS)

VM/SP now provides a new method for storing your files. If you wish, you can continue to store files on minidisks. Or, you can choose to store your files in a file pool, a large amount of DASD space containing the files for many users. The part of CMS that manages file pools is called the Shared File System (SFS).

Within a file pool, each user is assigned a file space for storing files. In this file space, you can create directories to hierarchically organize your files.

SFS also allows you to easily share your files or directories with other users. To allow another user to read from or write to one of your files or directories, you simply issue a command to grant them authority to do so. In turn, other users can grant you read or write authority on their files and directories. If you no longer wish to share your files, you can issue a command to revoke the authority you have granted.

Callable Services Library (CSL)

VM/SP now includes a callable services library, named VMLIB, that contains a set of assembler routines. Application programs can call these CSL routines to perform services like calling SFS routines, issuing VM commands through a REXX exec, and accessing REXX variables.

Programmers can also write their own routines and build their own callable services libraries.

Integration of Between-Release Support Information to VM/SP Release 6

VM Terminal Usability Enhancements, GC24-5309

VM/SP 9370 Processors, 9332 and 9335 Direct Access Storage Devices, and 9347 Tape Drive, GC24-5315

New Commands for Release 6 of VM/SP

The following CMS commands are new or changed for this release:

ACCESS	Allows you to access a minidisk or an SFS directory with a file mode letter.
COPYFILE	Copies CMS files from one minidisk to another, one SFS directory to another, or between minidisks and directories.
CREATE ALIAS	Places an additional name for a file in a specified directory.
CREATE DIRECTORY	Creates an SFS directory.

CREATE LOCK	Creates an explicit lock on an SFS directory or a file in an SFS directory.
CREATE NAMEDEF	Assigns a temporary name for a user which can be used by a program, instead of a file name and file type or a fully-qualified directory name.
CSLLIST	Lists information about all members of a specified callable services library.
DELETE LOCK	Releases the explicit lock placed on a file or directory by the CREATE LOCK command.
DELETE NAMEDEF	Deletes the temporary name given to a file or directory by the CREATE NAMEDEF command, and makes it no longer usable by a program.
DIRLIST	Lists directories of a specified directory structure in a full-screen environment.
DISK	Performs disk-to-card and card-to-disk operations for CMS files. Can be used for files residing on minidisk or in directories.
EDIT	Uses the VM/SP System Product Editor in CMS editor (EDIT) compatibility mode to create or modify a file residing on a minidisk or in an SFS directory.
ERASE	Deletes CMS files from a minidisk and deletes both files and subdirectories from an SFS directory.
EXECMAP	Lists storage-resident execs and display execs in saved segments.
FILEDEF	Defines an OS <i>ddname</i> and relates that <i>ddname</i> to any device supported by CMS or to a file residing in an SFS directory or on a minidisk.
FILELIST	Lists information about CMS files in an SFS directory or a minidisk, with the ability to edit and enter commands from the list.
FINIS	Closes an open file on a minidisk or in an SFS directory.
GRANT AUTHORITY	Authorizes other users to read and/or modify your SFS directories or the files in the directories.
HELP	Displays information about CP, CMS, or user commands, EDIT or XEDIT, EXEC, EXEC 2 and System Product Interpreter control statements, and descriptions of CMS and CP messages.
HELPCONV	Converts a CMS file into an acceptable form to be used by the HELP facility.
LISTDIR	Lists directories in a specified directory structure.
LISTFILE	Lists information about CMS files stored on a minidisk or in an SFS directory.
NUCXMAP	Identifies existing nucleus extensions, including those residing in saved segments.
PROGMAP	Displays or places on the program stack information on programs currently loaded in storage or in a saved segment.
PSERV	Copies a procedure from the VSE procedure library onto a CMS minidisk or an SFS directory, displays the procedure at the terminal, or spools the procedure to the virtual punch or printer.

QUERY	Requests information about CMS files, minidisks, or SFS directories, or other virtual machine characteristics.
RECEIVE	Reads to your SFS directory or minidisk a file or note that is in your virtual reader.
RELEASE	Releases an SFS directory or minidisk previously accessed.
RELOCATE	Moves a file or directory from one directory to another, that you own within the same file pool.
RENAME	Changes the name of a CMS file or directory.
REVOKE AUTHORITY	Cancels authorities that you granted for a directory or files in a directory.
RSERV	Copies a VSE relocatable module onto a CMS minidisk or SFS directory, displays it at the terminal, or spools a copy to the virtual punch or printer.
RTNDROP	Cancels the binding of a callable services library routine.
RTNLOAD	Searches for, loads, and binds a callable services library routine to a fixed location in storage, and makes it available for invocation.
RTNMAP	Displays information about the callable services library routines that are currently loaded and bound to an address.
RTNSTATE	Obtains the status of one or more specific callable services library routines.
SEGMENT ASSIGN	Indicates the logical segment to be associated with the physical segment.
SEGMENT LOAD	Loads a saved segment.
SEGMENT PURGE	Purges a saved segment.
SEGMENT RELEASE	Releases the storage held by a segment space.
SEGMENT RESERVE	Creates a segment space for subsequent loading.
SSERV	Copies a VSE source statement book onto a CMS minidisk or SFS directory, displays it at the terminal, or spools a copy to the virtual punch or printer.
STATE	Verifies the existence of a CMS file on a minidisk or in an SFS directory.
STATEW	Verifies a file on a read/write file mode.
TAPE	Transfers files between tapes and minidisks (or SFS directories), positions tapes, and displays or writes VOL1 labels.

How to Obtain Prior Editions of This Publication

To obtain editions of this publication that pertain to earlier releases of VM/SP, you must use the pseudo-number found in the *VM/SP Library Guide and Master Index*.

Miscellaneous

This major revision incorporates minor technical and editorial changes.

Summary of Changes for SC19-6210-4 for VM/SP Release 5

This publication has been restructured for Release 5 to meet the needs of general users. Much of the specific programming information has been relocated to the system

programmer's reference books to better organize the VM/SP library as a whole. The following chart shows which chapters have been relocated and the destination books:

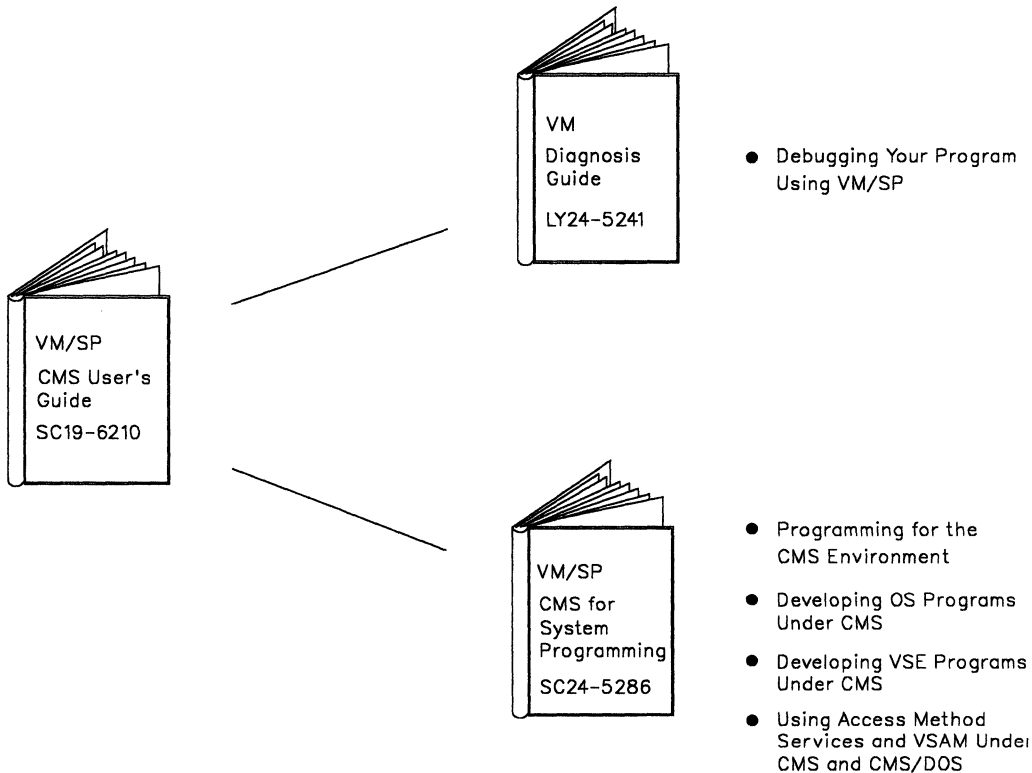


Figure 91. Restructuring of the CMS User's Guide

The remaining information within this publication has been revised as follows:

- "Part 1: Getting Acquainted with VM/SP" has been retitled and now includes "Chapter 5, Using the HELP Facility" (previously Chapter 19 in Part 4).
- "Part 2: Working with VM/SP" has been retitled and now includes "Chapter 6, Editing Your Files" (previously Chapter 5 in Part 1), "Chapter 7, Using Real Printers, Punches, Readers, and Tapes" (previously Chapter 6 in Part 1), "Chapter 8, Communicating with Other Computer Users" (previously Chapter 7 in Part 1), "Chapter 9, Looking at VM/SP Through Windows" (new for this release), and "Chapter 10, Using the CMS Batch Facility" (previously Chapter 12 in Part 2).
- In "Part 3: Learning to Use Execs," the chapter entitled "Exchanging Data Between Programs Through the Stack" (previously Chapter 17) has been deleted.
- "Part 4: Tailoring Your System" has been retitled and now includes "Chapter 15, Customizing Full-Screen CMS" and "Chapter 16, Tailoring the HELP Facility."

How to Obtain Prior Editions of This Publication

To obtain editions of this publication that pertain to earlier releases of VM/SP, you must use the pseudo-number found in the *VM/SP Library Guide and Master Index*.

New Commands for Release 5 of VM/SP

The following CMS commands are new for this release:

ALARM VSCREEN Sounds the terminal alarm when the display is refreshed

CLEAR VSCREEN	Erases data in the virtual screen
CLEAR WINDOW	Scrolls past all data in the virtual screen
CONVERT COMMANDS	Converts a CMS file containing Definition Language for Command Syntax (DLCS) statements into an internal form for the parsing facility
CMSSERV	Starts IBM Cooperative Processing Communications between your VM/SP host system and your workstation (IBM Personal Computer)
CURSOR VSCREEN	Positions the cursor on a specified line and column in a virtual screen
DEFINE VSCREEN	Creates a virtual screen
DEFINE WINDOW	Creates a window
DELETE VSCREEN	Removes a virtual screen definition
DELETE WINDOW	Removes a window definition
DROP WINDOW	Moves a window down in the order of displayed windows
GENMSG	Converts a message repository into an internal form
GET VSCREEN	Writes data from a CMS file to the specified virtual screen
HIDE WINDOW	Prevents the specified window from being displayed, and connects the window to a virtual screen
LANGGEN	Saves all text files for a language in DCSS, and/or saves CP message repository
LANGMERG	Combines all language-related files for an application into one text file
MAXIMIZE WINDOW	Expands a window to the physical screen size
MINIMIZE WINDOW	Reduces the size of the window to one line
MOREHELP	Obtains either additional or related information about the latest valid HELP command you issued
PARSECMD	Calls the parsing facility from within an exec
POP WINDOW	Moves a window up in the order of display windows
POSITION WINDOW	Changes the location of a window on the physical screen
PUT SCREEN	Makes a copy of the physical screen and writes the image to a CMS file
PUT VSCREEN	Writes the data from the data area of a virtual screen to a CMS file
QUERY	The following QUERY command options have been added or changed for Release 5: APL, BORDER, CHARMODE, CMSPF, CURSOR, DISPLAY, FULLREAD, FULLSCREEN, HIDE, INSTSEG, KEY, LINEND, LOCATION, LOGFILE, NONDISP, REMOTE, RESERVED, ROUTE, SHOW, TEXT, TRANSLATE, VSCREEN, WINDOW, and WMPF
REFRESH	Updates virtual screens and their associated windows and refreshes the screen
RESTORE WINDOW	Returns a maximized or minimized window to its original size and location
ROUTE	Directs data of a particular message class to a virtual screen
SCROLL	Moves a window to a new location on the virtual screen

SET	The following SET command options have been added or changed for Release 5: ABBREV, APL, BORDER, CHARMODE, CMSPF, FULLREAD, FULLSCREEN, INSTSEG, LANGUAGE, LINEND, LOCATION, LOGFILE, NONDISP, REMOTE, RESERVED, TEXT, TRANSLATE, VSCREEN, WINDOW, and WMPF
SHOW WINDOW	Places a window on top of all other displayed windows and connects the window to a virtual screen
SIZE WINDOW	Changes the number of lines and columns for a specified window
WAITREAD VSCREEN	Used from an exec to update the virtual screen with data, refresh the physical screen, and wait for the next attention interrupt
WAITT VSCREEN	Updates the virtual screen with data
WRITE VSCREEN	Enters information in a virtual screen
XMITMSG	Retrieves a message from a CMS message repository file or your own message repository file

In addition, several single-character Border commands have been added to manipulate windows in the full-screen CMS environment:

B	Scrolls the window backward
C	Clears the window of data
D	Drops the window
F	Scrolls the window forward
H	Hides the window
L	Scrolls the window to the left
M	Changes the location of the window
N	Minimizes the window
O	Restores the window
P	Pops the window
R	Scrolls the window to the right
S	Changes the size of the window
X	Maximizes the window

Integration of Functional Enhancements to Release 4

VM/SP now supports the IBM 3380 Direct Access Storage Device, Models AE4/BE4.

National Language Support

National Language Support (NLS) has been added to VM/SP to provide support for languages other than American English:

- SET and QUERY commands determine the languages supported and set your virtual machine to operate within a particular language environment
- The Central Message Facility, a consolidated file of all system messages and responses, is used to simplify the translation process
- The parser facility parses and translates command name arguments, eliminating the need for syntax checking in programs and letting users enter programs in their own language by modifying the Definition Language for Command Syntax (DLCS).

Enhanced 3270 Support

The following changes have been made to VM/SP to provide enhanced support for 3270-type terminals:

- A protected application environment protects interactive application users from accidentally entering the CP environment
- The VM logo message at the top of the screen has been changed from "VM/370" to "VIRTUAL MACHINE/SYSTEM PRODUCT"
- Users can now directly log on from the logo screen
- A new prompting screen and error message (DMKCFM288E LOGON FROM THE INITIAL SCREEN WAS UNSUCCESSFUL) have been added.

Execs and Macros in a Discontiguous Shared Segment

An optional facility has been added to VM/SP to allow installations to build an Installation Discontiguous Shared Segment (DCSS) to contain frequently used execs and Editor Macros. Users can access the DCSS and share the same executing copy of the execs.

CMS Session Services

VM/SP now provides the end-user with window and virtual screen functions. When the command SET FULLSCREEN ON is entered, CMS is in a window. This lets the user enter commands from almost anywhere on the physical screen, scroll through data, and log data into files. The user can also view messages and other information through windows on the physical screen.

System Profile

SYSPROF EXEC has been added to VM/SP to contain some of the CMS initialization functions, such as setting up the default CMS environment, which were previously done in a module. This lets the system administrator tailor the CMS environment for the users simply by changing the exec.

Advanced Printer Subsystem

Advanced Printer Subsystem (APSS) has been added to VM/SP to provide a destination option for spool files. This lets you select a specific printer or punch to process print, punch, or console files.

HELP Facility

The VM/SP HELP Facility has been enhanced to improve performance and usability and to include National Language Support:

- HELP screens and syntax notation have been simplified
- A BRIEF layer of HELP has been added to provide information on frequently-used commands
- New HELP command options (BRIEF, DETAIL, RELATED, EXTEND, TYPE, and NOTYPE) have been added
- The MOREHELP command has been added to provide additional detail or related HELP information
- New keywords have been added for specifying control sections in HELP files
- The ability to toggle between the BRIEF, ALL, and RELATED layers of HELP has been added.

Miscellaneous

- This major revision incorporates minor technical and editorial changes.

**Summary of Changes
for SC19-6210-3
for VM/SP Release 4**

How to Obtain Prior Editions of This Publication

To obtain editions of this publication that pertain to earlier releases of VM/SP, you must use the pseudo-number found in the *VM/SP Library Guide and Master Index*.

New Commands for Release 4 of VM/SP

The following CMS commands are new for this release:

EXECDROP- Purges storage resident execs.

EXECLOAD - Loads execs into storage.

EXECMAP - Provides a list of storage resident execs.

EXECSTAT - Provides the status of a specified exec.

HELPCONV - Converts a specified file into a formatted HELP file, leaving the .CS, .CM, and .MT control words in the file.

MACLIST - Displays a list of all members in a specified MACLIB, with the ability to edit and enter commands from the list.

Integration of Functional Enhancements to Release 3

Document support of the following:

- IBM 3800 Printing Subsystems, Models 1 and 3
- IBM 3370 Direct Access Storage Device, Models A2 and B2
- 3290 Information Panel
- IBM 4248 Printer.

CMS Macro Library Enhancements

Document the changes to CMS when you are using CMS macro libraries:

- The new MACLIST command displays a list of all members in a specified macro library, with the ability to edit and issue commands from the list. You can use the DEFAULTS command to customize the options for the MACLIST command.
- The new XEDIT MEMBER option lets you specify the name of a member to be created and/or edited in a specified macro library.
- Individual member names can now be specified with the MAP operand of the MACLIB command. MACLIB MAP output can be stacked using the new STACK, LIFO, and FIFO options.

XEDIT Mixed Case Messages

Document changes to XEDIT screens for messages that appear in mixed case.

Logon Procedure

Document the logon instructions that are displayed after you clear the logo.

DDR Command Enhancement

Document support of the DDR command COMPACT option used for compacting tape output.

HELP Facility

Document the changes to the HELP Facility:

- The addition of HELP Task Menus
- The addition of the IPCS component
- The new HELPCONV command used to convert a file into a formatted HELP file
- The new operands and options of the HELP command
- The addition of the .MT (MENU TYPE) HELPCONV format word.

EXECIO command VAR and STEM options

The STEM and VAR options allow you to use the EXECIO command directly with REXX or EXEC 2 variables.

Execs in Storage

Execs that you want to remain storage resident can be loaded into storage with the new EXECLOAD command. The new EXECDROP command will drop a storage resident exec from storage. Information about storage resident execs can be displayed or stacked with the new EXECMAP command and the status of a specific exec can be determined by using the EXECSTAT command.

CMS Command Search Order

Document the search for storage resident execs in the CMS Command Search Order description.

Migration of CMS Commands into the Nucleus

The following commands are now nucleus resident commands:

- ACCESS
- DLBL
- FILEDEF
- RELEASE
- SET

Tape Support

Document the changes in tape support:

- Support of the IBM 3480 Magnetic Tape Subsystem
- New TAPE command options, TRANSFER BUFFERED and TRANSFER IMMEDIATE, for writing tapes to the 3480 Magnetic Tape Subsystem
- Support of 38K density for 18-track tapes
- Support of multi-volume files for tapes with standard labels used in OS simulation.

New VM/SP Component

Document support of the Interactive Problem Control System (IPCS) component of VM/SP.

Miscellaneous

- This major revision incorporates minor technical and editorial changes.

Glossary of Terms and Abbreviations

A

abend dump. The contents of main storage, or part of main storage, written to an external medium for debugging an error condition that resulted in the termination of a task before its regular completion.

alias. A pointer to a base file. An alias can be in the same directory as the base file or in a different directory. There must always be a base file for the alias to point to. The alias references the same data as the base file. Data is not moved or duplicated.

attention interrupt. An I/O interrupt caused by a terminal user pressing the attention key (or equivalent). See *attention key (ATTN key)* and *signaling attention*.

attention key (ATTN key). A function key on terminals that, when pressed, causes an I/O interruption in the processing unit. See *signaling attention*.

authority. In SFS, the permission to access a file or directory. You can have read authority or write authority (which includes read authority). You can also have file pool administration authority, which is the highest level of authority in a file pool.

auxiliary directory. In CMS, an extension of the CMS file directory for a minidisk, which contains the names and locations of certain CMS modules not included in the minidisk's CMS file directory.

B

base file. The first occurrence of a file. It remains the base for the life of the file, even if the file has been renamed. Aliases point to base files.

block. A unit of DASD space on FB-512 devices. For example, FB-512 devices can be the IBM 9335, 9332, 9313, 3370, and 3310 DASD using fixed-block architecture.

border. A boundary around a window. The user can enter one-letter BORDER commands from the corners of the border. For example, the letter *P* entered from a border corner pops the window. The border corners are indicated by a + (plus) sign.

buffer. An area of storage, temporarily reserved for performing input or output, into which data is read, or from which data is written.

byte. A unit of storage, consisting of eight adjacent binary digits that are operated on as a unit and constitute the smallest addressable unit in the system.

C

callable services library (CSL). A package of CMS assembler routines that can be stored as an entity and made available to application programs.

CMS. Conversational Monitor System.

CMS batch facility. A facility that lets the user run time-consuming or noninteractive CMS jobs in another CMS virtual machine dedicated to that purpose, thus freeing the user's own terminal and virtual machine for other work.

CMS/DOS. The functions of CMS that become available when the user enters the command: SET DOS ON. CMS/DOS is a part of the regular CMS system and is not a separate system. Users who do not use CMS/DOS are sometimes called OS users, because they use the OS simulation functions of CMS. Synonymous with *DOS simulation under CMS*. Contrast with *OS simulation under CMS*.

CMS editor. A CMS facility that lets the user create, change, insert, delete, or rearrange lines of data in a CMS file. See *edit mode* and *input mode*.

CMS EXEC processor. The component of the VM/SP operating system that interprets and executes procedures and EDIT macros written in the CMS EXEC language.

CMS file directory. A directory on each CMS disk that contains the name, format, size, and location of each of the CMS files on that disk. When a disk is accessed by the ACCESS command, its directory is read into virtual storage and identified with any letter from A through Z. Synonymous with *master file directory block* and *minidisk directory*.

CMS files. Refers exclusively to files in the fixed-block format used by CMS file system commands. VSAM and OS data sets and DOS files are not compatible with the CMS file format and cannot be manipulated using CMS file system commands.

CMS file system. A way to create files in the CMS system. CMS files are created by using an identifier consisting of three fields: file name, file type, and file mode. These files are unique to the CMS system and cannot be read or written using other operating systems.

CMS nucleus. The portion of CMS that is resident in the user's virtual storage whenever CMS is executing. Each CMS user receives a copy of the CMS nucleus when the user IPLs CMS. See *saved system* and *shared segment*.

command. A request from a user at a terminal for the execution of a particular CP, CMS, IPCS, GCS, TSAF, or AVS function. A CMS command can also be the name of a CMS file with a file type of EXEC or MODULE. See *subcommand* and *user-written CMS command*.

command abbreviation. A short form of the command name, operand, or option that is not a truncation of the word. For example, MSG instead of MESSAGE, RDR instead of READER. Contrast with *truncation*.

command line. The line at the bottom of display panels that lets a user enter commands or panel selections. It is prefixed by an arrow (= = = >).

component. A collection of objects that together form a separate functional unit. A product may contain many components (for example, VM/SP has components of CP, CMS, GCS, TSAF, IPCS, AVS). A component can be part of many products (CP spans both VM/SP and VM/HPO products).

console spooling. Synonym for *virtual console spooling*.

control program. A computer program that schedules and supervises the program execution in a computer system. See *Control Program (CP)*.

Control Program (CP). A component of VM/SP that manages the resources of a single computer so multiple computing systems appear to exist. Each virtual machine is the functional equivalent of an IBM System/370.

control statement. A statement that controls or affects program execution in a data processing system.

control unit. A device that controls I/O operations at one or more devices.

Conversational Monitor System (CMS). A virtual machine operating system and component of VM/SP that provides general interactive time sharing, problem solving, program development capabilities, and operates only under the control of the VM Control Program (CP).

CP. Control Program.

CP command. A command available to all VM users. Class G CP commands let the general user reconfigure their virtual machine, control devices attached to their virtual machine, do input and output spooling functions, and simulate many other functions of a real computer console. Other CP commands let system operators, system programmers, system analysts, and service representatives manage the resources of the system.

CP directory. Synonym for *VM/SP directory*.

CP read. The condition when CP is waiting for a response or request for work from the user. On a typewriter terminal, the keyboard is unlocked; on a display terminal, the screen status area indicates CP READ.

CP READ screen status. For a display terminal used as a virtual console under VM/SP, an indicator located in the lower right of the screen, that indicates CP is waiting for a response or request for work from the user.

CSL. Callable services library.

current line pointer (CLP). A pointer that indicates the line of a CMS file on which the CMS Editor or the System Product Editor (XEDIT) is currently working.

D

DASD. Direct access storage device.

direct access storage device (DASD). A storage device in which the access time is effectively independent of the location of the data.

directory. See *auxiliary directory*, *CMS file directory*, *SFS directory*, or *VM/SP directory*.

directory identifier (dirid). A fully-qualified directory name (in which the file pool ID and user ID can be allowed to default), a file mode letter, or plus (+) or minus (-) file mode syntax (used in commands).

directory name (dirname). A fully-qualified directory name that can incorporate a period (.) to indicate the user's own top directory (used in commands).

dirid. Directory identifier.

dirname. Directory name.

discontiguous saved segment. One or more 64K segments of storage that were previously loaded, saved, and assigned a unique name. The segment(s) can be shared among virtual machines if the segment(s) contains reentrant code. Discontiguous segments used with CMS must be loaded into storage at locations above the address space of a user's CMS virtual machine. They can be detached when no longer needed.

disk operating system (DOS). An operating system for computer systems that use disks and diskettes for auxiliary storage of programs and data.

Disk Operating System/Virtual Storage Extended (DOS/VSE). An operating system that is an extension of DOS/VS. A VSE system consists of: (a) licensed VSE/Advanced Functions support, and (b) any IBM-supplied and user-written programs required to

meet the data processing needs of a user. VSE and the hardware it controls form a complete computing system.

display mode. A type of editing at a display terminal in which an entire screen of data is displayed at once and in which the user can access data through commands or by using a cursor. Contrast with *line mode*.

display terminal. A terminal with a component that can display information on a viewing surface such as a CRT or gas panel.

DOS. Disk operating system.

DOS/VSE. Disk Operating System/Virtual Storage Extended.

dump. To write the contents of part or all of main storage, or part or all of a minidisk, to auxiliary storage or a printer. See *abend dump*.

E

edit mode. The environment in which CMS EDIT subcommands and System Product Editor (XEDIT) subcommands can be entered by the user to insert, change, delete, or rearrange the contents of a CMS file. Contrast with *input mode*.

EOF. End of file.

EXEC 2 language. A general-purpose, high-level programming language, particularly suitable for EXEC procedures and XEDIT macros. The EXEC 2 processor runs procedures and XEDIT macros (programs) written in this language. Contrast with *CMS EXEC language* and *Restructured Extended Executor (REXX) language*.

EXEC procedure. (1) A procedure defined by a frequently used sequence of CMS and CP commands to do a commonly required function. A user creates the procedure to save repetitious rekeying of the sequence, and invokes the entire procedure by entering a command (that is, the exec file's file name). The procedure could consist of a long sequence of CMS and CP commands, along with REXX, EXEC 2, or CMS EXEC control statements to control processing within the procedure. (2) A CMS file with a file type of EXEC.

EXEC 2 processor. A program in VM/SP that interprets and executes procedures, EDIT macros, and XEDIT macros written in the EXEC 2 language.

explicit lock. A lock on a file or directory that a user explicitly created by entering a CREATE LOCK command or executing a DMSCRLOC CSL routine.

F

FIFO (first-in-first-out). A queuing technique in which the next item to be retrieved is the item that has been on the queue for the longest time. Contrast with *LIFO (last-in-first-out)*.

file ID. A CMS file identifier that consists of a file name, file type, and file mode. The file ID is associated with a particular file when the file is created, defined, or renamed under CMS. See *file name*, *file type*, and *file mode*.

file mode. A two-character CMS file identifier field comprised of the file mode letter (A through Z) followed by the file mode number (0 through 6). The file mode letter indicates the minidisk or SFS directory on which the file resides. The file mode number indicates the access mode of the file. See *file access mode*.

file name. A one-to-eight character alphanumeric field, comprised of A through Z, 0 through 9, and special characters \$ # @ + - (hyphen) : (colon) _ (underscore), that is part of the CMS file identifier and serves to identify the file for the user.

file pool. A collection of minidisks managed by SFS. It contains user files and directories and associated control information. Many users' files and directories can be contained in a single file pool.

file pool ID. The name of a file pool. It is part of a fully-qualified directory name, identifying where the directory and all files in it are located. It has up to eight characters, followed by a colon (:).

file space. A user's allocation of space within a file pool.

file type. A one-to-eight character alphanumeric field, comprised of A through Z, 0 through 9, and special characters \$ # @ + - (hyphen) : (colon) _ (underscore), that is used as a descriptor or as a qualifier of the file name field in the CMS file identifier. See *reserved file types*.

full-screen CMS. When a user enters the command SET FULLSCREEN ON, CMS is in a window and can take advantage of 3270-type architecture and windowing support, and various classes of output are routed to a set of default windows. Also, users can type commands anywhere on the physical screen and scroll through commands and responses previously displayed. See *windowing*.

H

HOLDING screen status. For a display terminal used as a virtual console under VM/SP, an indicator located in the lower right of the screen that displays that the current contents of the screen remain on the screen until the user requests that the screen be erased. This status occurs either by pressing the ENTER key, or it is triggered by a message or warning displayed on the screen.

I

immediate command. A type of CMS command that, when entered after an attention interruption, causes program execution, tracing, or terminal display to stop. Another immediate command can be entered to resume tracing or terminal display. The immediate commands are HB (halt batch execution), HI (halt all System Product Interpreter or EXEC 2 programs or macros), HO (halt tracing), HT (halt typing), HX (halt execution), RO (resume tracing), RT (resume typing), SO (suspend tracing), TE (trace end), and TS (trace start). They are called immediate commands because they are executed as soon as they are entered; they are not stacked in the console stack. Within an exec, immediate commands can be established or cancelled by the CMS command IMMCMD.

implicit lock. A lock automatically acquired and freed when you run CMS commands and program functions against files or directories that reside in an SFS file pool. Multiple readers and one writer can access the file or directory.

initial program load (IPL). The initialization procedure that causes an operating system to begin operation. A VM user must IPL the specific operating system into the virtual machine that will control the user's work. Each virtual machine can be loaded with a different operating system.

input line. For typewriter terminals, information keyed in by a user between the time the typing element of the terminal comes to rest following a carriage return until another carriage return is typed. For display terminals, the data keyed into the user input area of the screen. See *user input area*.

input mode. In the CMS Editor or System Product Editor (XEDIT), the environment that lets the user key in new lines of data. Contrast with *edit mode*.

input/output (I/O). (1) Pertaining to a device whose parts can do an input process and an output process at the same time. (2) Pertaining to a functional unit or channel involved in an input process, output process, or both, concurrently or not, and to the data involved in such a process.

interface. A shared boundary between two or more entities. An interface might be a hardware or software component that links two devices or programs together.

interrupt. A suspension of a process, such as execution of a computer program, caused by an external event and done in such a way that the process can be resumed.

invoke. To start a command, procedure, or program.

I/O. Input/output.

L

LIFO (last-in-first-out). A queuing technique in which the next item to be retrieved is the item most recently placed in the queue. Contrast with *FIFO (first-in-first-out)*.

line end symbol. Synonym for *logical line end symbol*.

line mode. The mode of operation of a display terminal that is equivalent to using a typewriter-like terminal. Contrast with *display mode*.

load. In reference to installation and service, to move files from tape to disk, auxiliary storage to main storage, or minidisks to virtual storage within a virtual machine.

lock. A tool for controlling concurrent usage of SFS objects. Implicit locks are acquired and automatically released when you run CMS commands and program functions in SFS. Explicit locks let you control the type and duration of the lock.

logical saved segment. A portion of a physical saved segment that CMS can manipulate. Each logical segment can contain different types of program objects, such as modules, text files, execs, callable services libraries, language repositories, user-defined objects, or a single minidisk directory. A system segment identification file (SYSTEM SEGID) associates a logical saved segment to the physical saved segment in which it resides. See *physical saved segment* and *saved segment*.

logoff. The procedure by which a user ends a terminal session.

logon. The procedure by which a user begins a terminal session.

M

master file directory. A directory on each CMS disk that contains the name, format, size, and location of all the CMS files on the disk. When a disk is accessed by the ACCESS command, the directory is read into main storage and identified with one of the 26 disk mode letters (A through Z).

minidisk. A logical subdivision (or all) of a physical disk pack that has its own virtual device address, consecutive virtual cylinders (starting with virtual cylinder 0), and a VTOC or disk label identifier. Each user virtual disk is preallocated and defined by a VM/SP directory entry as belonging to a user.

MORE screen status. For a display terminal used as a virtual console under VM/SP, an indicator located in the lower right of the screen that displays when the user's display screen is full and more data will be displayed. After 60 seconds, the screen is automatically erased and the next screen is displayed. To immediately clear the screen, press the Clear, Cancel, or PA2 key. To hold the data on the screen longer than 60 seconds, press the Enter key to enter HOLDING status. See *HOLDING screen status*.

N

NOT ACCEPTED screen status. For a display terminal used as a virtual console under VM/SP, an indicator in the lower right of the screen that displays that the user is: (1) trying to enter another command line, but the terminal buffer still contains a previous command line, and (2) using the copy function to copy the contents of the screen onto an associated hardcopy printer; however, the printer is busy, nonexistent, or otherwise unavailable.

nucleus. The part of CP or CMS resident in main storage.

null line. A logical line with a length of zero that usually signals the CMS Editor to end input mode and enter edit mode. In VM/SP, a null line for typewriter terminals is a terminal input line consisting of a return character as the first and only information, or a logical line end symbol as the last character in the data line. For display devices, a null line is indicated by the cursor positioned at the beginning of the user input area or the data in the user input area ending with a logical line end symbol.

P

parameter list (PLIST). In CMS, a string of 8-byte arguments that call a CMS command or function. The first argument must be the name of the command or function to be called. General register 1 points to the beginning of the parameter list.

parent directory. (1) The directory for a CMS disk that has a disk extension defined for it by the ACCESS command. (2) In SFS, the next higher-level directory in which the current directory is defined.

password. In computer security, a string of characters known to the computer system and a user, who must specify it to gain full or limited access to a system and

to gain full or limited access to a system and to the data stored within it.

PF key. Program function key.

physical saved segment. One or more pages of storage that have been named and retained on a CP-owned volume (DASD). Once created, it can be loaded within a virtual machine's address space or outside a virtual machine's address space. Multiple users can load the same copy. A physical saved segment can contain one or more logical saved segments. A system segment identification file (SYSTEM SEGID) associates a physical saved segment to its logical saved segments. See *logical saved segment* and *saved segment*.

physical screen. Synonym for *screen*.

PROFILE EXEC. A special EXEC procedure with a file name of PROFILE that a user can create. The procedure is usually executed immediately after CMS is loaded into a virtual machine (also known as IPL CMS).

program function (PF) key. On a terminal, a key that can do various functions selected by the user or determined by an application program.

R

read authority. The authority to read the contents of a file without being able to change them. For a directory, read authority lets the user view the names of the objects in the directory.

read-only access. An access mode associated with a virtual disk or SFS directory that lets a user read, but not write or update, any file on the disk or SFS directory.

read/write access. An access mode associated with a virtual disk or SFS directory that lets a user read and write any file on the disk or SFS directory.

reserved file types. (1) File types recognized by the CMS editors (EDIT and XEDIT) as having specific default attributes that include: record size, tab settings, truncation column, and uppercase or lowercase characters associated with that particular file type. The CMS Editor creates a file according to these attributes. (2) File types recognized by CMS commands; that is, commands that only search for and use particular file types or create one or more files with a particular file type.

Restructured Extended Executor (REXX) language. A general-purpose programming language, particularly suitable for EXEC procedures, XEDIT macros, or programs for personal computing. Procedures, XEDIT macros, and programs written in this language can be

interpreted by the System Product Interpreter. Contrast with *CMS EXEC language* and *EXEC 2 language*.

REXX language. Restructured Extended Executor language.

RUNNING screen status. For a display terminal used as a virtual console under VM/SP, an indicator located in the lower right of the screen. It displays that the user's virtual machine is in control (but not necessarily executing a program or command) and that the terminal can receive messages.

S

saved segment. A segment of storage that has been saved and assigned a name. The saved segment(s) can be physical saved segment(s) that CP recognizes or logical saved segments that CMS recognizes. The segments can be loaded and shared among virtual machines, which helps use real storage more efficiently, or a private, nonshared copy can be loaded into a virtual machine. See *logical saved segment* and *physical saved segment*.

screen. An illuminated display surface; for example, the display surface of a CRT. Synonymous with *physical screen*.

screen status area. For a display terminal used as a virtual console under VM/SP, an indicator of the current status of the display screen. This indicator is located in the lower right of the display screen. See *CP READ screen status*, *HOLDING screen status*, *MORE screen status*, *NOT ACCEPTED screen status*, *RUNNING screen status*, and *VM READ screen status*.

SCRIPT/VS. A component of the IBM Document Composition Facility program product available from IBM for a license fee.

scrolling. (1) Moving a display image vertically or horizontally in order to view data not otherwise visible within the boundaries of the display screen.

(2) Performing a scroll up, scroll down, scroll right, or scroll left operation.

SFS. Shared file system.

SFS directory. A group of files. SFS directories can be arranged to form a hierarchy in which one directory can contain one or more subdirectories as well as files.

shared file system (SFS). A part of CMS that lets users organize their files into groups known as *directories* and to selectively share those files and directories with other users.

shared segment. A feature of a saved system or physical saved segment that lets one or more segments of reentrant code or data in real storage be shared among

many virtual machines. For example, if a saved CMS system was generated, the CMS nucleus is shared in real storage among all CMS virtual machines loaded by name; that is, every CMS machine's segment of virtual storage maps to the same 64K of real storage. See *discontiguous saved segment* and *saved system*.

signaling attention. An indication that a user has pressed a key or keyed in a CP command to present an attention interrupt to CP or to the user's virtual machine.

spool file class. A one-character class associated with each virtual unit record device. For input spool files, the spool file class lets the user control which input spool files are read next; and, for output spool files, it lets the spooling operator better control or reorder the printing or punching of spool files having similar characteristics or priorities. The spool file class value can be A through Z or 0 through 9. See *virtual console spooling*.

storage group. A subset of minidisks within a file pool. Each storage group is identified by a number.

subcommand. The commands of processors such as EDIT or System Product Editor (XEDIT) that run under CMS.

subdirectory. Any directory below a user's top directory. The CREATE DIRECTORY command creates subdirectories. There can be up to eight levels of subdirectories with no limit on the number of them at each level, other than overall DASD space limits. Each level of a subdirectory is an additional identifier of up to 16 characters that is appended to next higher level subdirectory.

system administrator. The person responsible for maintaining a computer system.

System Product Editor. The CMS facility, comprising the XEDIT command and XEDIT subcommands and macros, that lets a user create, change, and manipulate CMS files.

System Product Interpreter. The language processor of the VM/SP operating system that processes procedures, XEDIT macros, and programs written in the REXX language.

T

terminal. A device, usually equipped with a keyboard and a display, capable of sending and receiving information.

terminal session. The period of time from logon to logoff when a user and the virtual machine can use the facilities of VM/SP or the operating system or both. This also includes any period of time that the virtual

machine is running in disconnect mode. See *disconnect mode*.

tokenized PLIST (parameter list). A string of doubleword aligned parameters occupying successive doublewords.

top directory. The directory created for a user when the user is enrolled in a file pool. The name of the top directory is the same as the person's user ID.

topmost window. With the window support, the highest window in the display order such that: (1) The window name is not WM or STATUS. (2) The window currently displays at least one virtual screen data line or reserved line. For example, a vsize window connected to a virtual screen such that there are no scrollable data being displayed, is NOT the topmost window.

Note: It may not be obvious by looking at the screen which is the topmost window.

truncation. A valid shortened form of CP, CMS, GCS, IPCS, RSCS, TSAF (Query only) command names, operands, and options that can be keyed in. When the shortened form is used, the number of key strokes is reduced. For example, the ACCESS command has a minimum allowable truncation of two, so AC, ACC, ACCE, ACCES, and ACCESS are all recognized by CMS as the ACCESS command. Contrast with *command abbreviation*.

typewriter terminal. Printer-keyboard devices that produce hardcopy output only, such as: the IBM 2741 Communication Terminal; the IBM 3215 Console Printer-Keyboard; the IBM 3767 Communication Terminal, Model 1 or 2, operating as a 2741. This term also refers to the IBM 3101 Display Terminal operating as a 2741.

U

universal class card reader. A virtual card reader that can read any class of reader, printer, or punch files spooled or transferred to it.

user ID. User identification.

user input area. On a display device, the lines of the screen where the user is required to key in command or data lines. See *display mode*, *input line*, and *line mode*.

user-written CMS command. Any CMS file created by a user that has a file type of MODULE or EXEC. Such a file can be executed as if it were a CMS command by issuing its file name, followed by any operands or options expected by the program or EXEC procedure.

V

virtual address. The address of a location in virtual storage. A virtual address must be translated into a real address in order to process the data in processor storage.

virtual card reader. CP's simulation on disk of a real card reader. A virtual card reader can read card, punch, or print records of up to 151 characters in length. The virtual device type and I/O device address are usually defined in the VM/SP directory. See *spool file class* and *universal class card reader*.

virtual console. A console simulated by CP on a terminal such as a 3270. The virtual device type and I/O address are defined in the VM/SP directory entry for that virtual machine.

virtual console spooling. The writing of console I/O on disk as a printer spool file instead of, or in addition to, having it typed or displayed at the virtual machine console. The console data includes messages, responses, commands, and data from or to CP and the virtual machine operating system. The user can invoke or terminate console spooling at anytime. When the console spool file is closed, it becomes a printer spool file. Synonymous with *console spooling*.

virtual machine (VM). A functional equivalent of a real machine.

Virtual Machine/System Product (VM/SP). An IBM licensed program that manages the resources of a single computer so that multiple computing systems appear to exist. Each virtual machine is the functional equivalent of a *real* machine.

virtual printer (or punch). A printer (or card punch) simulated on disk by CP for a virtual machine. The virtual device type and I/O address are usually defined in the VM/SP directory entry for that virtual machine.

virtual screen. A functional simulation of a physical screen. A virtual screen is a *presentation space* where data is maintained. The user can view pieces of the virtual screen through a window on the physical screen.

virtual storage. Storage space that can be regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses. The size of virtual storage is limited by the addressing scheme of the computing system and by the amount of auxiliary storage available, and not by the actual number of main storage locations.

VM. Virtual machine.

VM READ screen status. For a display terminal used as a virtual console under VM/SP, an indicator located in the lower right of the screen that displays when the

user's virtual machine is not executing, but is waiting for a response or a request for work from the user.

VM/SP. Virtual Machine/System Product.

VM/SP directory. A CP disk file that defines each virtual machine's typical configuration; the user ID, password, regular and maximum allowable virtual storage, CP command privilege class or classes allowed, dispatching priority, logical editing symbols to be used, account number, and CP options desired. Synonymous with *CP directory*.

vscreen. Virtual screen.

W

window. An area on the physical screen where virtual screen data can be displayed. Windowing lets the user do such functions as defining, positioning, and overlaying windows; scrolling backward and forward through data; and writing data into virtual screens.

windowing. A set of functions that lets the user view and manipulate data in user-defined areas of the physical screen called *windows*. Windowing support lets the user define, position, and overlay windows; scroll backward and forward through data; and write data into virtual screens.

write authority. The authority to read or change the contents of a file or directory. Write authority implies read authority.

X

XEDIT. See *System Product Editor*.

XEDIT macro. (1) A procedure defined by a frequently used command sequence to do a commonly required editing function. A user creates the macro to save repetitious rekeying of the sequence, and invokes the entire procedure by entering a command (that is, the macro file's file name). The procedure can consist of a long sequence of XEDIT commands and subcommands or both, and CMS and CP commands or both, along with REXX or EXEC 2 control statements to control processing within the procedure. (2) A CMS file with a file type of *XEDIT*.

Numerics

2741. Refers to the IBM 2741 Terminal. Information on the 2741 also applies to the IBM 3767 Terminal, unless otherwise noted.

3262. Refers to the IBM 3262 Printer, Models 1 and 11.

3270. Refers to a series of IBM display devices; for example, the IBM 3275, 3276 Controller Display Station, 3277, 3278, and 3279 Display Stations, the 3290 Information Panel, and the 3287 and 3286 printers. A specific device type is used only when a distinction is required between device types. Information about display terminal usage also refers to the IBM 3138, 3148, and 3158 Display Consoles when used in display mode, unless otherwise noted.

3284. Refers to the IBM 3284 Printer. Information on the 3284 also pertains to the IBM 3286, 3287, 3288, and 3289 printers, unless otherwise noted.

3289. Refers to the IBM 3289 Model 4 Printer.

3330. Refers to the IBM 3330 Disk Storage Device.

3370. Refers to the IBM 3370 Direct Access Storage Device.

3380. Refers to the IBM 3380 Direct Access Storage Device.

3422. Refers to the IBM 3422 Magnetic Tape Subsystem.

3480. Refers to the IBM 3480 Magnetic Tape Subsystem.

3725. Refers to the IBM 3725 Communications Controllers.

3800. Refers to the IBM 3800 Printing Subsystems. A specific device type is used only when a distinction is required between device types.

4245. Refers to the IBM 4245 Printer.

4248. Refers to the IBM 4248 Printer.

9313. Refers to the IBM 9313 Direct Access Storage Device.

9332. Refers to the IBM 9332 Direct Access Storage Device, Model 400.

9335. Refers to the IBM 9335 Direct Access Storage Device, Models A01 and B01.

9347. Refers to the IBM 9347 Tape Drive.

9370. Refers to a series of processors, namely the IBM 9373 Models 20 and 30, the IBM 9375 Models 40, 50, and 60, and the IBM 9377 Models 80 and 90.

Bibliography

Prerequisite Publications

Virtual Machine/System Product:

Introduction, GC19-6200

Terminal Reference, GC19-6206

A new user of CMS might refer to the *VM/SP CMS Primer*, SC24-5236, for introductory tutorial information on using CMS in display mode. If you are using a line-oriented terminal, then refer to the *VM/SP CMS Primer for Line-Oriented Terminals*, SC24-5242.

Corequisite Publications

Virtual Machine/System Product:

CMS Command Reference, SC19-6209

CP General User Command Reference, SC19-6211

EXEC 2 Reference, SC24-5219

System Messages and Codes, SC19-6204

System Messages Cross-Reference, SC24-5264

System Product Editor Command and Macro Reference, SC24-5221

System Product Editor User's Guide, SC24-5220

System Product Interpreter Reference, SC24-5239

System Product Interpreter User's Guide, SC24-5238

Quick References

There are publications available as quick reference material when you use VM/SP and CMS. They are:

Virtual Machine/System Product:

CMS Command Reference Summary, SX24-5220

Quick Reference, SX20-4400

System Product Editor Command Language Reference Summary, SX24-5122

System Product Interpreter Reference Summary, SX24-5126.

Related VM/SP Publications

Additional descriptions of various CMS functions and commands that are normally used by system support personnel are described in the following publications:

Virtual Machine/System Product:

Application Migration Guide for CMS, SC24-5366

Application Development Guide for CMS, SC24-5286

Application Development Reference for CMS, SC24-5284

Application Development Guide for FORTRAN and COBOL, SC24-5247

Administration, SC24-5285

CMS Shared File System Administration, SC24-5367

Diagnosis Guide, LY24-5241

Installation Guide, SC24-5237

Operator's Guide, SC19-6202

Planning Guide and Reference, SC19-6201

Virtual Machine:

Running Guest Operating Systems, SC19-6212

System Facilities for Programming, SC24-5288

Details on the use of OS/VS EREP operands for the CMS command CPERP are contained in the *OS/VS, DOS/VSE, VM/370 Environmental Recording, Editing, and Printing Program*, GC28-0772.

Related Publications for OS Users

For information on OS/VS tape label processing, discussed with "Label Processing in OS Simulation" in this publication, refer to:

OS/VS1 Data Management Services Guide, GC26-3874

OS/VS2 MVS Data Management Services Guide, GC26-3875

OS/VS Tape Labels, GC26-3795.

Information on the linkage editor is contained in *MVS/XA Linkage Editor and Loader User's Guide*, GC26-4143.

Related Publications for VSAM and Access Method Services Users

CMS support of Access Method Services is based on VSE and VSE/VSAM. The control statements that you can use are described in *Using VSE/VSAM Command and Macros*, SC24-5144.

Error messages produced by the Access Method Services program, and return codes and reason codes, are listed in *VSE/VSAM Messages and Codes*, SC24-5146.

For a detailed description of VSE/VSAM macros and macro parameters, refer to the *VSE/AF Macro User's Guide*, SC24-5210.

For information on OS/VS VSAM macros, refer to *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide*, GC26-3838.

Information on formatting virtual minidisks using the Device Support Facility Program is found in the *Device Support Facilities User's Guide and Reference*, GC35-0033.

Related Publications for CMS/DOS Users

The CMS ESERV command invokes the VSE ESERV program, and uses, as input, the control statements that you would use in VSE. These control statements are described in *Guide to the DOS/VSE Assembler*, GC33-4024.

Linkage editor control statements, used when invoking the linkage editor under CMS/DOS, are described in *Guide to the DOS/VSE Assembler*, GC33-4024.

For information on DOS/VSE and CMS/DOS tape label processing, refer to the following publications:

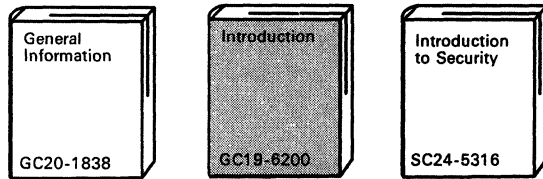
VSE/AF Tape Labels, SC24-5212

VSE/AF Macro User's Guide, SC24-5210.

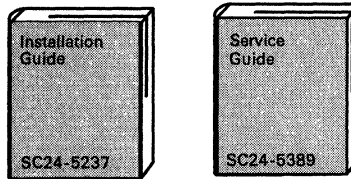
For information about using DL/I in the CMS/DOS environment, see *DL/I DOS/VS Data Base Administration*, SH24-5011.

VM/SP RELEASE 6 LIBRARY

Evaluation



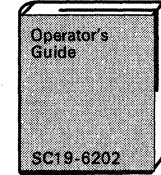
Installation and Service



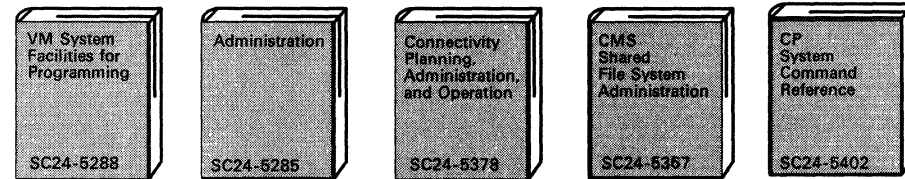
Planning



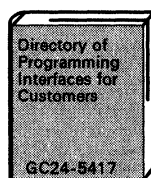
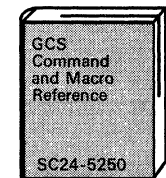
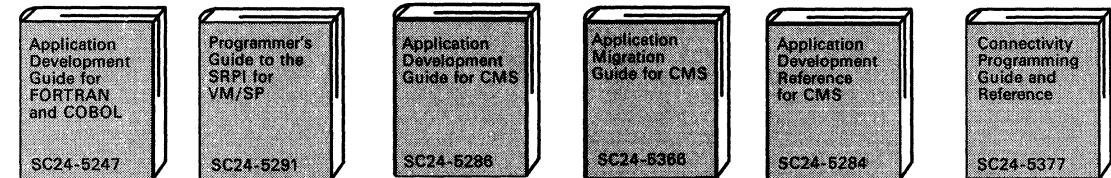
Operation



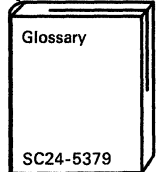
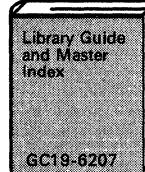
Administration



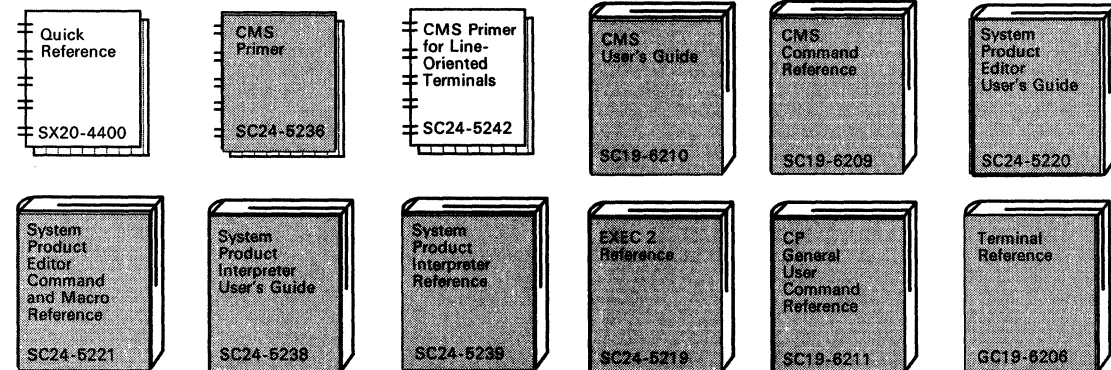
Application Development



Index/Glossary



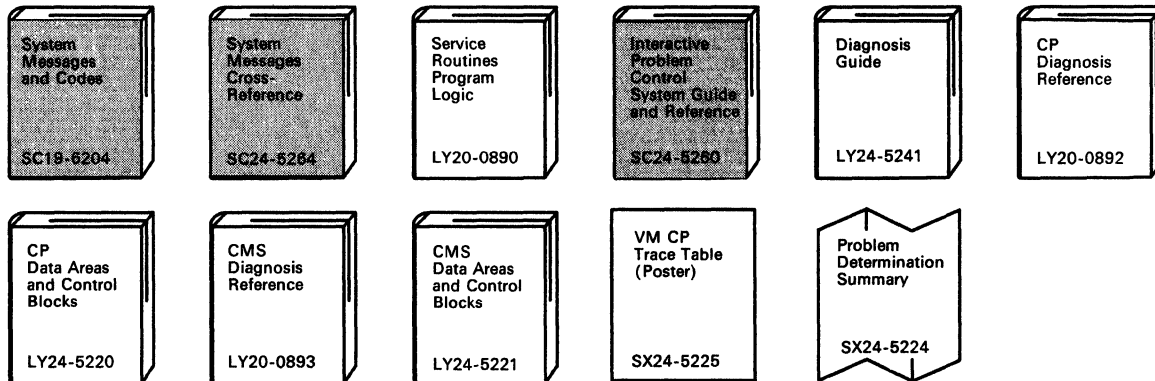
End Use



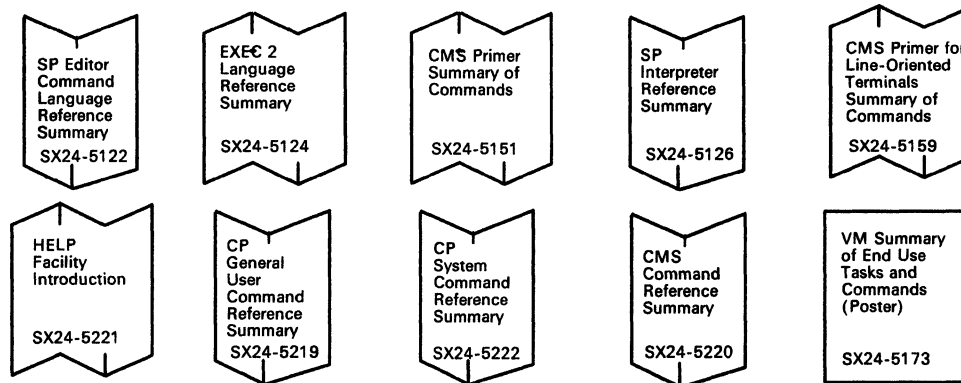
 one copy of each shaded manual received with product tape

VM/SP RELEASE 6 LIBRARY

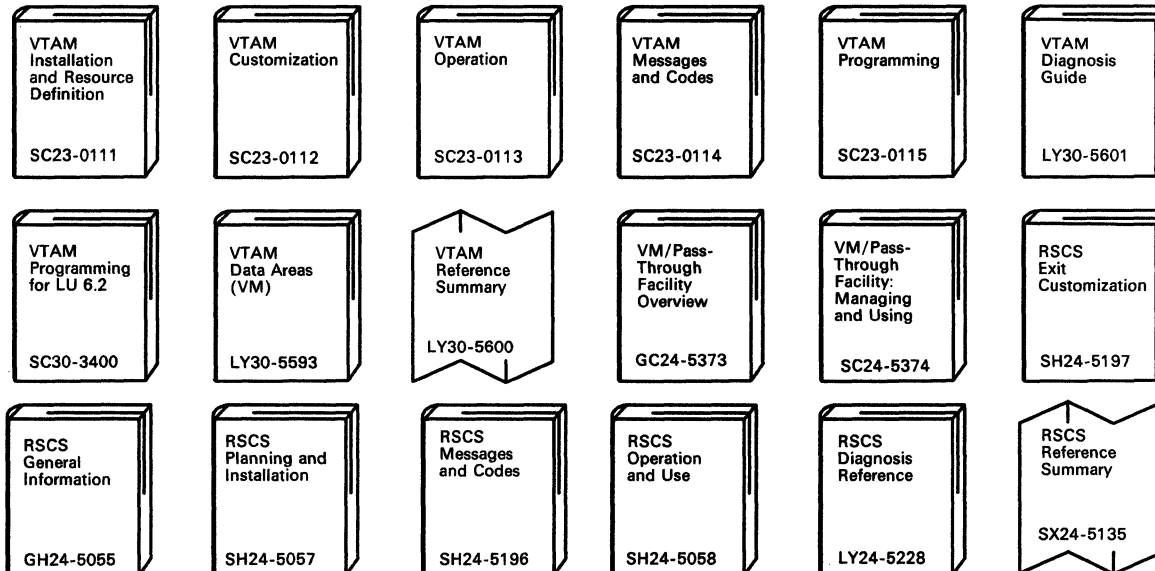
Diagnosis



Reference Summaries



Auxiliary Communication Support





Index

A

A-disk 12, 48
ACCESS command 74
accessing
 directories 23
 file modes
 as read-only extensions 50
 in CMS batch virtual machine 254
 master file directories for minidisks 130
 minidisks 23
adding
 BRIEF section to HELP files 317
 DESCRIPT section to HELP files 317
 DETAIL section to HELP files 317
 ERRORS section to HELP files 317
 FORMAT section to HELP files 317
 HELP components 320
 NOTES section to HELP files 317
 OPTIONS section to HELP files 317
 PARMS section to HELP files 317
 RELATED section to HELP files 318
address
 virtual, for unit record devices 171
administrator authority 108
ADSTOP command (CP) 345
ALARM VSCREEN command 350
ALIALIST command
 description 103
 PF keys 105
alias
 creating
 description 95
 to another user's file 99
 to your own file 96
 locking 121
 querying
 using ALIALIST 103
 using QUERY ALIAS 102
ALL option of DETAIL HELP 148
altering
 characteristics of spool files 174
 existing HELP files 330
AMSERV command 44
applications in full-screen CMS 361
ASM3705 file type usage in CMS/DOS 41
ASSEMBLE command usage in CMS 41
assigning file mode letters 48
attention interruptions
 causing 32
 virtual machine 33
ATTN command (CP) 345
AUTHLIST display, PF keys 114

authority
 administrator 108
 granting 109
 to a group of users 203
 querying
 using AUTHLIST command 112
 using QUERY AUTHORITY command 112
 read
 on a directory 107
 on a file 107
 revoking 110
 from a group of users 203
 write
 on a directory 108
 on a file 108
automatic
 IPL 9
 save function for editors 164
AUTOREAD operand of CMS SET command for
 display terminals 15
AUTOSAVE command usage in CMS 41
AUXxxxx file type usage in CMS 41

B

base files
 description 95
 erasing 105
batch facility (CMS)
 control cards 249
 controlling spool files 255
 description 249
 housekeeping done after executing job 254
 how jobs are processed 254
 ID card 249
 jobs for non-CMS users 262
 using exec procedure to submit jobs 258
 /JOB 252
 /SET 253
 /* 252
batch jobs
 for CMS batch facility 249
 for Non-CMS users 262
 purging 257
 reordering 257
batch processing in CMS 249
BEGIN command (CP) 28, 345
beginning
 virtual machine execution 28
 your terminal session 6
BLP
 See bypass label processing, tapes
Border commands
 scrolling forward and backward 238

Border commands (*continued*)
 scrolling right and left 241
BRIEF HELP 147
bulletin board, creating 117
bypass label processing, tapes 184

C

Callable Services Library (CSL) 365
caller ID in tape label processing 185
canceling changes during editing session 165
card punch
 used to send jobs to CMS batch facility 249
card reader
 restriction on use in job for CMS batch facility 256
 spooling punch or printer files to 173
cards
 as input to CMS batch facility 249
 /* as end-of-file indicator 252
causing breaks in text 329
CHANGE command (CP) 174, 345
changing
 characteristics of spool files 171
 characteristics of unit record devices 171
 file mode numbers 57
 output representation of a character 329
 the HELP facility 309
CLASS operand of SPOOL command 172
classes
 CP command privilege 343
 of CP SPOOL files 172
cleanup functions for VSAM 267
CLEAR VSCREEN command 350
CLEAR WINDOW command 350
clock indicator in full-screen CMS 223
CLOSE command (CP) 345
CMS Editor
 environment 30
CMS environment 29
CMS EXEC procedures
 submitting jobs to CMS batch facility 251
CMS file 19, 37
CMS files
 See files
CMS in full-screen
 See full-screen CMS
CMS installation saved segment 274, 280
 discontinuing 281
 use of 274, 280
 using IPL command to link 281
CMS subset
 environment 30
 using 168
CMS (Conversational Monitor System)
 considerations for migrating to full-screen CMS 358
 description 3
 file system 37
 files
 See files, CMS

CMS (Conversational Monitor System) (*continued*)
 loading into your virtual machine 8
 migrating to full-screen CMS 357
 understanding it 3
CMS (Conversational Monitor System) commands
 execution characteristics 60
 general information 4
 processing tape labels 191
 search order 59
 summary 333
 transient area 60
 used with execs written in REXX language 277
 user area 60
CMSUT1 file and CMS commands that create 47
CMS/DOS
 end-of-tape processing 193
 overview 31
 tape label processing 188
CNTRL command, file type usage in CMS 41
command abbreviations used in HELP 319
command HELP 146, 150
command line in full-screen CMS 222
commands
 environments 27
 how to enter 3
 language 3
 CMS 4
 CP 3
 search order 59
 translation synonyms 67
 translations 67
COMMANDS command (CP) 345
comments
 in HELP text files 324
 in REXX language 269
communicating
 with other computer users 201
 with VM/SP 3
COMPACT option of DDR command 178
compacting tape output 178
COMPARE command to compare contents of two CMS files 64
compilers supported in CMS 4
component MENU in HELP 153
components of VM/SP 3
composing notes 210
CONCEAL option of SET command 30
conditionally displaying text 325
considerations for windowing and full-screen CMS 349
console
 log
 creating file from 24
 printing 24
 produced by CMS batch facility 257
 output spooling for display terminal 24
CONT
 operand, of CP SPOOL command 173

continuous spooling 173
 control cards for CMS batch facility
 See batch facility (CMS)
 Control Program (CP)
 basic description 3
 command information 3
 considerations for migrating to full-screen CMS 360
 environment, entering 28
 privilege classes 343
 spooling facilities 171
 conventions, notational for HELP 317
 Conversational Monitor System (CMS)
 considerations for migrating to full-screen CMS 358
 description 3
 file system 37
 loading into your virtual machine 8
 migrating to full-screen CMS 357
 understanding it 3
 converting
 file to a HELP file 321
 fixed-length file to variable-length format 66
 variable-length file to fixed-length format 65
 COPY
 file type usage in CMS 41
 function on display terminals 25
 operand of CP SPOOL command 172
 COPYFILE command 81
 changing file mode numbers 53, 57
 copying files 65
 creating small files from large one 167
 used to change record formats of files 65
 COPYFILE command, REPLACE option 53
 copying
 contents of a display screen 25
 files, from one device to another 176
 files, to a directory 81
 file, with COPYFILE command 65
 from tape to minidisk or SFS directory 176
 spool files 172
 corrections
 of lines as you enter them 11
 COUPLE command (CP) 345
 CP command 345
 CP READ status on display terminal 14
 CP (Control Program)
 basic description 3
 command information 3
 considerations for migrating to full-screen CMS 360
 environment, entering 28
 privilege classes 343
 spooling facilities 171
 CP (Control Program) commands
 summary 345
 used in job for CMS batch facility 256
 used in System Product Interpreter execs 272
 CREATE LOCK command 120
 creating
 aliases 94
 creating (*continued*)
 aliases to files 95
 an alias to an alias 95
 an alias to another user's file 99
 an alias to your own file 96
 command HELP files 316
 file with System Product Editor 161
 HELP message files 318
 HELP text files 330
 HELPMENU files 313
 HELPTASK files 314
 menus, HELP file 312
 notes 210
 one spool file from many files being printed or
 punched 173
 PROFILE EXEC 273
 System Product Interpreter Execs 269
 CSL (Callable Services Library) 365
 CSLCNTRL file type
 usage in CMS 41
 CSLLIB file type
 usage in CMS 41
 CSLSEG file type
 usage in CMS 41
 CURSOR VSCREEN command 350

D
 data area in full-screen CMS 222
 DCSSMAP file type usage in CMS 41
 DDR command
 COMPACT option for tape output 178
 DDR program used to dump to tape 178
 default
 notebook 211
 setting with DEFAULTS command 211
 DEFINE command
 defining temporary minidisks 127
 to increase virtual storage size 166
 DEFINE command (CP) 345
 DEFINE VSCREEN command 290, 350
 DEFINE WINDOW command 290, 350
 defining
 tapes
 nonstandard 188
 standard 189
 unlabeled 188
 temporary minidisks 127
 Definition Language for Command Syntax (DLCS)
 file 67
 DELETE LOCK command 122
 DELETE VSCREEN command 350
 DELETE WINDOW command 350
 deleting HELP files 330
 density of tapes 198
 DESCRIP option of DETAIL HELP 148
 DETACH command (CP) 345

- detaching minidisks 130
- DETAIL HELP 148
- DIAL command (CP) 346
- DIRECT file type usage in CMS 41
- directories
 - accessing another user's 74
 - copying files to 81
 - copying to directory from tape 176
 - creating 79
 - definition 69
 - determining if locked 122
 - determining ownership 115
 - displaying status 23
 - erasing files from 142
 - extensions 49
 - granting read authority on 54
 - hierarchical structure 72
 - providing for CMS batch virtual machine 254
 - querying accessed 19
 - read-only extensions 49
 - referring to 73
 - releasing 24
 - releasing, during a terminal session 75
 - search order 23, 49
 - sharing 140
 - structure, viewing with DIRLIST 76
 - structure, viewing with LISTDIR 78
 - writing CMS files to 52, 161
 - writing files to 164
- directory identifier
 - commands that accept 75
 - ways to specify 75
 - when to specify a file mode as 75
- directory name 38
- dirid 75
- DIRLIST command 76
- DIRLIST display, PF key functions 77
- dirname 73
- DISCARD command 63
- discarding files, command environments 63
- DISCONN command 10
- DISCONN command (CP) 346
- disconnecting your terminal from your virtual machine 10
- discontinuing access to exec in saved segment 281
- DISK command
 - DUMP operand, using 176
 - LOAD operand restricted in job for CMS batch facility 256
 - LOAD operand, using 176
 - loading files 211
- display
 - full-screen, in XEDIT 162
 - multiple views 161
- DISPLAY command (CP) 346
- display screen status notices 13
- display terminals
 - characteristics 13

- display terminals (*continued*)
 - entering commands 10
 - example of display screen 26
 - Extended highlight feature 17
 - line-mode 170
 - retrieving previously entered data 11
 - setting PF keys 12
 - signalling interruptions 32
- displaying
 - list of CMS files 61
 - screens of data 15
- disposition of spool files 171
- DLBL command
 - assigning file mode numbers 57
- DLCS (Definition Language for Command Syntax) file 67
- DMSVSR command 267
- drawing boxes 322
- DROP WINDOW command 231, 350
- DUMP command (CP) 346
- duplicate file names or file types 38
- DVOLL function in tape command processing 192

E

- ECHO command (CP) 346
- edit mode, returning from input mode 162
- editing
 - CMS files 161
 - session 161
 - shared files 119
- end of file
 - indicating for input stream to batch virtual machine 252
- end-of-tape, processing 193
- end-of-volume, processing 193
- ending
 - editing session 21
 - terminal session 9
- enrolling in a file pool 71
- entering
 - CMS commands, in CMS subset environment 31
 - CMS environment 8
 - CMS/DOS environment 31
 - commands
 - from the screen 234
 - on display terminal 10
 - CP commands
 - from CMS command environment 28
 - from edit environment 30
 - CP environment
 - from CMS environment 28
 - HELP facility 5
 - Immediate commands 32
 - null lines 3
- environments of VM/SP 27
- ERASE command 22, 87

- erasing
 - base files 105
 - CMS files 22
 - directories 87
 - files to clear file space during editing session 58, 169
 - files to clear minidisk space during editing session 168
- error messages
 - controlling whether you receive them 208
- error processing
 - messages 196
 - NSL routines 196
 - OS simulation 196
 - standard label processing 196
- ERRORS option of DETAIL HELP 148
- EXEC
 - file type used in CMS 41
 - loading into saved segment 274
 - loading into storage 280
 - procedures
 - using to submit jobs to CMS batch facility 259
- EXEC 2
 - comparison to exec 265
 - comparison to System Product Interpreter 265
 - files
 - attributes 269
 - format 269
 - running 269
 - used with System Product Editor 161
 - &TRACE statement 269
- EXECDROP command
 - description 277
 - discontinuing access to exec in saved segment 281
 - example 281
- EXECIO command
 - description 277
 - example 278
- EXECLOAD command
 - description 277
 - example 280
 - improving performance of HELP 145
- EXECMAP command
 - description 277
 - example 281
 - listing execs in storage 281
- EXECSTAT command
 - description 277
 - example 280
- executing
 - command using program function (PF) keys 11
 - Exec procedures 269
 - Immediate commands in execs 277
 - PROFILE EXEC 273
- execution characteristics of CMS commands 60
- execution state indicator in full-screen CMS 223
- exiting from
 - editing session, saving changes 21

- exiting from (*continued*)
 - editing session, without saving changes 165
 - execution of a program 34
 - terminal session 9
- explicit locks 119, 122
- EXTEND option of DETAIL HELP 150
- extensions, read-only, using 48
- EXTERNAL command (CP) 346

F

- file
 - converting to a HELP file 321
 - identifier
 - changing with SAVE subcommand 165
 - CMS, rules for assigning 37
 - coded as equal sign (=) 39
 - master file directories 130
 - system 37
- file manipulation using System Product Editor 161
- file mode
 - displaying status 23
 - extensions 49
 - in file identifier 37
 - letters
 - assigning 38
 - when to specify for reading files 50
 - when to specify for writing files 52
 - numbers
 - changing 53
 - commands used to change 57
 - default 53
 - descriptions 48
 - used to manipulate subsets of files 54
 - when to specify 53
 - providing for CMS batch virtual machine 254
 - read-only extensions 49
 - search order 23
- file mode A
 - definition 12
 - status 48
 - storing files 48
- file mode determination
 - default for reading files
 - commands that search all accessed file modes 50
 - commands that search only file mode A 51
 - commands that search only file mode A and its extensions 51
 - default for writing files
 - commands for which you must specify file mode 52
 - commands that write output to file mode A 52
 - commands that write output to read/write file mode 52
- file mode letter change during a terminal session 38
- file mode numbers
 - in SFS
 - file mode number 0 54
 - file mode number 1 54

file mode numbers (*continued*)
 in SFS (*continued*)
 file mode number 2 54
 file mode number 3 54
 file mode number 4 55
 file mode number 5 55
 file mode number 6 55
 on minidisks
 file mode number 0 55
 file mode number 1 55
 file mode number 2 55
 file mode number 3 56
 file mode number 4 56
 file mode number 5 56
 file mode number 6 56
 file name 37
 file pool
 availability 9
 default 72, 125
 definition 69
 enrolling in 71
 IBM-supplied (VMSYSU) 71
 primary 126
 using more than one 125
 file pool identifier 73
 file space
 determining amount used 124
 determining limits 58
 full, recovering from in editing session 169
 managing 58
 size of 58
 file status table (FST)
 FILE subcommand to write CMS files 161
 file type
 created by assembler and language processors 40
 HELP facility 310
 HELPABBR 42
 HELPCMS 42
 HELPCMSQ 42
 HELPCMSS 42
 HELPCP 42
 HELPCPOT 42
 HELPCPQU 42
 HELPCPSE 42
 HELPEDIT 42
 HELPEXC2 42
 HELPEXEC 42
 HELPGROU 42
 HELPHELP 42
 HELPIPCS 42
 HELPMENU 42
 HELPMMSG 42
 HELPPREF 42
 HELPPVM 42
 HELPQUER 42
 HELPREXX 42
 HELPSET 42
 HELPSQLD 42
 HELPSRPI 42

file type (*continued*)
 HELP facility (*continued*)
 HELPTASK 42
 HELPTSFAF 42
 HELPXEDI 42
 \$HLPABBR 43
 \$HLPCMS 43
 \$HLPCMSQ 43
 \$HLPKMSS 43
 \$HLPKP 43
 \$HLPKPQU 43
 \$HLPKPSE 43
 \$HLPDEBU 43
 \$HLPEDIT 43
 \$HLPXEC2 43
 \$HLPXEC 43
 \$HLPGROU 43
 \$HLPHELP 43
 \$HLPKPCS 43
 \$HLPMENU 43
 \$HLPMSG 43
 \$HLPREF 43
 \$HLPQUER 43
 \$HLPREXX 43
 \$HLPSET 43
 \$HLPSQLD 43
 \$HLPSTASK 43
 \$HLPSTAF 43
 \$HLPXEDI 43
 in file identifier 37
 reserved for language processors 40
 temporary work files 46
 used by CMS commands 40
 FILEDEF command
 assigning file mode numbers 57
 OS simulation 181
 standard tape labels 182
 tape label processing 186
 FILELIST command
 in full-screen CMS 61
 in SFS 61
 FILELIST command used to list files in full-screen
 environment 61
 FILELIST SEARCH screen
 description 96
 PF keys 98
 FILELIST SHARE screen
 description 92
 PF keys 93
 FILELIST STATS screen
 description 90
 PF keys 92
 filepoolid 73
 files
 authority 107
 CMS
 creating files that are erased after being read 54,
 56
 erasing 63

files (*continued*)

- CMS (*continued*)
 - format 37
 - identifiers 37
 - renaming 65
- determining if locked 122
- determining ownership 115
- discarding after being read 63
- logical grouping 46
- requesting information about 61
- sharing
 - across file pools 99
 - commands used 140
 - creating aliases to files 95
 - how to 94
- splitting into smaller files 167
- too large to edit, what to do 167
- with users on other systems 126

FOR operand of CP SPOOL command, usage 172

FORMAT command to format a CMS minidisk 128

FORMAT option of DETAIL HELP 148

format words

- summary 322
- .IL (Indent Line) 326
- .BX (BOX) 322
- .CM (Comment) 324
- .CS (Conditional Section) 325
- .FO (Format Mode) 325
- .IN (Indent) 326
- .MT (Menu Type) 327
- .OF (Offset) 328
- .SP (Space Lines) 329
- .TR (Translate Character) 329

format-mode, processing 330

formatting

- CMS minidisks, example 128

full file space during editing session 169

full minidisk during editing session 168

full-screen CMS 221

- Border commands 235, 351
- commands 349
- considerations for migrating 357
- entering commands from the screen 234
- location information 224
- maximizing windows 294
- messages 231
- migrating from line-mode CMS 357
- PA and PF keys
 - Backward 227, 355
 - Clear Top 227, 355
 - Cursor 227, 355
 - Filelist 227, 355
 - Forward 227, 355
 - Help 227, 355
 - Left 227, 355
 - Pop Msg 227, 355
 - Quit 227, 355
 - Rdrlist 227, 355
 - Retrieve 227, 355

full-screen CMS (*continued*)

- PA and PF keys (*continued*)
 - Right 227, 355
- PA1 key 229
- physical screen characteristics
 - command line 222
 - data area 222
 - location information 222
 - PF key definition area 222
 - status area 222
 - title line 222
- positioning windows 293
- restoring windows 296
- setting the reserved line 299
- setting window borders 297
- sizing windows 294
- status information 223
- status notices
 - enter a command or press a PA or PF key 16
 - enter your response in vscreen vname 16
 - executing a command 16
 - scroll forward for more information in vscreen vname 16
 - system-defined windows and virtual screens 224
 - using 221

function types, CP command 343

G

GCS file type used in CMS 41

GENMSG command 44

GET subcommand of XEDIT

- creating small files from large one 167

GET VSCREEN command 350

GLOBALV command used with REXX language

- System Product Interpreter execs 282

GLOBALV file type usage in CMS 41

granting authority 109

GROUP file type use in CMS 42

H

halting

- program execution 34
- screen status 15
- System Product Interpreter execs 278
- terminal displays 34

HDR1 tape label 192

HELP

- command
 - BRIEF 147
 - DETAIL 148
 - RELATED 149
- format words
 - summary 322
- menus
 - component 153
 - task 152

HELP (*continued*)
 message 151
 notational conventions 317

HELP Facility
 components 145, 320
 file types 310
 keys, PF and PA2 155
 naming conventions 309
 tailoring 309
 using 145
 using System Product Editor 156

HELP file
 adding 330
 altering existing 330
 BRIEF section, adding 317
 changing menus 331
 converting to a formatted HELP file 321
 creating a new file 330
 creating and storing in SFS 309
 deleting 330
 DESCRIPT section, adding 317
 DETAIL section, adding 317
 ERRORS section, adding 317
 for messages 318
 FORMAT section, adding 317
 HELPMMSG file type 318
 how to name 309
 NOTES section, adding 317
 OPTIONS section, adding 317
 PARMs section, adding 317
 printing 155, 157
 RELATED section, adding 318

HELPABBR file type usage in CMS 42
 HELPAVS file type usage in CMS 42
 HELPCMS file type usage in CMS 42
 HELPCMSQ file type usage in CMS 42
 HELPCMSS file type usage in CMS 42
 HELPCONV command
 description 321
 how it treats comments 324
 summary of control words 322
 using to create additional HELP files 320

HELPCP file type usage in CMS 42
 HELPCPOT file type usage in CMS 42
 HELPCPQU file type usage in CMS 42
 HELPCPSE file type usage in CMS 42
 HELPEDIT file type usage in CMS 42
 HELPEXC2 file type usage in CMS 42
 HELPEXEC file type usage in CMS 42
 HELPGROU file type usage in CMS 42
 HELPHelp file type usage in CMS 42
 HELPIPCS file type usage in CMS 42
 HELPMACR file type usage in CMS 42
 HELPMENU file type usage in CMS 42
 HELPMENU files, creating 313
 HELPMMSG file type usage in CMS 42
 HELPPREF file type usage in CMS 42

HELPPVM file type usage in CMS 42
 HELPQUER file type usage in CMS 42
 HELPREXX file type usage in CMS 42
 HELPROUT file type usage in CMS 42
 HELPRSCS file type usage in CMS 42
 HELPSET file type usage in CMS 42
 HELPSQLD file type usage in CMS 42
 HELPSRPI file type usage in CMS 42
 HELPTASK file type usage in CMS 42
 HELPTASK files, creating 314
 HELPTSAF file type usage in CMS 42
 HELPXEDI file type usage in CMS 42

HI (Halt Interpretation) Immediate command 278
 HIDE WINDOW command 350
 highlighting words in a file using HELP 319
 HOLD operand of SPOOL command 172

holding
 display on terminal 15
 spool file to keep them from being processed 172

HOLDING screen status 15
 HT (Halt Type) Immediate command 34
 HX (Halt Execution) Immediate command
 effect in CMS subset 31

I
 IBM-supplied VMSYSU file pool 71
 ID card to submit job to CMS batch facility 249
 IEBTPCH utility program to create CMS files from
 tapes created by 197
 IEBUPDTE utility program to create CMS files from
 tapes created for 198
 IEHMOVE utility program to create CMS files from
 tapes created by 198

Immediate commands
 entering on display terminal 32
 using with System Product Interpreter
 programs 278

IMPCP operand of CMS SET command, setting 29
 IMPEX operand of CMS SET command, usage 270
 implicit locks 119, 123

implied
 CP function, SET IMPCP, usage 29
 exec function, SET IMPEX, usage 270

increasing virtual machine storage 166
 indenting text 326
 INDICATE command (CP) 346
 initial program load (IPL)
 entering CMS environment 8
 linking CMS installation saved segment 281

input mode
 on display terminal in line-mode 170
 when in full-screen XEDIT environment 162

INPUT subcommand
 used to enter data in XEDIT 162
 inserting lines in a file being edited 162

interrupting
 program execution 32

- interruptions
 - signaling on display terminal 32
- invoking
 - System Product Editor 161
 - System Product Interpreter Execs 269
- IPL command (CP) 346
- IPL (initial program load)
 - entering CMS environment 8
 - linking CMS installation saved segment 281

J

- jobname for job sent to CMS batch facility
 - specifying 252
 - used to identify spool files 257
- jobs for CMS batch facility, submitting 249

K

- keywords, translations of 67

L

- label off processing, tapes 184
- label processing, general description 181
- LABELDEF command
 - description 192
 - in tape processing 191
 - standard labels 182
 - use of 192
- labels
 - writing on CMS minidisks 128
- LABOFF (label off) processing 184
- LANGGCTL file type usage in CMS 43
- LANGMAP file type usage in CMS 43
- LANGMCTL file type usage in CMS 43
- language statements
 - in EXEC 2 language 266
 - in REXX language, for System Product Interpreter 265
- large files split into smaller files 167
- LASTING lock 121
- leaving
 - CMS subset environment 31
 - CMS/DOS environment 31
 - input mode 163
 - XEDIT environment 161
- length of CMS ready message, changing 17
- line-mode
 - migrating to full-screen CMS 357
 - using the editor 170
- LINK command
 - format, in job for CMS batch facility 256
 - linking to other user's minidisks 128
- LINK command (CP) 346
- linking
 - CMS installation saved segment 281
 - to other user's minidisks 128

linking (continued)

- to your own minidisks 129
- LISTCPDS file type usage in CMS 44
- LISTDIR command 78
- LISTFILE command 63
- LISTFILE command used to list your files 61
- listing
 - files using asterisk or percent signs 39
 - files with the same character string 39
 - information
 - about accessed file modes 19
 - about CMS files 61
 - about your virtual machine 208
 - requested 136
 - members of MACLIB
- LISTING file type
 - usage in CMS 44
- LISTING files
 - created by assembler and language processors 44
- LIST38PP file type usage in CMS 43
- LIST3800 file type usage in CMS 43
- LIST3820 file type usage in CMS 43
- LKEDIT file type usage in CMS 44
- loading
 - CMS into your virtual machine 8
 - execs into storage 280
- LOADLIB file type usage in CMS 44
- LOADVFCB command (CP) 346
- location information in full-screen CMS 222, 224
- locking files (SFS)
 - explicit locks 119
 - implicit locks 119
- locking options of XEDIT command 165
- locking the terminal keyboard 10
- locks
 - deleting, on files and directories 122
 - exclusive locks 120
 - lasting 121
 - session 121
 - share locks 120
 - update locks 120
- logging of messages 235
- logging of warnings 235
- logging off VM/SP 9
- logging on to VM/SP
 - at a 3270-type terminal 6
 - at other types of terminals 8
- logo screen, logging on from 6
- LOGOFF command 9
- LOGOFF command (CP) 346
- LOGON command to contact VM/SP 8
- LOGON command (CP) 346
- LRECL option of COPYFILE command 66
- LSEG file type usage in CMS 44
- LSEGMAP file type usage in CMS 44

M

MACLIB file type usage in CMS 44
MACRO file type usage in CMS 44
managing
 your file space 58
 your minidisks 59
map file type usage in CMS 45
master file directory 130
MAXIMIZE WINDOW command 294, 350
MEMO file type 48
 for documentation 48
menu
 changing 331
 component 153
 creating 312
 example, of creation 314
 task 152, 314
message class indicator in full-screen CMS 223
MESSAGE command (CP) 346
message HELP 151
MESSAGE LOGFILE 235
MESSAGE window
 dropping 231
 popping 231
 using to edit names file 232
 viewing messages 231
messages
 controlling whether you receive them 4
 displayed when logging on 8
 from CMS batch facility 254
 in full-screen CMS 231
 logging 235
 sending to other virtual machine users 207
 when display screen is full 15
migration to full-screen CMS 357
minidisks
 copying to minidisk from tape 176
 defined in VM/SP directory entry 127
 defining temporary minidisks for terminal session 127
 definition 127
 displaying status 23
 erasing files from 142
 extensions 49
 full, recovering from in editing session 168
 how much space is used 59
 linking 130
 managing 59
 master file directory 130
 providing for CMS batch virtual machine 254
 query status of 59
 querying accessed 19
 read-only extensions 49
 releasing 24
 search order 23, 49
 sharing 128
 sharing, commands used 139

minidisks (*continued*)
 writing a CMS file to 161
 writing CMS files to 52, 164
MINIMIZE WINDOW command 350
mode
 edit and input 21
 setting with CP TERMINAL command 32
 switching 27
modifying CMS files, commands to use 143
MODULE file type usage in CMS 45
MOREHELP command 156
MORE... status on display screen 15
MOVEFILE command
 copying CMS files from tapes created by 177
 copying tape files 197
 description 192
 reading files from virtual card reader 176
 use of 192
moving CMS files, commands to use 143
multi-volume tape processing
 See end-of-volume, processing
multiple file pools 125

N

NAMES file
 definition 201
 editing with MESSAGE window 232
 special characters to avoid 201
 storage in SFS directory or minidisk 202
NAMES file type usage in CMS 45
naming
 CMS files 37
 conventions for HELP files 309
 user commands 59
NETLOG file type use in CMS 45
nicknames, creating with a NAMES file 201
NL processing for tapes
 See no label processing
no label processing 181, 184
NOCLEAR option of XEDIT command used in EXEC procedure 170
nonstandard label processing, tapes 184
nonstandard label routine, writing 184
nonstandard labeled tapes, defining 188
NOPROF option of ACCESS command to suppress execution of PROFILE EXEC 273
NOT ACCEPTED status on display screen 15
notational conventions for HELP 317
NOTEBOOK file type use in CMS 45
NOTES option of DETAIL HELP 148
NOTREADY command (CP) 346
NSL (nonstandard label) processing 184
NUCXLOAD command 44, 45, 333
null line
 entering to determine environment 27
 input data from terminal 3
 to resume program execution after attention interruption 34

O

- offsetting text 328
- OPEN macros OS simulation 181
- OPTIONS option of DETAIL HELP 148
- ORDER command
 - selecting files for processing 174
- ORDER command (CP) 346
- OS (Operating System)
 - cleanup 267
 - simulation, end-of-tape processing 193
 - utility programs to create CMS files from tapes created by 197
- output
 - from CMS batch facility 257
 - from virtual console, spooling 24

P

- parent, of read-only extension 49
- PARMS option of DETAIL HELP 148
- passing
 - global variables between execs with GLOBALV 282
- password
 - for your virtual machine 6
 - supplying on LINK command line 129
 - suppression on command line 129
- PA1 key in full-screen CMS 229
- PA2 key in HELP 155
- PER command (CP) 346
- PF keys
 - See program function (PF) key
- PF keys on a DIRLIST display 77
- plus (+) and minus (-) file mode letter notation 75
- POP WINDOW command 231, 350
- POSITION WINDOW command 293, 350
- preparing jobs for CMS batch facility 254
- PRINT command to print CMS files 46
- printer files
 - produced by job running in batch virtual machine 255
 - querying status of 174
 - spooling 173
- printing
 - CMS files 46
 - HELP files 157
 - multiple copies 172
- privilege classes for CP commands 343
- processing, tapes
 - BLP 184
 - LABOFF 184
 - NL 184
 - NSL 184
- PROFILE EXEC
 - creating 273
 - description 273
 - sample, using REXX language 273
 - suppressing execution 273
- PROFILE EXEC (*continued*)
 - use to make EXECS storage-resident 274
- program function (PF) key
 - All 155
 - Backward 155
 - Brief 155
 - Clocate 155
 - Cursor 156
 - Forward 155
 - Help 155
 - Morehelp 155
 - PF Key 155
 - Print 155
 - Quit 155
 - Related 155
 - Return 155
 - toggling between keys 154
 - Top 155
 - ? 155
- program function (PF) keys
 - definition area in full-screen CMS 222
 - in full-screen CMS
 - Backward 227, 355
 - Clear Top 227, 355
 - Cursor 227, 355
 - Filelist 227, 355
 - Forward 227, 355
 - Help 227, 355
 - Left 227, 355
 - Pop Msg 227, 355
 - Quit 227, 355
 - Rdrlist 227, 355
 - Right 227, 355
 - in WM environment
 - Backward 230, 357
 - Clear 230, 357
 - Copy 230, 357
 - Forward 230, 357
 - Help 230, 357
 - Left 230, 357
 - Maximize 230, 357
 - Quit 230, 357
 - Restore 230, 357
 - Retrieve 230, 357
 - Right 230, 357
 - Top 230, 357
 - setting 12, 229
 - COPY function 25
 - in PROFILE EXEC 273
 - to retrieve previous line entered 11
 - using 12
 - using in FILELIST 62
 - using to send notes 211
 - using when composing a note 210
 - using when receiving files when in RDRLIST 215
- programs
 - written in REXX language for System Product Interpreter 265

- protected application environment 30
- protecting files from being accessed 55
- PSEG file type usage in CMS 45
- PSEGMAP file type usage in CMS 45
- PUNCH command
 - example 176
- punch files produced by job running in batch virtual machine 257
- punching
 - CMS files 176
 - jobs to batch virtual machine 251
- PURGE command to delete spool files 174
- PURGE command (CP) 346
- purging batch jobs 257
- PUT SCREEN command 350
- PUT VSCREEN command 350

Q

- QUERY
 - command (CMS)
 - display search order 49
 - how much space is on a minidisk 59
 - proportion of file space used 58
 - query CMSPF keys 227
 - command (CP)
 - display color and extended highlight values 17
 - query status of CP SET MSG function 4
- QUERY ACCESSED command 71
- QUERY ALIAS command 102
- QUERY CPLANG command 5
- QUERY FILEPOOL CONFLICT command 123
- QUERY LANGLIST command 5
- QUERY LANGUAGE command 5
- QUERY LIMITS command 58
- QUERY LOCK command 122
- QUIT subcommand to terminate an edit session 161

R

- read authority
 - on a directory 107
 - on a file 107
- read-only extensions, using 50
- READCARD command
 - restriction in CMS batch facility 256
 - used to assign file mode numbers 52
- READER operand
 - of ASSGN command, restriction in job for CMS batch facility 256
 - of FILEDEF command, restriction in job for CMS batch facility 256
- reader, holding user files in 172
- reading
 - cards from your virtual card reader 175
 - real card decks into your virtual machine 175
- READY command (CP) 347

- ready message
 - controlling how it is displayed 17
 - not displayed after #CP function is used in CMS 29
- read/write
 - displaying status of file modes 23
- read, to virtual console, definition 32
- record length of CMS file 65
- REFRESH command 350
- RELATED HELP 149
- RELEASE command
 - updating master file directory 130
- releasing
 - file modes 24
 - read-only extensions 50
- relocating files and directories 84
- Remote Spooling Communications Subsystem (RSCS)
 - networking 137
- remote terminals, using an editor 170
- RENAME command
 - changing file mode numbers 53, 57, 65
 - renaming CMS files 65
 - renaming files in another user's directory 83
- renaming, CMS files 65
 - in another user's directory 83
 - using aliases 116
- reordering batch jobs 257
- REQUEST command (CP) 347
- RESET command (CP) 347
- responses
 - from CMS commands 17
 - from VM/SP 8
- restarting batch jobs 257
- RESTORE WINDOW command 296, 350
- restrictions
 - on commands used in CMS batch facility 255
- resume
 - after an attention interruption 33
 - program execution
 - after regaining control following a disconnect 10
 - terminal displays 34
- RETRIEVE function for display terminals 11
- retrieving previously entered data 11
- RETURN CMS subset command to leave subset 31
- return code in CMS ready message 18
- revoking authority 110
- REWIND command (CP) 347
- ROUTE command 350
- RT (Resume Type) Immediate command 34
- RUNNING status on display screen 15

S

- SAVE subcommand
 - changing file identifier 165
 - writing a CMS file 161
- saved segment 274, 280
- saving files 164

screen

- example of 3270 screen display 26
- full-screen status notices
 - enter a command or press a PA or PF key 16
 - enter your response in vscreen vname 16
 - executing a command 16
 - scroll forward for more information in vscreen vname 16
- VM status notices
 - CP READ 14
 - HOLDING 15
 - MORE... 15
 - NOT ACCEPTED 15
 - RUNNING 15
 - VM READ 14
- SCREEN command (CP) 347
- SCREEN/NOSCREEN option of DETAIL HELP 150
- SCRIPT
 - file type usage in CMS 48
 - files 48
- SCROLL command 350
- scrolling forward and backward using Border commands 238
- scrolling right and left using Border commands 241
- SDATE synonym to sort FILELIST 63
- SDIR synonym to sort FILELIST 63
- search order
 - for CMS commands
 - displaying 59
 - summary 59
 - for file mode extensions 49
 - for file modes 49
- searching
 - file modes for CMS files
 - See file mode determination
 - read-only extensions 49
- SEND command (CP) 347
- sending
 - files to other virtual machine users
 - from SENDFILE menu 212
 - using DISK DUMP command 176
 - using SENDFILE command 211
 - messages to other virtual machine users
 - using CP MESSAGE command 208
 - using TELL command 207
 - notes to other virtual machine users
 - using SENDFILE command 211
- SESSION lock 121
- SET ABBREV command 67
- SET BORDER command 297
- SET command
 - CONCEAL option 30
- SET command (CMS)
 - controlling ready message display 17
 - controlling whether you receive messages 4
 - invalid forms in job for CMS batch facility 256
 - operands invalid in job for CMS batch facility 256
 - set tracing on or off for System Product Interpreter execs 277
- SET command (CMS) (*continued*)
 - setting full-screen CMS 221
 - setting implied CP function 29
 - setting implied exec function 270
 - setting program function keys 12
 - setting the reserved line in full-screen CMS 299
 - setting window borders in full-screen CMS 297
- SET command (CP) 347
- SET FILEPOOL command 72
- SET FILEWAIT command 122
- SET INSTSEG
 - description 277
 - use of 274, 281
- SET LANGUAGE command 5
- SET LOGFILE command 209, 235
- SET RESERVED command 299
- SET TRANSLATE command 67
- setting
 - defaults for SENDFILE command, example 211
 - length of ready message 17
 - limits on system resources during batch jobs 253
 - program function keys 12, 229
 - screen colors and highlighting features 17
- SFS (Shared File System)
 - definition 69
 - directory 69
 - file organization 72
 - file pool 69
 - file space 69
 - top directory 71
- shared disk, creating 117
- Shared File System
 - See SFS (Shared File System)
- Shared File System (SFS) administrator 108
- shared files
 - editing 119
- sharing
 - files and directories, commands used 140
 - files, how-to 94
 - minidisks 128
 - minidisks, commands used 139
- SHOW WINDOW command 351
- signing onto VM/SP 6
- simulated data sets
 - file mode number of 4 55, 56
- SIZE WINDOW command 294, 351
- SLEEP command (CP) 347
- SLREC synonym to sort FILELIST 63
- SMODE synonym to sort FILELIST 63
- SMSG command (CP) 208, 347
- SNAME synonym to sort FILELIST 63
- SORT command to specify file mode numbers 39
- sorting
 - CMS files 19
 - files in FILELIST 63
 - spacing between lines of text 329
 - special characters
 - in file names and file types 37

special characters (*continued*)
 3270 Text feature 35

special messages, controlling whether you receive them 208

specifying file mode numbers on DLBL and FILEDEF commands 57

splitting CMS files into smaller files 167

SPOOL command
 changing characteristics of unit record devices 171
 spooling console output 24
 used to combine multiple spool files 173

SPOOL command (CP) 347

spool files
 controlling in job for CMS batch facility 257
 determining status of 171
 produced by CMS batch facility, controlling 257

spooling
 basic description 171
 console output 24
 multiple copies 172

SRECF synonym to sort FILELIST 63

SSIZE synonym to sort FILELIST 63

stacking
 null lines
 after attention interruption 34
 at your terminal 3

standard label processing, CMS/DOS 189

standard labels, OS simulation 181

START command

status area in full-screen CMS 222

status information in full-screen CMS 223

status notices in full-screen CMS
 enter a command or press a PA or PF key 16
 enter your response in vscreen vname 16
 executing a command 16
 scroll forward for more information in vscreen vname 16

storage group
 definition 22
 sharing 170
 what to do when full 169

STORE command (CP) 347

storing files
 in SFS directories
 allocation of file space 22
 file space 19
 methods 4
 on both SFS directories and minidisks 19
 access with file mode A 72
 on minidisks 19
 allocation of DASD space 22

STYPE synonym to sort FILELIST 63

subdirectories 69

submitting jobs to CMS batch facility 249
 non-CMS users 262

subsetting options in DETAIL HELP
 ALL 148
 DESCRIPT 148

subsetting options in DETAIL HELP (*continued*)
 ERRORS 148
 FORMAT 148
 NOTES 148
 OPTIONS 148
 PARMS 148

summary
 of CMS commands 333
 of CP command privilege classes 343
 of CP commands 345
 of HELP and HELPCONV control words 322
 of Immediate commands 333

suppressing
 messages sent by the TELL command 208
 special messages 208

suppression of passwords on the command line 129

SYNONYM
 command to invoke synonym tables 66
 file type usage in CMS 45
 used to define synonyms for CMS and user-written commands 66

synonyms used to sort FILELIST
 SDATE 63
 SLREC 63
 SMODE 63
 SNAME 63
 SRECF 63
 SSIZE 63
 STYPE 63

SYSPROF EXEC 13

SYSTEM command (CP) 347

system disk, files available 55

System Product Editor
 considerations for migrating to full-screen CMS 360
 example, of using 162
 full-screen display 162
 invoking 162
 line-mode on display terminals 170
 using to display HELP files 156

System Product Interpreter
 basic description 265
 REXX language, interpreted by 265
 running 269
 sample execs 270
 writing execs for the 269

system profile 13, 134

SYSUTx file type 47
 SYSUT1 file type 46
 SYSUT2 file type 46
 SYSUT3 file type 46
 SYSUT4 file type 46
 SYS00x file type 47
 SYS001 file type 46
 SYS002 file type 46
 SYS003 file type 46
 SYS004 file type 46
 SYS005 file type 46

SYS006 file type 46

T

TAG command (CP) 347

TAPE command

creating CMS files from tapes created by 177

sample terminal display 178

using 178

tape file

DCB address 185

FCBSECT address 185

tape files, in CMS 177

tape handling options, specifying 198

tape label

by CMS commands 191

EOT 193

EOV 193

LABELDEF command 192

MOVEFILE command 197

under CMS/DOS 188

DTFMT macro 188

under OS simulation 181

under OS/VS simulation 186

tape labels

in CMS 180

limitations 181

tape label, processing, IBM standard 182

TAPECTL macro used in tape label process 191

TAPEMAC command 191

tapes

bypass label, description 184

compacting output 178

density, when to specify 198

label processing 181

labels

in CMS 180

in CMS/DOS 188

in OS simulation 181

nonlabeled, description 184

nonstandard label, description 184

optional handling 198

special handling 198

virtual addresses 177

TAPESL macro, description 191

TAPPDS command 191

copying files from tapes 197

creating CMS files from tapes created by 177

TASK menu 152

TASK menus 314

TCLOSE command in tape label processing 188

TE (Trace End) Immediate command 278

TEMPLATE file type usage in CMS 45

TERMINAL command (CP) 347

terminal output, controlling 138

terminals

characteristics 13

disconnecting 10

terminals (*continued*)

display

See display terminals

mode, setting 32

requesting information about 136

terminating

editing session 21

execution of a program 34

terminal session 9

TEXT

file type

usage in CMS 45

files

created by assembler language processors 46

text feature for 3270 terminals 35

title line in full-screen CMS 222

TO operand of SPOOL command 173

toggling between pf keys 154

top directory

abbreviated form 72

definition 71

name 71

referring to 73

TRACE command (CP) 348

TRANSFER command

moving reader files 174

TRANSFER command (CP) 348

transferring

data files 174

transient area

CMS commands that execute in 60

translating output characters 329

translation synonyms for commands 67

translations for commands 67

TRUNC option of COPYFILE command 66

truncating trailing blanks 66

TS (Trace Start) Immediate command 278

TXTLIB file type usage in CMS 45

type call in tape label processing 186

TYPE command 21

TYPEWRITER subcommand used to edit in

line-mode 170

TYPE/NOTYPE option of DETAIL HELP 150

U

underscore

highlighting feature, controlling with CP SCREEN

command 17

in file name and file type 38

unit record, devices 171

unlabeled tapes, defining 188

UPDATE file type usage in CMS 45

updating

master file directories 130

UPDLOG file type usage in CMS 46

UPDTxxxx file type usage in CMS 46

- user file directory 130
- user hold status of spool files 172
- user ID
 - for CMS batch virtual machine 249
 - for your virtual machine 6
 - specifying for output spool files 173
- user program area
 - commands that execute in 60
- user-written
 - execs, samples 270
- user-written commands 59
- using XEDIT subcommand in HELP 156

V

- vector
 - cleanup 267
 - resetting using EXECOS 282
- virtual addresses
 - for tapes 177
 - for unit record devices 171
- virtual machines
 - definition 3
- Virtual Machine/System Product
 - See* VM/SP (Virtual Machine/System Product)
- Virtual Machine/System Product (VM/SP)
 - basic description 3
 - command summaries 333
 - components of 3
 - environments 27
- virtual screen
 - default characteristics 305, 352
 - defining 290
 - general description 219
 - system-defined 224
- virtual storage
 - requesting information about 136
 - used by editor, what to do when it is full 166
- VM READ status on display screen 14
- VMDUMP command (CP) 348
- VMSYSU file pool 71
- VM/SP System Product Editor
 - See* System Product Editor
- VM/SP System Product Interpreter
 - See* System Product Interpreter
- VM/SP (Virtual Machine/System Product)
 - basic description 3
 - command summaries 333
 - components of 3
 - environments 27
- VOLID parameter FILEDEF command 182
- VSAM cleanup 267
- VSE
 - differences between CMS/DOS tape label processing 189
 - TLBL card in tape label processing 189

W

- WAITREAD VSCREEN command 277, 286, 351
- WAITT VSCREEN command 351
- WARNING LOGFILE 235
- window
 - characteristics of 220
 - default characteristics 303
 - defining 290
 - dropping 231
 - general description 219
 - maximizing 294
 - popping 231
 - positioning 293
 - restoring 296
 - setting borders 297
 - sizing 294
 - system-defined 224
- windowing and full-screen CMS considerations 349
- WM window
 - customizing 289
 - messages 225
 - PF keys
 - Backward 230, 357
 - Clear 230, 357
 - Copy 230, 357
 - Forward 230, 357
 - Help 230, 357
 - Left 230, 357
 - Maximize 230, 357
 - Quit 230, 357
 - Restore 230, 357
 - Retrieve 230, 357
 - Right 230, 357
 - Top 230, 357
 - using 225, 242
- write authority
 - on a directory 108
 - on a file 108
- WRITE VSCREEN command 290, 351
- writing
 - applications in full-screen CMS 361
 - CMS files onto file mode, file mode determination 52
 - labels on CMS minidisks 128
- writing files to a directory or minidisk 164
- WVOL1 function in tape command processing 192

X

- XEDIT
 - changing file mode number with 53
 - CLOCATE subcommand 156
 - command to invoke System Product Editor 162
 - example 162
 - file type usage in CMS 46
 - LOCK option (default) 165
 - NOLOCK option 165

XEDIT (*continued*)
subcommands, invoking 4

Z

ZAP file type usage in CMS 46

Numerics

19E minidisk address accessed with file mode Y 48
190 minidisk address accessed with file mode S 48
192 minidisk address accessed as file mode D 48
3270 screen display, example 26
3270 terminals
See display terminals

Special Characters

.BX (BOX) format word 322
.CM (Comment) control word 324
.CS (Conditional Section) control word 325
.FO (Format Mode) control word 325
.IL (Indent Line) control word 326
.IN (indent) control word 326
.MT (Menu Type) control word 327
.OF (Offset) control word 328
.SP (Space Lines) control word 329
.TR (Translate Character) control word 329
&PUNCH control statement
punching jobs to CMS batch facility 258
&TRACE statement in EXEC 2 267
\$HLPABBR file type usage in CMS 42
\$HLPAVS file type usage in CMS 42
\$HLPCCMS file type usage in CMS 42
\$HLPCCMSQ file type usage in CMS 42
\$HLPCCMSS file type usage in CMS 42
\$HLPCCP file type usage in CMS 42
\$HLPCCPOT file type usage in CMS 42
\$HLPCCPQU file type usage in CMS 42
\$HLPCCPSE file type usage in CMS 42
\$HLPDDEBU file type usage in CMS 42
\$HLPDEDIT file type usage in CMS 42
\$HLPDEXEC2 file type usage in CMS 42
\$HLPDEXEC file type usage in CMS 42
\$HLPDGROU file type usage in CMS 42
\$HLPDHELP file type usage in CMS 42
\$HLPDIPCS file type usage in CMS 42
\$HLPDMACR file type usage in CMS 42
\$HLPDMENU file type usage in CMS 42
\$HLPDMSG file type usage in CMS 42
\$HLPDPREF file type usage in CMS 42
\$HLPDPVM file type usage in CMS 42
\$HLPDQUER file type usage in CMS 42
\$HLPDREXX file type usage in CMS 42
\$HLPDROUT file type usage in CMS 42
\$HLPDRSCS file type usage in CMS 42
\$HLPDSET file type usage in CMS 42

\$HLPDSQLD file type usage in CMS 42
\$HLPDSRPI file type usage in CMS 42
\$HLPDTASK file type usage in CMS 42
\$HLPDTSAF file type usage in CMS 42
\$HLPDXEDI file type usage in CMS 42
* command (CP) 345
* (asterisk)
as file ID on command line 39
in file mode field 51
in FILELIST command 63
in LISTFILE command 63
*/ (end of a REXX comment statement) 269
/JOB control card description 252
/SET control card description 253
/* (slash asterisk)
CMS batch facility control card used to signal end of
job 252
end-of-file indicator in batch job 262
in REXX language interpreted as comment 269
%, used as file ID on command line 39
(logical line end character)
using when setting PF (program function) keys 12
#CP command (CP) 345
#CP function
used when setting PFnn RETRIEVE 11
using on display terminals 10
@ (logical character delete character)
using when setting PF (program function) keys 13
= (equal sign)
entered in file IDs on command lines 39
" (logical escape character)



Program Number
5664-167

File Number
S370/4300-39

VM/SP
CMS User's Guide
Order No. SC19-6210-05

**READER'S
COMMENT
FORM**

Is there anything you especially like or dislike about this book? Feel free to comment on specific errors or omissions, accuracy, organization, or completeness of this book.

If you use this form to comment on the online HELP facility, please copy the top line of the HELP screen.

_____ **Help Information** line ____ of ____

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you, and all such information will be considered nonconfidential.

Note: Do not use this form to report system problems or to request copies of publications. Instead, contact your IBM representative or the IBM branch office serving you.

Would you like a reply? __YES __NO

Please print your name, company name, and address:

IBM Branch Office serving you:

Thank you for your cooperation. You can either mail this form directly to us or give this form to an IBM representative who will forward it to us.

Reader's Comment Form

CUT
OR
FOLD
ALONG
LINE

Fold and tape

Please Do Not Staple

Fold and tape



BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



INTERNATIONAL BUSINESS MACHINES CORPORATION
DEPARTMENT G60
PO BOX 6
ENDICOTT NY 13760-9987



Fold and tape

Please Do Not Staple

Fold and tape





Program Number
5664-167

File Number
S370/4300-39

2

SC19-6210-05

