

z/VM
7.3

CMS User's Guide



Note:

Before you use this information and the product it supports, read the information in [“Notices” on page 303](#).

This edition applies to version 7, release 3 of IBM® z/VM® (product number 5741-A09) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2025-06-02

© **Copyright International Business Machines Corporation 1990, 2025.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures.....	ix
Tables.....	xv
About This Document.....	xvii
Intended Audience.....	xvii
Where to Find More Information.....	xvii
Links to Other Documents and Websites.....	xvii
How to provide feedback to IBM.....	xix
Summary of Changes for z/VM: CMS User's Guide.....	xxi
SC24-6266-73, z/VM 7.3 (June 2025).....	xxi
SC24-6266-73, z/VM 7.3 (September 2023).....	xxi
SC24-6266-73, z/VM 7.3 (September 2022).....	xxi
Part 1. Introduction to CMS.....	1
Chapter 1. Introduction to CMS and the z/VM Environment.....	3
Using CP and CMS Commands.....	3
CP Command Language.....	3
CMS Command Language.....	4
z/VM Environments and Mode Switching.....	5
CP Environment.....	7
CMS Environment.....	8
The XEDIT Environment, Input Mode, and CMS Subset.....	9
CMS/DOS.....	10
Protected Application Environment.....	11
Entering Commands.....	11
How z/VM Responds to Your Commands.....	13
Some Sample CP and CMS Command Responses.....	14
Storing Your Files.....	14
Console Output.....	15
Spooling Console Output.....	15
Copying Your Screen.....	16
Interrupting Program Execution.....	17
Using the Attention Key.....	17
Interrupting Your Programs.....	17
Halting Screen Displays.....	18
Control Program Interruptions.....	19
The CP TRACE Command.....	19
Using the 3270 Text Feature.....	20
Error Situations.....	20
Syntax, Message, and Response Conventions.....	20
Part 2. Working with CMS Files.....	25
Chapter 2. CMS File System.....	27
CMS File Formats.....	27

How CMS Files Get Their Names.....	28
Duplicate File Names or File Types.....	28
Working with CMS Files.....	29
Creating a New File.....	29
Editing a File.....	29
Displaying a List of Your CMS Files.....	29
Erasing a File.....	33
Copying Files.....	33
Using Asterisks (*) and Percent Signs (%) in File IDs.....	35
Equal Signs in Output File IDs.....	36
Chapter 3. Using the Shared File System.....	37
What is the Shared File System?.....	37
Getting Started - Accessing Your Top Directory.....	39
Releasing Directories.....	41
Managing Your File Space.....	41
Organizing Your Files.....	42
Working with Directories.....	43
Using the Abbreviated Form of Your Top Directory.....	43
Accessing Another User's Directory.....	44
Specifying a Directory Identifier.....	45
Listing the Structure of a Directory with DIRLIST.....	45
Using the LISTDIR Command.....	48
Creating a Directory.....	49
Putting Files into a Directory.....	52
Copying Files to a Directory.....	52
Creating New Files.....	53
Renaming Your Files and Directories.....	54
Relocating Your Files and Directories.....	54
Erasing a Directory.....	58
Navigating Through Your Directories.....	58
Sharing Files.....	64
DFSMS/VM and SFS File Management.....	64
Creating Aliases to Files.....	65
Using the QUERY ALIAS Command.....	71
Using the ALIALIST Command.....	72
Erasing Your Base Files.....	74
Authorizing Others to Access Your Files and Directories.....	75
FILECONTROL Directory Authority.....	76
DIRCONTROL Directory Authority.....	77
Granting Authority.....	78
REVOKE AUTHORITY Command.....	79
Determining Who Has Authority for a File or Directory.....	80
Using the AUTHLIST Command.....	81
Determining Ownership of a File or Directory.....	83
Using Aliases to Share Files.....	83
Creating a Bulletin Board or Shared Disk.....	84
Locking Files and Directories.....	86
Determining If a File or Directory is Locked.....	89
Using Directory Level Control.....	90
Application Considerations.....	99
Using Several File Pools at One Time.....	99
Chapter 4. Storing Your Files on Minidisks.....	101
Minidisks and How They Are Defined.....	101
Defining Temporary Minidisks.....	101
Defining Virtual Disks in Storage.....	102
Formatting Minidisks.....	102

Linking and Sharing Minidisks.....	103
Accessing Minidisks.....	104
Releasing and Detaching Minidisks.....	105
Minidisk File Directories.....	105
Managing Your Minidisks.....	106
Data Compression.....	107
Chapter 5. More on the CMS File System.....	109
What Are Reserved File Types?.....	109
File Types for CMS Commands.....	109
File Types for Output Files: TEXT and LISTING For Example.....	117
File Types for Temporary Work Files.....	117
File Types for Documentation.....	117
File Mode Letters and Numbers.....	117
How File Mode Letters Are Used.....	118
When to Specify File Mode Letters: Reading Files.....	120
When to Specify File Mode Letters: Writing Files.....	122
How File Mode Numbers Are Used.....	122
File Mode Numbers in SFS.....	123
File Mode Numbers for Minidisks.....	124
Commands Used to Assign File Mode Numbers.....	126
Accessing Your Directories or Minidisks.....	126
Linking and Accessing with VMLINK.....	127
NAMES Files for VMLINK.....	128
VMLINK Control File.....	130
VMLINK Control File Example.....	135
VMLINK Programming Interface.....	136
EXIT, PREEXIT and INVOKE Examples.....	140
Testing with the VMLNICXT EXEC.....	145
NAMES File Exit Examples.....	146
VMLINK Control File Exit Examples.....	146
VMLINK Command Examples.....	147
VMLINK Linking Behaviors.....	152
Using Synonyms.....	159
Using Translations.....	160
CMS Command Search Order.....	161
CMS Command Execution Characteristics.....	162
Changing the Record Format of a File.....	162
Chapter 6. Using Real Printers, Punches, and Readers.....	163
CMS Unit Record Device Support.....	163
Using the CP Spooling System.....	163
Some Options Available on the CP SPOOL Command.....	163
Altering Spool Files.....	165
Using the RECEIVE Command to Receive a File.....	166
Sending Files or Notes.....	167
Using Your Card Punch and Card Reader in CMS.....	167
Chapter 7. Using Tapes.....	169
Using the TAPE Command.....	170
Examples of the TAPE Command.....	170
Using the VMFPLC2 Command.....	173
Using the VMFPLC2 Command.....	174
Using the TAPPDS Command.....	174
Using the TAPEMAC Command.....	174
Using the MOVEFILE Command.....	174
User Programs.....	175
CMS Native Tape Macros.....	175

OS Simulation.....	175
VSE Simulation.....	176
VSE/VSAM (AMSERV Command).....	176
Using the DDR Command (DASD Dump Restore).....	176
Tape Recording Formats.....	176
Compacted and Noncompacted Recording Format.....	178
Device Recording Format Capabilities.....	179
Selecting the Recording Format with CMS.....	180
Tape Marks on TAPE DUMP tapes.....	181
Using Tape Library Dataservers under OS Simulation.....	182
OS Utility Programs.....	183
IEBPTPCH.....	183
IEBUPDTE.....	184
IEHMOVE.....	184
Part 3. z/VM HELP Facility.....	185
Chapter 8. Using the HELP facility.....	187
Getting HELP on Messages.....	188
Getting HELP on Commands.....	188
HELP Menus.....	192
Display and Search Options for HELP files.....	193
Using the PA2 and PF Keys.....	194
Using the MOREHELP Command.....	196
Displaying HELP Files Using XEDIT.....	196
Printing HELP Screens.....	197
Working with Your HELP Files.....	197
Chapter 9. Tailoring the HELP Facility.....	199
HELP Components Definition and Purpose.....	199
z/VM HELP Components.....	202
Restrictions for all HELP files.....	207
Creating HELP Files.....	207
Creating Menus for HELP Files.....	207
Creating Command HELP Files.....	215
Creating HELP Files for Messages.....	220
Highlighting Words within a File.....	220
Using Command Abbreviations.....	221
Adding Your Own HELP Components.....	224
Using HELPCONV to Create HELP Files.....	225
Changing Existing HELP Files.....	232
Adding HELP Files.....	232
Deleting HELP Files.....	233
Altering Existing HELP Files.....	233
Changing Menus.....	233
Part 4. Using Full-Screen CMS.....	235
Chapter 10. Introducing Full-Screen CMS.....	237
What Are Windows and Virtual Screens?.....	237
What Can You Do with a Window?.....	238
Using Full-Screen CMS.....	238
Status Area Information.....	240
Location Information.....	241
Your Default Windows and Virtual Screens.....	241
Special Keys.....	243
Messages in Full-Screen CMS.....	247

Reentering Commands.....	249
Working with Border Commands.....	250
Using the WM Window.....	254
Chapter 11. Customizing Full-Screen CMS.....	259
Tailoring System Defaults.....	259
WINDOW POSITION.....	262
WINDOW SIZE.....	263
WINDOW MAXIMIZE and WINDOW RESTORE.....	263
Using the SET Command.....	265
SET BORDER.....	265
SET RESERVED.....	266
SET WINDOW.....	267
Window and Virtual Screen Tables.....	269
Considerations When Disconnecting and Reconnecting.....	272
Message Routing.....	272
Migration Considerations.....	273
CMS Considerations.....	273
CP Considerations.....	275
XEDIT Considerations.....	275
Considerations for Writing Applications.....	276
Part 5. Using Execs and Programs in CMS.....	277
Chapter 12. Introduction to the Exec Processors.....	279
REXX/VM Interpreter.....	279
EXEC 2 Processor.....	280
CMS EXEC Processor.....	280
Relationship of the Exec Interpreters.....	280
Running Execs.....	281
Attributes of Exec Files.....	281
Chapter 13. Creating REXX Execs.....	283
Running Your Exec Files.....	283
Sample REXX Execs.....	284
Chapter 14. Creating a PROFILE EXEC.....	287
Chapter 15. Commands Used with REXX Execs.....	289
Using EXECIO.....	290
Using EXECDROP, EXECLOAD, EXECMAP, and EXECSTAT.....	292
Using IPL, SET INSTSEG, EXECMAP, and EXECDROP.....	293
Using EXECOS.....	293
Using GLOBALV.....	293
Using IDENTIFY.....	294
Using IMMCMD.....	294
Using LISTFILE.....	295
Using NAMEFIND.....	295
Using PIPE.....	295
Using QUERY and RDR.....	296
Using SET EXECTRACE.....	296
Using Windows and Virtual Screens.....	296
Using XEDIT.....	297
Writing XEDIT macros.....	297
Exchanging Data Between Programs Through the Stack.....	297
Chapter 16. Developing Programs in CMS.....	299

Creating a Program.....	299
Compiling a Program.....	300
Running a Program.....	301
Notices.....	303
Trademarks.....	304
Terms and Conditions for Product Documentation.....	304
IBM Online Privacy Statement.....	305
Bibliography.....	307
Where to Get z/VM Information.....	307
z/VM Base Library.....	307
z/VM Facilities and Features.....	308
Prerequisite Products.....	310
Related Products.....	310
Index.....	311

Figures

1. Paths From One Environment to Another.....	6
2. 3270 Screen Display.....	16
3. Sample FILELIST Screen.....	31
4. Sample FILELIST STATS Screen.....	31
5. Sample Hierarchical Directory.....	38
6. Another Sample Directory.....	43
7. Entering the DIRLIST Command.....	46
8. The .PARTY Directory.....	50
9. The .PARTY.TREATS Directory.....	51
10. Using DIRLIST to List All Directories.....	52
11. Files Within the .PARTY Directory.....	53
12. Moving a File to the .PARTY.TREATS Directory.....	55
13. Creating the .PARTY.FAVORS Directory.....	56
14. Relocating the .PARTY.FAVORS Directory.....	57
15. Listing All Your Directories.....	58
16. Copying More Files to .PARTY and PARTY.TREATS.....	59
17. Using PF11 from DIRLIST.....	60
18. Sample FILELIST Screen with ALLDATES Option.....	60
19. Using PF11 from FILELIST.....	61
20. The FILELIST SHARE Screen.....	62
21. Example of Alias Pointers Between SFS Files.....	66
22. The FILELIST SEARCH Screen.....	68
23. Using FILELIST SEARCH to List All Your Files.....	69

24. Listing All the Files in a Directory.....	70
25. Using PF11 to XEDIT a File.....	71
26. Using PF10 for Information on Aliases.....	71
27. Using the PF Keys on the FILELIST SHARE Screen.....	73
28. Entering the ALIALIST Command.....	73
29. Erased Indicator on the FILELIST SHARE Screen.....	74
30. Using the PF Keys on the FILELIST SHARE Screen.....	82
31. Entering the AUTHLIST Command.....	82
32. Determining the Owner of a File.....	83
33. Using Directory Control Directories.....	94
34. Using Directory Control Directories.....	95
35. Using Directory Control Directories.....	96
36. VMLINK Control File Example.....	135
37. VMLINK and the REXX Program Stack Example.....	138
38. VMLINK Variables and REXX Stem Variables Example.....	138
39. Specifying VMLINK Variables as Parameters Example.....	140
40. VMLINK PRODUCT A NAMES File Entry.....	141
41. VMLINK PRODUCT A Minidisk 191 NAMES File Entry.....	141
42. VMLINK PRODUCT B NAMES File Entry.....	142
43. VMLINK PRODUCT B Minidisk 191 NAMES File Entry.....	142
44. VMLINK PRODUCT B Minidisk 193 NAMES File Entry.....	143
45. VMLINK PRODUCT B Minidisk 195 NAMES File Entry.....	143
46. VMLINK PRODUCT C NAMES File Entry.....	144
47. VMLINK PRODUCT C Minidisk 191 NAMES File Entry.....	144
48. VMLINK PRODUCT C Minidisk 193 NAMES File Entry.....	145

49. PREXITB EXEC Example.....	146
50. EXITB EXEC Example.....	146
51. INVOKEB EXEC Example.....	146
52. PEXITCTL EXEC Example.....	147
53. EXITCTL EXEC Example.....	147
54. VMLINK PRODB with NAMES, but Without VMLINK CONTROL Activated.....	148
55. VMLINK PRODB with NAMES and VMLINK CONTROL Activated (Part 1 of 2).....	149
56. VMLINK PRODB with NAMES and VMLINK CONTROL Activated (Part 2 of 2).....	150
57. VMLINK PRODA with Invoke EXEC.....	151
58. SAYHI EXEC Example.....	152
59. Basic Tape Layout for Tape 500.....	173
60. Sample of DETAIL HELP for the SENDFILE Command.....	190
61. Sample of RELATED HELP for the ERASE Command.....	191
62. Sample HELP TASK Menu.....	192
63. Sample Component MENU for CMS.....	193
64. Example of a HELPMENU file.....	210
65. A HELPMENU file for PerfKit BASIC mode subcommands.....	211
66. Sample of HELPMENU File Creation.....	211
67. Sample HELPMENU File Displayed by the HELP Facility.....	212
68. Sample of HELPTASK File Creation.....	214
69. Sample HELPTASK File Displayed by the HELP Facility.....	214
70. Sample HELP file structure.....	218
71. Format of the RELATED Section of the CMS ERASE Command.....	219
72. A Window into a Virtual Screen.....	238
73. Full-Screen CMS.....	239

74. Displaying the CMSPF Key Settings.....	244
75. Setting CMSPF 9 to TELL.....	246
76. Adding an Entry to the Names File.....	248
77. MESSAGE Window in the Names File.....	249
78. Popping the MESSAGE Window.....	250
79. Looking at the Corners of a Window Border.....	251
80. Using a Border Command to Scroll Forward.....	252
81. Result of Scrolling Forward.....	252
82. Scrolling Backward through a Window Border.....	253
83. Scrolling to the Right through a Window Border.....	253
84. Scrolling to the Left through a Window Border.....	254
85. Popping the WM Window.....	255
86. Dropping the MESSAGE Window.....	255
87. Displaying the MESSAGE Window.....	256
88. Changing the MESSAGE Window.....	256
89. WM Window.....	257
90. Restoring the MESSAGE Window.....	257
91. Your Newly-Defined MESSAGE Window.....	261
92. Using the POPDROP EXEC.....	262
93. Using the WINDOW POSITION Command.....	263
94. Popping the MESSAGE Window.....	264
95. Maximizing a Window.....	264
96. The Window after WINDOW RESTORE.....	265
97. Changing Only the Top Border.....	266
98. Changing the Bottom and Left Window Borders.....	266

99. Deleting a Window Title.....	267
100. Deleting a Blank Reserved Line.....	267
101. Message Class Indicator.....	268
102. MESSAGE Window.....	269
103. Sample REXX Exec - Copy a File.....	284
104. Sample REXX Exec - Send a File.....	285
105. Sample PROFILE EXEC.....	287
106. The RDPRIND EXEC, an Example of the EXECIO Command Used in a REXX Exec.....	291

Tables

1. Commands Used to Move from One Environment to Another.....	6
2. Determining your Environment from the System Response.....	7
3. Using CP Commands in a CMS Environment.....	8
4. Examples of Syntax Diagram Conventions.....	21
5. Examples of Using * and % in CMS commands.....	36
6. DIRLIST PF Keys.....	47
7. FILELIST STATS PF Keys.....	61
8. FILELIST SHARE PF Keys.....	63
9. FILELIST SEARCH PF Keys.....	69
10. ALIALIST PF Keys.....	74
11. AUTHLIST PF Keys.....	82
12. Lock Functions on Files and Directories.....	87
13. Reserved File Types.....	109
14. Commands Used to Assign File Mode Numbers.....	126
15. VMLINK Linking Behavior When No Link to the Disk.....	152
16. VMLINK Linking Behavior When Already Linked in READ (Not Accessed).....	153
17. VMLINK Linking Behavior When Already Linked in READ (Accessed).....	154
18. VMLINK Linking Behavior When Already Linked in WRITE (Not Accessed).....	155
19. VMLINK Linking Behavior When Already Linked in WRITE (Accessed).....	156
20. VMLINK Linking Behavior When Already Linked Multiple Times (READ and WRITE).....	157
21. VMLINK Linking Behavior When Already Linked But Directory Has Moved.....	158
22. VMLINK Linking Behavior When Already Linked But No Longer Have Authority.....	159
23. Toggling between layers of HELP.....	194

24. PA and PF keys in the HELP facility.....	195
25. Short suffix HELP components.....	200
26. Four-character suffix HELP components.....	200
27. File types, HELP components, and high-level HELP menus.....	202
28. Examples of menu files of components with 4-character file type suffixes.....	208
29. Examples of menu files that contain an .MT control statement.....	209
30. Text equivalents for special characters.....	215
31. Example file names and HELP commands.....	215
32. Examples of command HELP file names.....	216
33. Layering options, HELP file sections, and z/VM HELP section titles.....	217
34. The order of entries in a HELPABBR file and the retrieved command file.....	222
35. HELP command component_name operands that match an abbreviations file name.....	222
36. HELP command cmd_name operands that match an abbreviation or synonym definition.....	223
37. Abbreviations file names that identify HELP components.....	223
38. HELP and HELPCONV Control Word Summary.....	226
39. Default Windows and Virtual Screens.....	241
40. CMSPF Key Settings.....	243
41. WMPF Key Settings.....	247
42. Default Windows.....	269
43. Default Virtual Screens.....	271
44. Default Settings for Message Routing.....	272

About This Document

This topic collection provides information about using the IBM z/VM conversational monitor system (CMS). It will show you how to:

- Use CMS to manage your files
- Create and update help files managed by the z/VM HELP Facility
- Use windowing commands and full-screen CMS to customize your CMS session
- Create and run programs in the CMS environment.

Intended Audience

This information is intended for users who want to use CMS functions and facilities. If you are not familiar with basic CMS functions, you might first want to read *z/VM: CMS Primer*. Those who will use this information for assistance in running programs should be familiar with the programming language they will use.

Where to Find More Information

For more information about CMS and the other parts of z/VM, see the documents listed in the [“Bibliography” on page 307](#).

Links to Other Documents and Websites

The PDF version of this document contains links to other documents and websites. A link from this document to another document works only when both documents are in the same directory or database, and a link to a website works only if you have access to the Internet. A document link is to a specific edition. If a new edition of a linked document has been published since the publication of this document, the linked document might not be the latest edition.

How to provide feedback to IBM

We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information. See [How to send feedback to IBM](#) for additional information.

Summary of Changes for z/VM: CMS User's Guide

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line (|) to the left of the change.

SC24-6266-73, z/VM 7.3 (June 2025)

This edition includes terminology, maintenance, and editorial changes.

SC24-6266-73, z/VM 7.3 (September 2023)

This edition includes terminology, maintenance, and editorial changes.

SC24-6266-73, z/VM 7.3 (September 2022)

This edition supports the general availability of z/VM 7.3. Note that the publication number suffix (-73) indicates the z/VM release to which this edition applies.

Part 1. Introduction to CMS

Learning how to use CMS is not an end in itself; you must learn how CMS interacts with the other environments of VM. The information contained in the following topic is organized to help you quickly familiarize yourself with CMS and the other environments of VM, so that you can learn how to take advantage of CMS to simplify your work.

Chapter 1, “Introduction to CMS and the z/VM Environment,” on page 3 introduces you to VM 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. During a terminal session, commands and requests that you enter are processed by different parts of the system. This section describes how and when you can communicate with these different programs.

Chapter 1. Introduction to CMS and the z/VM Environment

The Conversation Monitor System (CMS) is a component of z/VM. To understand what CMS is, and how it works in z/VM, first you must understand the components (environments) of z/VM.

z/VM is an interactive, multiple-access operating system. Interactive means two-way communication between users and z/VM. Multiple-access means many people can use a z/VM system at the same time. Therefore, productivity is increased by sharing data more quickly and easily between you and other users.

With z/VM, you have a functional simulation of a real computer and its associated devices at your fingertips. This functional equivalent of a computing system is called a *virtual machine*.

Virtual machines are not real, but do work like a real system. Your entire organization can use z/VM to share the resources of a single processor, while at the same time every user accesses the system as if they were the only user.

The Control Program (CP) is a component that manages the resources of a single computer so that multiple computing systems appear to exist. When you are working in the CP environment, you are provided with processor functions, input and output devices, and processor storage.

This brings us to the CMS environment. CMS performs two roles: (1) as an end-user interface, it is the part of z/VM that is most often seen by your users, and (2) it is the part of the operating system that supports the running of your programs, thus, it is an application programming interface.

Using CP and CMS Commands

You can use CP and CMS commands, either alone or together, to accomplish a variety of tasks in z/VM. Following are some of the ways you can use CP and CMS commands.

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 DEFINE command. CP takes care of the space allocation for you and then lets your virtual machine use it.

Receiving Messages

If you are receiving printed output at your terminal and do not want to be interrupted by messages from other z/VM users, you can use the CP SET MSG OFF command 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.

You can also modify the configuration of devices in your virtual machine. CP commands are available to all virtual machines using z/VM. You can enter these commands when you are in the virtual machine environment using CMS (or some other operating system) in your virtual machine.

Not all CP commands are available to all users; some commands are only available to system administrators or other privileged users. Such commands are not discussed in this document.

Because many CP commands are used with CMS commands when performing a task, 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. These commands and other general CP commands are discussed in detail in [z/VM: CP Commands and Utilities Reference](#).

CMS Command Language

The CMS command language lets you create, modify, and debug problems or application programs and, in general, manipulate data files. Many CMS commands are discussed and used as examples in this document. For more information on a command, see [z/VM: CMS Commands and Utilities Reference](#).

Storing Files

You can take advantage of two methods for storing your files; you can store them on minidisks or in a file pool. *Minidisks* are areas of direct access storage device (DASD) space assigned to individual users. A *file pool* is a large amount of DASD 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)*. Chapter 3, “Using the Shared File System,” on page 37 discusses file pools and directories (individual files organized in hierarchical structures) in more detail.

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 XEDIT, the Editor

When you want to create, modify, or manipulate CMS files, you call the editor, XEDIT. After XEDIT is started, you may process XEDIT subcommands and use the REXX/VM interpreter or EXEC 2 macro facility. The REXX/VM 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 macrolanguage. A *macrolanguage* 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.

Using HELP

You use the HELP command to display information on how to use CP commands and CMS commands, subcommands, execs, and explanations of CP and CMS messages. You can enter 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 Full-Screen CMS

CMS also lets you 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 lets you scroll forward and backward through your CMS session to see commands you entered previously and CMS responses to those commands.

Creating and Running Programs

z/VM supports many programming language environments, such as Ada, AD/Cycle, APL2®, Assembler, C, C++, COBOL, FORTRAN, Pascal, PL/I, REXX, RPG, and VisualAge® Generator. You can find a

comprehensive list of language processors that are run under CMS and relevant publications in *z/VM: CMS Application Development Guide*. CMS runs the assembler and the compilers when you load them with CMS commands.

Using Non-English Languages

If your z/VM 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:

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

z/VM lets you 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 z/VM in another language. SET LANGUAGE also lets you add language information for applications.

For more information about the SET LANGUAGE, QUERY LANGLIST, and QUERY LANGUAGE commands, see *z/VM: CMS Commands and Utilities Reference*. For more information on the QUERY CPLANG command, see *z/VM: CP Commands and Utilities Reference*.

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

Using CP from CMS

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

Using CMS Pipelines

CMS Pipelines lets you solve a complex problem by breaking it up into a series of smaller, less complex programs called *stages*. A series of stages is called a *pipeline*.

A pipeline operates on a set of data. The output resulting from one stage is the input to the next stage. Each stage manipulates or handles this data. Each stage consists of a stage subcommand and its operands.

You can call CMS Pipelines by issuing the CMS PIPE command interactively or by invoking it from an exec procedure.

z/VM Environments and Mode Switching

This document covers the tasks you can perform using CMS. Many of these tasks involve the interaction of CMS commands and commands or programs used in other environments of z/VM. Because you will be moving from one environment to another, or *mode switching*, you need to know how the environments of z/VM are related, and the paths you need to take to achieve your task.

Figure 1 on page 6 and Table 1 on page 6 summarize the paths between the z/VM environments, list the commands and actions you need to move between environments, and describe what to expect when entering commands from each. Additional information on entering commands will be given in “[Entering Commands](#)” on page 11.

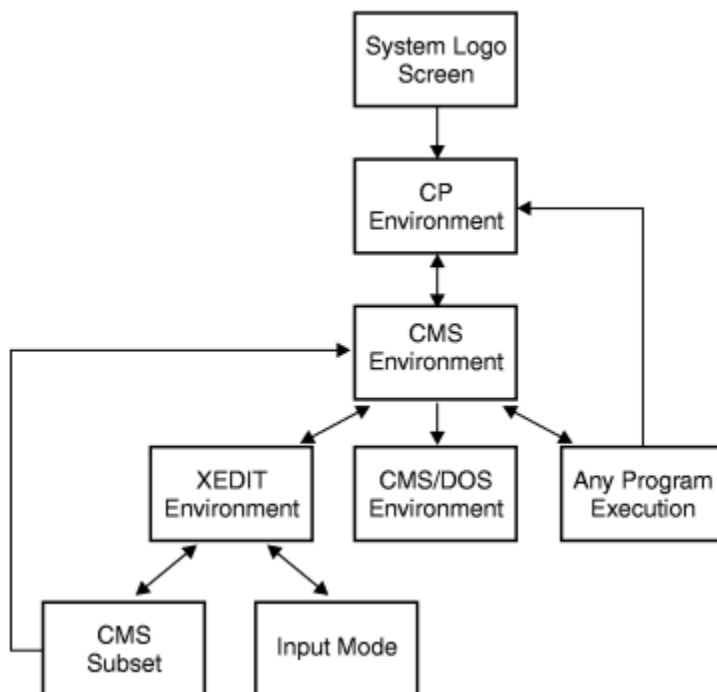


Figure 1. Paths From One Environment to Another

Table 1. Commands Used to Move from One Environment to Another

To Go From...	To...	Enter This...
system logo screen	CP environment	LOGON
CP environment	CMS environment	IPL CMS or Begin
CMS environment	XEDIT environment	XEDIT <i>fn ft fm</i> or XEDIT <i>fn ft</i>
CMS environment	CMS/DOS environment	SET DOS ON
CMS environment	a program execution	run any OS or CMS program
XEDIT environment	CMS subset	CMS
XEDIT environment	input mode	INPUT
input mode	XEDIT environment	any input line carriage return, or null line
CMS/DOS environment	CMS environment	SET DOS OFF
CMS subset	XEDIT environment	RETURN
CMS subset	CMS environment	HX
XEDIT environment	CMS environment	FILE or QUIT
any program execution	CMS environment	HX (or abend)

Table 1. Commands Used to Move from One Environment to Another (continued)

To Go From...	To...	Enter This...
---------------	-------	---------------

Notes:

1. The CP environment may be entered from any other environment using your terminal's Attention key (PA1 or equivalent), or by entering: `#cp`
2. A CP command can be entered at any time; if you are not in the CP environment, you must prefix your command with `#CP`. For more information, see Table 3 on page 8.
3. If you return to the environment you were in when you typed CP, you can do so using the BEGIN command. For example, if you were:
 - In the XEDIT environment or in input mode, the current line pointer remains unchanged.
 - Executing a program, execution resumes at the instruction address indicated in the Program Status Word (PSW).

With the exception of input mode in the XEDIT environment, you can always determine which environment your virtual machine is in by pressing the Return or Enter key (either once or twice) with a null line. Table 2 on page 7 lists responses you receive and the environments they indicate:

Table 2. Determining your Environment from the System Response

Response	Environment
CP	CP
CMS	CMS
CMS DOS on	CMS/DOS
CMS subset	CMS Subset

CP Environment

After you log on z/VM, 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. Only CP commands are valid terminal input in the CP environment. You can find detailed information about each CP command in *z/VM: CP Commands and Utilities Reference*.

You can also enter CP commands from other z/VM 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 CP READ status message (displayed in the lower right corner of your screen) indicates that your virtual machine is in the CP environment.

After entering whatever CP commands you need to use, you can 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: After 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 initial setting), use the CP TERMINAL command. For more information, see the BRKKEY option of the CP TERMINAL command in [z/VM: CP Commands and Utilities Reference](#).

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 error forces you into the CP environment.

When you enter the IPL command, specify the name of the CMS named saved system 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 run many of your own OS or DOS programs. For more information on running OS or DOS programs, see [z/VM: CMS Application Development Guide for Assembler](#).

Table 3 on page 8 lists the ways that you can enter CP commands from the CMS environment.

Table 3. Using CP Commands in a CMS Environment	
Command	Description
Using the implied CP function of CMS	<div>This function is controlled by the SET IMPCP command. If IMPCP=ON, command names unrecognizable to CMS are considered a CP command and are sent to CP. You can determine whether the implied CP function is in effect for your virtual machine by entering the command: <pre>query impcp</pre> If the response is IMPCP = OFF, you can change it by entering: <pre>set impcp on</pre></div>
Using a CP command	<div>For example, you can enter the following: <pre>cp close punch</pre></div>
Using the #CP function	<div>This function lets you preface any CP command with the characters #CP, followed by one or more blanks, which passes the command to the control program <i>immediately</i>, bypassing any processing by the CMS virtual machine. For example: <pre>#cp set msg off</pre> The #CP function can be used in any z/VM environment, and you can enter it even when a program is executing.</div>

Table 3. Using CP Commands in a CMS Environment (continued)

Command	Description
Notes:	
1.	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 process in EXEC procedures should be prefaced by the CP command, regardless of the implied CP setting.
2.	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, you do not receive the ready message.

CMS Virtual Machine Environments

z/VM provides two versions of CMS:

ESA/390 CMS (generally referred to simply as CMS)

CMS runs in the following virtual machine architectures:

ESA/390 (ESA or XA virtual machine)

An ESA virtual machine simulates IBM Enterprise Systems Architecture/390 (ESA/390), which is a superset of IBM Enterprise Systems Architecture/370 (ESA/370), which is a superset of IBM System/370 Extended Architecture (370-XA). The XA virtual machine designation is supported for compatibility; an XA virtual machine is functionally equivalent to an ESA virtual machine.

ESA/XC (XC virtual machine)

An XC virtual machine processes according to IBM Enterprise Systems Architecture/Extended Configuration (ESA/XC), which is an architecture unique to virtual machines. Although XC virtual machines run with dynamic address translation off, they can take advantage of a subset of dynamic address translation architectural features, and in particular, data spaces.

z/Architecture® CMS (z/CMS)

z/CMS runs in the following virtual machine architectures:

z/Architecture (ESA or XA virtual machine)

z/Architecture uses 31-bit addressing mode in an ESA, XA, or Z virtual machine. CMS programs can use z/Architecture instructions, including those that operate on 64-bit registers, while permitting existing ESA/390 architecture CMS programs to continue to function without change.

When z/CMS is IPLed in an ESA/390 (ESA or XA) virtual machine, z/CMS switches the virtual machine to z/Architecture mode and thereafter executes in z/Architecture mode.

z/XC (XC virtual machine)

A z/XC guest uses VM Data Spaces with z/Architecture in the same way that an ESA/XC guest uses VM Data Spaces with Enterprise Systems Architecture. CMS applications that run in z/Architecture can use multiple address spaces. z/CMS can use VM Data Spaces for accessing Shared File System (SFS) Directory Control (DIRCONTROL) directories. z/XC supports programs that employ z/Architecture instructions and registers (within the limits of z/CMS support) and programs that exploit data spaces in the same CMS session.

When z/CMS is IPLed in an XC virtual machine, z/CMS switches the virtual machine to z/XC mode and thereafter executes in z/XC mode.

Unless otherwise indicated, "CMS" means either version, and descriptions of CMS functions apply to both CMS and z/CMS. For information on the differences between z/CMS and CMS, see [z/VM: CMS Planning and Administration](#).

The XEDIT Environment, Input Mode, and CMS Subset

XEDIT is a z/VM facility that lets you create and modify data files that reside on minidisks or SFS directories. The editor, or 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 editor is found in [z/VM: XEDIT Commands and Macros Reference](#) and [z/VM: XEDIT User's Guide](#). For introductory tutorial information about editing, see [z/VM: CMS Primer](#).

For example, 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, you would enter:

```
xedit party supplies
```

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 press Enter twice.

If you enter the XEDIT subcommand:

```
cms
```

the editor gives the following response to indicate you are in CMS subset mode:

```
CMS subset
```

At this point, you can enter any CMS command that is allowed in CMS subset mode. Commands that run 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 HX (Halt Execution) command, your editing session abends and your virtual machine returns to the CMS environment.

For more information on CMS subset, see [z/VM: XEDIT Commands and Macros Reference](#).

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 SFS directory. Otherwise, you can enter the subcommand QUIT, which tells the editor not to save any modifications or insertions made after the last time the file was written.

CMS/DOS

If you are a Virtual Storage Extended (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 programs, run programs, or both, 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.

Note: The CMS/DOS environment does not support System/370 (370 mode) applications. You will need to use the CP 370 Accommodation Facility to execute 370 applications in this environment. For information about the CP 370 Accommodation Facility, see [z/VM: CP Programming Services](#).

For more information about the CMS/DOS environment, see [z/VM: CMS Application Development Guide for Assembler](#).

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 start an automatic re-IPL upon encountering an error that causes CMS to 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 to place you in a protected application at log on time. You can use the CONCEAL option of the SET command to enter a protected application at any time during your terminal session by entering:

```
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 Virtual Telecommunications Access Method (VTAM®) 3270 graphics terminals.

For more information on the SET CONCEAL command, see [z/VM: CP Commands and Utilities Reference](#).

Entering Commands

When you are running your virtual machine under z/VM, 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.

After you have typed 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 z/VM.

If you make a mistake entering a command, z/VM 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; z/VM translates your input to uppercase.

To enter a command from a display terminal when you use it as a virtual machine console under z/VM, type the command and press Enter. The keyboard is never locked during the execution of a command or program, so 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.

If there are commands that you use frequently, you can set the *program function keys* (PF keys) on your terminal to process 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 10, “Introducing Full-Screen CMS,” on page 237 provides details on PF keys in full-screen CMS and in the Window Manipulation (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 process a #CP function with a PF key, or you want a PF key to run 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. For example, you can change the setting of the PF5 key by entering:

```
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, z/VM will XEDIT a file called TEST FILE, input the word line, and write the file to file mode A.

Note: Throughout this document, you may see references to the term *file mode A*. This term refers to a minidisk or SFS directory 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 PF 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 CP 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*, which results in the command or data line being placed in the user input area. Then, you must press Enter to process the command. For example:

```
cp set pf1 "#cp query rdr all
```

would place the following command in the user input area when PF1 was entered:

```
#cp query rdr all
```

The user would then press Enter for the command to be processed. The default setting is DELAY for PF keys.

With delay, it is possible to modify the line before you enter it. 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 Enter to process 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, see the section on "Logical Line Editing Symbols" in [z/VM: CP Commands and Utilities Reference](#).

How z/VM 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 entered the command:

```
set rdymsg smsg
```

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

```
Ready;
```

When you enter a command incorrectly, you receive a message describing the error. The ready message contains the return code from the command. By default, a return code of up to 10 digits in length displays as part of the ready message. For example:

```
Ready(00028);
```

indicates that the return code from the command was 28.

```
Ready(200000);
```

indicates that the return code from the command was 200000.

Note that at least 5 digits are displayed. Leading zeros are added on the left as needed. 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. Note that at least 4 digits are displayed when the return code is negative.

The number of digits displayed for the return code depends on the SET RDYMSG command. For more information, see [z/VM: CMS Commands and Utilities Reference](#).

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. For example:

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 enter 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. For example, 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. For example, the following command will return no response from CP:

```
spool e class x hold
```

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

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

It prompts you with the message:

```
DMSRT604R Enter sort fields:
```

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

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 DASD storage. This storage is either on a minidisk or within an SFS file pool.

A minidisk is a location on a real DASD that has been allocated for storage of a user's files. Minidisks are defined using CP, and can also be formatted for use as an OS (Operating System) or DOS (Disk Operating System) disk.

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 (Operating System) data sets. There are two kinds of directories, those that have the *file control* (FILECONTROL) attribute, and those that have the *directory control* (DIRCONTROL) attribute. Later the differences between the two kinds of directories will be discussed. For now, just remember that there are two kinds of directories that let you do different kinds of work on the files within them.

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 more information on the size of your SFS file space, you can use the QUERY LIMITS command. For more information on the QUERY LIMITS, see [z/VM: CMS Commands and Utilities 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.

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 CMS commands that resulted in an error condition. There are two techniques you can use in z/VM 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. For more information, see [“Some Options Available on the CP SPOOL Command”](#) on page 163.

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 an accessed minidisk or SFS directory as a CMS file. You would do this by receiving the console file using the RDRLIST command. 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.

For 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.

- 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. For more information on the SET LOGFILE command, see [z/VM: CMS Commands and Utilities Reference](#).

Copying Your Screen

If you are using a 3270 or 3290 display terminal, and you have a 3286, 3287, 3288, or 3289 printer, you can copy the entire screen display currently appearing on the screen. To copy the screen, you have to assign the copying function to a PF key with the CP SET PFnn COPY command.

Then, whenever you want to copy a screen display, you can press a key (whichever PF key you set). The display is printed on the printer 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.

When you use the copy function to copy a screen, all 24 lines of the display screen are copied. For additional information, see [z/VM: CP Commands and Utilities Reference](#).

Within the edit environment, you can copy your screen by assigning a PF key using the XEDIT subcommand, SET PFn with COPYKEY option. For example, you could enter the following subcommand:

```
set pf9 copykey
```

To print an edit screen, you could then press PF9. [Figure 2 on page 16](#) is an example of a 3270 screen display that could be copied on the printer. The line:

```
X E D I T 1 File
```

is the screen status area for a remotely attached 3270; if locally attached, this area is blank. You can use the user input area of your screen to type in comments, or your name or user ID, if several users are spooling copy files.

```
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 8/04/00 21:12:04
BANANA  DATA      A1 V          95      29      1 8/04/00 20:58:07
BEAR    NOTE         A1 V         107     281      5 8/04/00 17:59:00
HONEY   DATA      A1 V          92     101      2 8/02/00 15:33:05
LION    NOTE         A2 V          75      28      1 7/25/00 12:10:03
ALL     NOTEBOOK  A0 V         120     277      4 7/24/00  9:14:02
TIGER   NOTE         A1 V          26       7      1 7/23/00 16:50:06
ZOOKEEP NETDATA   A1 V          80     489     10 6/26/00 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= XEDIT/LIST 12= Cursor

====> John Doe - sample screen to be copied

X E D I T 1 File
```

Figure 2. 3270 Screen Display

For more information about copying screens in XEDIT, see the COPYKEY option of the SET PF command in *z/VM: XEDIT Commands and Macros Reference*.

If you are using full-screen CMS, you can copy your screen with the PSCREEN PUT command. This command sends a copy of your physical screen to a CMS file which you can later XEDIT or print. For more information, see *z/VM: CMS Commands and Utilities Reference*.

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 either 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
- 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 Enter key sends an interruption to your virtual machine, and one depression of the PA1 key results 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, either PA1 or Enter, 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
```

Notes:

1. In any fullscreen environment, like Fullscreen CMS or XEDIT, all PF keys, all PA keys, the Enter key, and the Clear key cause virtual machine I/O interruptions.
2. After 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 initial setting), enter the CP TERMINAL command.

Interrupting Your Programs

HX (Halt Execution), HT (Halt Typing), and RT (Resume Typing) are three of the CMS *Immediate commands*. They are immediate because they are processed as soon as they are entered. Unlike other

commands, they are not stacked in the console stack. On a display terminal, you can enter an Immediate command, such as HT or HX, whenever your virtual machine is in a running status, without having to signal an interruption before you enter the command. On a typewriter terminal, you must press the Attention key once to cause a virtual machine-check interruption (if the terminal mode is set to VM) before you can enter an Immediate 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 Enter. 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 with prefix #. 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 with a prefix of #, 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 processed (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 processed first and ABCD will be passed to the exec.

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 PA1 (or equivalent) to place your terminal into the CP environment (indicated by the CP READ in the screen status area). 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 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, to suppress output from the TAPE DUMP command, you would enter:

```
tape dump #ht
```

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.

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 have a prefix of #. If they are, they will not be processed at the time they are entered. Instead, they will be processed 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 [z/VM: CP Commands and Utilities Reference](#).

The CP TRACE Command

You can use the CP TRACE command to monitor events that occur in your virtual machine. This lets you analyze the operation of your virtual machine and debug problems. z/VM lets you trace a number of events, including:

- Instruction execution
- Storage alteration
- Register alteration
- I/O activity.

Each traced event results in a trace entry, a command response that you can have sent to your virtual console, to a virtual printer, or to both. The trace entry contains a significant amount of information about

the event. For a full explanation of how to use the trace facility, along with examples, see [z/VM: Virtual Machine Operation](#).

Using the 3270 Text Feature

If you have a 3277 or 3278 display station equipped with the Data Analysis Text keyboard, you can type in, as well as display, all of the special text characters. For a description of these characters, see [z/VM: CMS Application Development Guide](#). These characters are in addition to those available with standard EBCDIC 3270 terminals. To obtain a printed copy of your screen, see “Copying Your Screen” on page 16.

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
```

For more information on using the SET TEXT commands to select appropriate translation tables for special characters, see [z/VM: CMS Commands and Utilities Reference](#) and [z/VM: XEDIT Commands and Macros Reference](#).

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 more information on displaying nondisplayable characters, see the description of the SET NONDISP commands in [z/VM: CMS Commands and Utilities Reference](#) and [z/VM: XEDIT Commands and Macros Reference](#).

Syntax, Message, and Response Conventions

The following topics provide information on the conventions used in syntax diagrams and in examples of messages and responses.

How to Read Syntax Diagrams

Special diagrams (often called *railroad tracks*) are used to show the syntax of external interfaces.

To read a syntax diagram, follow the path of the line. Read from left to right and top to bottom.

- The ►►— symbol indicates the beginning of the syntax diagram.
- The —► symbol, at the end of a line, indicates that the syntax diagram is continued on the next line.
- The ►— symbol, at the beginning of a line, indicates that the syntax diagram is continued from the previous line.
- The —►◄ symbol indicates the end of the syntax diagram.

Within the syntax diagram, items on the line are required, items below the line are optional, and items above the line are defaults. See the examples in [Table 4 on page 21](#).

Table 4. Examples of Syntax Diagram Conventions	
Syntax Diagram Convention	Example
Keywords and Constants A keyword or constant appears in uppercase letters. In this example, you must specify the item KEYWORD as shown. In most cases, you can specify a keyword or constant in uppercase letters, lowercase letters, or any combination. However, some applications may have additional conventions for using all-uppercase or all-lowercase.	►► KEYWORD ◄◄
Abbreviations Uppercase letters denote the shortest acceptable abbreviation of an item, and lowercase letters denote the part that can be omitted. If an item appears entirely in uppercase letters, it cannot be abbreviated. In this example, you can specify KEYWO, KEYWOR, or KEYWORD.	►► KEYWOrd ◄◄
Symbols You must specify these symbols exactly as they appear in the syntax diagram.	* Asterisk : Colon , Comma = Equal Sign - Hyphen () Parentheses . Period
Variables A variable appears in highlighted lowercase, usually italics. In this example, <i>var_name</i> represents a variable that you must specify following KEYWORD.	►► KEYWOrd — <i>var_name</i> ◄◄

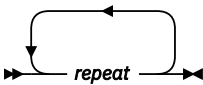
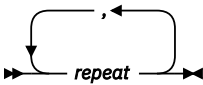
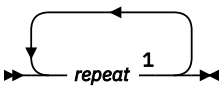

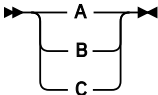
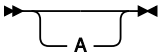
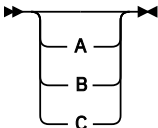
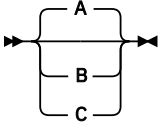
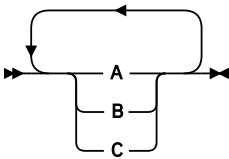
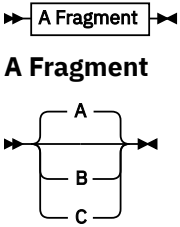
Table 4. Examples of Syntax Diagram Conventions (continued)	
Syntax Diagram Convention	Example
Repetitions An arrow returning to the left means that the item can be repeated. A character within the arrow means that you must separate each repetition of the item with that character. A number (1) by the arrow references a syntax note at the bottom of the diagram. The syntax note tells you how many times the item can be repeated. Syntax notes may also be used to explain other special aspects of the syntax.	   Notes: <div>¹ Specify <i>repeat</i> up to 5 times.</div>
Required Item or Choice When an item is on the line, it is required. In this example, you must specify A. When two or more items are in a stack and one of them is on the line, you must specify one item. In this example, you must choose A, B, or C.	 
Optional Item or Choice When an item is below the line, it is optional. In this example, you can choose A or nothing at all. When two or more items are in a stack below the line, all of them are optional. In this example, you can choose A, B, C, or nothing at all.	 
Defaults When an item is above the line, it is the default. The system will use the default unless you override it. You can override the default by specifying an option from the stack below the line. In this example, A is the default. You can override A by choosing B or C.	
Repeatable Choice A stack of items followed by an arrow returning to the left means that you can select more than one item or, in some cases, repeat a single item. In this example, you can choose any combination of A, B, or C.	

Table 4. Examples of Syntax Diagram Conventions (continued)

Syntax Diagram Convention	Example
Syntax Fragment <p>Some diagrams, because of their length, must fragment the syntax. The fragment name appears between vertical bars in the diagram. The expanded fragment appears in the diagram after a heading with the same fragment name.</p> <p>In this example, the fragment is named "A Fragment."</p>	 <p>The example shows two parts. The top part is a box labeled 'A Fragment' with arrows on either side. The bottom part is a heading 'A Fragment' followed by a diagram where three sub-elements labeled A, B, and C are grouped by a large right-facing curly bracket, with arrows on either side of the bracket.</p>

Examples of Messages and Responses

Although most examples of messages and responses are shown exactly as they would appear, some content might depend on the specific situation. The following notation is used to show variable, optional, or alternative content:

xxx

Highlighted text (usually italics) indicates a variable that represents the data that will be displayed.

[]

Brackets enclose optional text that might be displayed.

{ }

Braces enclose alternative versions of text, one of which will be displayed.

|

The vertical bar separates items within brackets or braces.

...

The ellipsis indicates that the preceding item might be repeated. A vertical ellipsis indicates that the preceding line, or a variation of that line, might be repeated.

Part 2. Working with CMS Files

File management is one of the primary functions of CMS. You need to know how to name files, organize them, store and print them, and share them with other users. The following topics teach you these tasks and more.

Chapter 2, “CMS File System,” on page 27 discusses the data and programs that you create that are stored in *files*. These files are stored in Shared File System (SFS) directories or on minidisks. This section introduces you to the creation and handling of CMS files.

Chapter 3, “Using the Shared File System,” on page 37 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 4, “Storing Your Files on Minidisks,” on page 101 provides details on how to manage files stored on minidisks.

Chapter 5, “More on the CMS File System,” on page 109 contains more detailed discussion on CMS files and CMS commands. Information on reserved file types, how file mode letters and numbers are used in SFS and minidisk environments, and CMS command search order and execution characteristics, are examples.

Chapter 6, “Using Real Printers, Punches, and Readers,” on page 163 discusses how to store and retrieve CMS files on punched cards, and how to use your virtual printer and punch to get real output.

Chapter 7, “Using Tapes,” on page 169 discusses how to store and retrieve CMS files on tape, using tape devices, and the use of tape commands.

Chapter 2. 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 (file ID). The file ID 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 refer to 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.

Under z/VM, 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 section provides general information on CMS files and how to display and manipulate 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. For more information on managing your files in SFS, see [Chapter 3, “Using the Shared File System,”](#) on page 37 and for more information on using minidisks, see [Chapter 4, “Storing Your Files on Minidisks,”](#) on page 101.

z/VM OpenExtensions (POSIX support) includes another type of file called a byte file system (BFS) file. BFS files are organized in a hierarchy, as in a UNIX system. All files are members of a *directory*. Each directory is in turn a member of another directory at a higher level in the hierarchy. The highest level of the hierarchy is the *BFS file space*. Typically, a user has all or part of a BFS file space mounted as the *root directory*.

VM views an entire file hierarchy as a *Byte File System*. Each Byte File System is a mountable file system. The root file system is the first file system mounted. Subsequent file systems can be mounted on any directory within the root file system or on a directory within any mounted file system.

All files in the Byte File System are called BFS files. BFS files are byte-oriented, rather than record-oriented, like CMS record files on minidisks or in the Shared File System. You can copy BFS files into CMS record files, and you can copy CMS record files into the Byte File System.

For more information on BFS files, see “An Introduction to the Byte File System” in [z/VM: OpenExtensions User's Guide](#).

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.

If you need to, however, you can specify logical record length and record format when you create an SFS file with the CREATE FILE command, or you can change the record format of an existing file as explained in [“Changing the Record Format of a File”](#) on page 162.

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 109.

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, \$ (dollar sign), # (pound sign), @ (at sign), + (plus sign), - (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 z/VM system, z/VM translates your input line by either the user-defined input table or by the uppercase table. For more information on the SET INPUT command, see *z/VM: CMS Commands and Utilities Reference*. If you do not have an input table, you can just enter the command in lowercase and z/VM 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 unsupported characters.

The @ and @ characters are line editing symbols in z/VM; when you use them to identify a file, you must precede them with the logical escape symbol (^). For more information on logical line editing symbols, see *z/VM: CP Commands and Utilities Reference*.

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 can be an uppercase or lowercase letter (A-Z), a number (0-9), or one of several special characters, \$ (dollar sign), # (pound sign), @ (at sign), - (hyphen), or _ (underscore). The remaining characters can be A-Z and 0-9. Lowercase letters are allowed, as they are translated to uppercase. For more information about directory names, see Chapter 3, “Using the Shared File System,” on page 37 and *z/VM: CMS Commands and Utilities Reference*.

The file mode is made up of:

File Mode	Description
Letter	<p>Specifies an alphabetic character (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.</p> <p>The file mode letter, for any file, can change during a terminal session. Suppose when you log on, your top directory (or minidisk) is accessed with a file mode of A. (Your <i>top directory</i> 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 (or minidisk), named SPECIAL EVENTS, would have a file identifier of:</p> <pre>SPECIAL EVENTS A</pre> <p>If, however, you later access another directory (or minidisk) with a file mode of A, and access your top directory (or minidisk) with a file mode of B, then the SPECIAL EVENTS file will have a file identifier of:</p> <pre>SPECIAL EVENTS B</pre>
Number	<p>Specifies a number from 0 to 6. Each number has a particular meaning to CMS. For more information, see “File Mode Letters and Numbers” on page 117.</p>

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 process the CMS command whose name you duplicated.

For more information on how CMS identifies a command name, see [“CMS Command Search Order” on page 161](#).

For the most part, 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* and should be avoided, if possible. For more information on reserved file types, see [“What Are Reserved File Types?” on page 109](#).

Working with CMS Files

In *z/VM: CMS Primer*, many examples were used to explain how to create, display, and manipulate your CMS files. This section is an overview or review of that material. It is not intended as a complete source of information on any of the commands discussed. It is merely a refresher to ready you for upcoming chapters. For more information on these commands see [z/VM: CMS Commands and Utilities Reference](#).

Creating a New File

There are many ways to create a new file under CMS. The most common method is with the editor.

To create a new file called NEW FILE A and store it on your A disk, you would enter the following command either from the command line within the FILELIST display, or from the command line of CMS:

```
xedit new file a
```

A new, empty file called NEW FILE A would be displayed for your input. After your modifications were complete, you would **file** the new file for later retrieval.

Note: When a file is created using XEDIT, the editor assumes many default characteristics. These will be discussed in [Chapter 5, “More on the CMS File System,” on page 109](#).

For more information on using XEDIT to create files, see [z/VM: XEDIT User's Guide](#).

A few other methods of creating a new file are:

- Use the COPYFILE command to create a new file from an existing file.
- Modify an existing file and **file** or save it under a new name.
- For SFS users, the CREATE FILE command can be used which includes many formatting options. For more information, see [“Creating New Files” on page 53](#).

Editing a File

To modify the contents of an existing file on one of your accessed disks or SFS subdirectories, you would use the editor.

For example, to edit a file called CHANGE CONTENTS A, the following command would be entered:

```
xedit change contents a
```

As a result, the current contents of the file would be displayed on your screen, ready for your modifications. When complete, you would enter, **file**, to save your alterations.

Displaying a List of Your CMS Files

Once you have created or gained access to CMS files, you can use CMS commands to display information about them. For this purpose, these two commands are available:

- LISTFILE
- FILELIST

Listing Your Files with the LISTFILE Command

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

```
listfile
```

displays the file name, file type, and file mode of all files on your A disk in the following format:

```
ALL      NOTEBOOK A0
ANIMAL   DATA      A1
BANANA   DATA      A1
BEAR     NOTE       A1
HONEY    DATA      A1
TIGER    NOTE       A1
ZOOKEEP  NETDATA    A1
```

For SFS, the LISTFILE command has various options that allow you to display different information about your files and directories. An example is the SHARE option. When you enter:

```
listfile (share
```

Your output will be similar to this:

```
FILENAME FILETYPE FM OWNER   TYPE    R W
ALL      NOTEBOOK A0  yourid  BASE    X X
ANIMAL   DATA      A1  CROCKETT BASE    X X
BANANA   DATA      A1  BREEZY  ALIAS   X X
BEAR     NOTE       A1  BREEZY  ALIAS   X -
HONEY    DATA      A1  STONE   ALIAS   X -
TIGER    NOTE       A1  BREEZY  ALIAS   X -
ZOOKEEP  NETDATA    A1  yourid  BASE    X X
```

In addition to the file name, file type, and file mode, this display shows you the owner of the file, whether your file is a base file or an alias, and what type of authority you have to the file (read or write). For more information on these terms, see [Chapter 3, “Using the Shared File System,”](#) on page 37.

For more information on the LISTFILE command, see [z/VM: CMS Commands and Utilities Reference](#).

Using the FILELIST Command

Use the FILELIST command to display information about your CMS files. If your files are stored in an SFS file pool, you can use FILELIST for files in accessed directories.

Entering the FILELIST command will display a screen of information similar to the one shown in [Figure 3 on page 31](#).

```

yourid FILELIST A0 V 108 Trunc=108 Size=7 Line=1 Col=1 Alt=0
Cmd Filename Filetype Fm Format Lrecl Records Blocks Date Time
ANIMAL DATA A1 V 95 34 1 8/04/00 21:12:04
BANANA DATA A1 V 95 29 1 8/04/00 20:58:07
BEAR NOTE A1 V 107 281 5 8/04/00 17:59:00
HONEY DATA A1 V 92 101 2 8/02/00 15:33:05
ALL NOTEBOOK A0 V 120 277 4 7/24/00 9:14:02
TIGER NOTE A1 V 26 7 1 7/23/00 16:50:06
ZOOKEEP NETDATA A1 V 80 489 10 6/26/00 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= XEDIT/LIST 12= Cursor

====> _

X E D I T 1 File

```

Figure 3. Sample FILELIST Screen

This FILELIST screen shows information about the files on file mode A.

- The first line has information about the FILELIST display itself. Your user ID appears instead of *yourid*. Size shows the number of files in the list. Line tells which file in the list is the first on this screen.
- There are several columns in the FILELIST display:
 - Cmd is where you would enter commands for a specific file or directory listed.
 - Filename, Filetype, and Fm indicate the file name, file type, and file mode of the file. For SFS, names of subdirectories are listed in the Filename column, but the Filetype column is blank.
 - Format shows the format of a file. An F is displayed for a fixed format file, V for a variable format file, and DIR for an SFS directory.
 - Lrecl, Records, and Blocks, carry information on the size of the file shown. Listings for SFS directories show a dash (–) in the columns for Lrecl, Records, and Blocks.
 - Date and Time show when a file was last updated. For SFS directories, these columns show when the directory was created.
- The bottom of the screen shows the current PF key settings.

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 also enter XEDIT subcommands to manipulate the list itself.

For SFS, like LISTFILE, the FILELIST command has several options that allow you to display different information about your files and directories. The default option is FILELIST STATS. The FILELIST STATS screen is similar to one displayed for a non-SFS user except for the directory name identifier, appearing in the top left portion of the screen.

```

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 8/04/00 21:12:04
BANANA DATA A1 V 95 29 1 8/04/00 20:58:07
BEAR NOTE A1 V 107 281 5 8/04/00 17:59:00
HONEY DATA A1 V 92 101 2 8/02/00 15:33:05
ALL NOTEBOOK A0 V 120 277 4 7/24/00 9:14:02
TIGER NOTE A1 V 26 7 1 7/23/00 16:50:06
ZOOKEEP NETDATA A1 V 80 489 10 6/26/00 16:05:08

```

Figure 4. Sample FILELIST STATS Screen

SFS also provides you with SHARE, SEARCH, and ALLDATES options of FILELIST to display other information about your files and directories. Chapter 3, “Using the Shared File System,” on page 37 shows example screens for the FILELIST options. For more information, see [z/VM: CMS Commands and Utilities Reference](#).

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.

PF7

Scrolls backward one screen.

PF8

Scrolls forward one screen.

2. Use any of the appropriate PF keys to sort the displayed output.
3. Use the XEDIT subcommand LOCATE if you know the file name, file type, or both, 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 or LISTFILE to List Some of Your Files

The LISTFILE and FILELIST commands let you obtain various lists of your files and subdirectories. 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. Following are various ways that you might use the FILELIST and LISTFILE commands:

Command	Description
listfile	Lists the files on file mode A.
filelist	

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

Erasing a File

Use the ERASE command to delete one or more of your CMS files or SFS directories. If a file is to be deleted, the file name, file type, and file mode (or directory identifier) are operands. If a directory is to be deleted, the directory identifier is the only operand needed.

For example, the following command will delete the file, DATA FILE A1:

```
erase data file a
```

Use the DISCARD command to erase a file or subdirectory that is displayed in a list (such as the FILELIST display). The DISCARD command is equivalent to the ERASE command. DISCARD can either be entered in the command area (Cmd) 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. For more information on the DISCARD or ERASE command, see [z/VM: CMS Commands and Utilities 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.

You can copy a file over a file which already exists (overlaying the contents of the file), using the REPLACE option. For more information on the COPYFILE command, see [z/VM: CMS Commands and Utilities Reference](#).

Notes for SFS Users

To copy a file into a:

- File into a file that already exists in an SFS directory, and you want to use the REPLACE option, you need proper authority to the existing file. The authorization needed varies depending on the kind of directory the file resides in. If the file resides in a directory with the file control (FILECONTROL) attribute, you need write authority to the file. You also need read authority to the directory in which the file resides (so you can access it). If the file resides in a directory with the DIRCONTROL attribute, you need directory control write (DIRWRITE) authority to the directory.

- FILECONTROL directory by creating a new file, you need write authority to the directory in which you are creating the file. If you are copying into a DIRCONTROL directory, you need DIRWRITE authority on the directory.

Note: Types of directories and authority will be discussed in [Chapter 10, “Introducing Full-Screen CMS,”](#) on page 237.

Comparing Contents of Files

To compare the contents of two files to see if they are identical, use the COMPARE command from the CMS command line. The comparison is made on a record-for-record basis and dissimilar records are displayed to the terminal.

For example, suppose two files exist with the following contents:

EMPLOYEE LIST A EMPLOYEE LIST B

Kathy	Kathy
Diana	Diana
Kate	Kate

To compare the contents of EMPLOYEE LIST A with EMPLOYEE LIST B, the following command would be entered:

```
compare employee list a employee list b
```

The following message would be returned indicating that contents of the files are identical:

```
DMSCMP179I Comparing EMPLOYEE LIST A with EMPLOYEE LIST B
Ready; T=0.30/0.03 08:06:21
```

If, in the second file, employee Kate was replaced by new employee Terri, the COMPARE command would return the following:

```
DMSCMP179I Comparing EMPLOYEE LIST A with EMPLOYEE LIST B
Kate
Terri
DMSCMP209W Files do not compare
Ready(00004); T=0.01/0.01 08:08:15
```

The line from each file which did not match is displayed.

Since the comparison is done a record-for-record basis, the insertion or deletion of a line may return misleading results. For example, if EMPLOYEE LIST2 A was an identical copy of EMPLOYEE LIST A, and the name of a new employee, Tom, was inserted in the file, so the contents of the files were:

EMPLOYEE LIST A EMPLOYEE LIST2 A

Kathy	Tom
Diana	Kathy
Kate	Diana
	Kate

The result of a compare on these files would be:

```
DMSCMP179I Comparing EMPLOYEE LIST A with EMPLOYEE LIST B
Kathy
Tom
Diana
Kathy
Kate
Diana
DMSCMP010E Premature EOF on file EMPLOYEE LIST A
Ready(00004); T=0.01/0.01 08:14:09
```


The COMPARE command did not recognize that the files were identical except for one line. It simply compared each line on a line-by-line or record-for-record basis and displayed each dissimilar record. It did, however, indicate that one file contained more records than the other.

For more information on the COMPARE command, see [z/VM: CMS Commands and Utilities Reference](#).

Renaming Files

You can change the file identifier of a file with the RENAME command. For example, to change the name of TEST FILE A1 to GOOD FILE A1, the following command would be used:

```
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
```

This command 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. File mode numbers will be discussed in [“File Mode Letters and Numbers”](#) on page 117.

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. You can also rename a file in another user's FILECONTROL directory, if the user has granted you write authority to the file and to the directory. To rename a file in another user's DIRCONTROL directory, you need DIRWRITE authority. Types of directories and authority will be discussed in [Chapter 3, “Using the Shared File System,”](#) on page 37.

For more information on the RENAME, COPYFILE, or RELOCATE commands, see [z/VM: CMS Commands and Utilities Reference](#).

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

As mentioned in the discussions on LISTFILE and FILELIST, some CMS commands that manipulate files allow you to enter the file name or file type fields or both 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	

The same commands allow you to use the percent sign (%) as a place holder to mean any single character.

Some examples are given in [Table 5 on page 36](#).

Table 5. Examples of Using * and % in CMS commands.

Command Entered	Result
<code>listfile t* assemble</code>	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.
<code>listfile tr* a*</code>	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.
<code>listfile %%% stock</code>	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.
<code>listfile t%% cat</code>	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, for example: <div data-bbox="480 569 579 638" data-label="Text"> <pre>top cat the cat tom cat</pre> </div>
<code>listfile %tr*s *ri% %</code> (See Note 1.)	This command would produce a list of files with: <ul style="list-style-type: none"> • File name starting with a character, followed by TR, and ending with an S. • File type starting with an RI or one or more characters preceding the RI, followed by two characters. • File mode A, since unspecified.

Notes:

1. The LISTFILE command allows a great deal of flexibility in the use of percent (%) and asterisk (*). They may appear together, have multiple occurrences, or both, within a file type or file mode.
2. Because CMS checks for open files first, you may get unexpected results when specifying an asterisk for the file mode if there are open files matching the file name and file type specified.

For more information on the commands listed in this table see [z/VM: CMS Commands and Utilities Reference](#).

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 they are 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 = b
```

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

Chapter 3. Using the Shared File System

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 section contains information specific to the Shared File System. For more information on how to manage your files if your files are stored on minidisks, see [Chapter 4, “Storing Your Files on Minidisks,”](#) on page 101.

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 section 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 additional subdirectories to organize individual chapters of the book.

[Figure 5 on page 38](#) shows how this might look.

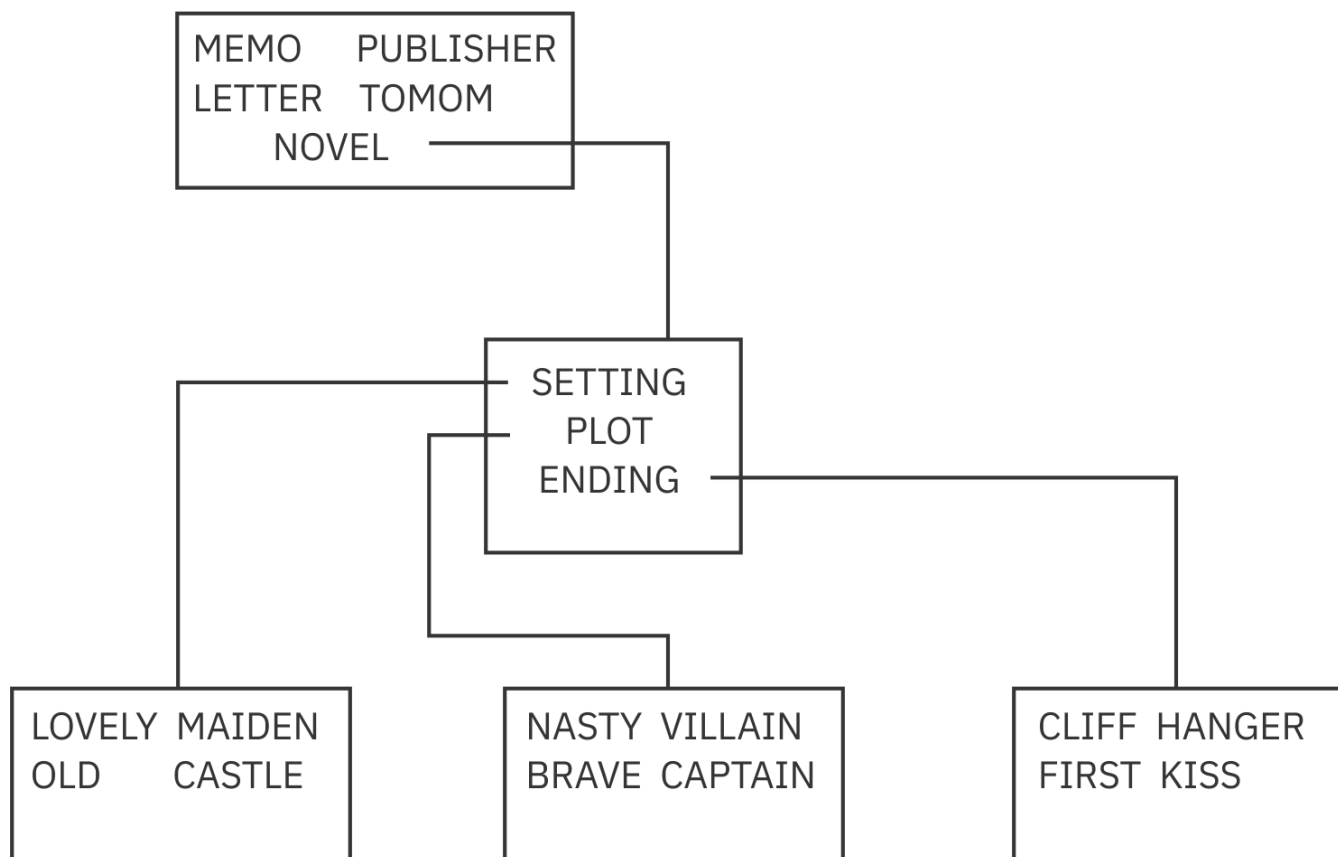


Figure 5. Sample Hierarchical Directory

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 grant other users authority to them. The kinds of authorities you can grant depend on the kind of directory you wish to share.

There are two types of directories: directories having the file control attribute, called *FILECONTROL directories*, and directories having the directory control attribute, called *DIRCONTROL directories*.

For *FILECONTROL* directories, you can choose to grant authority for one or more individual files, or you can grant authority for all the files currently in a directory. You can even authorize users to read or write any files that will be added to the directory in the future. 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.

For *DIRCONTROL* directories, you grant authorizations at the directory level—you cannot grant authorities on individual files. You either grant *directory control read* (*DIRREAD*) authority or *directory control write* (*DIRWRITE*) authority for the directory. When you grant *DIRREAD* authority, others can read any file in the directory (even files that are added in the future). *DIRWRITE* authority lets others read or write any file in the directory (even future files). *DIRWRITE* authority also lets users create or erase files in the directory.

Detailed information will be given about types of authority in sections, [“FILECONTROL Directory Authority”](#) on page 76, and [“DIRCONTROL Directory Authority”](#) on page 77.

As you can see, *FILECONTROL* directories provide more flexibility. When you create a directory it is, by default, a file control directory. You need to specify an option to create a *DIRCONTROL* directory. In most cases, your directories will be *FILECONTROL* directories.

DIRCONTROL directories have a somewhat specialized use. They are intended primarily for data that is seldom updated or that is updated in a controlled manner. Although you may choose to not create DIRCONTROL directories, you should be aware of them, because it is likely that system data or applications you use will reside in those directories. Unless otherwise noted, all of the instructions in this section apply to both FILECONTROL and DIRCONTROL directories.

The rest of this section provides information on how to manage the files and directories you store within an SFS file pool.

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. If this is the case, you may want to read this section and [Chapter 4, “Storing Your Files on Minidisks,”](#) on page 101, for additional information on how to manage CMS files on minidisks.

Getting Started - Accessing Your Top Directory

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. When first assigned to you, your top directory is always a FILECONTROL directory. You can change it to a DIRCONTROL directory, if you wish, by using the DIRATTR command.

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 indicates 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 section, you will need to determine if you are enrolled in a file pool, and know your file pool identifier (*filepoolid*). If you do not know, see your system administrator now. You may be enrolled in the IBM-supplied VMSYSU file pool, or another file pool supplied by your system administrator. The file pool ID, VMSYSU, appears in the examples in this section. If this is not the file pool you are enrolled in, you will see your file pool ID 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 section.

Once you are enrolled, you should determine if your top directory is accessed as A. To find out, enter the following command which will display the modes for all the directories and minidisks to which you have access.

```
query accessed
```

Note: You can alternatively use the MDISK option of the DIRLIST command to display not only your accessed minidisks, but also any minidisks that are linked. These are listed, in a full screen environment, along with the directory structure that was specified (see [“Listing the Structure of a Directory with DIRLIST”](#) on page 45).

When you enter the QUERY ACCESSED command, 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: <i>yourid</i> .
S	R/O	1321	190	MNT190
Y/S	R/O	337	19E	MNT19E

This information is organized as follows:

- Mode indicates the mode letter used to access the directory.

- Stat gives the status of the directory: R/W (read/write) or R/O (read/only). When you access an SFS directory without forcing the status to read-only or read/write, the status is determined by who owns it. If you:
 - Are the directory owner, then the directory is accessed as read/write. For DIRCONTROL directories, however, only one user may have the directory accessed in read/write status. If someone already has the directory accessed in read/write status, and you (the directory owner) attempt to get a read/write access, you will, instead, get a read-only access.
 - Are not the directory owner, then the directory is accessed as read-only.
 - Attempt to access a directory for which you are not authorized, the directory is not accessed.
- **Note:** Most CMS commands that issue file mode letters will not allow you to write to files if the directory is accessed as read-only. However, COPYFILE and XEDIT will allow you to write to a file control directory accessed read-only, unless you use the SET RORESPECT ON command.
- Files displays the number of entries in the directory. This total can include subdirectories, revoked or erased aliases, and files.
- Vdev displays DIR for FILECONTROL directories, DIRC for DIRCONTROL directories. (These terms will be discussed in this section.)
- Label/Directory shows the complete name of the directory. In this example, 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 you are a SFS user, and 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 section, 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 your default file pool ID.

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.

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.
- 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.

For more information on accessing directories, see [“Accessing Another User's Directory”](#) on page 44.

Releasing Directories

When you no longer need a directory that you temporarily accessed, you can use the RELEASE command. For example, to release the directory accessed with file mode B, enter:

```
release b
```

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

When you log off, any directories that you temporarily accessed are automatically released.

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

Managing Your File Space

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 (substituting the name of your file pool for the *filepoolid*):

```
query limits * filepoolid:
```

If you defined a default file pool, then just enter:

```
query limits
```

Your output will look like this:

Userid	Storage	Group	4K Block Limit	4K Blocks Committed	Threshold
<i>yourid</i>		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 more information on the SET THRESHOLD command, see [z/VM: CMS Commands and Utilities Reference](#).

Note: This warning message will be seen only once for a file pool in between console reads. An example of a console read occurring is when you press the Enter key.

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.

You can find out how much space you will free by erasing an SFS file by using the QUERY BLOCKS command. For example, if you enter:

```
query blocks myfile script a
```

for file `myfile script` in the `VMSYSU:SMITH.ALL.FILES` directory, accessed as A, you will see:

```
Directory = VMSYSU:SMITH.ALL.FILES
Filename  Filetype  Fm  Type  Datablocks  Systemblocks
MYFILE    SCRIPT    A   BASE 10      2
```

The file is taking up 12 blocks of file space, so if you erase this file, you will free 12 blocks of space.

You may have files in your file space whose data has been moved into storage controlled by DFSMS/VM. They are referred to as being in *migrated status* (this will be discussed in more detail in “[DFSMS/VM and SFS File Management](#)” on page 64). You should be aware that although these files take up logical storage in your file space, they no longer occupy real storage as long as they are in migrated status. If they are recalled from migrated status, they will once again require physical space in your storage group. This recall happens automatically when you reference the file data if the SET RECALL command is set to ON. For more information about SET RECALL, see [z/VM: CMS Commands and Utilities Reference](#).

If you cannot erase any of the files in your file space, there are several alternative 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 can overlay the contents of a file with a *packed* version of the file, using the COPYFILE command with PACK option. To browse or modify the contents of a packed file, you must recopy the file to itself using the COPYFILE command with UNPACK option.
4. You may contact your file system administrator to request that more storage be added to your file space.

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. Subdirectories can either have the FILECONTROL or DIRCONTROL attribute.

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 the example described in [Figure 6 on page 43](#).

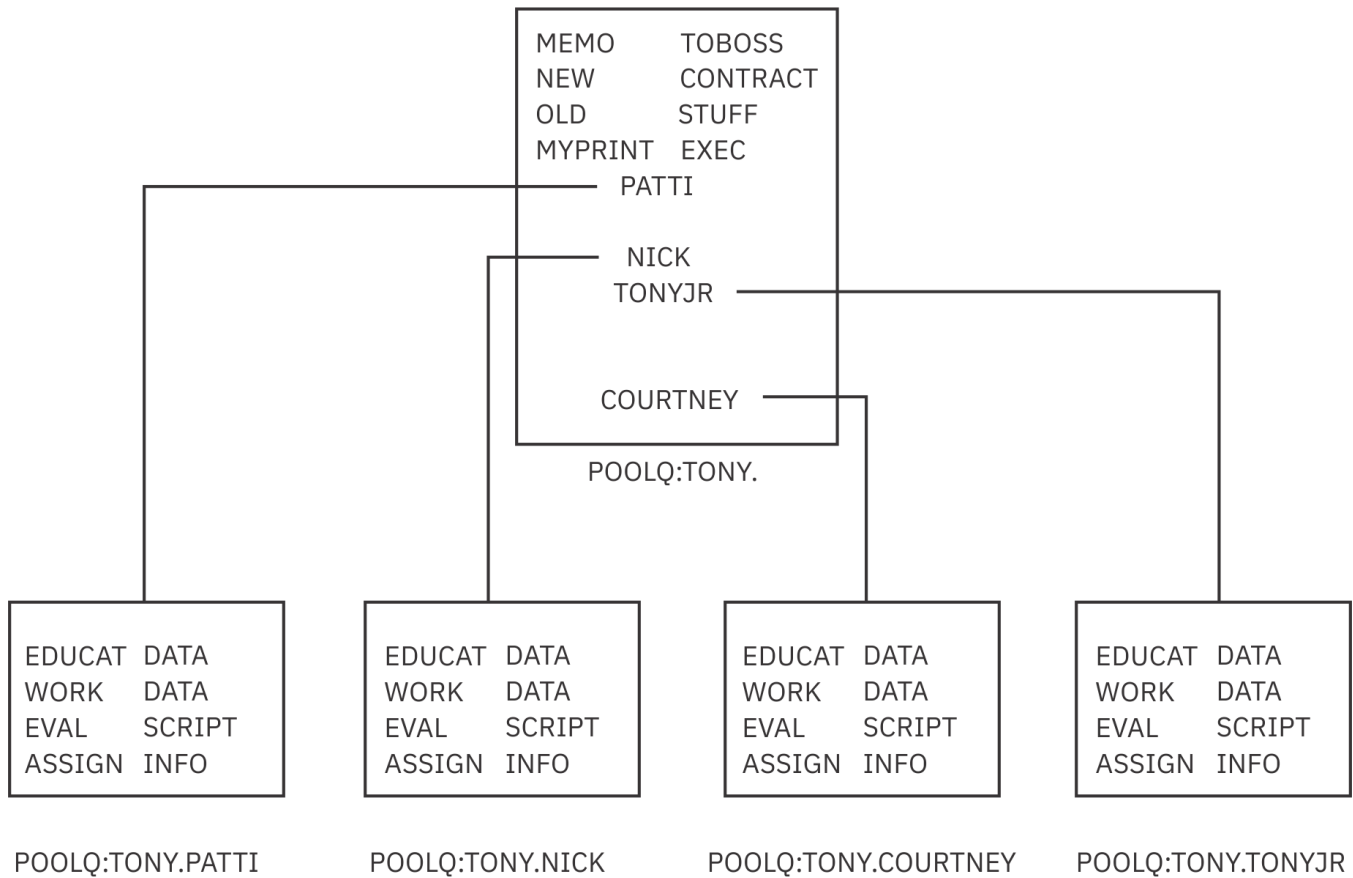


Figure 6. Another Sample Directory

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.

Likewise, 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 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 (if he has not issued a SET FILESPACE command) because it will default to his user ID. He must, however, be sure to retain the period to indicate his top directory. Therefore, Tony could 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.

Notice that you refer to FILECONTROL and DIRCONTROL directories in the same way. The .NICK directory could have either the FILECONTROL or DIRCONTROL attribute.

Accessing Another User's Directory

Note: If your system responds differently than indicated in this section, request assistance from the administrator that set up the SFS files and directories.

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 reside in the IBM-supplied VMSYSU file pool. Usually, other users will need to grant you authority for 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 as 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 — filepool: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 ENROLL command. For more information on QUERY ENROLL, see [z/VM: CMS Commands and Utilities 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 (discussed in “[File Mode Letters and Numbers](#)” on page 117), 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, you can enter the QUERY ACCESSED command, your screen will look something like this:

Mode	Stat	Files	Vdev	Label/Directory
A	R/W	2	DIR	VMSYSU: <i>yourid</i> .
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. Also notice that the MAINT. directory has the FILECONTROL attribute. If it had the DIRCONTROL attribute, you would see DIRC instead of DIR in the Vdev column. The MAINT. directory only remains accessed for the duration of your CMS session; a LOGOFF command will automatically release it.

Also notice that the MAINT. directory is accessed in read-only status. Most CMS commands that issue file mode letters will not allow you to write to files if the directory is accessed as read-only. However, COPYFILE and XEDIT will allow you to write to a file control directory accessed read-only, unless you use the SET RORESPECT ON command.

Another command that might be useful when you are accessing another user's directory is the SET FILESPACE command. This command is similar to the SET FILEPOOL command except it allows you to default the user ID portion of the *dirid* (rather than the file pool ID portion). In the above example, you could have entered:

```
set filespace maint
```

followed by:

```
access . b
```

and because you set the default user ID to MAINT, CMS would have accessed MAINT's top directory. There is a QUERY FILESPACE command you can use to find the default setting. For more information on SET FILESPACE and QUERY FILESPACE, see [z/VM: CMS Commands and Utilities Reference](#).

Note: The examples that follow assume this has not been done.

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 refer to 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.SAMPLES). It could be an abbreviated form of the directory name, as discussed in [“Using the Abbreviated Form of Your Top Directory”](#) on page 43. After you have accessed a directory, you can also use the file mode letter in commands which accept a directory name.

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.

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 [z/VM: CMS Commands and Utilities Reference](#).

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

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 just specify the file mode letter. Enter:

```
dirlist b
```

Your screen will look like the example shown in [Figure 7 on page 46](#).

```
yourid  DIRLIST  A0  V 319   l=1 Alt=0
Cmd      Fm Directory Name
-        B  VMSYSU:MAINT.
-        -  VMSYSU:MAINT.SAMPLES

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 7. 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 processed 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 can also display only a subset of directories, or minidisks linked to your virtual machine. Listed below are some of the options available for the DIRLIST command. For more information, see [z/VM: CMS Commands and Utilities Reference](#).

ALL

the default for DIRLIST; it lists all the directories in the specified directory structure for which you have some authority. Directories are listed regardless of their attributes, and even if they are not accessed.

ACCESSED

to list only directories that are accessed.

DIRCONTROL

to list only directory control directories

FILECONTROL

to list only file control directories

MDISK

to list the minidisks that you have linked. If this option is used with the ACCESSED option, only the minidisks you have accessed will be listed along with the directories that are listed. When the MDISK option is used with FILECONTROL, DIRCONTROL, or ALL, all of the minidisks you have linked will be listed along with all of the FILECONTROL, DIRCONTROL, or ALL directories that are listed. Examples of using this option will be given in the next section.

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*.

Displaying Minidisks and Directories with DIRLIST

The MDISK option of DIRLIST command is an alternative to using QUERY ACCESSED command for both minidisk and SFS users. Because it shows all the minidisks that you have linked to your virtual machine in a full screen format, you can do automatic access and release of a disk. For example, the following is a sample DIRLIST (MDISK display):

```
yourid  DIRLIST A0 V 319 Trunc=319 Size=4 Line=1 Co l=1 Alt=0
Cmd    Fm Directory Name
      - VMSYSU:yourid
      C 0090
      - 019A
      - 019D
      G 019F
      A 0191
      R 0399
```

You can tell whether a directory or minidisk 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. For example, to access the minidisk at address 019A with file mode of B, you would enter the following ACCESS command:

```
yourid  DIRLIST A0 V 319 Trunc=1 Col=1 Alt=0
Cmd    Fm Directory Name
      - VMSYSU:      C 0090
acc / b 019A
      - 019D
      G 019F
      A 0191
      R 0399
```

Note: This command is also very useful when looking for free file modes.

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. You can use these PF keys to find useful information about directories and files.

Table 6. DIRLIST PF Keys

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 end results of any previously entered 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.

Table 6. DIRLIST PF Keys (continued)

Key	Meaning	Usage
PF6	Auth	Pressing the PF6 key processes 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 authority. For more information, see “Using the AUTHLIST Command” on page 81.
PF7	Backward	This key scrolls the DIRLIST display backward one screen.
PF8	Forward	This key scrolls the DIRLIST display forward one screen.
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.

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, you can use the options ACCESSED, DIRCONTROL, and FILECONTROL, to list a subset of directories. Also 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 this information:

```
Fm Directory Name
B  VMSYSU:MAINT.
-  VMSYSU:MAINT.SAMPLES
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 6 on page 43](#), Tony would have used the following series of commands:

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

In these commands, the period specifies Tony's top directory. Therefore, the first 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. By default, your directory will have the FILECONTROL attribute. To create a directory with the directory control attribute, you must specify the DIRCONTROL option of the CREATE DIRECTORY command.

[Figure 8 on page 50](#) represents your directory structure after entering the CREATE DIRECTORY command.

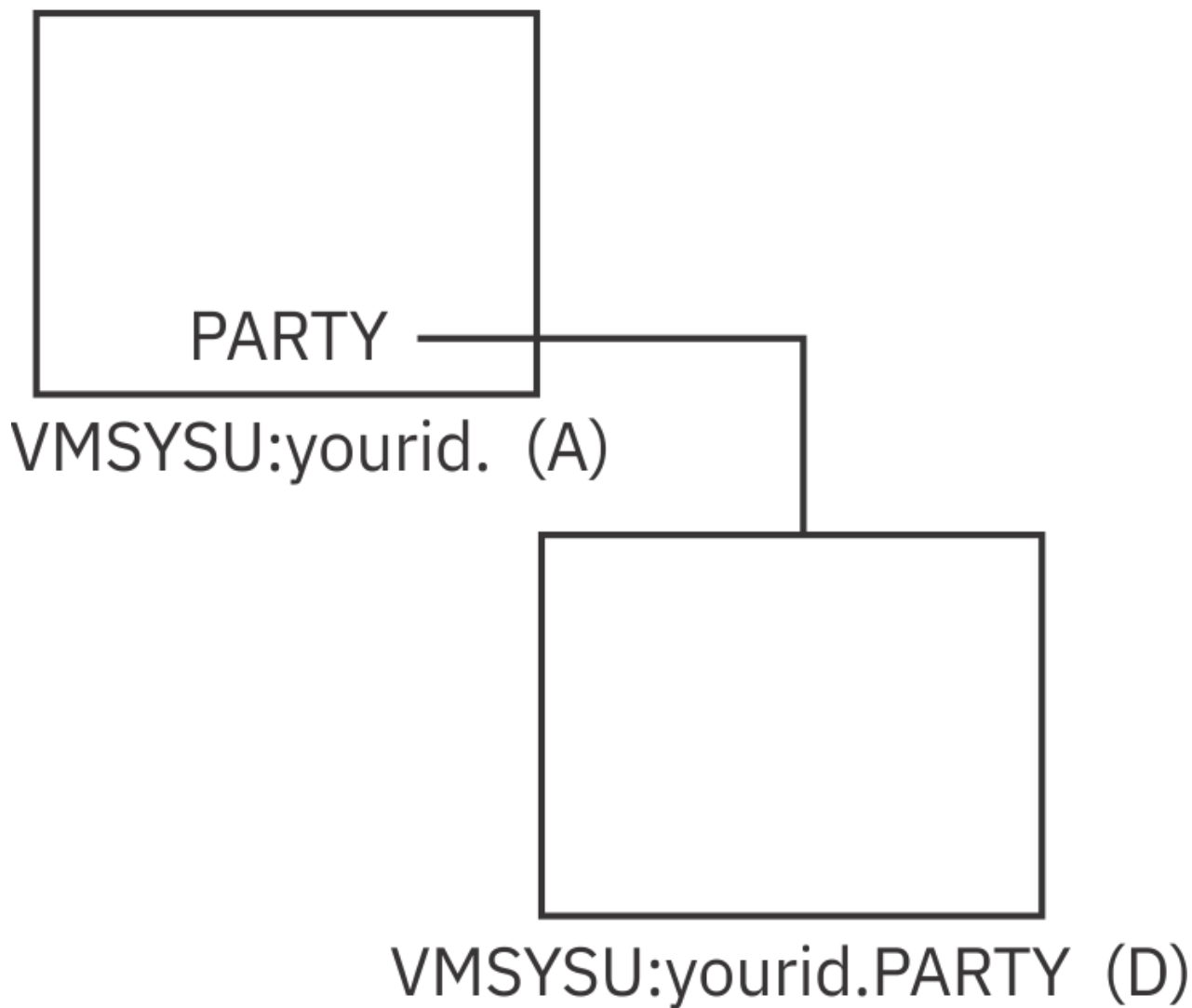


Figure 8. The .PARTY Directory

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

```
create directory .party.treats
```

[Figure 9 on page 51](#) represents your directory structure.

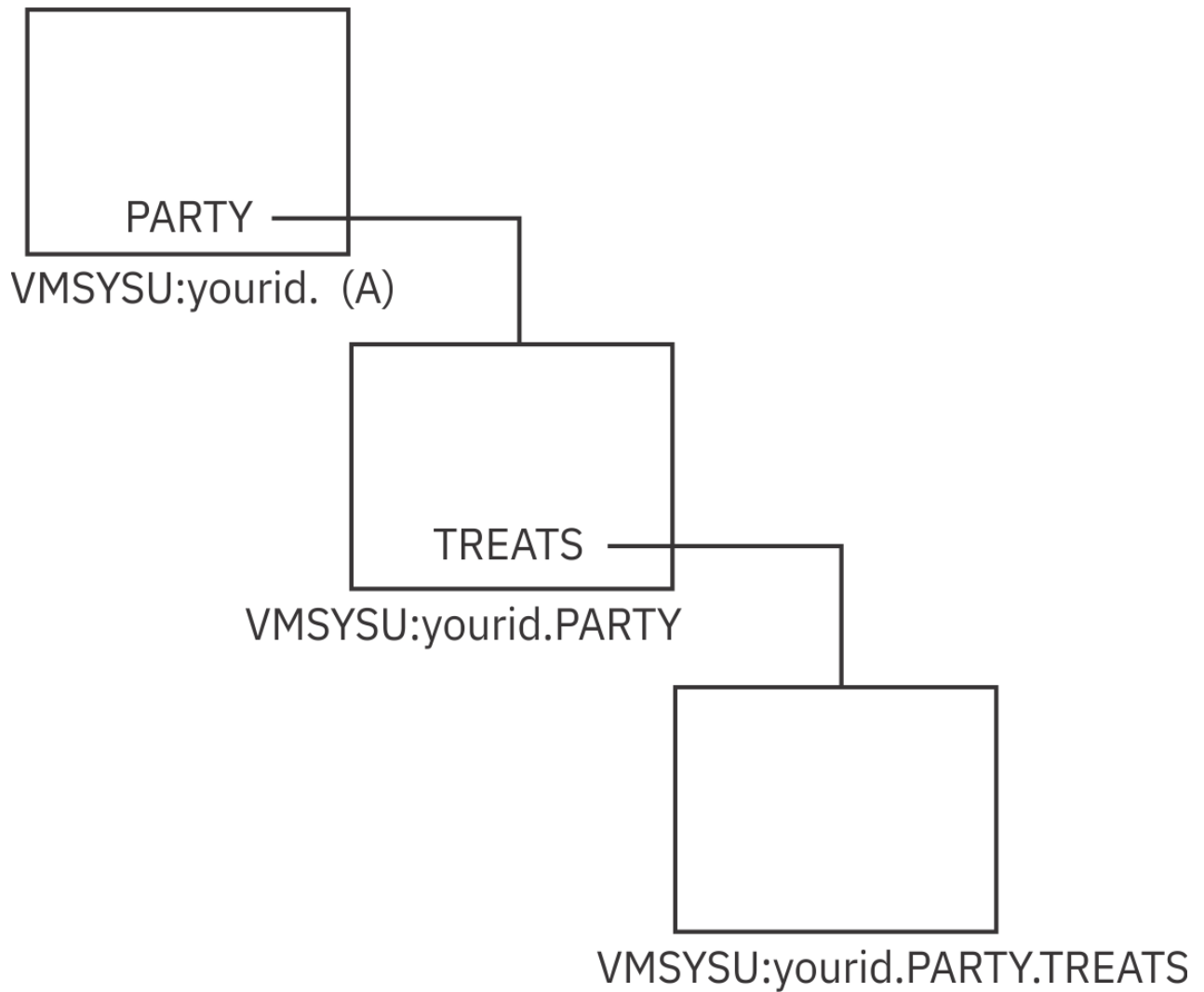


Figure 9. The .PARTY.TREATS Directory

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

```

yourid  DIRLIST  A0  V 319   l=1 Alt=0
Cmd    Fm Directory Name
-      A  VMSYSU:yourid.
-      -  VMSYSU:yourid.PARTY
-      -  VMSYSU:yourid.PARTY.TREATS

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 10. Using DIRLIST to List All Directories

The .PARTY and .PARTY.TREATS 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 [z/VM: CMS Commands and Utilities Reference](#).

Putting Files into a Directory

Although you have just created the .PARTY and .PARTY.TREATS 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. To determine which file modes you have used, enter the QUERY ACCESSED command to display information similar to the following:

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.

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.

To work with the sample files provided, we'll need to release MAINT's top directory and access MAINT's .SAMPLE directory.

```
release b
access maint.samples b
```

The MAINT.SAMPLES 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
```

Figure 11 on page 53 represents your directory structure.

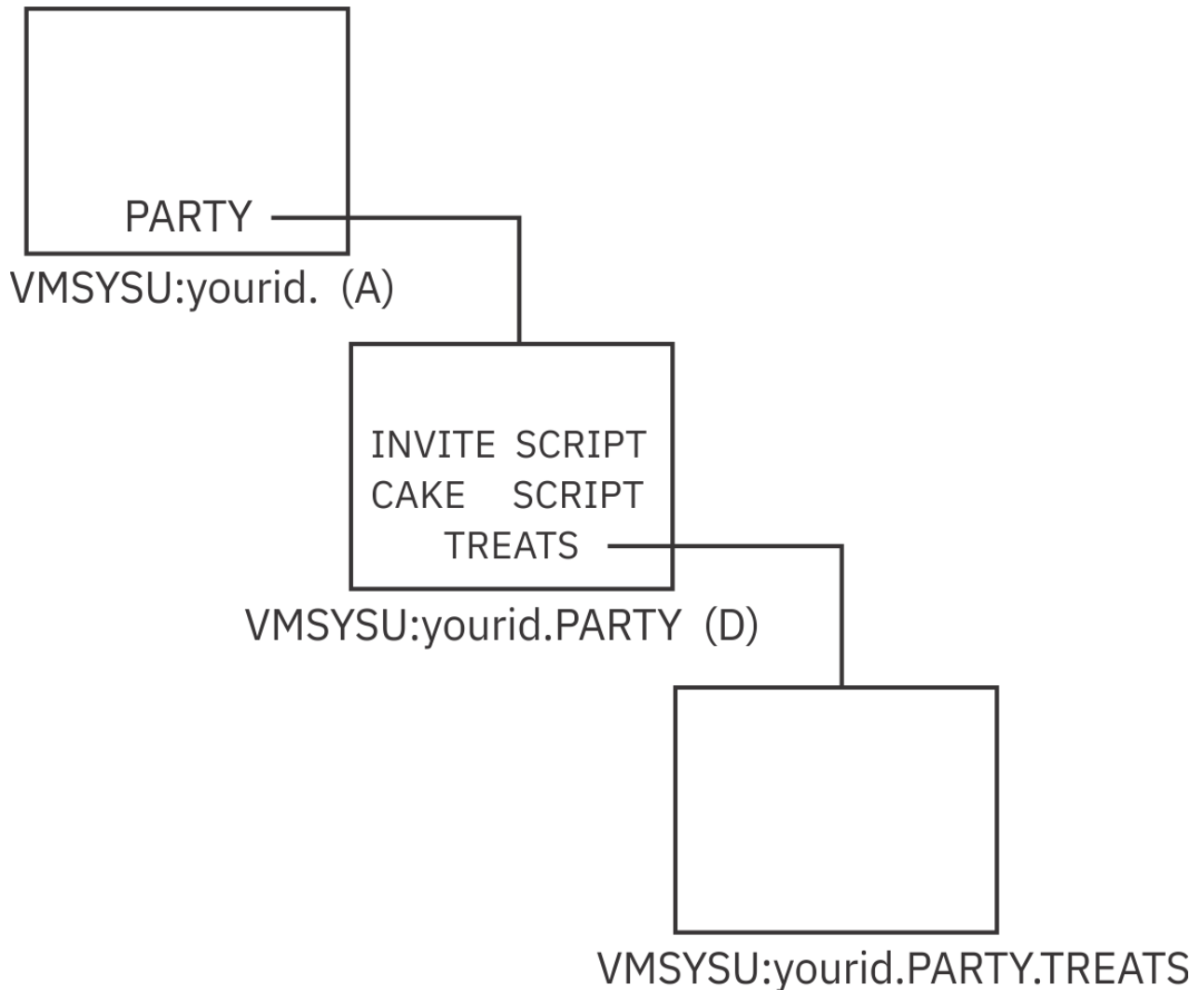


Figure 11. Files Within the .PARTY Directory

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 from a directory to minidisk. You just 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 New Files

For more information on other ways to add files to your directories including creating new files using XEDIT or using the CREATE FILE command, see [“Creating a New File” on page 29](#).

CREATE FILE creates new empty files in SFS directories only. Empty files are necessary as place holders for future input, and also for setting up authorizations and grants that will be used later. You cannot create new files on minidisks with CREATE FILE.

For example, you could create FUNTIMES SCRIPT by entering:

```
create file funtimes script d
```

CREATE FILE also lets you specify the record format, record length, and recoverability and overwrite attributes for the files you create. For additional information, see [z/VM: CMS Commands and Utilities Reference](#).

Renaming Your Files and Directories

You can use the RENAME command to rename your own file, a file in another user's directory (if you are properly authorized), or a directory.

To rename your own file, use the RENAME command format as follows (for help with reading syntax diagrams, see [“Syntax, Message, and Response Conventions”](#) on page 20):

```
➤ rename — fn1 — ft1 — dirid — fn2 — ft2 — dirid ➤
```

The *dirid* is the same for both the old and new file IDs. You cannot 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”](#) on page 54.

If you are renaming a file in another user's directory, you must either use the directory name (not the file mode) in the RENAME command, or you must access the other user's directory in read/write mode. To access another user's directory in read/write mode, use the FORCERW option of the ACCESS command.

If you wish to rename a directory, use the following sample format (for help with reading syntax diagrams, see [“Syntax, Message, and Response Conventions”](#) on page 20):

```
➤ rename — dirid1 — dirid2 ➤
```

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
- Rename directories you do not own
- Use RENAME to move a directory or subdirectory from one level to another. For example, the following command is *not* allowed:

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

Relocating Your Files and Directories

You can use the RELOCATE command to:

- Relocate a file from one directory to another
- Relocate an entire directory and all the files it contains.

A sample format of the RELOCATE command follows (for help with reading syntax diagrams, see [“Syntax, Message, and Response Conventions”](#) on page 20):

```
➤ relocate —  — dirid1 — TO — dirid2 ➤
```

You specify the file ID (including the *dirid* where the file is currently located), and the destination *dirid*.

For example, you can relocate the CAKE SCRIPT file (currently in your .PARTY directory) to your .PARTY.TREATS directory with the command:

```
relocate cake script .party to .party.treats
```

Or, you can relocate files using file modes. You could access the .PARTY directory as D, and the .PARTY.TREATS directory as E, and enter the following command:

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

After you have entered this command, the CAKE SCRIPT file will be in the .PARTY.TREATS directory. CAKE SCRIPT .PARTY.TREATS is the long version of the new name of the file.

Note: All the directories in the following series of relocations are FILECONTROL directories. There are restrictions for DIRCONTROL directories:

- Files cannot be relocated into or out of DIRCONTROL directories. However, DIRCONTROL directories can be relocated to other parts of the directory structure, and other directories can be relocated as subdirectories of DIRCONTROL directories.
- You cannot relocate a DIRCONTROL directory if anyone other than you has the directory accessed.

Figure 12 on page 55 represents your directory structure.

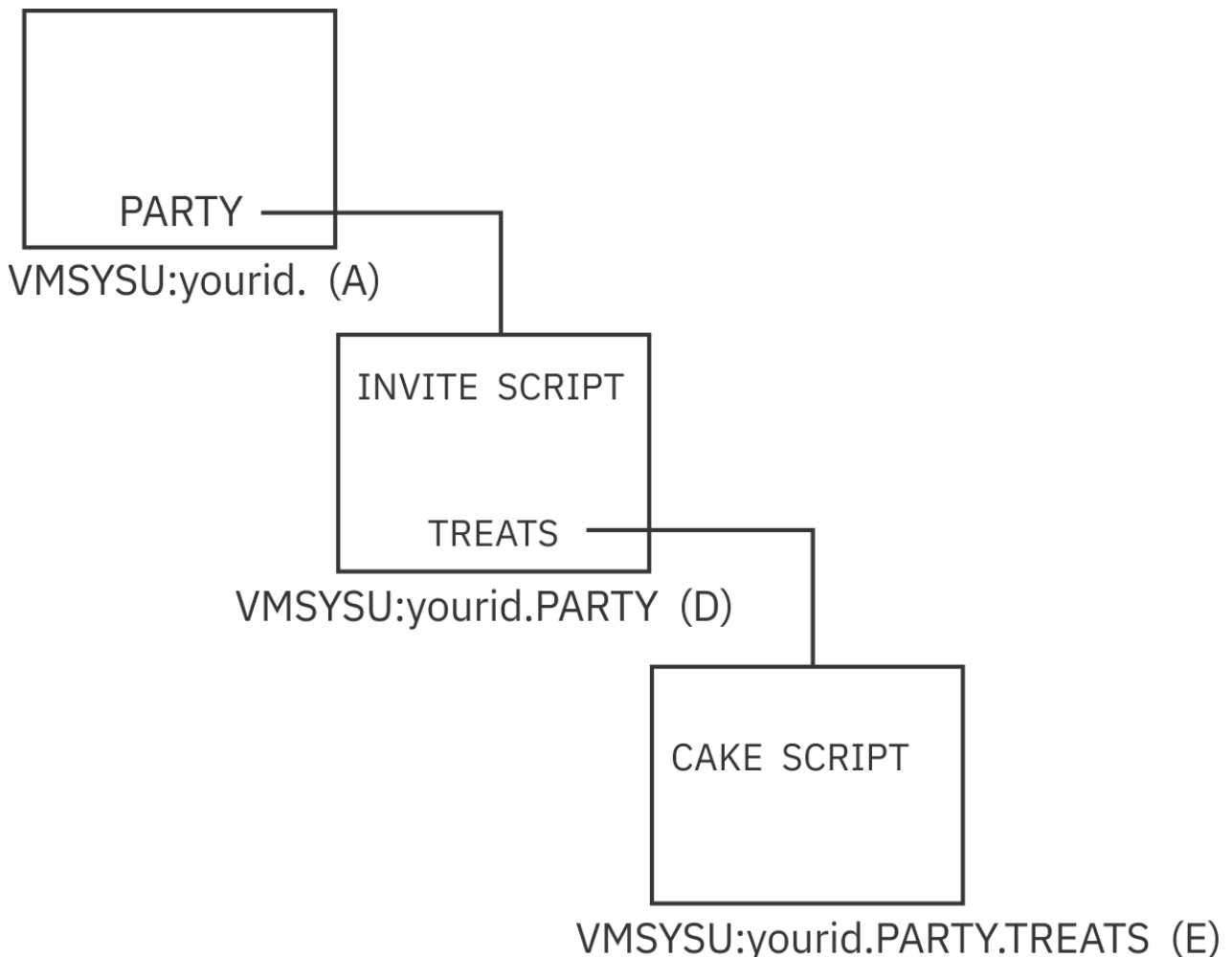


Figure 12. Moving a File to the .PARTY.TREATS Directory

To relocate an entire directory and all the files it contains, you would specify the following:

```
➡ relocate — dirid1 — TO — dirid2 →
```

For example, a new directory called .PARTY.FAVORS. will be created, then relocated. Enter:

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

Figure 13 on page 56 represents your directory structure.

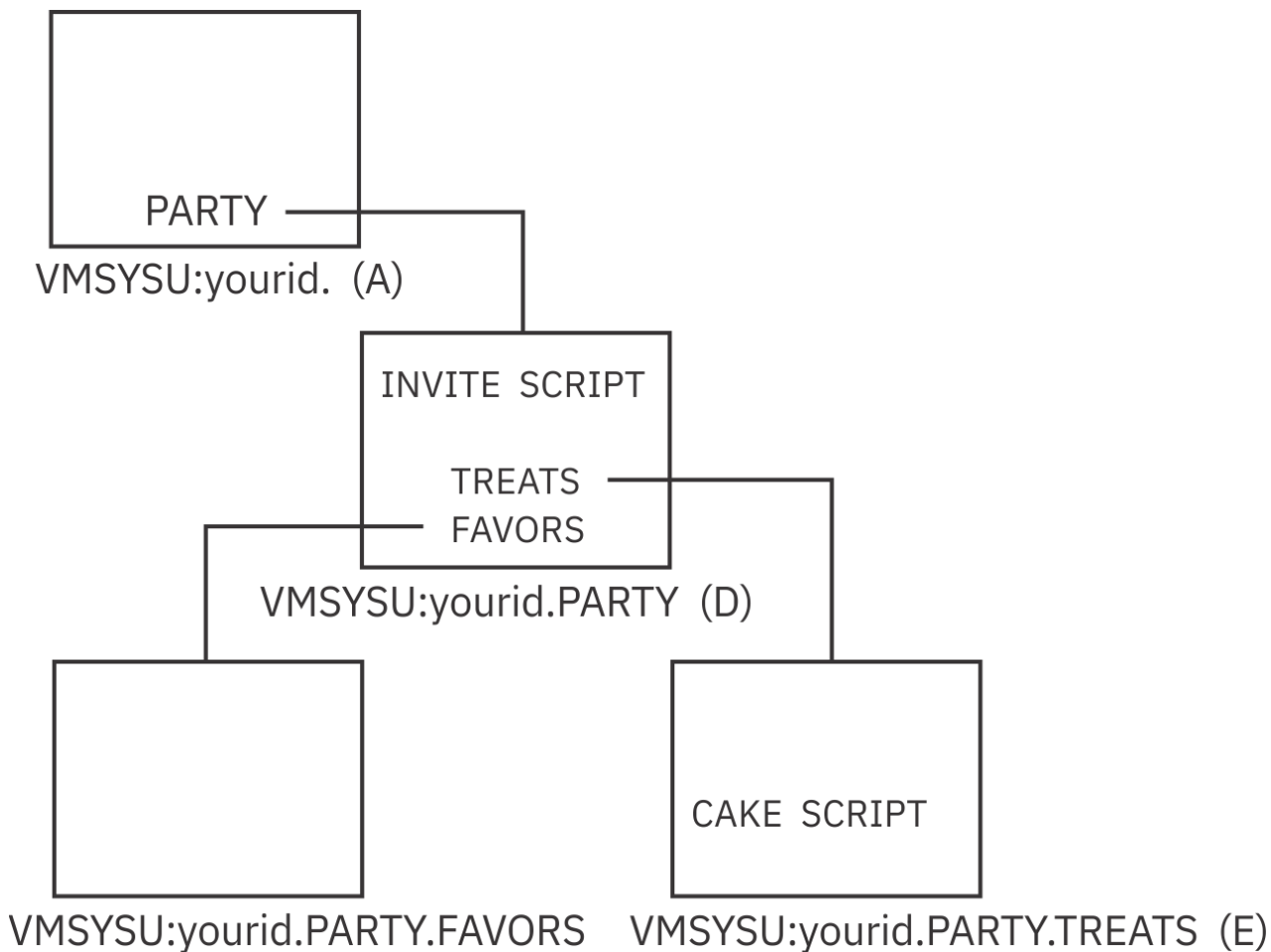


Figure 13. Creating the `.PARTY.FAVORS` Directory

Now relocate the directory with:

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

Your directory structure now looks like [Figure 14 on page 57](#).

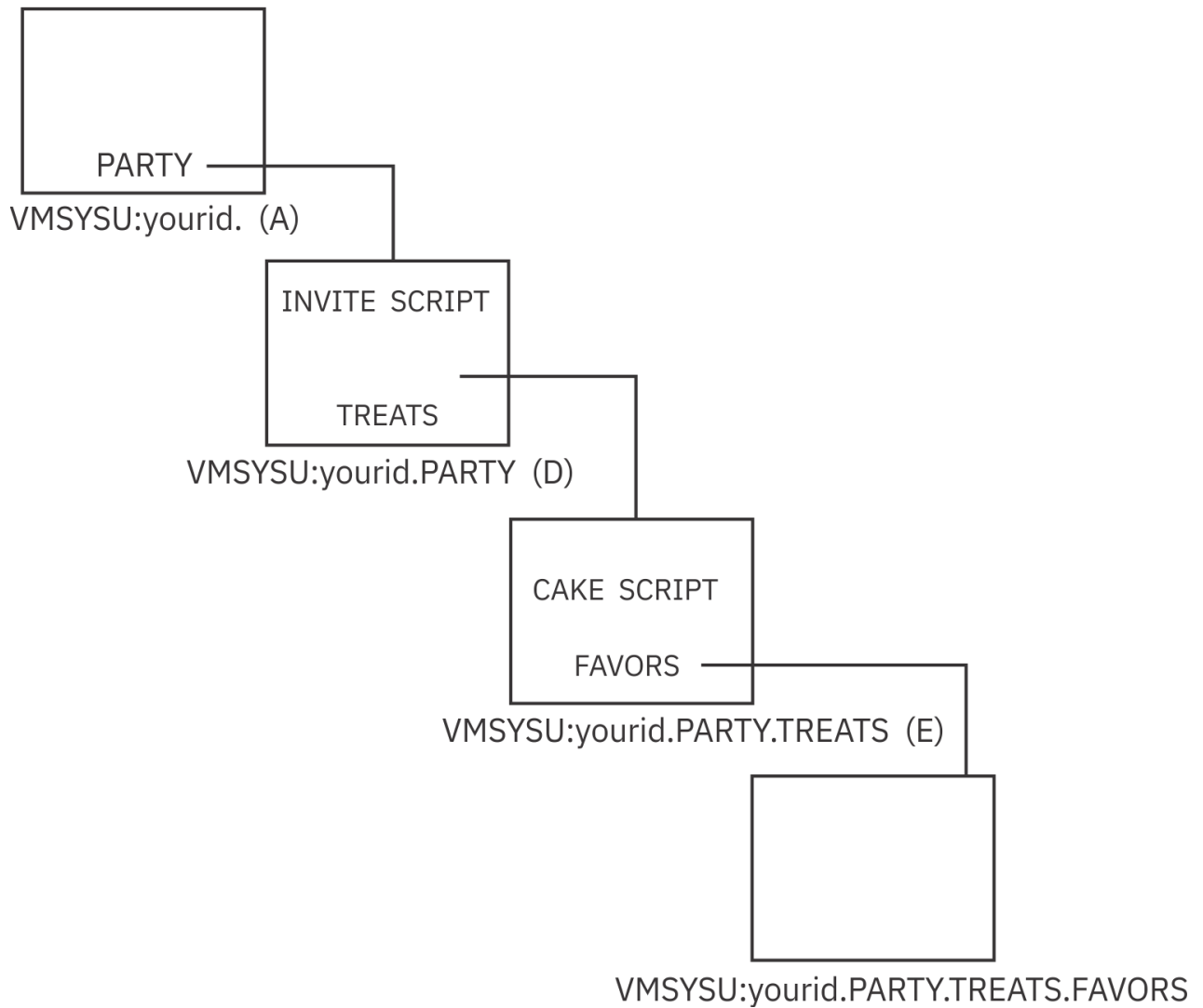


Figure 14. Relocating the `.PARTY.FAVORS` Directory

After 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 whether the commands worked, you could check `DIRLIST`.

```

yourid  DIRLIST A) V 319 Trunc=319 Size=5 Line=1 Co1=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.

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 15. Listing All Your Directories

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. (We will discuss aliases and authorities later in this section.)

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.

Erasing a Directory

When you decide to erase a directory, you must first move any files or subdirectories you wish to keep. (Use the RELOCATE command to move files in FILECONTROL directories, and COPYFILE for files in DIRCONTROL directories.)

A sample format of the ERASE command follows:

➤ **erase** — *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. NOFILES is the default for the ERASE command.

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

Navigating Through Your Directories

The options of the FILELIST command, and the FILELIST screen PF keys, can be used to navigate through your directories and see the files they contain.

For an example, we will copy some additional files from the MAINT.SAMPLES directory to the .PARTY and .PARTY.TREATS directories with the following commands:

```

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

```

Figure 16 on page 59 represents your directory structure.

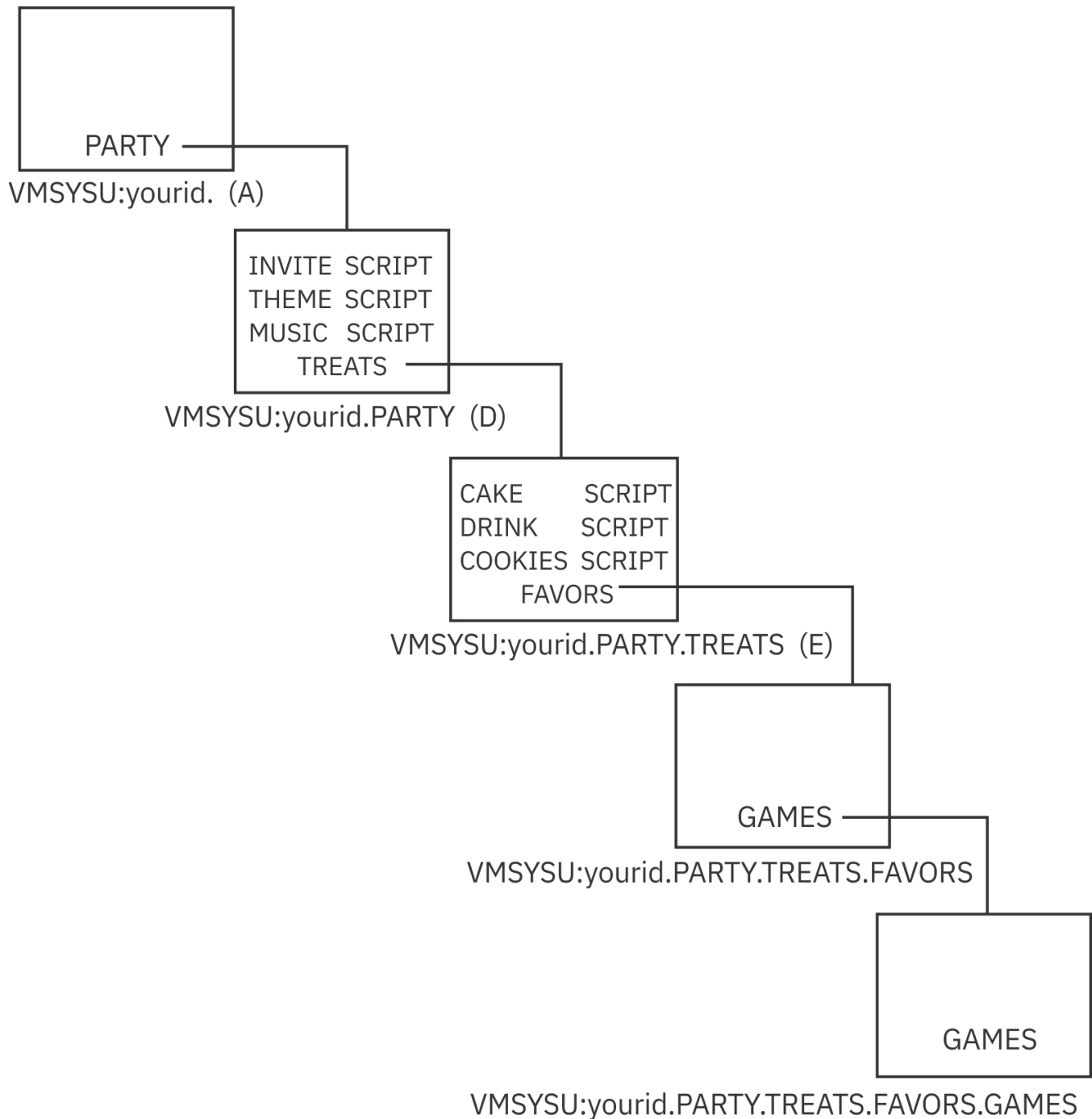


Figure 16. Copying More Files to .PARTY and PARTY.TREATS

For a FILELIST display of the .PARTY.TREATS directory, press PF11 from DIRLIST when your cursor is on the line for .PARTY.TREATS.

Your screen will look like the example shown in [Figure 17 on page 60](#).

```
yourid  FILELIST A0 V 1 08 Trunc=108 Size=4 Line=1 Col=1 A
Directory = VMSYSU:yourid.PARTY.TREATS
Cmd  Filename Filetype Fm Format Lrecl  Records  Blocks  Date  Time
-    COOKIES  SCRIPT  E1 V      37      6        1  1/06/00 16:03:52
    DRINK    SCRIPT  E1 V      13      8        1  1/06/00 16:03:42
    FAVORS   E DIR    -      -        -  1/06/00 15:59:38
    CAKE     SCRIPT  E1 V      55     13        1  1/06/00 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= XEDIT/LIST 12= Cursor
```

```
====>
```

```
X E D I T 1 File
```

Figure 17. Using PF11 from DIRLIST

This displays 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 .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.

If you need information on the date and time the files were created, the date the files were last referenced, or the date and time the files were last updated, you can use the ALLDATES option of the FILELIST command. For example, to get these additional dates for the PARTY.TREATS directory, you would enter on the command line:

```
filelist * * e (alldates
```

Your screen would look like the example shown in [Figure 18 on page 60](#).

```
yourid  FILELIST A0 V 174 Size=4 Line=1 Col=1 Alt=0
Directory = VMSYSU:yourid.PARTY.TREATS
Cmd  Filename Filetype Fm Create-Dt Create-Tm Lref-Dt  Update-Dt Update-Tm
    COOKIES  SCRIPT  E1 12/00/00 12:24:46 1/06/00 1/06/00 16:03:52
    DRINK    SCRIPT  E1 12/00/00 12:35:26 1/06/00 1/06/00 16:03:42
    FAVORS   E 12/00/00 12:40:32 1/06/00 1/06/00 15:59:38
    CAKE     SCRIPT  E1 12/03/00 11:15:23 1/06/00 1/06/00 15:56:16
```

```
1= Help      2= Refresh  3= Quit    4= Cancel    5= Sort(updt) 6= Sort(lrdt)
7= Backward  8= Forward  9= S(cdt) 10= Stats   11= XEDIT/LIST 12= Cursor
```

```
====>
```

```
X E D I T 1 File
```

Figure 18. Sample FILELIST Screen with ALLDATES Option

There are several PF key changes on this screen:

PF5

Sorts by date of last update

PF6

Sorts by date of last reference

PF9

Sorts by creation date

PF10

Toggles between the FILELIST STATS and FILELIST ALLDATES screen

For a complete description of ALLDATES and other FILELIST options, see [z/VM: CMS Commands and Utilities Reference](#).

To display the contents of the .PARTY.TREATS.FAVORS directory, you can move your cursor to the line for the FAVORS directory and press PF11.

```
yourid FILELIST A0 V 108 Tre=1 Line=1 Col=1 Alt=0
Directory = VMSYSU:yourid.PARTY.TREATS.
Cmd  Filename Filetype Fm Format Lrecl  Records  Blocks  Date  Time
-    GAMES          Z  DIR      -      -      -    1/06/00 16:00:29

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

Figure 19. Using PF11 from FILELIST

The .PARTY.TREATS.FAVORS directory contains the GAMES subdirectory. Note the message at the bottom of the screen. Because the directory was not accessed, FILELIST temporarily accessed the .PARTY.TREATS.FAVORS directory with the first available file mode letter, starting at the end of the alphabet.

The .PARTY.TREATS.FAVORS.GAMES directory is shown here with a file mode of Z (the file mode of its parent).

If you were to press PF11 from the Cmd area of the GAMES line, you would see the message:

```
Directory is empty
```

You can always tell where you are in the hierarchy by referring to the line of the FILELIST screen labeled Directory.

Table 7 on page 61 lists several PF keys on the FILELIST STATS screen that provide you with information about the files and directories.

Table 7. 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.

Table 7. FILELIST STATS PF Keys (continued)

Key	Meaning	Usage
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.
PF11	XEDIT/LIST	When you press PF11, if your cursor is located on a file, PF11 will XEDIT the file. If your cursor is for 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 the example shown in [Figure 20 on page 62](#).

```

yourid  FILELIST A0  V 149 Size=6 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    E1 yourid    DIR       X X
     CAKE      SCRIPT    E1 yourid    BASE      X X
     ICECREAM  SCRIPT    E1 yourid    ALIAS     X -
     CANDLE    SCRIPT    E1 yourid    EXTRNL    - -

1= Help      2= Refresh  3= Quit    4= Cancel    5= Sort(dir)  6= Auth
7= Backward  8= Forward  9= Alias  10= Stats    11= XEDIT/LIST 12= Cursor

====>

X E D I T  1 File

```

Figure 20. The FILELIST SHARE Screen

The SHARE display differs from the FILELIST STATS screen with the columns labeled Owner and Type. The Owner column lists the owner of each file or directory displayed. The Type column shows the type of item displayed, such as:

DIR

File control directory

DIRC

Directory control directory

MDISK

Minidisk

BASE

Base file

ALIAS

Alias

ERASED

Erased Alias

REVOKED

Revoked alias

BASE*

Migrated base file

ALIAS*

Migrated alias

EXTRNL

External object

Note: External objects are only displayed if you have also specified the ALLfile option.

The terms: base files, aliases, migration, and erased and revoked files will be discussed in upcoming sections.

If you see an asterisk (*) next to BASE or ALIAS, this means that this file has been migrated and is now located in DFSMS/VM-owned storage. These files may take longer than usual to reference. For more information about file migration, see [“DFSMS/VM and SFS File Management” on page 64](#).

External objects appear to be in the directory shown by FILELIST, but the data in these files is actually located outside of the directory, perhaps in another SFS file pool or in a database. These may be used by application programs.

For FILECONTROL directories, files contained in them, and external objects, the last two columns indicate whether you have read or write authority for each file or directory displayed. An X indicates that you have authority; a dash (-) means that you do not.

For DIRCONTROL directories, the last two columns indicate whether you have directory control read (DIRREAD) or directory control write (DIRWRITE) authority for the directory. For files and external objects within DIRCONTROL directories, the last two columns indicate whether you have DIRREAD or DIRWRITE authority for the parent directory. Regular read and write authorities cannot be granted on DIRCONTROL directories or the files within them.

A quick way to bring up the FILELIST SHARE screen is to enter the FILELIST command with the SHARE option. For more information, see [z/VM: CMS Commands and Utilities Reference](#).

The FILELIST SHARE screen provides several PF key functions that differ from the FILELIST STATS display.

Table 8. FILELIST SHARE PF Keys

Key	Meaning	Usage
PF6	Auth	This key processes the AUTHLIST command, displaying 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. For more information, see “Using the AUTHLIST Command” on page 81 .

Table 8. FILELIST SHARE PF Keys (continued)

Key	Meaning	Usage
PF9	Alias	This key processes the ALIALIST command to 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. For more information, see “Using the ALIALIST Command” on page 72.
PF10	Stats	This is the toggle to the FILELIST STATS 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 enrolled in your file pool, you can use the QUERY ENROLL command. A sample format of the command follows:

►► query — enroll — user — 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 enroll user for mary
```

Your output might look like this:

```
Userid    Enrolled
MARY      YES
```

To see a list of all the users currently connected to your file pool, enter one of the following:

```
query enroll user for all
```

or

```
query enroll user
```

This section provides information on how to share files with users enrolled in your file pool. However, you can also use the QUERY ENROLL command to determine which users are enrolled in other file pools. For more information, see [z/VM: CMS Commands and Utilities Reference](#).

DFSMS/VM and SFS File Management

You may want to ask your SFS administrator if *DFSMS/VM*, a storage management feature of z/VM, has been installed on your system, and is being used to manage SFS files, because this can affect the behavior of files in an SFS file pool.

DFSMS/VM provides the capability for the system to automatically manage SFS files and directories by assigning them to a management class. A *management class* is a set of attributes that define storage management criteria for the files. Files can be automatically deleted after a certain number of days, for

example, or moved to DFSMS/VM-owned storage. This can simplify SFS administration and enable more efficient use of storage.

File Migration

Some SFS files controlled by DFSMS/VM may appear to reside in your file pool, but actually reside in a storage repository managed by DFSMS/VM. (Note that migrated files still are considered by SFS to consume their usual amount of room in your file space.) These files are said to be in *migrated status* in your file pool. (Files in DIRCONTROL directories are never migrated.) You can identify these files by issuing FILELIST (SHARE or LISTFILE. Migrated files are shown with an asterisk in the Type column.

These files act exactly like regular SFS files, but they must be *recalled* (either automatically or explicitly) into your actual file pool before you can access the data. This may cause a delay, depending on your system configuration and workload. Automatic recall is governed by the SET RECALL command, while explicit recall is performed with the DFSMS RECALL command. If SET RECALL is ON (the default), recall will happen automatically when the file data is referenced. (If you wish, you can add the SET RECALL setting to your PROFILE EXEC.) If SET RECALL is OFF, the file will not be recalled. You will get an error indicating that the file is migrated and will not be implicitly recalled. You can enter the DFSMS RECALL command to explicitly recall the file in this case. For more information about SET RECALL, see [z/VM: CMS Commands and Utilities Reference](#), and for more information on the DFSMS RECALL, see [z/VM: DFSMS/VM Storage Administration](#).

A file does not need to be recalled unless you need to access the file data itself (for example, with the XEDIT command.) You may query or change a file's attributes (LISTFILE, FILELIST FILEATTR, GRANT AUTHORITY, and so forth) without recalling the file. The file also does not need to be recalled to be erased, or to have its data replaced with COPYFILE (REPLACE.

Automatic File Migration and Expiration

As mentioned above, DFSMS/VM can automatically cause SFS files to be migrated or *expired* (erased) at certain predetermined times, in order to free up DASD in your storage group. The criteria for file migration and expiration are established by your installation's storage management policies. File expiration criteria are usually related to how long the file has existed, or how long since it has last been referenced. The entire file may be erased, or only the data in the file. (Files in DIRCONTROL directories *may* be expired.) For more information about DFSMS/VM, see [z/VM: DFSMS/VM Storage Administration](#).

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 is 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 refer to a single file in more than one directory, or more than once in one directory. Aliases also let two different users refer to 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.

Aliases can be created only in FILECONTROL directories. They cannot exist in DIRCONTROL directories. You can, however, create an alias in a FILECONTROL directory that refers to a base file in a DIRCONTROL directory.

Aliases are useful for pointing to the same information from two directories, or from two different places within the same FILECONTROL 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, called PROJA. When Jim is assigned a second project, he creates a separate FILECONTROL 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, Project NEW. Internally, CMS ensures that all aliases point directly to a base file, regardless of how they are created.

To illustrate this point, after Jim created aliases with the following commands, the pointers would be set as shown in [Figure 21](#) on page 66.

```
create alias price list .proja ez prices .projez
create alias ez prices .projez new prices .projnew
```

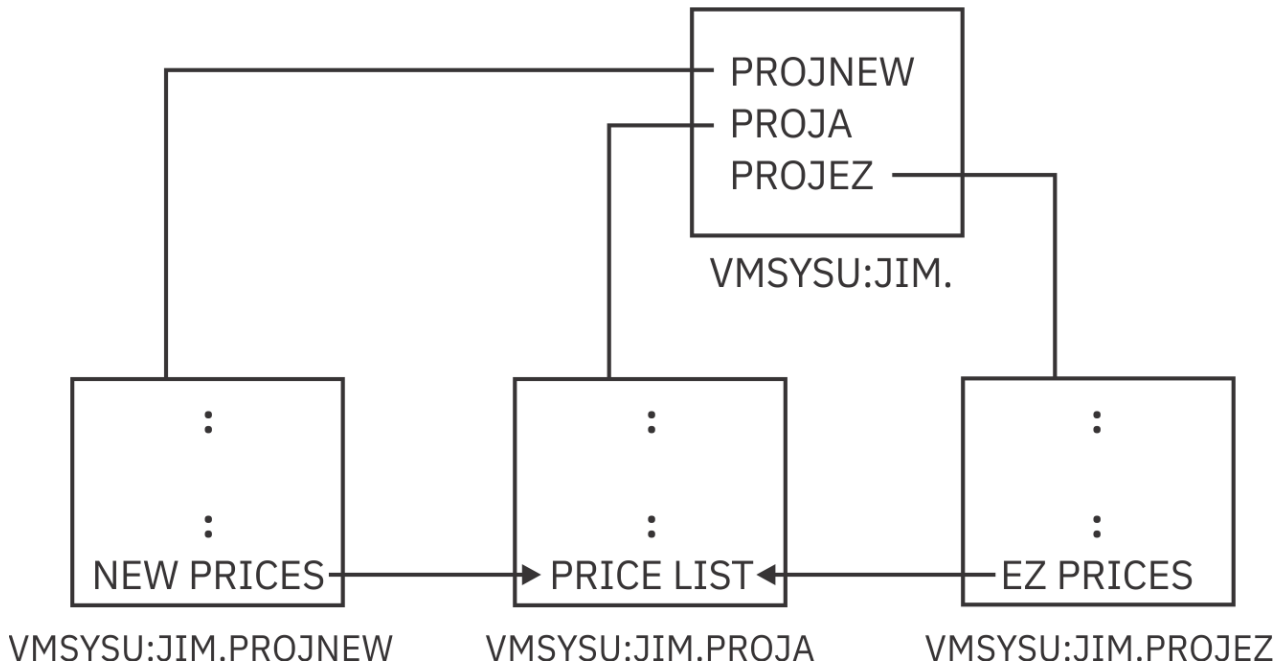


Figure 21. Example of Alias Pointers Between SFS Files

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 in one of his directories. The directory has the FILECONTROL attribute. 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 for the LOTS TODO file. Terry could then create an alias to the file in one of her own FILECONTROL 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 for one of her FILECONTROL 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 INVITE SCRIPT, THEME SCRIPT, and MUSIC SCRIPT, from the MAINT.SAMPLES directory to the .PARTY directory that you created, and accessed, with a file mode of D.

Suppose you also wanted to group your INVITE SCRIPT file in the directory with the files for your party favors. All of your directories have the FILECONTROL attribute, so you know you can create aliases in any of them. You decide to create an alias for the INVITE SCRIPT file in the .PARTY.TREATS.FAVORS directory by using the CREATE ALIAS command.

A sample format of the CREATE ALIAS command follows (for help with reading syntax diagrams, see [“Syntax, Message, and Response Conventions”](#) on page 20):

```
►► create — alias — fn1 — ft1 — dirid — fn2 — ft2 — dirid ►►
```

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. For more information on file mode numbers, see [“File Mode Letters and Numbers”](#) on page 117.

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 invite 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 INVITE SCRIPT file. You can do this by using the SEARCH option of the FILELIST command. Enter:

```
filelist invite script a (search
```

This command example tells SFS to display every occurrence of the INVITE 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 users' directories.

Your screen will look like the example shown in [Figure 22 on page 68](#).

```

yourid  FILELIST A0  V 355  2 Line=1 Col=1 Alt=0
Cmd    Filename Filetype Fm Directory Name
-      INVITE   SCRIPT  D1 VMSYSU:yourid.PARTY
      INVITE   SCRIPT  F1 VMSYSU:yourid.PARTY.FAVORS

1= Help      2= Refresh  3= Quit    4= Dirlist    5= Sort(name)  6= Auth
7= Backward  8= Forward  9= Alias  10= Filelist  11= XEDIT/LIST 12= Cursor

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

```

Figure 22. The FILELIST SEARCH Screen

The FILELIST SEARCH screen contains a column labeled `Directory Name` which 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.

INVITE SCRIPT appears in your .PARTY directory (currently accessed as D) and your .PARTY.TREATS.FAVORS directory (currently accessed as F).

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 * * A (SEARCH 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.

Your screen will look like the example shown in [Figure 23 on page 69](#).

```

yourid  FILELIST A0  V 355  T7 Line=1 Col=1 Alt=0
Cmd  Filename Filetype Fm Directory Name
-    INVITE    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
    INVITE    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/LIST 12= Cursor

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

```

Figure 23. Using FILELIST SEARCH to List All Your Files

To list all of your files, in all of your directories, enter the command FILELIST * * A (SEARCH. This will begin with your top directory (accessed as A) and list the files in the top directory and each directory below.

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 the keys listed in [Table 9 on page 69](#):

Table 9. FILELIST SEARCH PF Keys

Key	Meaning	Usage
PF4	Dirlist	Enters the DIRLIST command for the directory on the line where your cursor is located.
PF5	Sort(name)	Sorts the output currently displayed on your screen alphabetically by file name and file type.
PF10	Filelist	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.
PF11	XEDIT/LIST	Lets you XEDIT the file on the line where your cursor is located. 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.

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 referring to information in a file.

For example, the MAINT. top directory contains the MAINT.SAMPLES subdirectory. Within the MAINT.SAMPLES subdirectory is a file called GIFTS SCRIPT. If you have read authority to this file and to the MAINT.SAMPLES directory, you could access the directory to see what the GIFTS SCRIPT file contains. However, instead of accessing the entire directory, it may be easier and quicker to create an alias to the specific file you want to use.

Another reason to create an alias to another user's file is that it 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.SAMPLES 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 refer to the information in the file through one of your own FILECONTROL directories.

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.

For example, enter the following command to create an alias to the GIFTS SCRIPT file in the MAINT.SAMPLES directory and put it in your .PARTY.TREATS.FAVORS directory, which you previously accessed with a file mode of F (giving your alias the name PRIZES SCRIPT):

```
create alias gifts script maint.samples 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.SAMPLES 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.

```
yourid  FILELIST A0  V 108  3 Line=1 Col=1 Alt=0
Directory = VMSYSU:yourid.PARTY.TREATS.
Cmd  Filename Filetype Fm Format Lrecl  Records  Blocks  Date  Time
-    GAMES      F DIR      -      -      -      1/06/00 16:00:29
-    INVITE     SCRIPT  F1 V      29      13      1      1/06/00 15:56:04
-    PRIZES     SCRIPT  F1 V      16      6       1      1/06/00 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= XEDIT/LIST 12= Cursor
```

```
====>
```

```
X E D I T  1 File
```

Figure 24. 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 the example shown in [Figure 25 on page 71](#).

```
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 25. Using PF11 to XEDIT a File

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.

Notice the message at the top of the screen, warning you that you are not authorized to lock the file. For more information on file locking, see [“Locking Files and Directories”](#) on page 86.

From the FILELIST STATS screen, you can enter the FILELIST SHARE screen to find information about the GIFTS SCRIPT base file (PF10 from PRIZES SCRIPT line).

```
yourid  FILELIST A0  V 149  3 Line=1 Col=1 Alt=5
Directory = VMSYSU:yourid.PARTY.TREATS.
Cmd  Filename Filetype Fm Owner      Type      R W
-    GAMES      F      yourid  DIR       X X
-    INVITE     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= XEDIT/LIST 12= Cursor

```
=====>
```

X E D I T 1 File

Figure 26. 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 labeled Owner, because MAINT owns the base file.

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, write, DIRREAD, or DIRWRITE 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, INVITE 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 invite script .party
```

Your screen will look like this:

```
Directory = VMSYSU:yourid.PARTY
Filename  Filetype  Fm  T  Userid      Num  Filename  Filetype  Directory
INVITE    SCRIPT    D1  B  yourid       1  INVITE    SCRIPT    .PARTY.TREATS.FAVORS
```

The QUERY ALIAS display shows you have an alias to the INVITE 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, INVITE 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 invite script .party.treats.favors
```

This command would show you that INVITE 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 the ALIALIST Command

There is also another way to determine who has an alias to a file using one of the PF keys that appears on both the FILELIST SHARE and SEARCH screens.

For example, to see if there are any aliases to the INVITE SCRIPT file in your .PARTY directory, enter:

```
filelist invite script d (share
```

Your screen will look like the example shown in [Figure 27 on page 73](#).

```

yourid  FILELIST A0  V 149  1 Line=1 Col=1 Alt=0
Directory = VMSYSU:yourid.PARTY
Cmd  Filename Filetype Fm Owner  Type    R W
_    INVITE   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= XEDIT/LIST 12= Cursor

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

```

Figure 27. Using the PF Keys on the FILELIST SHARE Screen

Position your cursor on the line for the INVITE SCRIPT file and press PF9 to enter to the ALIALIST command. Your screen will look like the example shown in [Figure 28 on page 73](#).

```

yourid  ALIALIST A0  V 190  1 Line=1 Col=1 Alt=0
Base file = INVITE SCRIPT VMSYSU:yourid.PARTY
Userid   Num Filename Filetype Directory
yourid   1 INVITE   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 28. 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 INVITE 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, INVITE 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 INVITE 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.

[Table 10 on page 74](#) lists the following PF keys on the ALIALIST screen to let you sort the entries displayed:

Table 10. ALIAlIST PF Keys

Key	Meaning	Usage
PF4	Sort(type)	Sorts the files displayed, first by file type, then by file name.
PF5	Sort(name)	Sorts the files displayed, first by file name, then by file type.
PF6	Sort(dir)	Sorts the display alphabetically, first by directory name, then by file name, then by file type.
PF9	S(user)	Sorts the display alphabetically by user ID.

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 [z/VM: CMS Commands and Utilities 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.

Note: This is true unless the DATAONLY option was specified with ERASE. ERASE with DATAONLY erases only the data in a base file. All aliases and authorizations of the file remain intact, and its status is not shown as changed in FILELIST or QUERY ALIAS output.

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 CURTISJ erases the base file, when you enter the command:

```
filelist * * p (share
```

Figure 29 on page 74 displays what your screen should look like.

```
yourid  FILELIST A0  V 149  T3 Line=1 Col=1 Alt=0
Directory = VMSYSU:yourid.PARTY.PEOPLE.
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= XEDIT/LIST 12= Cursor

====>

X E D I T  1 File
```

Figure 29. Erased Indicator on the FILELIST SHARE Screen

You would see ERASED in the column labeled 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.PEO PTTEND
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.

Authorizing Others to Access Your Files and Directories

Users can share your files or directories if you grant them authority to do so with the GRANT AUTHORITY command. The kinds of authorities you can grant differ for FILECONTROL directories and DIRCONTROL directories. You can grant another user READ or WRITE authority for any base file contained in a FILECONTROL directory. You can also grant READ or WRITE authority for the FILECONTROL directory itself. And, you can grant NEWREAD or NEWWRITE authority for FILECONTROL directories. NEWREAD authority means that the user is automatically granted READ authority to any file added to the directory in the future. NEWWRITE authority means that the user is automatically granted WRITE authority to any future file. For more information on the GRANT AUTHORITY command, see [“Granting Authority” on page 78](#).

With DIRCONTROL directories, there are only two authorizations: directory control read (DIRREAD) authority and directory control write (DIRWRITE) authority. DIRREAD authority means that the user can read the directory, any file in the directory, and any file added to the directory in the future. DIRWRITE authority lets you write to the directory, its files, and its future files. You cannot grant authority for individual files within a DIRCONTROL directory.

Naturally, other users can also share any of their files or directories with you by granting the above authorities to 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 certain irrevocable authorizations. The authorizations vary with the kinds of objects you create.

When you create a FILECONTROL directory or files within FILECONTROL directories, you automatically have WRITE authority to those objects. (WRITE authority always includes READ authority.) When you create a DIRCONTROL directory, you automatically have DIRWRITE authority for that directory. (DIRWRITE authority always includes DIRREAD authority.) Files in DIRCONTROL directories have no individual authorizations—your ability to read from or write to those files is derived from your DIRWRITE authority for the parent directory.

These authorities give you control over your files and directories. You can, for example, read from or write to the file or directory, rename it, or relocate it. (You cannot, however, relocate files into or out of DIRCONTROL directories—no user can.) 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.

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.

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 with the REVOKE AUTHORITY command. For more information, see [“REVOKE AUTHORITY Command” on page 79](#).

It is important to remember that only the person who owns a file or directory can grant a user authority for it. (The only exception to this is that an SFS administrator can grant authority to another user. For more information, see [“Administrator Authority” on page 78](#). If, for example, you grant authority for one of your files to a user named Bill, he cannot grant that authority to someone else.

FILECONTROL Directory Authority

In this section, READ and WRITE authority to a file and a directory will be discussed, in addition to dynamic authorization, NEWREAD and NEWWRITE.

READ Authority for a File

READ authority for 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 section.) If that user tries to write to the file, however, SFS prevents the write and returns an error message.

READ Authority for a Directory

READ authority for a directory means that the user can read the names of objects within the directory (base files, aliases, subdirectories, and external objects). Users with READ authority can access the FILECONTROL directory in read-only status, or they can use Callable Services Library (CSL) routines in application programs to read files without accessing. 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. For more information about CSL, see [*z/VM: CMS Application Development Guide for Assembler*](#).

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 only READ authority for a directory, you will not automatically have READ authority for the files (objects) in the directory.

Note: You do, however, have READ authority for any external objects in the directory. Since external objects do not have authorities of their own, they inherit their parent directory's authority. You can only 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 for a File

WRITE authority for 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 for each individual file.

If you have WRITE authority for a file, you can lock the file in any mode (EXCLUSIVE, UPDATE, or SHARE).

WRITE authority always implies READ authority. You cannot have the ability to write to a file without having the ability to read it.

WRITE Authority for a Directory

WRITE authority for a directory gives the grantee a fair amount of authority over the contents of the directory. If you are granted WRITE authority for 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 at least READ authority for the file, and you cannot change, erase or rename any file, unless you have WRITE authority for 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 for 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 for 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 (SHARE, EXCLUSIVE, or UPDATE).

WRITE authority always implies read authority. You cannot have WRITE authority for a directory without implicitly having READ authority for that directory as well.

Dynamic Authorizations (NEWREAD and NEWWRITE)

SFS lets you grant authority for a FILECONTROL directory such that the grantee is automatically authorized for files added to the directory in the future. This ability, known as *dynamic authorization*, spares you from having to grant authority whenever you add a file to the directory.

There are two kinds of dynamic authorization: NEWREAD and NEWWRITE authority. When a base file is added to a directory, SFS determines whether any users have NEWREAD or NEWWRITE authority for the directory. The server then grants READ authority for the base file to any user having NEWREAD authority. Users with NEWWRITE authority are automatically granted WRITE authority for the base file.

The READ and WRITE authorizations that the server grants automatically are identical to those you grant yourself. It's as though the server issued a GRANT command for you. When you enter a QUERY AUTHORITY command, for example, authorizations that you granted are indistinguishable from those that the server granted for you.

All other rules regarding these READ and WRITE authorizations are the same. Suppose, for example, user Joe has read authority to TEST SCRIPT because you granted Joe NEWREAD authority for the directory prior to adding TEST SCRIPT to it. If you relocate TEST SCRIPT to another directory, Joe does not lose his READ authority for the file, even if the file is moved to a directory for which Joe has no authority. Remember that once SFS grants the authority, it is no different from you entering the command yourself.

When files are relocated into the directory from other directories, the server again automatically grants file authorities to those having NEWREAD or NEWWRITE authority for the directory. Moreover, the server retains the authorizations granted before the relocation (as it always does).

NEWREAD and NEWWRITE authority causes grants only on base files, not on aliases.

Both NEWREAD and NEWWRITE authority apply only to files that are added *after* the authority is granted. The authorities do not apply to any file that already exists in the directory. Consequently, when you are granting authority to a new user for the directory, you will probably want to enter at least two GRANT commands: one that grants authority for the existing files, and one that grants dynamic authorization for files added in the future. You might also want to grant READ or WRITE authority for the directory itself.

DIRCONTROL Directory Authority

In this section, DIRREAD and DIRWRITE authority to a file and a directory will be discussed, in addition to SFS administrator authority.

READ Authority for a File

You cannot grant READ authority to specific files in DIRCONTROL directories, only for files in FILECONTROL directories.

DIRREAD Authority for a Directory

You cannot grant READ authority for DIRCONTROL directories. Instead, you can grant DIRREAD authority.

DIRREAD authority applies only to DIRCONTROL directories. It lets the grantee read the directory and all files within the directory. It also lets the grantee read any file added in the future. Users with DIRREAD authority can access the DIRCONTROL directory in read-only status or they can use Callable Services Library (CSL) routines in application programs to read files without accessing. For more information about CSL, see [z/VM: CMS Application Development Guide for Assembler](#).

WRITE Authority for a File

You cannot grant WRITE authority to files in DIRCONTROL directories, only for files in FILECONTROL directories.

DIRWRITE Authority for a Directory

DIRWRITE authority applies only to DIRCONTROL directories. It lets the grantee read from, and write to, the directory and all files within it. It also lets the grantee read from, and write to, any file added to the directory in the future. DIRREAD authority and DIRWRITE authority are mutually exclusive. Users with DIRWRITE authority can access the directory in read-only or read/write mode. Or, they can use CSL routines to read and write files without accessing. For more information about CSL, see [z/VM: CMS Application Development Guide for Assembler](#).

Administrator Authority

The SFS 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 performs additional tasks first. For example, the administrator cannot create an alias for a user if that user does not already have authority for 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 [z/VM: CMS Commands and Utilities Reference](#).

Granting Authority

To grant another user authority, use the GRANT AUTHORITY command. A sample format of the command follows (for help with reading syntax diagrams, see [“Syntax, Message, and Response Conventions”](#) on page 20):

```
➔ grant — authority ————— dirid — TO ————— userid —▶
                   |         |
                   fn — ft   PUBLIC
```

Use PUBLIC to grant authority to everyone with access to your file pool.

To specify what type of authority you want to grant, you follow the GRANT AUTHORITY command with the options READ, WRITE, NEWREAD, NEWWRITE, DIRREAD, or DIRWRITE. READ authority is the default for FILECONTROL directories and the files that reside in them. DIRREAD is the default for DIRCONTROL directories.

For example, Craig is working on a project with Debbie. She needs to modify his file, SPECIAL PROJECT, which is in a FILECONTROL directory that Craig has accessed as Q. Because the file resides in a FILECONTROL directory, Craig can grant WRITE authority to that file as follows:

```
grant authority special project q to debbie (write
```

After Craig entered this command, Debbie would be able to read from and write 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, discussed in the following section.

REVOKE AUTHORITY Command

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 (for help with reading syntax diagrams, see [“Syntax, Message, and Response Conventions”](#) on page 20):

```
➤ revoke — authority ————— dirid — FROM — userid ➤
                    |         |
                    fn — ft
```

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 (keepread
```

This command specifies that Debbie's authority should be changed from write (which Craig originally granted) to read. Other options include:

KEEPNEWREAD

Changes a user's authorization for a FILECONTROL directory from NEWWRITE to NEWREAD.

KEEPDIRREAD

Changes a user's authorization for a directory control directory from DIRWRITE to DIRREAD.

NEWAUTH

Removes NEWREAD or NEWWRITE authority from a file control directory while retaining any READ or WRITE authority to that directory.

RWAUTH

Removes READ and WRITE authority from a file control directory while retaining any NEWREAD or NEWWRITE authority to that directory.

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 for 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, you would see:

```
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   - -
```

The word REVOKED in Type column 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* (ESM) 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 for 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 ————— dirid ➡
                        |
                        | fn — ft |
                        |
```

To determine who has authority for a specific file or files, specify the file name and file type, and the directory to be queried. To determine the authorities granted for a directory, specify only the directory identifier.

For example, 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 screen will look like this:

```
Directory = VMSYSU:yourid.PARTY.TREATS
Filename Filetype Fm Type   Grantee R W
CAKE      SCRIPT    E1 BASE   yourid X X
```

Your user ID is listed under the Grantee column and both the R and W columns 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 for 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.SAMPLES directory (to which you had read authority), the MAINT user ID does not maintain any authority for 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.

When you enter the QUERY AUTHORITY command for a directory, additional columns are displayed. For FILECONTROL directories, the following is displayed:

```
Directory = VMSYSU:IRA:MCDS
Grantee   R   W   NR   NW
IRA       X   X   X   X
JACK      X   -   X   -
MIKE      X   X   X   X
SUE       X   X   -   -
.         .   .   .   .
.         .   .   .   .
```

The NR and NW columns indicate whether you have NEWREAD or NEWWRITE authority for the FILECONTROL directory.

If the directory has the DIRCONTROL attribute, the following is displayed:

```
Directory = VMSYSU:IRA:NEWS
Grantee   R   W   DR   DW
IRA       X   X   X   X
JESSICA   X   X   X   X
MARY      X   -   X   -
MIKE      X   X   X   X
MORGAN    X   X   X   X
.         .   .   .   .
.         .   .   .   .
```

The DR and DW columns indicate whether you have DIRREAD or DIRWRITE authority for the DIRCONTROL directory.

Using the AUTHLIST Command

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 58, 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 for 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.

For example, enter the following command:

```
filelist cookies script e (share
```

[Figure 30 on page 82](#) displays what your screen should look like.

```
yourid  FILELIST A0  V 149  1 Line=1 Col=1 Alt=0
Directory = VMSYSU:yourid.PARTY.
Cmd    Filename Filetype Fm Owner    Type      R W
_      COOKIES  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= XEDIT/LIST 12= Cursor

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

Figure 30. Using the PF Keys on the FILELIST SHARE Screen

Positioning your cursor on the line for the COOKIES SCRIPT file and pressing PF6, would result in:

```
yourid  AUTHLIST A0  V 165  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 31. Entering the AUTHLIST Command

The information you receive is similar to the information you received when entering the QUERY AUTHORITY command.

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.

Table 11 on page 82 lists the following PF keys on the AUTHLIST screen that you can use to sort the information displayed on your screen:

Table 11. AUTHLIST PF Keys		
Key	Meaning	Usage
PF4	S(Grantee)	Sorts the files displayed alphabetically by grantee.

Table 11. AUTHLIST PF Keys (continued)

Key	Meaning	Usage
PF5	Sort(W)	Grantees with WRITE authority to the file are listed alphabetically, then other grantees are listed alphabetically.

When you enter an AUTHLIST command for a directory, you will see additional columns on the AUTHLIST screen. For FILECONTROL directories, these columns indicate whether a user has NEWREAD or NEWWRITE authority. For DIRCONTROL directories, the columns indicate DIRREAD or DIRWRITE authority. For more information on the AUTHLIST command, see [z/VM: CMS Commands and Utilities Reference](#).

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 ESM 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 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.

For example, 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
```

The Owner column shows you the owner of each base file, alias, or directory.

```
yourid  FILELIST A0  V 149  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 SMITH   ALIAS    X X
     FRIED   RICE     T1 WILLIAMS ALIAS    X -
     CHILI   BEANS    T1 yourid  BASE     X X
     PIZZA   SAUCE    T1 SIMKULET ALIAS    X -
     RECIPES DIR       T1 yourid  DIR      X X
     ICE     CREAM    T1 THOMPSON ERASED   - -

1= Help      2= Refresh  3= Quit    4= Cancel    5= Sort(dir)  6= Auth
7= Backward  8= Forward  9= Alias  10= Stats    11= XEDIT/LIST 12= Cursor

====>

X E D I T  1 File
```

Figure 32. Determining the Owner of a File

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 for 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. (For more information on using a NAMES file, see [z/VM: CMS Primer](#)).

Next, he could grant all the people within the MYDEPT group READ authority for AUTO4 ASSEMBLE. (Assume that the file resides in a FILECONTROL 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 for 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 for it or the base file.

Granting authority for an alias is the same as granting authority for 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 for one of their FILECONTROL 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 for 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 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 for a FILECONTROL 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 FILECONTROL directory and grant write authority for that directory to every member of your department. Eric could also grant read authority for 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 for 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 (by default, the directory will have the FILECONTROL attribute):

```
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 for the VMTOOLS directory, Eric would enter the following command:

```
grant authority .vmtools to toolprog (write newwrite
```

Eric would then grant read authority for the directory to everyone in the file pool:

```
grant authority .vmtools to public (read newread
```

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 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:

```
create alias projtrak module .goodstuff = = eric.vmtools
```

This would give all the users in the file pool read authority to the new tool, 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 log on. 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 simply 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 for that file. Eric would be the only one who could grant read authority for those files to the user community.

The techniques outlined previously work well for information that is updated frequently or at unpredictable times. If you have read-only data or infrequently updated data that you would like to share, consider using DIRCONTROL directories instead of FILECONTROL directories. DIRCONTROL directories provide better performance than FILECONTROL directories. For more information, see [“Using Directory Level Control”](#) on page 90.

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.

Locking Files and Directories

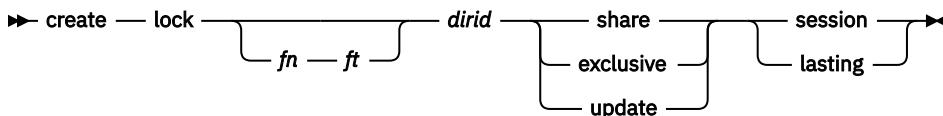
To share files and directories between users, you need to understand how CMS handles the simultaneous use of shared files. Also, you will need to know how to ensure that you and another user do not simultaneously edit a file and accidentally overwrite changes.

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 lock a file or directory with an *explicit* lock. (Except if it is an external object. These cannot be locked.) For example, you may want to perform a series of tasks on a file and not want anyone manipulating the file for a certain period of time. You can put an explicit lock on a file for the duration of your CMS session, or until you explicitly delete the lock.

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 (for help with reading syntax diagrams, see [“Syntax, Message, and Response Conventions”](#) on page 20):



You can create an EXCLUSIVE, SHARE, or UPDATE lock on a file in a FILECONTROL directory or on a FILECONTROL directory itself. You can create an UPDATE lock on a file in a DIRCONTROL directory. EXCLUSIVE and SHARE locks do not apply to files in DIRCONTROL directories. When creating a lock, you must also specify its duration: SESSION or LASTING. [Table 12 on page 87](#) describes the meaning of each lock as it applies to a file or directory:

Table 12. Lock Functions on Files and Directories

Lock Type	File	Directory
Exclusive	Other users cannot read or change the file. An EXCLUSIVE lock on a base file also prevents anyone from issuing any type of lock (SHARE, EXCLUSIVE, or UPDATE) on the file's parent directory. EXCLUSIVE locks cannot be created on files within DIRCONTROL directories.	Other users cannot read from or write to any of the base files or aliases in the directory. Also, they cannot 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. EXCLUSIVE locks cannot be created on DIRCONTROL directories.
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, no one can rename, relocate, or erase the file. A SHARE lock on a base file also prevents other users from issuing an EXCLUSIVE lock on the file's parent directory. SHARE locks cannot be created on files within DIRCONTROL directories.	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, can 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 SHARE lock prevents anyone from issuing an EXCLUSIVE or UPDATE lock on any base file, alias, or subdirectory within the directory. SHARE locks cannot be created on DIRCONTROL directories.
Update	Other users can read the file while you are reading or updating it. They cannot rename, relocate, or erase the file. An UPDATE lock on a base file also prevents anyone from issuing any type of lock (SHARE, EXCLUSIVE, or UPDATE) on the file's parent directory. UPDATE locks can be created on files within both DIRCONTROL and FILECONTROL directories. If locked, only the holders of the locks can access the directory in read/write mode and change files. Users who do not hold locks on files can access the directory only in read-only mode. However, you can lock a file in a DIRCONTROL directory that you have accessed in read-only mode.	For FILECONTROL directories, the person who locked the directory can read from or write to any files for which he is authorized. Other users can 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 cannot lock the directory in any mode; base files, aliases, and subdirectories can be locked only in SHARE mode. For DIRCONTROL directories, not even the person who holds the lock for the directory can access it in read/write mode.

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”](#) on page 88.

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 create only 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. If you have DIRWRITE authority to a directory, you can create UPDATE locks for the directory and on the files within it.

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 are affected by the lock. Thus, an alias that points to a file in a DIRCONTROL directory can have only an UPDATE lock put on it.

To allow others to read a file while you are updating it, you can create an UPDATE lock on the file or for 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 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.

You do not need to be concerned with locking files in DIRCONTROL directories when using XEDIT. In fact, XEDIT ignores the LOCK and NOLOCK options for files in DIRCONTROL directories. Rather than use locks for protection from unwanted updates, XEDIT depends on the access status. If you have a DIRCONTROL directory accessed in read/write status, you are the only user who can write to files in that directory—an UPDATE lock is not necessary. If you have the directory accessed in read-only status you cannot write to any file in that directory.

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 in effect, users with write authority can XEDIT the file with the NOLOCK option, but they cannot 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 ————— dirid ➡
 └── fn — ft ─┘

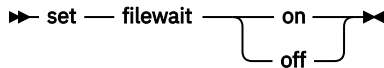
If Stan wanted to delete the UPDATE SESSION lock he placed on the BRUSHING TIPS file, he would enter the following command:

```
delete lock brushing tips a
```

To delete a lock for 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:



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 or directory. The request will wait until the required object becomes available or until you re-IPL or log off. 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.

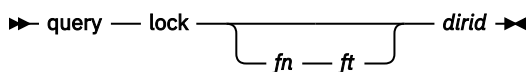
If the file or directory is explicitly locked, you will not wait, regardless of the SET FILEWAIT setting.

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:



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, a FILECONTROL 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 screen might look like this:

```
Directory = VMSYSU.yourid.PROBLEMS
Filename  Filetype Fm Type      Userid   Lock   Duration
PROBLEM   LOG          N1 BASE    ELLEN    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 incorrect 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 wanted to access, he would ask another user to enter the following command:

```
query filepool conflict crocketd poolq:
```

The result would look like this:

Requester	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 *Request* has requested a lock. The *Lock Type* column displays the type of lock that the *Request* 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 the lock will be deleted. If you choose not to continue waiting, enter:

```
#cp ipl 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.

Using Directory Level Control

Regular SFS processing allows control at a file level. You can, for example, grant authority to individual files. You can lock individual files. And you can see file changes made by other users as soon as they are committed; you do not need to reaccess directories to see changes as you do with minidisks. This level of control is referred to as *file level control*.

While file level control provides the most flexibility and concurrency, some applications do not need that level of control. In fact some applications that were coded to use minidisks may not work properly with file level control. Such applications might not, for instance, be able to cope with a locked file in a directory that is accessed in read/write mode, as this cannot occur on a minidisk.

Directory level control provides a level of control that is better suited to these applications. Directory level control provides, essentially, minidisk-like control of files within directories.

You get directory level control when you create the directory, or by using the DIRATTR command on existing directories.

In addition to minidisk-like function, DIRCONTROL directories offer performance benefits for users with XC virtual machines, if the server uses data spaces (see “Performance Benefits of Directory Control Directories” on page 92). Depending on the use of the directory, you might find that the performance benefit of a DIRCONTROL directory outweighs its functional restrictions.

An important fact to remember is that DIRCONTROL directories are intended for use with *read-only* information, or information that is infrequently updated. When the directory is changed, write activity should be confined to brief periods of time, preferably when concurrent read access is low. Frequent writing to a DIRCONTROL directory can degrade performance for read-only users, and can impair overall server performance. For directories with frequent write activity, it is strongly recommended that you use file level control, *not* directory level control.

The following sections describe the characteristics of DIRCONTROL directories. These characteristics involve:

- Authorizations
- Data consistency
- Concurrency
- Performance
- Other Functional Restrictions.

Authorizations for Directory Level Control

You cannot grant authority for individual files within a DIRCONTROL directory. Nor can you grant NEWREAD or NEWWRITE authority. Instead, *only* DIRREAD and DIRWRITE authority can be granted. As a directory owner, you have DIRWRITE authority for DIRCONTROL directories that you create.

DIRREAD authority lets the grantee read the directory and all files within the directory. It also lets the grantee read any file added in the future. Users with DIRREAD authority can access the directory in read-only mode, or they can use CSL routines to read files without accessing.

DIRWRITE authority lets the grantee read from and write to the directory and all files within it. It also lets the grantee read from and write to any file added to the directory in the future. Users with DIRWRITE authority can access the directory in read-only or read/write mode. Or, they can use CSL routines to read and write files without accessing.

This level of authorization control is similar to that of minidisks. If, for example, you allow another user to link to your minidisk in read/write mode, the effect is similar to granting DIRWRITE for a directory. That is, the user can read, write, and create files on the minidisk. Furthermore, the user can read from and write to any file that is created on the minidisk in the future. The authority lasts until the user is no longer allowed to link to your minidisk (perhaps you change the minidisk passwords).

Allowing a user to link to a minidisk in read mode is similar to granting DIRREAD authority for a directory. The user can read the entire minidisk and any files that are added to it.

The directory should contain read-only data or data that is rarely changed. If files in the directory are frequently updated, you should use file level control. With file level control, you still have the ability to grant authority for future files. (Use the NEWREAD or NEWWRITE options on the GRANT command.)

Data Consistency of Directory Control Directories

The data consistency of DIRCONTROL directories is also to that of minidisks. When you access a DIRCONTROL directory in read-only mode, you see a consistent view of the data until you reaccess the directory. If another user changes a file within the directory, you will not see the change until you reaccess. In FILECONTROL directories, on the other hand, you see file changes as soon as the other user commits the changes and you re-read the file.

When you access a DIRCONTROL directory in read/write mode, you see your own changes as you make them. This is identical with the way minidisks work.

It's important to remember that access-to-release consistency rules do not apply to aliases. Suppose you create an alias to a base file that resides in a DIRCONTROL directory. Then, you access the directory

containing the alias and the DIRCONTROL directory that contains the base file. If you use the alias to refer to the file, you will see the most current version of the file. Regular file control sharing rules apply.

If, however, you refer to the base file directly, you will see the version of the file that existed at the time the directory was accessed. That is, you will have access-to-release consistency. In fact, executing a FILELIST command for the base file and for the alias might show different sets of file statistics.

Another exception to access-to-release consistency is the *inplace* file. *Inplace* SFS files are files that have the INPLACE and NORECOVER attributes. For these files, SFS does not maintain access-to-release consistency. Nor does it try to immediately reflect all changes made to the file. While changed blocks may be seen, new blocks allocated to the file will not be seen until the user reaccesses the directory. So, while *inplace* files may reside in DIRCONTROL directories, they have little use outside of specialized applications.

Concurrency of Directory Control Directories

DIRCONTROL directories do not allow the same level of concurrency that FILECONTROL directories do. While read concurrency is the same, write activity is restricted. In particular, when a user has a DIRCONTROL directory accessed in read/write mode, no other user can write to the directory. If no one has the directory accessed in read/write mode, multiple users can concurrently write to different files within the directory by using CSL routines to directly refer to the files.

With FILECONTROL directories, on the other hand, any number of users can access the directory in read/write mode. Furthermore, these users can write to different files in the accessed directory at the same time. SFS ensures that no two users write to the same file at the same time.

Both DIRCONTROL and FILECONTROL directories let multiple users access the directory in read-only mode. Any number of users can be reading from a given file at the same time.

Another important concurrency difference between DIRCONTROL and FILECONTROL directories concerns updates of directories accessed in read-only mode. With FILECONTROL directories, some commands (such as XEDIT and COPYFILE) let you write to files even if the directory is accessed read-only. Applications that use CSL routines to directly reference files in FILECONTROL directories can write to files regardless of how the directory is accessed (so long as the user is authorized).

With DIRCONTROL directories, however, a read-only access prevents *all* writing activity. When you access a DIRCONTROL directory in read-only mode, there is no way for you to write to files in that directory. To write to files, you must access the directory in read/write mode. Or, if the command or application does not need the directory to be accessed, you can release the directory (that is, do not access it at all). When you do not have the directory accessed, you can write to files in it if no other user has the directory accessed read/write, or is currently writing to the files you need to write to.

The concurrency of DIRCONTROL directories is similar to that of minidisks. While it is possible to have several concurrent writers to a minidisk, the results are unpredictable. Minidisks, in effect, allow only one user to write to the minidisk at a time. Any number of users can access a minidisk in read-only mode.

Performance Benefits of Directory Control Directories

One compelling reason for using a DIRCONTROL directory is the potential performance benefit it provides. SFS delivers improved performance by using *data spaces*.

A *data space* is a separate area of virtual storage that can be directly addressed by your virtual machine. Your virtual machine does not need to communicate with the server to get file data. The use of data spaces can significantly improve performance of accessed DIRCONTROL directories.

Your file pool administrator controls which directories reside in data spaces. If you have a DIRCONTROL directory that you wish to have placed in a data space, you will need to ask your file pool administrator to do it. DIRCONTROL directories that are good candidates for data space residence meet these requirements:

- The directory should contain read-only data or data that is infrequently updated.
- If it is occasionally updated, write activity should be confined to brief periods of time, preferably when concurrent read access is low.

- Authorized users must access the directory *in read-only* mode to get the performance benefit.
- The directory must be *local* to the users accessing it for those users to get the performance benefit. (A data space cannot be referred to from another processor. CMS uses regular Advanced Program-to-Program Communication/Virtual Machine (APPC/VM) for remote file pools.)

Because your system has a limited number of available data spaces, your administrator will probably ensure that the directory meets all of the above requirements.

The read-only requirements are because of the data consistency SFS provides for DIRCONTROL directories.

For example, suppose users Cherie and Pam are accessing a directory in read-only mode. That directory happens to reside in a data space (refer to [Figure 33 on page 94](#)). If user Butch writes to a file in the directory, the data space is not changed (to provide data consistency to Cherie and Pam), but the data in the file pool *is* changed. SFS must now keep track of the version in the data space, and the data that Butch changed.

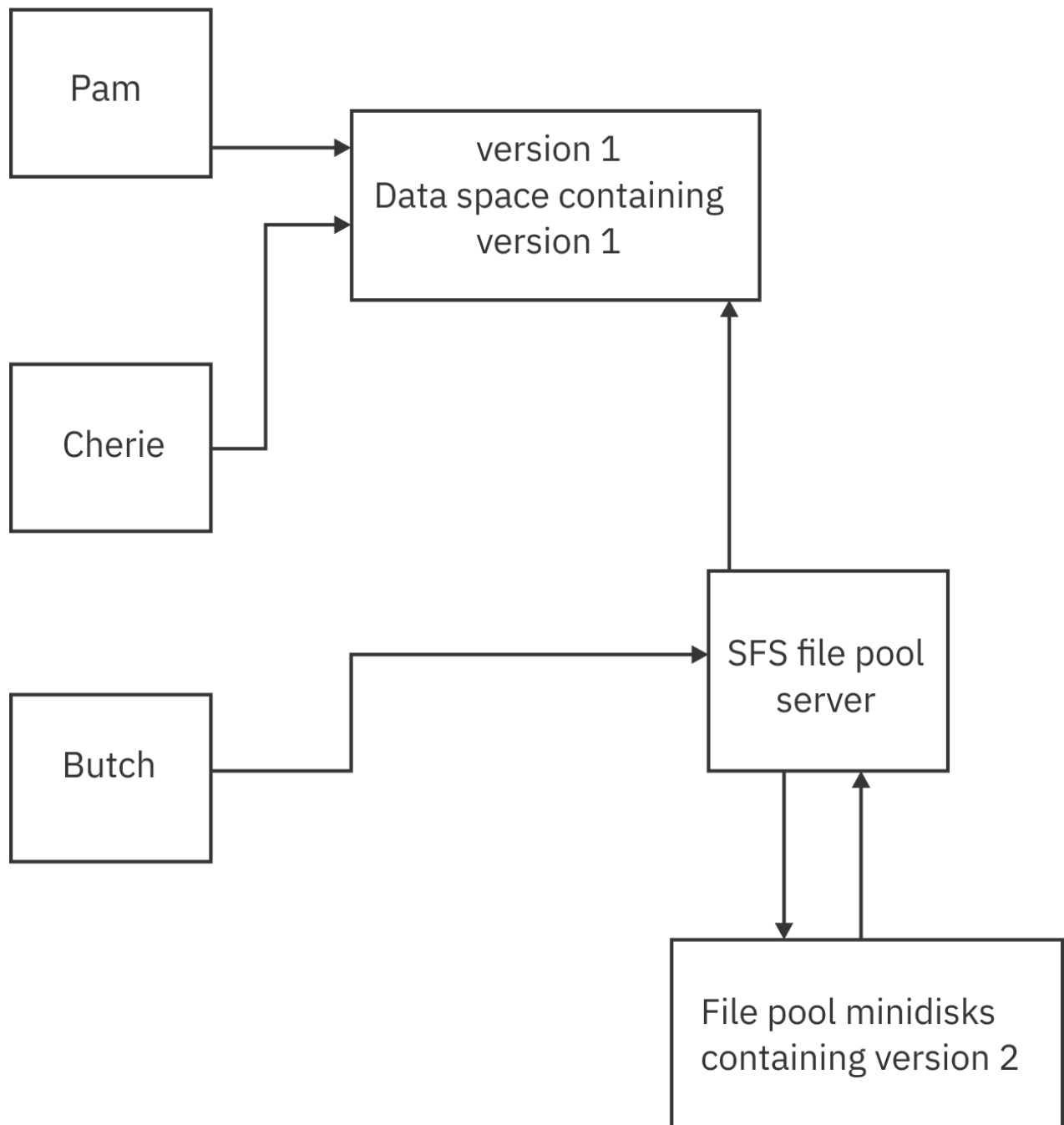


Figure 33. Using Directory Control Directories

Now suppose that Ed accesses the directory in read-only mode (refer to Figure 34 on page 95). If another data space is available, SFS loads the *changed* version of the directory into another data space and allows Ed to use it. SFS still maintains the original data space because Cherie and Pam still have it accessed. So, at this point, SFS is maintaining two data spaces for the directory. It also needs to keep track of the data that Butch changed.

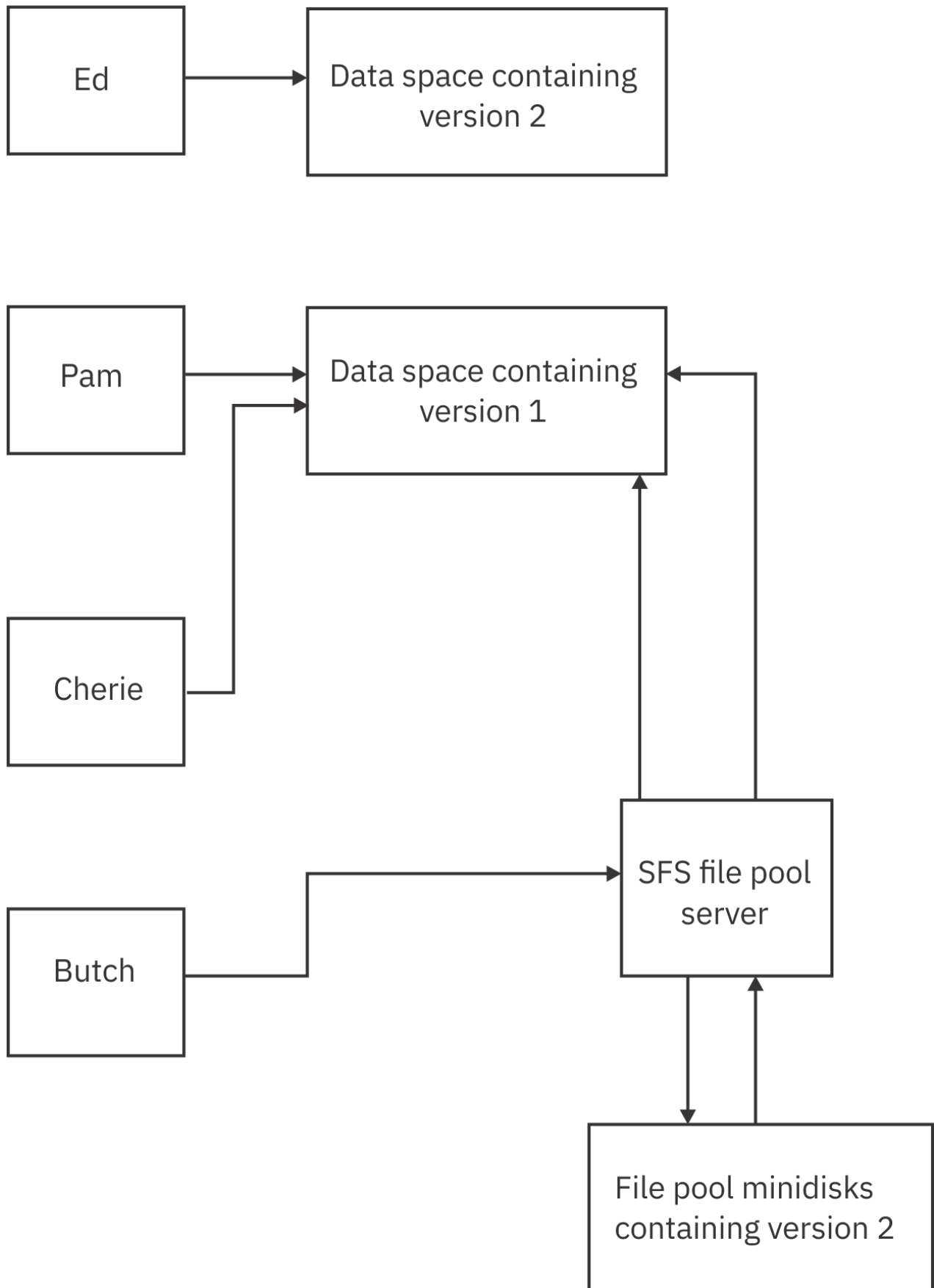


Figure 34. Using Directory Control Directories

Suppose Butch changes another file in the directory (refer to Figure 35 on page 96), creating a third version of the directory. If another user, Brad, accesses the directory, SFS loads yet another data space.

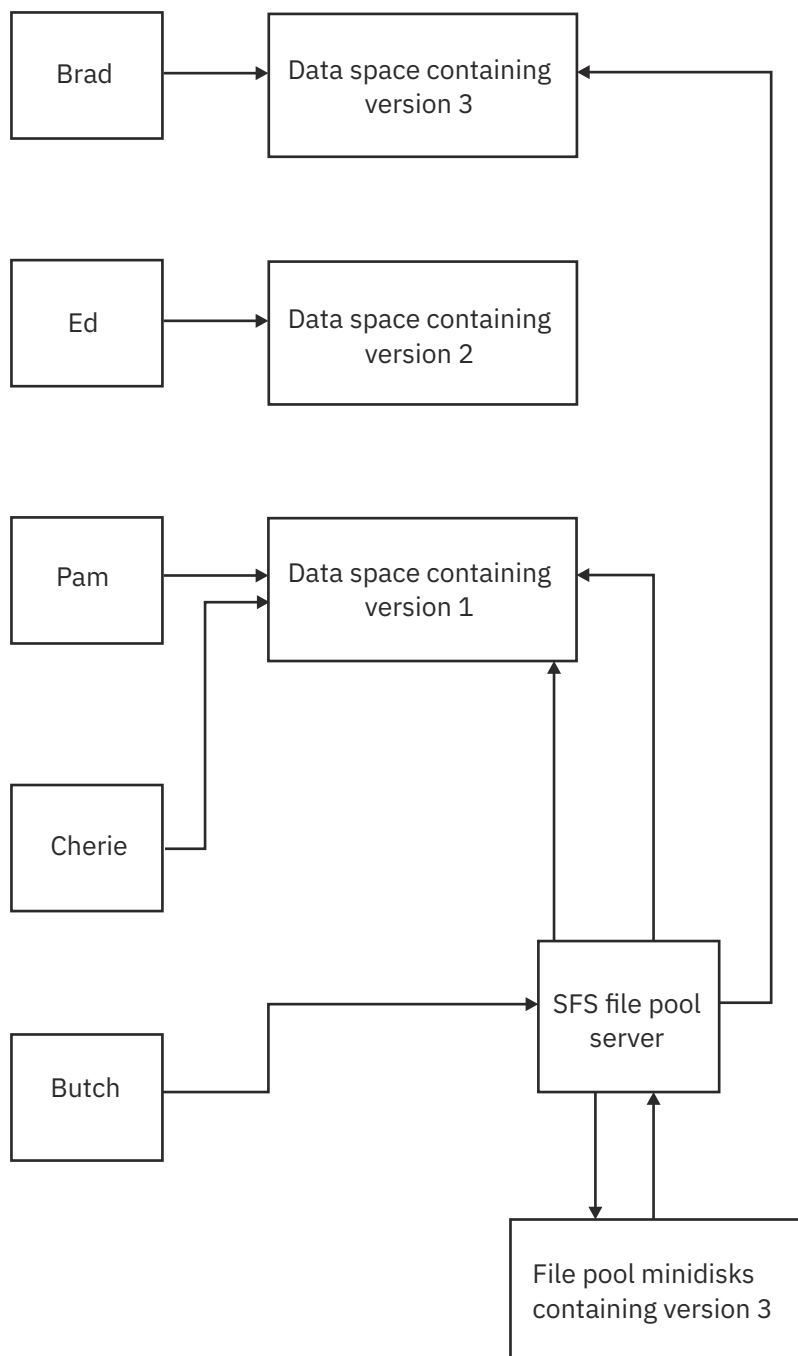


Figure 35. Using Directory Control Directories

SFS maintains each of the data spaces until all the users of a particular data space release or reaccess the directory. When all users of a back-level data space release any access of the directory, reaccess the directory it represents, or log off, SFS frees the data space for other uses. If, for instance, Cherie and Pam release the directory, the original data space would go away (assuming there were no other users of it). If Cherie happened to reaccess the directory, SFS would let her use the data space that had the most recent copy (the one Brad has accessed in our example).

It's also possible that Cherie and Pam neither reaccess the directory, nor release it all day. In this case, the server maintains the data space all day. The server does not force them to reaccess the directory, nor does it automatically *refresh* it. Remember that with DIRCONTROL directories, the server provides access-to-release consistency for read-only users at all times.

You can see how frequent writing activity on an often-used directory would create a lot of overhead in SFS processing. SFS would have to load data spaces and keep track of various levels of changed data. This processing overhead will reduce the performance gains, and the server could easily run out of data spaces. In this case, users who access the directory would use the SFS server processing—almost all performance benefits would be lost.

To get the most benefit from data space use, you should schedule all updates during periods of low directory use. You should make the updates in as brief a time as possible, and then *request all users of the directory to reaccess it*. You can determine who has your directory accessed by using the QUERY ACCESSORS command. While not all users will heed your advice, you can minimize the chance of the server running out of data spaces by sending the message.

File pool administrators can determine the number of versions of a directory that SFS is maintaining by using the QUERY ACCESSORS command.

Another requirement for DIRCONTROL directories in data spaces is that users must access the directory in read-only mode to get the benefit. Users who access in read/write mode do not use the data space. Nor do users who use CSL routines to refer to the directory without accessing it. Because CSL routines can refer to SFS files directly, some applications may not require the directory to be accessed. These applications will not use the data space and, consequently, will not get the performance benefit. The directory should be accessed before the application is started. Then, even though the application uses direct reference, the data space will be used (assuming the directory is local).

Another important point to remember is that aliases never use data spaces. Suppose, for example, you create an alias to a base file in a DIRCONTROL directory. The alias resides in one of your directories, and both directories are accessed. If you use the alias to refer to the base file, you will not use the data space. Instead, the server will process your request. If, however, you refer to the base file directly, you will use the data space. So, using the alias will give you the most current version of the file, which is the version the server maintains, but will not give you the benefits of the data space.

If your DIRCONTROL directory meets the requirements for use in a data space, you need to ask your file pool administrator to make the directory *data space eligible*. The administrator will probably verify that your directory does, indeed, meet the requirements for data space residence and enter a DATASPACE command for the directory. (Only users with file pool administration authority can use the DATASPACE command.)

Once the DATASPACE command is processed, your directory is said to be *data space eligible*. The next time a local user accesses your directory in read-only mode, the file pool server will load the directory into a data space if one is available. If a data space is not available, regular server processing is used.

So, even if your directory is data space eligible, there is no guarantee that it will be loaded into a data space. If your file pool administrator has correctly estimated the number of data spaces needed for the file pool, the odds are that one will be available. In any case, you might ask your administrator to enter a QUERY ACCESSORS command if you suspect that your directory is seldom placed in a data space (there is no noticeable performance improvement). The administrator can tell whether write activity on some other directory in a data space has caused all data spaces to be used.

Once a directory is made eligible for use in a data space, that eligibility persists (even across server shutdowns) until one of these things happens, the:

- Administrator explicitly removes eligibility by entering a DATASPACE RELEASE command.
- Directory is erased.
- Directory is changed from directory control to file control.

Other Functional Restrictions of Directory Control Directories

In addition to the characteristics described above, DIRCONTROL directories have these restrictions:

- UPDATE is the only type of explicit lock that the file is in.
- A file (or external object) cannot be the source or target of a RELOCATE command.
- The directory cannot contain aliases.

Example of a High-Performance Directory Control Directory

Suppose you work as a securities analyst in a brokerage firm. Your department analyzes businesses and creates a CMS file for each firm on which you have an opinion. Every day or two, your department completes a few new opinions. All the stockbrokers in your firm read the reports and advise their clients of your opinions. They typically read the files while discussing investment options on the telephone with clients, so they need good access performance.

These opinion files are good candidates for placement in a high-performance DIRCONTROL directory. Because opinions are generated every day or two, the directory would not have frequent write activity. Furthermore, you could schedule the updates to occur near the end of the business day when there is little read activity. Your department members might, for example, create CMS files in their own subdirectories while they are forming an opinion. When their analysis is complete, they would wait until the end of the business day and copy the file to the DIRCONTROL directory.

During the day, your company's numerous stockbrokers would keep the directory accessed in read-only mode. They would benefit from having the opinions in a high-speed directory because it would reduce delays when they are retrieving an opinion for a busy client.

While the example being used is for a brokerage firm, the same requirements for high-speed directories are satisfied by many situations: electronic bulletin boards, forum directories, document libraries, and so on. To illustrate how you would set up a high speed directory, we will continue with the brokerage example.

First, you need to create a DIRCONTROL directory. To do so, specify the DIRCONTROL option on a CREATE DIRECTORY command:

```
create directory vmsysu:yourid.opinions (dircontrol
```

If the directory already exists, you can convert it to a DIRCONTROL directory by doing the following:

1. Ensure that all explicit locks for the directory and its files are deleted.
2. Then enter: `dirattr vmsysu:yourid.opinions dircontrol (force`

The FORCE option revokes all individual authorizations you've granted for the directory and the files within it. Remember that individual authorizations are not allowed in DIRCONTROL directories. The FORCE option also erases all aliases from the directory. (Aliases cannot exist in a DIRCONTROL directory.)

You can verify that the directory is directory control by entering:

```
query dirattr vmsysu:yourid.opinions
```

If it is a DIRCONTROL directory, you will see:

```
DIRCONTROL
```

Otherwise you will see:

```
FILECONTROL
```

Next, you would grant DIRWRITE authority to your department. Assuming you have created a nickname, SA, for the securities analysis department, you would enter:

```
grant authority vmsysu:yourid.opinions to sa (dirwrite
```

DIRWRITE authority allows your department members to create new files or update existing files in the directory. It also allows them to write to any file that is added to the directory in the future.

Next, grant DIRREAD authority to all the stockbrokers and anyone else who can use your computer system:

```
grant authority vmsysu:yourid.opinions to public
```


DIRREAD is the default for GRANT AUTHORITY, when a DIRCONTROL directory is specified.

Finally, call the file pool administrator and ask to have your directory made data space eligible. The administrator will enter a DATASPACE ASSIGN command for your directory, and your stockbrokers will enjoy high-speed opinions.

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 [z/VM: CMS Application Development Guide](#).

- Many system facilities and applications, such as the DASD Dump Restore (DDR) Service Program, 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 it. In these cases, the facility may receive a nonzero return code and terminate, because it cannot use a file it needs.
- If you are running an application program without Coordinated Resource Recovery (CRR) and it fails, it could be a result of accessing directories in several 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. For more information on CRR, see [z/VM: CMS File Pool Planning, Administration, and Operation](#).

Using Several File Pools at One Time

It is possible for you to be enrolled in more than one file pool. If this is the case, you will have a top directory in each file pool. Your administrator might have set you up one of these file pools as your default file pool. 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
```

Sue can abbreviate the commands by omitting her user ID if she has not entered a SET FILESPACE command. CMS will assume she wants to access the top directories.

After these commands are processed, 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: Assuming your administrator has defined a default file pool for you, you can reset your current default to the one originally defined for you by entering:

```
set filepool primary
```

PRIMARY is a keyword. It means the default file pool in effect at the time of IPL.

Sharing Files with Users on Other Systems

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 that 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.

SFS lets you share your files with users on other systems. For example, you can share files with users in another building, another town, or state.

The Transparent Services Access Facility (TSAF) is a component of z/VM 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 z/VM systems that can share resources.

Inter-System Facility for Communications (ISFC) is a component of z/VM that handles program-to-program communications between VM systems and LAN-based domain controller workstations. A Communication Services (CS) collection is a group of one or more of these domains.

To share files owned by another user in a different file pool, or on a different system in the same TSAF or CS 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 for 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 or CS collection, your system administrator will need to use APPC/VM VTAM Support (AVS). For more information, see [*z/VM: CMS File Pool Planning, Administration, and Operation*](#), and [*z/VM: Connectivity*](#).

Chapter 4. Storing Your Files on Minidisks

z/VM uses disk storage, which can either be a directory (within the SFS file space) or a minidisk. This section contains information specific to minidisks. For more information on how to manage your files if your files are only stored in an SFS file space, see [Chapter 3, “Using the Shared File System,”](#) on page 37.

Minidisks and How They Are Defined

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

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 three types of minidisks: *permanent minidisks*, *temporary minidisks (T-disks)*, and *virtual disks in storage*.

Permanent minidisks

last across terminal sessions (logons); they are defined in the z/VM directory entry for your virtual machine.

Temporary minidisks

are automatically destroyed at logoff. You can define temporary minidisks for your own virtual machine by using the CP DEFINE command, or they can be attached to your virtual machine by the system operator.

Virtual disks in storage

are temporary simulations of minidisks in system storage; they are not allocated on a real DASD. Virtual disks in storage are defined in the z/VM directory entry for your virtual machine, or you can define them using the CP DEFINE command.

All three types of 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. You enter the CP DEFINE command with the device type of the storage you wish, the amount of cylinders or blocks you require, and the virtual device number you would like it assigned to.

Before entering the CP DEFINE command, you may wish to use the CP QUERY DASD command to see what virtual device numbers you are currently using. Entering:

```
query dasd
```

will result in a display of information similar to the following:

```
DASD 0191 3380 USG017 R/W      20 CYL  ON DASD 0954 SUBCHANNEL = 0003
DASD 0192 3390 USG01M R/O     120 CYL  ON DASD 0565 SUBCHANNEL = 0000
DASD 019E 3390 SYGEMC R/O     200 CYL  ON DASD 0541 SUBCHANNEL = 000A
```

The virtual device numbers currently being used are 191, 192 and 19E. In the CP DEFINE command example which follows, a virtual device number of 291 will be used.

The following command allocates a 10-cylinder temporary minidisk from a 3380 device and assigns it a virtual device number of 291:

```
define t3380 as 291 cyl 10
```

When complete, the system will return messages similar to the following:

```
DASD 291 DEFINED
Ready; T=0.15/1.39  11:21:07
```

You could check that your new temporary minidisk has been defined by reentering the CP QUERY DASD command:

```
DASD 0191 3380 USG017 R/W      20 CYL  ON DASD  0954 SUBCHANNEL = 0003
DASD 0192 3390 USG01M R/O     120 CYL  ON DASD  0565 SUBCHANNEL = 0000
DASD 019E 3390 SYGEMC R/O     200 CYL  ON DASD  0541 SUBCHANNEL = 000A
DASD 0291 3380 (TEMP) R/W      10 CYL  ON DASD  07D1 SUBCHANNEL = 0013
```

When you define a minidisk, you can choose any valid virtual device number that is not already assigned to a device in your virtual machine. Valid device numbers for minidisks are 0001 through FFFF.

For more information on the DEFINE or QUERY DASD command, see [z/VM: CP Commands and Utilities Reference](#).

Defining Virtual Disks in Storage

A virtual disk in storage is another type of temporary minidisk that you can attach to your virtual machine for the duration of a terminal session. However, because a virtual disk in storage is allocated from system storage instead of on a real DASD, and so avoids the I/O overhead of writing to the DASD, it may be faster to use than a regular temporary minidisk.

Use the CP DEFINE command to create a virtual disk in storage. A virtual disk in storage is always defined as an FBA minidisk, which is allocated in 512-byte blocks. You do not need to have a real FBA DASD on your system. For example, the following command allocates a 160-block virtual disk in storage and assigns it a virtual device number of 333:

```
define vfb-512 as 333 blk 160
```

For more information, see [z/VM: CP Commands and Utilities Reference](#).

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 and virtual disks in storage that you have defined with the CP DEFINE command. When you enter the FORMAT command, you must use the virtual device number you have defined for the minidisk and assign a file 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 by typing either a number or a word (YES or NO):

```
YES
```

CMS then asks you to assign a label for the minidisk, which you select. A valid label is considered to be any combination of one to six numeric (0-9) and alphabetic (A-Z) characters. However, be aware that the use of fewer than six characters will cause blanks to be filled in place of the missing rightmost characters. And, the use of more than six characters results in only the first six characters from the left being used. 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:

```
scratch
```

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
```

After the format of the disk is complete, entering the command:

```
query disk
```

will return a display similar to the following, including the newly formatted disk:

LABEL	VDEV	M	STAT	CYL	TYPE	BLKSIZE	FILES	BLKS	USED-(%)	BLKS	LEFT	BLK	TOTAL
DL0191	191	A	R/W	20	3380	2048	227		4554-84		846		5400
SCRATC	291	C	R/W	10	3380	4096	0		6-00		1494		1500
IDT00L	192	E/A	R/O	120	3390	1024	1797		55424-93		3976		59400
YDISK	19E	Y	R/O	200	3390	4096	1443		28661-80		7339		36000

The QUERY DISK command will be discussed further in section [“Managing Your Minidisks”](#) on page 106.

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.

For more information on the FORMAT or QUERY DISK commands, see [z/VM: CMS Commands and Utilities Reference](#).

Linking and Sharing Minidisks

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.

Since only one user can own a minidisk, and there are many occasions that require users to share data or programs, z/VM lets you share minidisks, on either a permanent or temporary basis, by *linking*.

Virtual disks in storage that are defined in the directory, although they are temporary, can be shared among users. A shareable virtual disk in storage exists from when the first user links to it until the last user detaches it or logs off. Because of its temporary nature, however, a shareable virtual disk in storage should not be used for permanent data.

Permanent links can be established for you in your z/VM 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.

Coding in the file of access For example, if you have a program that uses data that resides on a minidisk identified in the user ID INFO's configuration as 194, and you know that the password assigned to this minidisk is GO, you could enter the command:

```
link info 194 as 198 r pass= go
```

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 INFO in any event, you can reenter the LINK command using the access mode RR:

```
link info 194 198 rr go
```

Notes:

- Using the RR access lets one user read a minidisk while another is updating it at the same time. This can produce unpredictable results.
- The password cannot be entered on the command line if the password suppression facility is active.
- If RACF® or another external security manager is installed on your system and is in effect, the procedure for linking may be changed somewhat. For more information, see your system administrator.

Once successfully completed, INFO's 194 minidisk is then added to your virtual machine configuration at virtual device number 198.

To verify the command has been processed, you can enter the CP QUERY DASD command, which will result in a display similar to this:

```
DASD 191 3390 USG017 R/W      20 CYL  ON DASD  0541 SUBCHANNEL = 0003
DASD 198 3390 USG01M R/O      120 CYL  ON DASD  0565 SUBCHANNEL = 0001
```

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.

For complete information and options available for the CP LINK command, see [z/VM: CP Commands and Utilities Reference](#).

Accessing Minidisks

Once minidisks are linked, they can be accessed (assigned a file mode). In [“Accessing Your Directories or Minidisks”](#) on page 126, we discussed the reasons for accessing a minidisk or a directory. This section discusses how to display the minidisks you currently have access to, and how to assign a file mode to a minidisk virtual device number.

You may recall in [“Formatting Minidisks”](#) on page 102, an example was used where a new temporary disk was assigned a file mode as part of the formatting command. As part of the FORMAT, in addition to the assignment, the disk contents were erased. When linking to your own minidisks, or another user's minidisk, you would, generally, like to keep the information already on the minidisk. Therefore, the FORMAT command would not be applicable.

You can assign a file mode using the ACCESS command. A sample format of the ACCESS command follows:

```
➤ access — virtual device address — fm ➤
```

Before using the ACCESS command, you may want to use the QUERY ACCESSED command to determine the file modes (minidisks) you currently have access to, for example:

Mode	Stat	Files	Vdev	Label/Directory
A	R/O	765	191	DL0191
S	R/O	1321	190	CMS6.0
Y/S	R/O	337	19E	19ESP6

The information displayed is organized as follows:

- Mode indicates the mode letter used to access the minidisk.
- Stat gives the status of the minidisk: R/W (read/write) or R/O (read-only).
- Files displays the number of files on the minidisk.
- Vdev displays the virtual device number of the minidisks.
- Label/Directory shows the label assigned to the minidisk when it was formatted.

If you had already linked to a minidisk whose virtual device number is 194, you could access it with a file mode of B by entering:

```
access 194 b
```

Releasing and Detaching Minidisks

When you no longer need a minidisk that you temporarily accessed, you can release it by entering the RELEASE command. For example:

```
release b
```

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

When you log off, any minidisk that you temporarily accessed is automatically released.

You can also specify that the disk be detached from your virtual machine configuration by using the DETACH option of the RELEASE command. For example:

```
release b (detach or release 194 (detach
```

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

When you detach a virtual disk in storage or log off, if you are the last user of the virtual disk in storage, it is destroyed.

For more information on the RELEASE command, see [z/VM: CMS Commands and Utilities Reference](#).

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 SFS.) 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.



Attention: 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 QUERY LINKS command to determine if other users have links to any minidisks you want to access. For more information, see [z/VM: CP Commands and Utilities Reference](#). If you are using the Shared File System, files can be locked and unlocked to prevent simultaneous updates. For more information, see [“Locking Files and Directories”](#) on page 86.

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 implicitly releases the minidisk.



Attention: Any RELEASE attempted after a minidisk has been detached, whether it be implicit or explicit, cannot perform equivalent cleanup of open files as a RELEASE issued before the DETACH. Therefore, to ensure that all open files are cleaned up when detaching a CMS minidisk, issue the RELEASE command with the DETACH option, or issue the RELEASE command before issuing the CP DETACH command.

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.

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   3390  1024     171    1221-92    107    1328
```

Term	Description
LABEL	Identifies the label assigned to the disk when it was formatted.
VDEV	Identifies the virtual device number.
M	Identifies the access or file mode letter.
STAT	Identifies whether disk status is read/write or read/only.
CYL	Identifies the number of cylinders available on the disk.
TYPE	Identifies the device type of the disk.
BLKSIZE	Identifies the CMS disk block size when the minidisk was formatted.
FILES	Identifies the number of CMS files on the disk.
BLKS USED	Identifies the number of CMS disk blocks in use. The percentage of blocks in use is also displayed.
BLKS LEFT	Identifies the number of disk blocks left.
BLK TOTAL	Identifies the total number of disk blocks.

When a minidisk is becoming full, you should erase whatever files you no longer need, pack inactive files using the COPYFILE command with PACK option, 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 process 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.

Data Compression

You can save data in a compressed format to conserve DASD resources. The CSRCMPSC macro provides a pair of services that compress and expand data.

Compression takes an input string of data and, using a data area called a *dictionary*, produces an output string of compression symbols. Each symbol represents a string of one or more characters from the input.

Expansion takes an input string of compression symbols and, using a *dictionary*, produces an output string of the characters represented by those compression symbols.

To use the Data Compression Services for compression and expansion, the CSRCMPSC macro uses two dictionaries: the compression dictionary and the expansion dictionary. These dictionaries are logically and physically related. When you expand the data that has been compressed, you want the result to match the original data. Thus the dictionaries are complementary. When the compression is being done, the expansion dictionary must immediately follow the compression dictionary, because the compression algorithm examines entries in the expansion dictionary.

To help you use the compression services, the S-disk contains the following compiled REXX execs:

- CSRBDICV for building compression and expansion dictionaries
- CSRCMPEV to run a test, to compress and re-expand files using dictionaries created by the CSRBDICV EXEC. Reports are generated giving statistics on the efficiency of the compress and expand functions with the current dictionary set.

For more information on how to use these execs, see *z/VM: CMS Commands and Utilities Reference*. For more information about compression and using the execs, see *z/VM: CMS Application Development Guide* and *Enterprise Systems Architecture/390 Data Compression*.

Chapter 5. More on the CMS File System

This section is an extension of [Chapter 2, “CMS File System,” on page 27](#). It contains more detailed information on CMS files and commands, such as:

- Additional considerations when naming and storing your CMS files.
 - What are Reserved File Types?
 - How are File Mode Letters and Numbers used? For SFS, for minidisks?
- What are Synonyms and Translations and how are they used?
- CMS command search order and execution characteristics.

Many CMS command names are listed in the section; many have not been discussed or even mentioned previously in this document. For a full description on any command, see [z/VM: CMS Commands and Utilities Reference](#).

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 your files, but caution should be used if they are not being used for the specific CMS function usually associated with that file type.

Some reserved file types also have special significance to XEDIT. 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.

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 13 on page 109](#) lists the file types reserved for use by CMS commands, in addition to some special file types reserved for use by the language processors, which are IBM licensed programs.

Table 13. Reserved File Types

File Type	Command/ Environment	Usage	File Name	RECFM	LRECL	Comments
AMSERV	AMSERV	Input	<i>fn</i>	F	80	Input control statements for Access Method Services
ASM3705	ASM3705	Input	<i>fn</i>	F	80	3705 assembler source statements
	GEN3705	Output	<i>fn(nn)</i>	F	80	
ASSEMBLE	ASSEMBLE	Input	<i>fn</i>	F	80	Assembler language source statements
	VHFHASM	Input	<i>fn</i>	F	80	
	HASM	Input	<i>fn</i>	F	80	
AUTOSAVE	XEDIT	Input	<i>fn</i>	F	80	Copy of an edited file

Table 13. Reserved File Types (continued)

File Type	Command/ Environment	Usage	File Name	RECFM	LRECL	Comments
AUXxxxx	UPDATE XEDIT	Input	<i>fn</i>	F	80	List of file types used for update files
		Input	<i>fn</i>	F	80	
BASDATA	BASIC execution	Execu- tion time files	<i>fn</i>	V	255	
BASIC	BASIC	Input	<i>fn</i>	V	156	BASIC language source statements
C	CC	Input	<i>fn</i>	F	80	C source statements
CMSUT1	AMSERV	Inter- medi- ate work files	<i>fn</i>	F	121	
	COPYFILE		COPYFILE	V/F	ANY	
	DISK LOAD		DISK	V/F	ANY	
	DOSLIB		<i>fn</i>	V	1024	
	EDIT		EDIT			
	ESERV		<i>fn</i>	F	80	
	EXECUPDT		X\$EUPD\$X	V/F	ANY	
	EXPAND		CMSUT1			
	GENCMD		COMMANDS			
	HELP		HELP			
	HELPCONV		HELPCONV			
	INCLUDE		DMSLDR			
	IUCV		CMSCOMM			
	LANGMERG		'Appl_ID'NLS			
	LOAD		DMSLDR			
	MACLIB		DMSLBM			
	READCARD		READCARD	V/F	80	
	RECEIVE		RECEIVE	V/F	ANY	
	SENDFILE		SENDFILE	V/F	ANY	
	TAPE LOAD		TAPE	V/F	ANY	
	TAPPDS		TAPPDS			
	UPDATE		<i>fn</i>	V/F	ANY	
	VMFPLC2		TAPE	V/F	ANY	
	LOAD					
	XEDIT		XEDIT			
	ZAPTEXT		CMSUT1			
CMSUT2	ESERV	Work	ESERV			
	GENMOD LANGMERG		COMMANDS 'Appl_ID'NLS			
CNTRL	UPDATE XEDIT	Input	<i>fn</i>	F	80	List of update files
COBOL	COBOL	Input	<i>fn</i>	F	80	COBOL source statements
	COBOL2	Input	<i>fn</i>	F	80	

Table 13. Reserved File Types (continued)

File Type	Command/ Environment	Usage	File Name	RECFM	LRECL	Comments
COPY	MACLIB	Input	<i>fn</i>	F	80	COPY control cards and macro definitions A book from a DOS/VS source library
	SSERV	Output	<i>fn</i>	F	80	
CSLCNTRL	CSLGEN	Input	<i>fn</i>	V F	65535 2 ³¹⁻¹	Routine names, TEXT files, template files, and other CSL control files that are to be used in building the library
CSLLIB	CSLGEN	Output	<i>fn</i>	V	65535	Callable services library (CSL), generated by CSLGEN, for use on DASD
	CSLLIST	Input/ Output	<i>fn</i>			
	RTNLOAD	Input	<i>fn</i>			
CSLSEG	CSLGEN	Output	<i>fn</i>	V	65535	Callable services library (CSL), generated by CSLGEN, for use in a logical saved segment
	CSLLIST	Input/ Output	<i>fn</i>			
	RTNLOAD	Input	<i>fn</i>			
DCSSMAP	LANGGEN	Output	NLxy x-level id y-lang id	V	100	Output file from LANGGEN command
DIRECT	DIRECT	Input	<i>fn</i>	F	80	User directory entries
DLCS	GENCMD	Input	<i>fn</i> xxxSPAacc xxxUPAcc	F	80	Source statements for command syntax definition and synonym files
DOSLIB	DOSLIB	Input	<i>fn</i>	V	1024	CMS/DOS phase library
	DOSLKED	Input	<i>fn</i>	V	1024	
	FETCH	Output	<i>fn</i>	V	1024	
	GLOBAL	Input	<i>fn</i>			
DOSLNK	DOSLKED	Input	<i>fn</i>	F	80	Linkage editor control statements for input to CMS/DOS linkage editor
ESERV	ESERV	Input	<i>fn</i>	V/F	ANY	Input control statements for ESERV program
EXEC	EXEC	Input	<i>fn</i>	V	130	EXEC statements CMS EXEC and \$LISTIO EXEC files should not be shared.
	EXEC 2	Input	<i>fn</i>	V	256	
	GEN3705	Output	<i>fn</i>	F	80	
	LISTFILE	Output	CMS	F	80, 88	
	LISTIO	Output	\$LISTIO	F	80	
	REXX	Input	<i>fn</i>	V	2 ³¹⁻¹	
EXPAND	EXPAND	Input	<i>fn</i>	V/F	2 ³¹⁻¹	Control records that expand object files

Table 13. Reserved File Types (continued)

File Type	Command/ Environment	Usage	File Name	RECFM	LRECL	Comments
FORTRAN	FORTGI	Input	<i>fn</i>	V	80	FORTRAN source statements
	FORTHX					
	FORTVS2	Input	<i>fn</i>	V/F	2 ³¹⁻¹	
	GOFORT					
	TESTFORT					
FREEFORT	GOFORT	Input	<i>fn</i>	V	81	FREEFORM FORTRAN source statements
FT ⁿⁿ F001	FORTRAN execution	Input/ Output	<i>fn</i>	V/F	2 ³¹⁻¹	User input and output files
GCS	EXEC	Input	<i>fn</i>	V	130	EXEC statements
GLOBALV	GLOBALV	Input/ Output	Initial	F	80	Collection of named variables
	DEFAULTS		Session Lasting	V V	=<520 =<520	
GROUP	GROUP	Output	<i>fn</i>	F	80	Group Control System (GCS) data block entries used to describe a GCS virtual machine group
HELPxxxx	HELP	Input	<i>fn</i>	V	79	Input files for HELP Facility, Where: xxxx specifies the name of the component to which the file belongs. For more information on file types reserved for HELP, see Chapter 9, “Tailoring the HELP Facility,” on page 199.
\$HLPxxxx	HELPCONV	Output	<i>fn</i>	V	79	Output files for HELPCONV command Where: xxxx specifies the component to which the file belongs.
LANGGCTL	LANGGEN	Input	<i>fn</i>	V/F	2 ³¹⁻¹	Control file for LANGGEN command
LANGLIST	GENCMD GENMSG LANGMERG LANGGEN SET LANGUAGE VMFNLS	Input	VMFNLS	F	ANY	Contains CMS NLS country code and language ID mapping

Table 13. Reserved File Types (continued)

File Type	Command/ Environment	Usage	File Name	RECFM	LRECL	Comments
LANGMAP	LANGMERG	Output	<i>ay</i> <i>a-appl id</i> <i>y-lang id</i>	V	2 ³¹⁻¹	
LANGMCTL	LANGMERG	Input	<i>fn</i>	V/F	2 ³¹⁻¹	Control file for LANGMERG command
LISTCPDS	PRINT	Input	<i>fn</i>	V/F	ANY	File used for printing on a page printer
LISTING	AMSERV	Output	<i>fn</i>	F	121	Processor output
	ASSEMBLE	Output	<i>fn</i>	F	121	Processor output
	ASM3705	Output	<i>fn</i>			Processor output
	CC	Output				Processor output
	COBOL	Output	<i>fn</i>	V	121	Processor output
	COBOL2	Output				Processor output
	ESERV	Output	<i>fn</i>	V	ANY	Processor output
	FORTGI					
	FORTHX					
	FORTVS2	Output				Processor output
	GENMSG	Output	<i>fn</i>			Processor output
	GOFORT					
	LOADLIB	Output	<i>fn</i>			Processor output
	PLIC	Output				Processor output
	PLIOPT	Output	<i>fn</i>			Processor output
	TESTCOB	Input	<i>fn</i>			
	VSPASCAL	Output	<i>fn</i>			Processor output
LIST38PP	PRINT	Input	<i>fn</i>	V/F	ANY	File used for printing on 3800 model 3 page printer
LIST3800	PRINT	Input	<i>fn</i>	V/F	ANY	File used for printing on 3800 printer
LIST3820	PRINT	Input	<i>fn</i>	V/F	ANY	File used for printing on 3820 page printer
LKEDIT	LKED	Output	<i>fn</i>	F	121	Listing
LOGFILE	SET LOGFILE	Output	<i>fn</i> <i>vscreen name</i>	V V	132 80	
LOADLIB	LKED	Output	<i>fn</i>	F	≤260	3705 control program load modules
	ZAP	Input	<i>fn</i>			
	GLOBAL	Library				
LSEG	SEGEN	Input	<i>fn</i>	V/F	2 ³¹⁻¹	Logical segment definition file
LDTFD11	AMSERV	Work				
LDTFD12	AMSERV					
LSEGMAP	SEGEN	Output	<i>fn</i>	F	100	Logical segment load map output from SEGEN command

Table 13. Reserved File Types (continued)

File Type	Command/ Environment	Usage	File Name	RECFM	LRECL	Comments
MACLIB	GLOBAL	Library	<i>fn</i>	F	80	Macro definitions (dictionary and members)
	MACLIB	Input/ Output	<i>fn</i>	F	80	
	MACLIST	Input/ Output	<i>fn</i>	F	80	
MACRO	MACLIB	Input	<i>fn</i>	F	80	Macro definitions
	ESERV	Output	<i>fn</i>	F	80	
MAP	DOSLIB	Output	<i>libname</i>	F	24	Library map
	DOSLKED	Output	<i>fn</i>	F	121	VSE linkage editor map
	DSERV	Output	DSERV	F	120	Directory information from VSE libraries
	INCLUDE	Output	LOAD	F	100	Module map
	LOAD	Output	LOAD			Module map
	MACLIB	Output	<i>fn</i>			Library map
	TXTLIB	Output	<i>fn</i>	F	80	Library map
	TAPE	Output	<i>fn</i>			List of files dumped, loaded, scanned, or skipped
MDMPxxxx	DUMPLD2	Output	<i>fn</i>	F	4096	Segmented dump files
MEMO	XEDIT	Input/ Output	<i>fn</i>	V/F	≤132	For documenting program notes or writing reports
MODULE	DOSLKED	Input	<i>fn</i>	V	ANY	Nonrelocatable or relocatable Executable file
	GENMOD	Output	<i>fn</i>	F	80	
	LOADMOD MODMAP	Input Input	<i>fn</i> <i>fn</i>	V	65535	
NAMES	NAMEFIND NAMES	Input/ Output	<i>userid</i>	V	255	Information about users in communication
NETLOG	RECEIVE SENDFILE	Logging	<i>userid</i>	V	255	Records logging transmission of files sent or received
				V	255	
NOTE	NOTE	Input/ Output	<i>userid</i>	V	132	Creates a note to be sent to others
NOTEBOOK	RECEIVE SENDFILE	Input	<i>userid</i>	V	132	Notes sent or received by you
				V	132	
PASCAL	VSPASCAL	Input	<i>fn</i>	V	72	PASCAL source statements
PLI or PLIOPT	PLIOPT PLIC	Input Input	<i>fn</i> <i>fn</i>	F	80	PL/I source statements
PROC	PSERV	Output	<i>fn</i>	F	80	A procedure from the DOS/VS procedure library

Table 13. Reserved File Types (continued)

File Type	Command/ Environment	Usage	File Name	RECFM	LRECL	Comments
PSEG	SEGGEN	Input	<i>fn</i>	V/F	2 ³¹⁻¹	Physical segment definition file
PSEGMAP	SEGGEN	Output	<i>fn</i>	F	100	Physical segment load map output from SEGGEN routine
REPOS	GENMSG	Input	xxxMEScc xxxUMEcc	F	80	Source statements for message repositories
REXX	PIPE		<i>fn</i>			Contains stage commands and REXX statements
RPGII	RPGII	Input	<i>fn</i>			RPGII source statements
RTABLE	PROP	Input	<i>fn</i>	V	72	Routing table for Programmable Operating Facility
SCRIPT	SCRIPT	Input	<i>fn</i>	V	132	Input to SCRIPT processor
SYMDMP	FCOBOL	Output	<i>fn</i>	V	512	DOS/VS COBOL DEBUG file for SYMDMP option
SYNONYM	SYNONYM	Refer- ence	<i>fn</i>	F	80	
SYSUT1	ASM3705	Work	<i>fn</i>	F	8000	Created by the assembler and language processors for use as temporary workfiles.
	ASSEMBLE	Work	<i>fn</i>			
	DOSLKED	Work	<i>fn</i>	F	248	
SYSUT2	ASSEMBLE	Work	<i>fn</i>	F	13312	Created by the assembler and language processors for use as temporary workfiles
	COBOL	Work	<i>fn</i>			
	DOSLKED	Work	<i>fn</i>	F	1024	
SYSUT3	ASSEMBLE	Work	<i>fn</i>	F	13312	Created by the assembler and language processors for use as temporary workfiles
	COBOL	Work	<i>fn</i>			
	LKED	Work	<i>fn</i>			
	PLIOPT	Work	<i>fn</i>			
SYSUT4	COBOL	Work	<i>fn</i>	F	80	Used as input to TESTCOB. Created by the assembler and language processors for use as temporary workfiles.
	LKED					
	PLIC					
	PLICR					
	TESTCOB	Input		F	512	
SYSUT5	COBOL	Output	<i>fn</i>	F		Debug file
SYSUT6	COBOL	Work	<i>fn</i>	F		Work file
TEMPLATE	CSLGEN	Input	<i>fn</i>	F	80	Template information for parameters of CSL routines
TESTCOB	TESTCOB	Output	<i>fn</i>			COBOL workfile
TESTFORT	TESTFORT	Output	<i>fn</i>	VB	125	Processor printed output

Table 13. Reserved File Types (continued)

File Type	Command/ Environment	Usage	File Name	RECFM	LRECL	Comments
TEXT	ASSEMBLE	Output	<i>fn</i>	F	80	Object code
	ASM3705	Output	<i>fn</i>			3705 source code and JCL statements
	CC	Output	<i>fn</i>			Object code
	COBOL	Output	<i>fn</i>			Object code
	COBOL2	Output	<i>fn</i>			Object code
	DOSLKED	Input	<i>fn</i>			Object code
	FORTGI	Output	<i>fn</i>	F	80	Object code
	FORTHX	Output	<i>fn</i>			Object code
	FORTVS2	Output	<i>fn</i>			Object code
	GEN3705	Output	<i>fn(Ln)</i>			Linkage editor, control statements for 3705 control programs
	GOFORT	Output	<i>fn</i>			Object code
	INCLUDE	Input	<i>fn</i>			Object code
	LKED	Input	<i>fn</i>			Object code
	LOAD	Input	<i>fn</i>			Object code
	PLIC	Output	<i>fn</i>			Object code
	PLICR	Input	<i>fn</i>			Object code
	PLIOPT	Output	<i>fn</i>			Object code
	RSERV	Output	<i>fn</i>			Relocatable module file
	SET					
	LANGUAGE	Work	SET\$TEMP			Temporary work file
TXT <i>langid</i>	GENCMD	Output	<i>fn</i>	F	80	Object code for language files
TXTLIB	GLOBAL	Library	<i>fn</i>	F	80	
	TXTLIB	Output	<i>fn</i>	F	80	Object decks (dictionary and members)
UPDATE	UPDATE	Input	<i>fn</i>	F	255	UPDATE control statements
	XEDIT	Output				
UPDLOG	UPDATE	Output	<i>fn</i>	F	271	UPDATE log
UPDTxxxx	UPDATE	Input	<i>fn</i>	F	255	UPDATE control statements
VS BASIC	VS BASIC	Input	<i>fn</i>	F	≤256	VS BASIC language source statements
VSBDATA	VSBDATA	Execution time files	<i>fn</i>	V	140	
XEDIT	XEDIT	Input	<i>fn</i>	V	255	Exec/XEDIT statements
ZAP	ZAP ZAPTEXT	Input	<i>fn</i>	F	80	Control records that modify or dump files

File Types for Output Files: TEXT and LISTING For Example

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 Work Files

Temporary work files are those files which are listed as *Work* in the *Usage* column of Table 13 on page 109 (CMSUT1 and SYSUT1 are examples). These files are created during the execution of a command, program, or function. Space is allocated for these files on an as-needed basis. They are erased when processing is complete. If a program, command, or function you are executing is terminated before completion, these *workfiles* can remain on your minidisk or SFS directory. These files are reserved for system usage, and use of these files may cause unpredictable results, and should not be shared. You can erase them.

File Types for Documentation

There are three CMS reserved file types for which XEDIT accepts (by default) uppercase and lowercase input data. These are:

File Type	Description
MEMO	Specifies documenting program notes or writing reports.
NOTE	Specifies when sending notes to others.
SCRIPT	Specifies the SCRIPT command. This command calls a text processor that is part of the IBM Document Composition Facility licensed program.

File Mode Letters and Numbers

The file mode field of a CMS file identifier has these characters:

File Mode	Description
Letter	Is established by the ACCESS command and specifies the directory or minidisk where a file resides: A through Z.

File Mode	Description
Number	Specifies 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. File mode numbers have different meanings for SFS and minidisks; this will be discussed later in this section.

How File Mode Letters Are Used

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 an SFS 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.

You load CMS with the IPL command (short for Initial Program Load). 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.

Note for SFS Users

It is possible that you could be enrolled in a file pool and also be able to use minidisk storage. The status of the file pool server determines what is accessed as file mode A. (The *file server* is a device or computer which accesses and retrieves files for all users.) In this case, if your top directory was accessed with a file mode of A, and the file pool server is down, access is dropped. CMS will try to access your 191 minidisk with a file mode of A. The directory cannot be accessed if the file server is down. 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. (File mode numbers will be discussed later in this section.) Therefore, the access commands for these file modes are:

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

In addition, at CMS IPL time the following special rules hold for the formatting and accessing of a minidisk defined as virtual device number 192. If 192 is:

- An unformatted temporary minidisk or virtual disk in storage, CMS formats it and accesses it as file mode D.
- A CP-formatted temporary minidisk or virtual disk in storage, CMS reformats it for CMS use and accesses it as file mode D.
- A CMS-formatted temporary minidisk, virtual disk in storage, or permanent minidisk that is accessed as a file mode other than D, CMS reaccesses it as file mode D.
- An unformatted or CP-formatted permanent minidisk, CMS does not automatically format, reformat, access, or reaccess it.

When CMS accesses a 192 minidisk as file mode D, any minidisk or SFS directory already accessed as D is released.

The actual letter 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 alphabetic: 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 QUERY SEARCH command.

How Extensions Are Used

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.

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
```

For SFS, accessing a directory as read-only restricts writing to files in that directory for all commands that use file modes, even if you are properly authorized to write to the files. The only exceptions to this are for directories with the file control (FILECONTROL) attribute. With FILECONTROL directories, you can create an alias to a file (with the CREATE ALIAS command), and write to the file using the COPYFILE and XEDIT commands, even if you have the directory accessed in read-only status. Of course, you must have the proper authorization to write to the file. To have XEDIT and COPYFILE respect the read-only access, use the SET RORESPECT ON command.

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 more information on locking files, see [“Locking Files and Directories” on page 86](#).

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 re-enter 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. For more information on the search order for specific commands, see [z/VM: CMS Commands and Utilities Reference](#).

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 duplicate exists on 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 is reestablished. For example, if you enter:

```
access 191 a
access 198 c/a
release a
```

file mode C (virtual address 198) is still accessed. Note that if you enter:

```
query disk c
```

virtual address 198 still appears accessed as C/A. Since file mode A is no longer accessed, files on the 198 minidisk can only be referred to by using file mode C, not A. The fact that it still appears accessed as C/A merely indicates that if you subsequently access another directory or minidisk as A, the parent/extension relationship is again in effect, and you can again refer to the files on the 198 minidisk using either file mode A or C.

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 that CMS 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 process any of these commands or process 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 that you 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 searches 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	GRANT AUTHORITY	RELOCATE
CREATE LOCK	LISTFILE	RENAME
DELETE LOCK	QUERY ALIAS	REVOKE AUTHORITY
ERASE	QUERY AUTHORITY	
FILELIST	QUERY LOCK	

You must explicitly enter the file mode if you want to use these commands to process files that are on extensions.

Note: Note that ERASE and RENAME only operate on file modes accessed as read/write if a file mode letter is used.

These 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. For more information on these commands, see [Chapter 3, “Using the Shared File System,” on page 37](#).

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 do not allow you to specify the file mode. They write output files 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, such as the RENAME or COPYFILE commands. For a complete list, see [“Commands Used to Assign File Mode Numbers” on page 126](#).

To change the file mode number using COPYFILE, the REPLACE option is used. For example:

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

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

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


Other commands used to change file mode numbers are covered in the section, [“Commands Used to Assign File Mode Numbers”](#) on page 126.

Notes for SFS Users

- 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. You cannot specify file mode numbers on commands that operate on an entire minidisk or SFS directory, because file mode numbers are attributes of files, and not SFS directories or minidisks. For example, do not use file mode numbers on such commands as ACCESS, DIRLIST, RELEASE, and RELOCATE, which require a directory identifier, *dirid*, unless you are specifying a file or set of files.
- When using the RENAME command, the COPYFILE command with REPLACE option, or the XEDIT subcommand FFILE, 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.
 - 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.
 - 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.
 - 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.
 - FFILE with 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.

File mode numbers can be used in CMS commands to identify or operate on subsets of files. For example, to list only those files in an accessed directory or minidisk 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
```

In SFS, file mode numbers are not useful accessing subsets of directories. You can only 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 the same way you can with minidisks (using the ACCESS command). The proper way to access a subset of files is to create a subdirectory with those files in it, and then access the subdirectory.

File Mode Numbers in SFS

Following is a description of file mode numbers 0 through 6 for SFS directories. For more information on minidisks see, [“File Mode Numbers for Minidisks”](#) on page 124.

File Mode Number 0

File mode number 0 has no special meaning for SFS. In a minidisk environment, file mode number 0 makes the files private. Because all SFS files are private unless you explicitly grant authority to someone, file mode number 0 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. Users who can read from your SFS directories can see the names of the files within that directory, regardless of the file mode number.

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 processed 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.

For more information about OS simulated data sets, see [z/VM: CMS Application Development Guide for Assembler](#).

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

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

File Mode Number 6

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

Note: If you want your files to have the INPLACE attribute (*update-in-place*), which means that the existing records of a file are written back to their previous location on the file pool rather than in a new slot, you can create or change the extended file attributes for those files. For more information on manipulating extended file attributes, see [z/VM: CMS Application Development Guide](#).

File Mode Numbers 7 Through 9

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 usual 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 [z/VM: CMS Application Development Guide for Assembler](#).

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. To take advantage of the *update-in-place* capability, the existing file must be modified, not replaced. Many CMS commands, such as XEDIT, COPYFILE, and RECEIVE, will usually replace the file.

Attention: When modifying an existing file mode number 6 file, it is possible to damage the file, or even the entire minidisk on which it resides. This damage occurs when the following conditions are met:

- Some of the updates made to the file or disk by an application are updated in place.
- CMS terminates (requiring a re-IPL of CMS) before it can write all of the data to disk.

For a variable format file, *update-in-place* applies only if a record is replaced by a record of the same length. For more information on the file mode 6 update-in-place attribute and data integrity, see [z/VM: CMS Application Development Guide for Assembler](#).

File Mode Numbers 7 Through 9

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

Commands Used to Assign File Mode Numbers

Table 14 on page 126 lists the commands you can use to assign file mode numbers:

Table 14. Commands Used to Assign File Mode Numbers	
Command	Usage
COPYFILE	When you create a new file with the COPYFILE command, or change the file mode of an existing file or module with the COPYFILE command with REPLACE option.
DISK LOAD	When you load a file or files from the spooled card reader.
DLBL, FILEDEF	When you assign file definitions to files for programs or CMS command functions.
GENMOD	When a module file is created.
NETDATA RECEIVE	When you request that a current read spool file be processed, and that file is in NETDATA format.
READCARD	When you specify a file identifier with the READCARD command or on a READ control card.
RECEIVE	When receiving a file from your virtual reader.
RENAME	When you use RENAME to change the file mode number. For SFS, see notes in section “How File Mode Numbers Are Used” on page 122.
SORT	When you use the SORT command and specify file mode in output file ID.
XEDIT	When you create a file with the editor or change the file mode number of an existing file using FILE, SAVE, or SET FMODE subcommands. For SFS note, see “How File Mode Numbers Are Used” on page 122.

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. If your files are stored on minidisks and are not *permanently linked* you must link to the minidisks before you can access them. *Permanent links* are those which have been established for you through your z/VM directory entry. These minidisks are then a part of your virtual machine configuration every time you log on.

Note: For more information on linking additional minidisks, for example, temporarily to another user's minidisk, see [“Linking and Sharing Minidisks”](#) on page 103.

By accessing directories and establishing a file mode letter for them, you can 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 process the command.

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

- Which directories or minidisks are to contain the new files that you create.
- Whether you can write on a minidisk or whether you can only read from it (its read/write status).
- The command search order for programs executing in your virtual machine. (For more information, see [“File Mode Letters and Numbers”](#) on page 117.)

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

Note: For a detailed *column-by-column* description, see [“Getting Started - Accessing Your Top Directory”](#) on page 39 for SFS, or see [“Accessing Minidisks”](#) on page 104 for minidisks.

The disk accessed as mode A, your *A-disk*, is your *default disk*. If you use a CMS command and do not specify a mode, your A disk will be assumed. It is in R/W (read/write) mode. For minidisks, you can only store files on disks you have accessed in R/W mode. For SFS, you need to have write authority to the directory to create a new file, and write authority to the file to update an existing file. Generally the owner of the directory has such authority, the exception being when the file pool is under the control of an external security manager (ESM), which is a program which either augments or completely replaces the authorization checking done by the file pool server. You can check with your system administrator to see if you have an ESM active on your system.

You can tell, from this display, whether your A disk is space in an SFS file pool or a minidisk. If, in row A, in the Vdev column, you see a DIR or DIRC, rather than a virtual device number, you are an SFS user. You may also be a minidisk user, but your main disk space is within an SFS file pool.

For more information on SFS linking and accessing, see [“Getting Started - Accessing Your Top Directory”](#) on page 39 and [“Accessing Another User's Directory”](#) on page 44. For minidisks, see [“Linking and Sharing Minidisks”](#) on page 103.

Note: As alternatives to using the QUERY ACCESSED command:

- List your accessed directories and minidisks in a full screen format, using the DIRLIST command with the MDISK and ACCESSED options. For more information, see [“Listing the Structure of a Directory with DIRLIST”](#) on page 45, or see the DIRLIST command in *z/VM: CMS Commands and Utilities Reference*.
- Access minidisks and directories with the VMLINK command, which finds free access modes (see [“Linking and Accessing with VMLINK”](#) on page 127, and *z/VM: CMS Commands and Utilities Reference*).

Linking and Accessing with VMLINK

For more information on how to use CMS and CP commands to access and release SFS directories, and to link, access, release, and detach minidisks, see [“Getting Started - Accessing Your Top Directory”](#) on page 39 and [“Linking and Sharing Minidisks”](#) on page 103. The VMLINK command makes it easier to use the functions of these commands.

For example, if you want to use files on the disk owned by user SYSTOOLS at address 233, you would have to:

1. Find out whether you have the disk linked, and where it is linked.
2. If it is not linked, find a free virtual device number at which to link it to your virtual machine.
3. Find a free access mode letter and access the disk.

With VMLINK, you could enter:

```
vmLink systools 233 (nonames
```

VMLINK finds a free device number and access mode and performs the linking and accessing.

If you don't want to keep the SYSTOOLS 233 linked and accessed, you can release and detach the disk with:

```
vmLink systools 233 <detach> (nonames
```

You can also use VMLINK with SFS directories. To access the VMUSER's PARTY directory in file pool VMSYSU, enter:

```
vmLink .dir vmsysu:vmuser.party (nonames
```

The command examples included the NONAMES option. The NONAMES option causes VMLINK to avoid searching the NAMES files when using the VMLINK userid vdev or VMLINK .DIR dirid form of the command or the VMLINK .DIR dirid form of the command with SFS.

Notes:

1. Nicknames can be used instead of user IDs, device address, or directory names.
2. VMLINK without any options will display a menu of available nicknames.
3. A user can define his/her own nicknames, in addition to those provided by the system administrator.
4. VMLINK can be performed automatically when a user logs on, or re-IPLs, by using the AUTOLINK option.

NAMES Files for VMLINK

The system default search order is USERPROD NAMES * VMLINK *nodeid* * VMLINK NAMES *.

A nickname entry consists of a :NICK tag and a tag value. Other tags are optional, and if a tag is missing, its value is assumed to be null.

If minidisks or SFS directories are not specified on a :PRODUCT tag or nicknames are not specified on a :LIST tag, nothing is linked, but the contents of the tags in the entry can be retrieved by VMLINK. Attempting to do a FILELIST in this situation results in an error message. Exits are run when you use EXIT or PREEXIT.

You can use the NAMES command with the VMLINK option to create or modify the USERPROD NAMES file.

:category.name

associates this minidisk or SFS directory with other minidisks or SFS directories. If there is no :CATEGORY tag or if *name* begins with an at-sign (@) or a percent sign (%), the nickname entry is not displayed on the menu. VMLINK uses only the first eight characters of the category name.

You can display a menu of all nicknames associated with a category. Entering VMLINK SOMECA causes VMLINK to display a menu of all the nicknames with a :category.somecat tag, provided there is no SOMECA nickname.

There can be more than one :CATEGORY tag in a nickname entry. In this case the nickname would be listed in the VMLINK menu for either of the two categories.

:exit.execname parmlist

names an exit to be called after the minidisk or SFS directory is accessed. The content and behavior of the tag is the same as that of the EXIT option and *EXIT control statement. The NOEXIT option suppresses using this tag. The EXIT option overrides this tag. There is no way to suppress or override the *EXIT control statement.

:invoke.envir cmd parmlist ;...

names routines to be invoked after all products are processed. The content and behavior of the tag is the same as that of the INVOKE option. The INVOKE option overrides this tag and the NOINVOKE option suppresses the use of this tag.

Only the :INVOKE tag assigned to the first nickname specified on the command is processed; :INVOKE tags for nicknames on :LIST tags are not processed. The :INVOKE tag is not executed if a non-zero return code is encountered during the process of accessing the minidisks or SFS directories.

:list.disk_operands

names additional disk operands to be accessed. The disk operands are the same as for the VMLINK command, except it cannot be a category. Use the NOLIST option to suppress processing of the :LIST tag.

:nick.nickname

defines a nickname for a minidisk or SFS directory.

:node.node_list

lists the nodes for which the nickname is valid. If there is no :NODE tag, the nickname is valid on all nodes. The node list can contain node IDs, *symbols* defined on an *EQUATE record in the VMLINK CONTROL file, and node IDs qualified by an asterisk or a logical NOT sign:

- The asterisk matches any string at the beginning or end of a node ID. For example, :node . NAT* matches any node ID starting with "NAT", and :node . * matches any node ID. You cannot use an asterisk in the middle of a string; doing so causes a search for a node of that name. For example, NAT*12 would look explicitly for a node of NAT*12 only.
- The logical NOT sign (¬) or minus sign (-) before a node ID specifically excludes that node. For example :node . ¬NATVM12 will include all nodes except for NATVM12.
- An asterisk can be combined with the logical NOT sign or minus sign in one node ID. For example, ¬NAT* would exclude all nodes beginning with NAT.
- The logical NOT (¬) or minus sign (-) can be combined with a control file *EQUATE symbol to exclude the nodes represented by the symbol rather than include them. For example, with a *EQUATE symbol of MYNODES with nodes NATVM10 NATVM11 NATVM12, a :node . ¬MYNODES in a names file would make the nickname not valid on nodes NATVM10, NATVM11, and NATVM12.

:preexit.execname parmlist

names an exit to be called before a minidisk is linked or an SFS directory is accessed. Its behavior is the same as that of the PREEXIT option and the *PEXIT control statement. The NOEXIT option suppresses using this tag. The PREEXIT option overrides this tag. There is no way to suppress or override the *PEXIT control statement.

:product.disk_operands linking_details

is the linking information for the minidisk or SFS directory. The disk operands and linking details are the same as for the VMLINK command, with these exceptions:

- The disk operands cannot be a nickname or a category.
- Nothing is linked when the disk operands consist of one or two asterisks: * or * *.
- When the disk operands are an asterisk and a virtual device number (* vdev), VMLINK accesses the user's vdev minidisk.

:title.description

describes the minidisk or SFS directory. The title is displayed on the terminal when the minidisk or SFS directory is accessed and as the description on the menu. When there is no :TITLE tag, the menu description is blank; in messages, VMLINK generates a title from the minidisk or SFS directory specification.

:tagname.tag_value

is a user tag. VMLINK takes no action based on the presence of a user tag, but the information it contains is available to exit execs, and invoked routines.

NAMES File Processing

VMLINK searches for nicknames first in any NAMES files specified on the command with the ADDFILE option; second in any files specified on the *ADDFILE record; and finally in any files specified on the *FILES record. Files are searched in the order specified, and where a mode letter is not specified, in A-to-Z file mode order. Alternatively, you can limit the search to files specified with the ONLY option.

The search for a nickname entry ends only when a nickname entry is found that is valid for the node or when all the NAMES files have been searched. If no match is found for a disk operand entered on the command, VMLINK looks for matching :CATEGORY tags. If there are multiple disk operands on the command, and one is determined to be a category, VMLINK displays a menu of nicknames in the category before it processes the next disk operand.

When a nickname is found in a NAMES file:

1. VMLINK verifies that the entry applies to the current node. If the current node is included in the node IDs listed on the :NODE tag, or if there is no :NODE tag, the entry applies to your system. If the nickname does not apply to your system, VMLINK continues the search.
2. The contents of the :LIST tag are appended to the list of minidisks and/or SFS directories to be accessed.
3. The file mode and vdev to be used are found if not specified.
4. The exec listed on the :PREEXIT tag is run for that nickname.
5. The minidisk or SFS directory on the :PRODUCT tag is linked and accessed.
6. The exec listed on the :EXIT tag is executed for that nickname.
7. VMLINK repeats the process to this point for the next nickname in the list.
8. When all nicknames have been processed, the INVOKE option or the :INVOKE tag on the first nickname in the command is processed.

VMLINK Control File

The VMLINK control file, VMLINK CONTROL, sets defaults for VMLINK, including:

- The search ranges for free file modes (the default for the *filemode*) (*MODES)
- The search range for free virtual device numbers (the default for *vdev*) (*VDEV)
- The NAMES files to be searched (*FILES)
- NAMES files to add to the beginning of the default search order (*ADDFILE)
- VMLINK's behavior when it cannot access a minidisk or SFS directory (*ERROR)

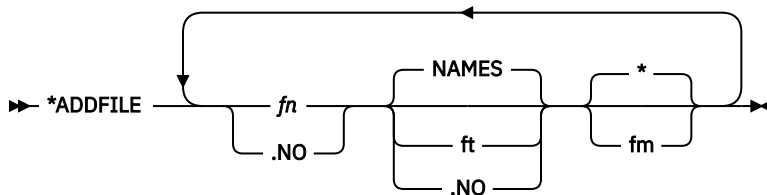
VMLINK scans all VMLINK CONTROL files in A-to-Z file mode order, but only the first acceptable instance of each type of control record, with the exception of the *EQUATE record, is used. Supported control records, and their acceptable values are described below. Thus you can have a VMLINK CONTROL file at file mode A containing only the records for which you want to override the system defaults. For the following control records, defaults set in the control files are overridden by values in a nickname entry, which in turn are overridden by operands specified with the command: *ADDFILE, *MODES and *VDEV.

Each control record name begins with an asterisk. Control records cannot be continued on the next line. VMLINK ignores blank lines and lines that do not begin with a valid record name.

These syntax diagrams show the defaults when the record is not in a control file.

***ADDFILE Record**

Use *ADDFILE to list NAMES files to be searched before files listed on the *FILES record. The file type and mode of the last file name in the list can default to NAMES *.

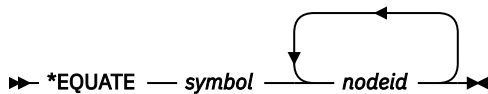


As long as any value is entered this control record will be used and no further searching is done for this record.

If no control record value is specified the control record is ignored and the control file search will continue for this record.

*EQUATE Record

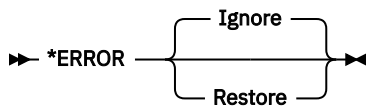
The *EQUATE record assigns a list of nodes to a symbol. When the symbol is used on the :NODE tag, the list of nodes is substituted for the symbol.



- All control files are scanned for *EQUATE records, and all *EQUATE records are used.
- *EQUATE records can be used to define more than one *symbol*, but VMLINK uses only the first instance of any particular value of *symbol*.
- A *symbol* defined on an *EQUATE record cannot be used as a *nodeid* on another *EQUATE record.
- An asterisk should not be used as a symbol. An asterisk in a names file node tag indicates that all nodes are valid. As a result, a symbol of asterisk will be ignored.

*ERROR Record

The *ERROR record determines whether VMLINK accesses any minidisks and/or SFS directories when it cannot access all the minidisks and/or SFS directories. VMLINK can access multiple minidisks and SFS directories by either specifying more than one minidisk, SFS directory or nickname on the command or by specifying a nickname which defines a list.

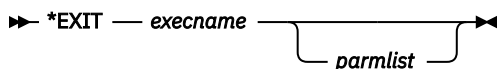


- IGNORE, the default, means to keep whatever was linked/accessed with this VMLINK.
- RESTORE specifies to get rid of whatever might have been linked/accessed by this VMLINK and restore any minidisks and/or SFS directories that were released.

If no control record value is specified the default IGNORE is used.

*EXIT Record

The *EXIT record names an exec to be called after each minidisk or SFS directory has been accessed.



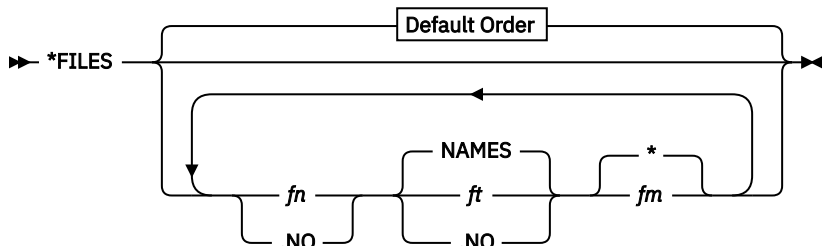
- This exit is called for each minidisk and SFS directory accessed before the exec named on an EXIT option or :exit tag.
- The *EXIT record is processed even if the NOEXIT option is specified.
- Just as with the EXIT option and :exit tag, the parameter list can include VMLINK variables.

As long as any value is entered this control record will be used and no further searching is done for this record.

If no control record value is specified the control record is ignored and the control file search will continue for this record.

*FILES Record

The *FILES record specifies the file ID and search order for the NAMES files. The file type and mode of the last file name in the list can default to NAMES *.



Default Order

➡ USERPROD — NAMES — * — VMLINK — .NO — * — VMLINK — NAMES — * —>

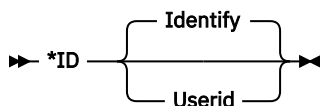
VMLINK replaces the VMLINK variable, .NO, with the system node ID.

As long as any value is entered this control record will be used and no further searching is done for this record.

If no control record value is specified the default order will be used.

*ID Record

The *ID record determines how VMLINK obtains the system node ID.

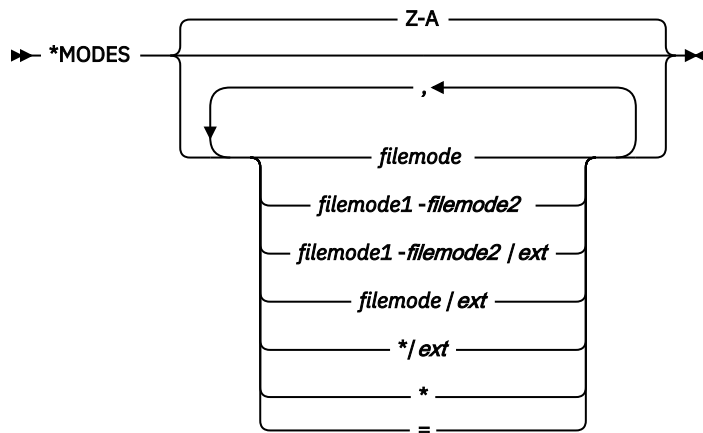


- IDENTIFY, the default, means to use the CMS IDENTIFY command to get the system node ID. See Usage Note 1 in the IDENTIFY command to see when this may not be the desired command.
- USERID means to use the CP QUERY USERID command to get the system node ID.

If no control record value is specified the default IDENTIFY is used.

*MODES Record

The *MODES record defines the default ranges and search orders for finding a free file mode letter. This defines the default search that is used when an asterisk is specified for the file mode in the linking details. A file mode "S" will not be used. If a file mode extension is specified, the extension is used unless overridden by the command line or by the :product tag in the NAMES file.

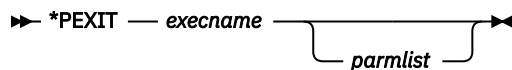


As long as any value is entered, this control record is used and no further searching is done for this record. If an asterisk (*) or equals sign (=) is specified in the control record, the default search order, Z-A, is substituted for the asterisk or equals sign.

If no control record value is specified, the default search order, Z-A, is used.

*PEXIT Record

The *PEXIT record names an exit exec to be called after VMLINK has located a free virtual device number and file mode letter and before the minidisk or SFS directory is accessed.



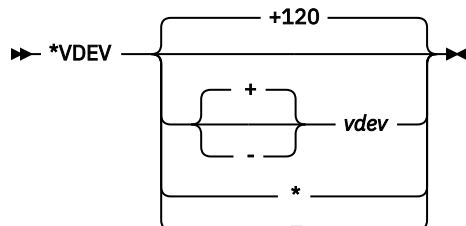
- This exit is called for each minidisk and SFS directory accessed before the exec named on the PREEXIT option or :preexit tag.
- The *PEXIT record is processed even if the NOEXIT option is specified.
- Just as with the PREEXIT option and :preexit tag, the parameter list can include VMLINK variables.

As long as any value is entered this control record will be used and no further searching is done for this record.

If no control record value is specified the control record is ignored and the control file search will continue for this record.

*VDEV Record

The *VDEV record defines where to start a search for a free virtual device number. To search in ascending order, enter the virtual device address number, or a plus sign (+) followed by the virtual device number. To search in descending order, enter a minus sign (-) followed by the virtual device number. This value is used when an asterisk (*) is specified for the virtual device number in the linking details on the command line or in the names file. The virtual device number is a 1- to 4-character hexadecimal.



As long as any value is entered, this control record will be used, and no further searching is done for this record.

The default range is +120 if no control record value is specified, or if an asterisk (*) or equals sign (=) is specified as the control record value.

VMLINK Control File Example

```

*****
** Name      - VMLINK    CONTROL                               **
**                                                    **
** Function - The definitions in this file will define the  **
**            system defaults.  If this file does not exist **
**            or a record is not specified, a default value **
**            will be used.                                **
**                                                    **
** For performance reasons, it is recommended that only   **
** values which are to be changed are included in this    **
** file.  Also, comment records can be removed.           **
**                                                    **
** Each record of this file is assumed to be a comment,   **
** unless it starts with a valid keyword.                 **
** Valid KEYWORDS are:                                    **
**                                                    **
** ADDFILE - EQUATE - ERROR - EXIT - FILES - ID - MODES   **
**            PEXIT - VDEV                                **
**                                                    **
** All of them preceded by an '*'                          **
*****
**EQUATE EAST      BOSTON NEWYORK ATLANTA
**EQUATE CENTRAL   STLOUIS DALLAS
**EQUATE WEST      FRISCO SEATTLE

*ERROR RESTORE

*EXIT EXITCTL
*PEXIT PEXITCTL

*ADDFILE NEWFILE
*FILES USERPROD NAMES * MYOWN NAMES * PRODUCT .NO * VMLINK NAMES Y

** For discussion purposes, we show a default value on *ID
**ID IDENTIFY

*MODES W-Z
*VDEV 900

```

Figure 36. VMLINK Control File Example

Some defaults were set in the example in [Figure 36 on page 135](#):

***EQUATE EAST BOSTON NEWYORK ATLANTA**

EAST is a symbol that was defined, through the *EQUATE control record, to be a synonym of BOSTON and NEWYORK and ATLANTA nodes. In the same way, **CENTRAL** was defined for STLOUIS and DALLAS, and **WEST** for FRISCO and SEATTLE.

The list of nodes assigned to a *symbol* might have significance when systems can be grouped by some common characteristics (for example: geographical location, system level maintenance, system level release, products activated, and so on).

***ERROR RESTORE**

RESTORE is an option on the *ERROR control record that tells VMLINK not to link as many minidisks as possible, but to restore any minidisk accessed if a nonzero return code is received.

***EXIT EXITCTL**

EXITCTL is the filename of an EXEC defined on the *EXIT control record. It is executed after a minidisk is linked and accessed.

***PEXIT PEXITCTL**

PEXITCTL is the filename of an EXEC defined on the *PEXIT control record. It is processed before a minidisk is linked and accessed.

***ADDFILE NEWFILE**

if the file *NEWFILE NAMES* * is found when running VMLINK, it will be searched before the files listed on the *FILES control record.

***FILES USERPROD NAMES * MYOWN NAMES * PRODUCT .NO * VMLINK NAMES Y**

*FILES is a control record on which many VMLINK NAMES files were specified. However, the order in which they were specified determines the search order. In [Figure 36 on page 135](#), the first NAMES file searched is the default, which is USERPROD NAMES, followed by MYOWN NAMES, and so on.

*** ID IDENTIFY**

IDENTIFY is an option of the *ID control record that causes VMLINK to use the CMS IDENTIFY command to obtain the system node ID.

***MODES W-Z**

is a control record that is defined so the first default file mode to be used to access a minidisk is **W**. But if W is unavailable, then the search for a free file mode continues incrementally towards **Z**. If you run out of access file modes, error message DMSVML1277E (no file mode is available) is issued.

***VDEV 900**

is a control record that defined so the first default virtual device address to be used to link a minidisk is **900**. But if 900 is unavailable, then the search for a free virtual device address continues incrementally towards **FFFF**. If you run out of virtual device addresses, error message DMSVML1277E (no virtual device address is available) is issued.

VMLINK Programming Interface

In addition to linking, accessing, and releasing minidisks or SFS directories for users transparently, VMLINK has a programming interface with REXX EXECs:

- Data about the nicknames or about the accessed minidisks or SFS directories can be returned on the stack or in a stem variable.
- Commands and routines can be run at different points in the accessing process. (PREEXIT, EXIT, FILELIST, and INVOKE options, :preexit, :exit, and :invoke tags, *PEXIT and *EXIT control statements or through use of VMLNICXT). The order in processing when these are run is described below.
 - VMLNICXT EXEC, if it exists, is called before any nickname file is searched.
 - *PEXIT control statement exec then the PREEXIT option or :preexit tag exec is called after VMLINK has located a free virtual device number and file mode letter but before the minidisk or SFS directory is accessed.
 - *EXIT control statement exec then the EXIT option or :exit tag exec is called after each minidisk or SFS directory has been accessed.
 - INVOKE option or :invoke tag routines are executed after all the minidisks and SFS directories have been accessed.
 - FILELIST option causes VMLINK to issue a FILELIST command for files on the first minidisk or SFS directory accessed. FILELIST overrides any commands on the :invoke tag, but is run after any INVOKE option commands specified on the VMLINK command. If INVOKE option is used with the FILELIST option, sure to specify the KEEP option also, otherwise, the FILELIST will fail because the minidisk or SFS directory it attempts to do the FILELIST on will have been released already because of the use of the INVOKE option.
- A call to VMLINK can specify parameters to be passed to commands and exit routines.
- A call to VMLINK can suppress tags in the nickname entry (for example: NOINVOKE, NOEXIT, and NOLIST options).
- Prior configuration of the virtual machine can be saved using the PUSH option.
- Prior configuration of the virtual machine can be restored using the POP option.

- VMLINK can search names files besides those listed in the VMLINK CONTROL file (*ADDFILE and *FILES control statements).

Since VMLINK is an EXEC, within REXX EXECs, VMLINK should be started under the CMS or command environments:

- Address command 'EXEC VMLINK ...'
- Address cms 'VMLINK ...'

There are three ways that VMLINK can pass data to a program:

- On the program stack
- In REXX stem variables
- As input parameters to a routine called during VMLINK processing.

Data can be identified by specifying VMLINK variables, such as .AR or the .MSG operand or the NAMES file tags in the options field. Variables can also be used in parameter lists.

The examples in this section use the following EXEC and the following entries for the nicknames PRODLIST, PRODA191 and PRODC193:

```
TESTER EXEC:
/*TESTER EXEC*/
parse arg in
say in
say 'Done'
```

```
:NICK.PRODLIST
:PRODUCT.
:TITLE.List of Disks
:LIST.PRODA191 PRODC193
```

```
:NICK.PRODA191
:PRODUCT..DIR VMSYSU:MAINT.ADISK
:TITLE.Directory for A-disk
:DOG.Fenris
:CAT.Snowball
:EXIT.TESTER .pa
:INVOKE.CP MSG * .TI1 accessed at .FM1
```

```
:NICK.PRODC193
:PRODUCT.PRODUCTC 193 <-900 Z-A RR>
:TITLE.Productc's disk
:DOG.Wolf
:CAT.Puff
:EXIT.TESTER .ta
```

Putting VMLINK Data on the Program Stack

To receive the messages generated by VMLINK in a REXX program, you can use the stack. Specifying the .MSG option will stack the messages in FIFO order by default, prefixed with *.MSG. An example of this is shown in [Figure 37 on page 138](#):

```

type vmlstack exec

/* */
Address command
'EXEC VMLINK PRODLIST (.MSG'
do queued()
  parse pull a
  say a
end

Ready; T=0.01/0.01 11:46:33

vmlstack
*.MSG DMSVML2060I Directory for A-disk accessed as file mode K
*.MSG DMSVML2060I Productc's disk linked RR as 900 file mode L
Ready; T=0.10/0.12 11:46:39

```

Figure 37. VMLINK and the REXX Program Stack Example

In [Figure 37 on page 138](#), the first item was defined as an SFS directory. Therefore, it was only accessed.

Notes:

1. The form of the disk identifier is determined by the way the minidisk or SFS directory is specified in the command. See VMLINK usage note 4 in CMS Command Reference for details on this form.
2. In the stacked lines for each minidisk or SFS directory, variables and tags are listed in the order they are specified in the option field, paired with their values: `.vv vv_value` and `:tag tag_value`
3. By default, VMLINK saves and clears the stack before it calls a routine from PREEXIT, EXIT, or INVOKE, so routines do not have access to data stacked by the VMLINK command, nor do they have access to data stacked by a program that calls VMLINK. To pass data to an exit or a routine called by VMLINK, use the .PA and .TA variables in the routines parameter list.

You can preserve the stack for routines called with the INVOKE option by using the NOSAVE option.

4. Do not leave data on the stack when you call VMLINK in XEDIT. XEDIT tries to execute the stacked lines as subcommands. To keep the stack clear:
 - In a macro, use the STEM option to suppress the stacking.
 - In a macro, create a new stack buffer before calling VMLINK, and drop or empty the buffer before the macro exits.

Putting VMLINK Variables in REXX Stem Variables

By adding the stem option to the VMLINK command, it is possible to assign data passed from VMLINK to a REXX stemmed-variable. An example of this is shown in [Figure 38 on page 138](#):

```

type vmlstem exec

/* */
Address command
'EXEC VMLINK PRODLIST (:PRODUCT .MSG STEM STEMVAR.'
do i = 1 to stemvar.0
  say stemvar.i
end

Ready; T=0.01/0.01 12:34:06

vmlstem
*.MSG DMSVML2060I Directory for A-disk accessed as file mode X
*.MSG DMSVML2060I Productc's disk linked RR as 122 file mode L
PRODLIST :PRODUCT
PRODA191 :PRODUCT .DIR VMSYSU:MAINT.ADISK
PRODC193 :PRODUCT PRODUCTC 193 -900 Z-A RR
Ready; T=0.08/0.09 12:34:10

```

Figure 38. VMLINK Variables and REXX Stem Variables Example

From [Figure 38 on page 138](#) we learn that even though PRODLIST is a list of products, VMLINK will still report a null :PRODUCT tag for it. Then it will resolve the two products that were part of the list.

Asking for additional tag values on the VMLINK command results in an increase in the length of the stemmed-variable.

Passing VMLINK Variables as Parameters

VMLINK variables and tags (but not .MSG) can be passed as parameters on the INVOKE, PARMS, PREEXIT, and EXIT options, or on the corresponding NAMES file tags and control file records. VMLINK replaces variables with their values before it calls the routine.

The PRODA191 nickname entry includes this :INVOKE tag:

```
:invoke.CP MSG * .ti1 accessed at .fm1
```

This call to VMLINK

```
address COMMAND 'EXEC VMLINK PRODA191 (NOTYPE '
```

produces this CP message with information about the first minidisk or SFS directory accessed:

```
10:14:36 * MSG FROM USER1 : DIRECTORY FOR A-DISK ACCESSED AT Q
```

The INVOKE option overrides the :INVOKE tag. This call to VMLINK

```
EXEC VMLINK PRODA191 (NOTYPE INVOKE CP MSG * .PR1 accessed at .FM1
```

substitutes the product and file mode for the minidisk or SFS directory into the CP message:

```
10:15:11 * MSG FROM USER1 : .DIR VMSYSU:MAINT.ADISK ACCESSED AT Q
```

The :EXIT tag for PRODA191 uses the .PA variable: :exit.tester .pa. Accessing the PRODA191 SFS directory with the PARMS option, like this,

```
address COMMAND 'EXEC VMLINK PRODA191 (NOTYPE PARMS X Y Z'
```

replaces the .PA variable on the :EXIT tag with the parameter string specified with the PARMS option. The parameter list starts with an identifier for the minidisk or SFS directory, in this case the nickname:

```
PRODA191 X Y Z
```

Notes:

1. VMLINK replaces variables in parameter lists with the values of the variables.
2. Tags cannot be specified in parameter lists, but the .TA variable can be specified; it is replaced by tag-value pairs for tags specified on the command invocation. For example, the :EXIT tag of the PRODC193 nickname is

```
:exit.tester .ta
```

This call to VMLINK, with the :DOG and :CAT tags specified as options,

```
address COMMAND 'EXEC VMLINK PRODC193 (:DOG :CAT'
```

passes this parameter list of tag-value pairs to the TESTER EXEC:

```
PRODC193 :DOG Wolf :CAT Puff
```

3. VMLINK always passes a parameter list to an exit. The first token is the disk identifier, which is the same as the identifier for stacked lines, except that there is no leading asterisk. See VMLINK usage note 4 in the CMS Command Reference. The rest of the tokens are the values of variables specified on the EXIT and PREEXIT options and :EXIT and :PREEXIT tags.
4. VMLINK processes the :EXIT and :PREEXIT tags for all nicknames in the command and on :LIST tags. An exec on an EXIT or PREEXIT option or an exec on a *PEXIT or *EXIT control statement is called for

each minidisk and SFS directory accessed. At each invocation, an exit exec has access to variable and tag data only for the particular minidisk or SFS directory.

5. Only the :INVOKE tag for the first nickname on the command is processed, and the INVOKE option is processed only once. A routine called with the INVOKE option or the :INVOKE tag has access to variables for all the minidisks and/or SFS directories accessed.
6. When VMLINK accesses an SFS directory, it replaces .CU with "DIR *dirname*". For example, this call to access MAINT's .SHIPS subdirectory specifies the TESTER EXEC as an exit:

```
vmLink .dir .ships (exit(tester .cu .fm)
```

VMLINK passes this string to the TESTER exec:

```
.DIR.SHIPS DIR SERVER:MAINT.SHIPS M
```

Where:

.DIR.SHIPS

specifies the disk identifier for the parameter list.

DIR SERVER:MAINT.SHIPS

specifies the value of .CU,

M

specifies the value of .FM.

We have already shown how to pass VMLINK variables to exits. It is possible to use the .PA tag in your names file to accept parameters passed from the command line as well. Parameters may be passed to the :EXIT, :INVOKE, or :PREEXIT tags in the names file. In [Figure 39 on page 140](#) the data PARM_LIST would be substituted where a .PA exists in the VMLINK names file:

```
VMLINK PRODLIST ( PARM PARM_LIST
```

Figure 39. Specifying VMLINK Variables as Parameters Example

EXIT, PREEXIT and INVOKE Examples

The following are NAMES file entries to be used in the EXIT, PREEXIT and INVOKE examples:

```

====> NAMES (Vmlink panel)      File: MYOWN      NAMES      A1      <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: PRODA      Description: Linking PRODUCT A with its
pre-req ...
Product Linking      :
Information (USERID:
cuu/.DIR dirname)   :
Category: PRODUCTS
      Invoke:
      Preexit:
      Exit:
      Valid Nodes: NYVM1 BOSTVM3
      :
      List of Names: PRODA191 PRODC193
      :
      :
      :
      Tag:      Value:
      Tag:      Value:

1= Help      2= Add      3= Quit      4= Clear      5= Find      6= Change
7= PrevNick  8= NextNick 9= Delete   10= PrevScrn 11= NextScrn 12= Cursor
=====> Screen 1 of 1 <=====

```

Figure 40. VMLINK PRODUCT A NAMES File Entry

```

====> NAMES (Vmlink panel)      File: MYOWN      NAMES      A1      <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: PRODA191      Description: Linkingproduct A Mdisk 191 ...
Product Linking      : PRODUCTA 191 <+800 G-A MR>
Information (USERID:
cuu/.DIR dirname)   :
Category: PRODUCTS
      Invoke:
      Preexit:
      Exit:
      Valid Nodes: NYVM1 BOSTVM3
      :
      List of Names:
      :
      :
      :
      Tag:      Value:
      Tag:      Value:

1= Help      2= Add      3= Quit      4= Clear      5= Find      6= Change
7= PrevNick  8= NextNick 9= Delete   10= PrevScrn 11= NextScrn 12= Cursor
=====> Screen 1 of 1 <=====

```

Figure 41. VMLINK PRODUCT A Minidisk 191 NAMES File Entry

```

====> NAMES (Vmlink panel)      File: MYOWN      NAMES      A1      <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: PRODB      Description: Linking PRODUCT B with no
pre-req ...
Product Linking      :
Information (USERID:
cuu/.DIR dirname)   :
Category: PRODUCTS
                Invoke: EXEC INVOKEB .no1
                Preexit: PREXITB .CA1
                Exit:
                Valid Nodes: NYVM1 BOSTVM1
                :
List of Names: PRODB191 PRODB193 PRODB195
                :
                :
Tag:                Value:
Tag:                Value:

1= Help      2= Add      3= Quit      4= Clear      5= Find      6= Change
7= PrevNick  8= NextNick 9= Delete   10= PrevScrn  11= NextScrn 12= Cursor
=====> Screen 1 of 1 <=====

```

Figure 42. VMLINK PRODUCT B NAMES File Entry

```

====> NAMES (Vmlink panel)      File: MYOWN      NAMES      A1      <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: PRODB191 Description: Linking Product B Mdisk 191 ...
Product Linking      : PRODUCTB 191 <+500 E-L RR>
Information (USERID:
cuu/.DIR dirname)   :
Category: PRODUCTS
                Invoke:
                Preexit:
                Exit: EXITB .CU1 .FM1
                Valid Nodes: NYVM1 BOSTVM1
                :
List of Names:
                :
                :
Tag:                Value:
Tag:                Value:

1= Help      2= Add      3= Quit      4= Clear      5= Find      6= Change
7= PrevNick  8= NextNick 9= Delete   10= PrevScrn  11= NextScrn 12= Cursor
=====> Screen 1 of 1 <=====

```

Figure 43. VMLINK PRODUCT B Minidisk 191 NAMES File Entry

```

====> NAMES (Vmlink panel)      File: MYOWN      NAMES      A1      <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: PRODB193  Description: Linking Product B Mdisk 193 ...
Product Linking      : PRODUCTB 193 <+500 E-L RR>
Information (USERID:
cuu/.DIR dirname)   :
Category: PRODUCTS
      Invoke:
      Preexit:
      Exit: EXITB .CU1 .FM1
      Valid Nodes: NYVM1 BOSTVM1
      :
      List of Names:
      :
      :
      :
      Tag:      Value:
      Tag:      Value:

1= Help      2= Add      3= Quit      4= Clear      5= Find      6= Change
7= PrevNick  8= NextNick 9= Delete   10= PrevScrn 11= NextScrn 12= Cursor
=====> Screen 1 of 1 <=====

```

Figure 44. VMLINK PRODUCT B Minidisk 193 NAMES File Entry

```

====> NAMES (Vmlink panel)      File: MYOWN      NAMES      A1      <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: PRODB195  Description: Linking Product B Mdisk 195 ...
Product Linking      : PRODUCTB 195 <+500 E-L RR>
Information (USERID:
cuu/.DIR dirname)   :
Category: PRODUCTS
      Invoke:
      Preexit:
      Exit: EXITB .CU1 .FM1
      Valid Nodes: NYVM1 BOSTVM1
      :
      List of Names:
      :
      :
      :
      Tag:      Value:
      Tag:      Value:

1= Help      2= Add      3= Quit      4= Clear      5= Find      6= Change
7= PrevNick  8= NextNick 9= Delete   10= PrevScrn 11= NextScrn 12= Cursor
=====> Screen 1 of 1 <=====

```

Figure 45. VMLINK PRODUCT B Minidisk 195 NAMES File Entry

```

====> NAMES (Vmlink panel)      File: MYOWN      NAMES      A1      <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: PRODC      Description: Linking PRODUCT C with no
pre-req ...
Product Linking      :
Information (USERID:
cuu/.DIR dirname)   :
Category: PRODUCTS
                Invoke:
                Preexit:
                Exit:
                Valid Nodes: NYVM1 BOSTVM3
                :
                List of Names: PRODC191 PRODC193
                :
                :
                :
Tag:                  Value:
Tag:                  Value:

1= Help      2= Add      3= Quit      4= Clear      5= Find      6= Change
7= PrevNick  8= NextNick 9= Delete  10= PrevScrn  11= NextScrn 12= Cursor
=====> Screen 1 of 1 <=====

```

Figure 46. VMLINK PRODUCT C NAMES File Entry

```

====> NAMES (Vmlink panel)      File: MYOWN      NAMES      A1      <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: PRODC191 Description: Linking roduct C Mdisk 191 ...
Product Linking      : PRODUCTC 191 <-200 X-Z RR>
Information (USERID:
cuu/.DIR dirname)   :
Category: PRODUCTS
                Invoke:
                Preexit:
                Exit:
                Valid Nodes: NYVM1 BOSTVM3
                :
                List of Names:
                :
                :
                :
Tag:                  Value:
Tag:                  Value:

1= Help      2= Add      3= Quit      4= Clear      5= Find      6= Change
7= PrevNick  8= NextNick 9= Delete  10= PrevScrn  11= NextScrn 12= Cursor
=====> Screen 1 of 1 <=====

```

Figure 47. VMLINK PRODUCT C Minidisk 191 NAMES File Entry

```

====> NAMES (Vmlink panel)      File: MYOWN      NAMES      A1      <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: PRODC193  Description: Linking oduct C Mdisk 193 ...
Product Linking      : PRODUCTC 193 <-200 X-Z RR>
Information (USERID:
cuu/.DIR dirname)   :
Category: PRODUCTS
      Invoke:
      Preexit:
      Exit:
      Valid Nodes: NYVM1 BOSTVM3
      List of Names:
      :
      :
      :
      :
      :
      Tag:          Value:
      Tag:          Value:

1= Help      2= Add      3= Quit      4= Clear      5= Find      6= Change
7= PrevNick  8= NextNick 9= Delete    10= PrevScrn 11= NextScrn 12= Cursor
=====> Screen 1 of 1 <=====

```

Figure 48. VMLINK PRODUCT C Minidisk 193 NAMES File Entry

Testing with the VMLNICXT EXEC

New product versions are usually introduced by making them available on a test basis, while keeping the old version available in case problems develop with the new version. VMLINK's VMLNICXT exit provides a way to introduce a new version while users and applications continue to use the same nickname. If there is a VMLNICXT EXEC, VMLINK calls it before it searches any NAMES files for the nickname. VMLINK passes the nickname to VMLNICXT. VMLNICXT either returns a nickname for a test minidisk or SFS directory or returns the input nickname. In either case, VMLINK uses the nickname returned by VMLNICXT.

A VMLNICXT SAMPEXEC is on the system tools minidisk. It can be customized to VMLNICXT EXEC, like this:

```

/* VMLNICXT EXEC--translate nicknames for beta test */
/* Input: nickname */
/* Output: substitute beta test nickname for */
/* input nickname */
/* The substitution list is a SELECT statement */
/* containing conditions in the form: */
/* when innick='PRODUCTION' */
/* then outnick='BETATEST' */
arg innick
select
  when innick='REXXCOMP' /*list of substitutions */
  then outnick='NEWREXXC' /*Rexx Compiler*/
  when innick='DMS' /*Display Management System*/
  then outnick='NEWDMS';
  otherwise outnick=innick; /*no substitution*/;
end
return(outnick) /*return substitute nickname*/

```

To use this VMLNICXT EXEC:

1. Install the new versions of REXXCOMP and DMS on new product minidisks or SFS directories.
2. Define the nicknames, NEWREXXC and NEWDMS, for the new product minidisks or SFS directory in a NAMES files accessed by the beta test users.
3. Make the VMLNICXT EXEC available to the beta test users. Now when the DMS or the REXXCOMP nickname, is used, the user or application gets the NEWDMS or NEWREXXC product minidisk or SFS directory.

When you are ready to put the test minidisks or SFS directories into production:

1. Update the NAMES files so the DMS and REXXCOMP nicknames refer to the new product minidisks or SFS directories.
2. Remove VMLNICXT EXEC.

NAMES File Exit Examples

To demonstrate how VMLINK handles the default overrides, simple PREEXITs, EXITs, and INVOKE EXECs are coded and run, and console output is displayed.

The code for VMLINK PREXITB exit example is shown in [Figure 49 on page 146](#):

```
/*                                     */
/*   This is the PREXITB EXEC example that is specified on the   */
/*   preexit tag in the VMLINK NAMES file for PRODUCTB          */
/*                                     */
Address command
parse arg nick_id cat
say
say "I am the PREXITB defined on the preexit tag of the NAMES file."
say "VMLINK passed me its category variable, .CA1. So I received: "
say "Nickname ID : " nick_id
say "Category    : " cat
say
```

Figure 49. PREXITB EXEC Example

The code for VMLINK EXITB exit example is shown in [Figure 50 on page 146](#):

```
/*                                     */
/*   This is the EXITB EXEC example that is specified on the     */
/*   exit tag in the VMLINK NAMES file for PRODUCTB             */
/*                                     */
Address command
parse upper arg nick_id vaddr filemode
say
say "I am the EXITB defined on the exit tag of the NAMES file."
say "VMLINK passed me its virtual address variable, .CU1, and its"
say "file mode variable, .FM1. So I received: "
say "Nickname ID      : " nick_id
say "Virtual Address  : " vaddr
say "File mode       : " filemode
say
```

Figure 50. EXITB EXEC Example

The code for VMLINK INVOKEB exit example is shown in [Figure 51 on page 146](#):

```
/*                                     */
/*   THIS IS THE INVOKEB EXEC EXAMPLE THAT IS SPECIFIED ON THE  */
/*   VMLINK NAMES File for PRODUCTB                             */
/*                                     */
Address command
parse arg nodes
say
say "I am the INVOKEB defined on the invoke tag of the NAMES file."
say "I am usually used to start up an application. However, in this"
say "case, VMLINK passed me its node variable, .N01. So I received: "
say "Nodes : " nodes
say
```

Figure 51. INVOKEB EXEC Example

VMLINK Control File Exit Examples

The code for the VMLINK PEXITCTL exit example is shown in [Figure 52 on page 147](#):


```

/*                                     */
/*      This is the PEXITCTL EXEC example that is used in the      */
/*                                     VMLINK CONTROL File           */
/*                                     */
say
say "I am the PEXITCTL defined on the VMLINK CONTROL control file."
say "But the preexit tag on the NAMES file will override me."
say

```

Figure 52. *PEXITCTL EXEC Example*

The code for the VMLINK EXITCTL exit example is shown in [Figure 53 on page 147](#):

```

/*                                     */
/*      This is the EXITCTL EXEC example that is used in the      */
/*                                     VMLINK CONTROL File           */
/*                                     */
say
say "I am the EXITCTL defined on the VMLINK CONTROL control file."
say "But the exit tag on the NAMES file will override me."
say

```

Figure 53. *EXITCTL EXEC Example*

VMLINK Command Examples

Before using the VMLINK command, you may require the VMLINK CONTROL and VMLINK NAMES files to be coded and activated by system support personnel. Advanced users can create their own. However, if you are using VMLINK with *userid* and *disk* or SFS options, you will not need NAMES or CONTROL files.

The examples in this section rely on the proper setup of the VMLINK CONTROL and VMLINK NAMES files. To show you VMLINK setup and usage, the examples in this section will use the VMLINK CONTROL file shown previously and the nicknames defined previously. While reading these examples, please refer to the mentioned sections whenever needed.

Example 1

[Figure 54 on page 148](#) shows the execution of the following command:

```
VMLINK PRODB
```

The characteristics are:

- The entries declared for PRODB are in effect.
- The VMLINK CONTROL control file did not override any default values.

```

vmlink prodb
I am the PREXITB defined on the preexit tag of the NAMES file.
VMLINK passed me its category variable, .CA1. So I received:
Nickname ID : PRODB
Category    : PRODUCTS

DMSVML2060I Linking Product B Mdisk 191 ... linked RR as 500 file mode E

I am the EXITB defined on the exit tag of the NAMES file.
VMLINK passed me its virtual address variable, .CU1, and its
file mode variable, .FM1. So I received:
Nickname ID : PRODB191
Virtual Address : 500
File mode    : E

DMSVML2060I Linking Product B Mdisk 193 ... linked RR as 902 file mode F

I am the EXITB defined on the exit tag of the NAMES file.
VMLINK passed me its virtual address variable, .CU1, and its
file mode variable, .FM1. So I received:
Nickname ID : PRODB193
Virtual Address : 501
File mode    : F

DMSVML2060I Linking Product B Mdisk 195 ... linked RR as 903 file mode G

I am the EXITB defined on the exit tag of the NAMES file.
VMLINK passed me its virtual address variable, .CU1, and its
file mode variable, .FM1. So I received:
Nickname ID : PRODB195
Virtual Address : 502
File mode    : G

I am the INVOKEB defined on the invoke tag of the NAMES file.
I am usually used to start up an application. However, in this
case, VMLINK passed me its node variable, .NO1. So I received:
Nodes : NYVM1 BOSTVM1

DMSVML2061I Linking Product B Mdisk 191 ... detached
DMSVML2061I Linking Product B Mdisk 193 ... detached
DMSVML2061I Linking Product B Mdisk 195 ... detached
Ready; T=0.15/0.19 15:05:27

```

Figure 54. VMLINK PRODB with NAMES, but Without VMLINK CONTROL Activated

The following occurred:

1. The **PREXITB**, which is the preexit EXEC defined on the preexit tag of the NAMES file, executed. It used the VMLINK variables passed to the EXEC as parameters with the category .CA1.
2. VMLINK linked and accessed the PRODB 191 minidisk with virtual address 500 and file mode E.
3. The **EXITB**, which is the exit EXEC defined on the exit tag of the NAMES file, executed for the preceding minidisk (191), showing the VMLINK variables passed to the EXEC as parameters with the virtual address .CU1 and the file mode .FM1.
4. VMLINK linked and accessed the PRODB 193 minidisk with virtual address 501 and file mode F.
5. The **EXITB**, which is the exit EXEC defined on the exit tag of the NAMES file, was executed for the preceding minidisk (193).
6. VMLINK linked and accessed the PRODB 195 minidisk with virtual address 502 and file mode G.
7. The **EXITB**, which is the exit EXEC defined on the exit tag of the NAMES file, was executed for the preceding minidisk (195).
8. The **INVOKEB**, which is the invoke EXEC defined on the invoke tag of the NAMES file, was executed as the last step, showing the VMLINK variables passed to the EXEC as parameters with the valid nodes .NO1.
9. All minidisks were detached because of the NOKEEP option is used when the INVOKE tag is used. To keep the minidisks linked and accessed, use the KEEP option of the VMLINK command.

Example 2

Figure 55 on page 149 and Figure 56 on page 150 show the execution of the following command:

```
VMLINK PRODB
```

The characteristics are:

- The entries declared for PRODB are in effect.
- The VMLINK CONTROL control file defined in Figure 36 on page 135 is active.

vmlink prodb

I am the **PEXITCTL** defined on the VMLINK CONTROL control file.
But the preexit tag on the NAMES file will override me.

I am the **PREXITB** defined on the preexit tag of the NAMES file.
VMLINK passed me its category variable, .CA1. So I received:
Nickname ID : PRODB
Category : PRODUCTS

I am the **EXITCTL** defined on the VMLINK CONTROL control file.
But the exit tag on the NAMES file will override me.

Figure 55. VMLINK PRODB with NAMES and VMLINK CONTROL Activated (Part 1 of 2)

```

I am the PEXITCTL defined on the VMLINK CONTROL control file.
But the preexit tag on the NAMES file will override me.

DMSVML2060I Linking Product B Mdisk 191 ... linked RR as 500 file mode E

I am the EXITCTL defined on the VMLINK CONTROL control file.
But the exit tag on the NAMES file will override me.

I am the EXITB defined on the exit tag of the NAMES file.
VMLINK passed me its virtual address variable, .CU1, and its
file mode variable, .FM1. So I received:
Nickname ID      : PRODB191
Virtual Address  : 500
File mode        : E

I am the PEXITCTL defined on the VMLINK CONTROL control file.
But the preexit tag on the NAMES file will override me.

DMSVML2060I Linking Product B Mdisk 193 ... linked RR as 900 file mode F

I am the EXITCTL defined on the VMLINK CONTROL control file.
But the exit tag on the NAMES file will override me.

I am the EXITB defined on the exit tag of the NAMES file.
VMLINK passed me its virtual address variable, .CU1, and its
file mode variable, .FM1. So I received:
Nickname ID      : PRODB193
Virtual Address  : 501
File mode        : F

I am the PEXITCTL defined on the VMLINK CONTROL control file.
But the preexit tag on the NAMES file will override me.

DMSVML2060I Linking Product B Mdisk 195 ... linked RR as 901 file mode G

I am the EXITCTL defined on the VMLINK CONTROL control file.
But the exit tag on the NAMES file will override me.

I am the EXITB defined on the exit tag of the NAMES file.
VMLINK passed me its virtual address variable, .CU1, and its
file mode variable, .FM1. So I received:
Nickname ID      : PRODB195
Virtual Address  : 502
File mode        : G

I am the INVOKEB defined on the invoke tag of the NAMES file.
I am usually used to start up an application. However, in this
case, VMLINK passed me its node variable, .N01. So I received:
Nodes : NYVM1 BOSTVM1

DMSVML2061I Linking Product B Mdisk 191 ... detached
DMSVML2061I Linking Product B Mdisk 193 ... detached
DMSVML2061I Linking Product B Mdisk 195 ... detached
Ready; T=0.17/0.23 15:30:36

```

Figure 56. VMLINK PRODB with NAMES and VMLINK CONTROL Activated (Part 2 of 2)

The following occurred:

1. These steps are followed by the resolution of **PRODB** nickname entry in the NAMES file:
 - a. The **PEXITCTL**, which is the preexit EXEC defined on the VMLINK CONTROL control file, executed.
 - b. The **PREXITB**, which is the preexit EXEC defined on the preexit tag of the NAMES file, executed, overriding the PEXITCTL exit defined on the VMLINK CONTROL control file.
 - c. The **EXITCTL**, which is the exit EXEC defined on the VMLINK CONTROL control file, executed.
2. These steps are followed by the resolution of **PRODB191** nickname entry in the NAMES file:
 - a. The **PEXITCTL**, which is the preexit EXEC defined on the VMLINK CONTROL control file, executed.
 - b. VMLINK linked and accessed the PRODB 191 minidisk with virtual address 500 and file mode E.
 - c. The **EXITCTL**, which is the exit EXEC defined on the VMLINK CONTROL control file, executed.
 - d. The **EXITB**, which is the exit EXEC defined on the exit tag of the NAMES file, executed, overriding the EXITCTL exit defined in the VMLINK CONTROL control file.

3. These steps are followed by the resolution of **PRODB193** nickname entry in the NAMES file:
 - a. The **PEXITCTL**, which is the preexit EXEC defined in the VMLINK CONTROL control file, executed.
 - b. VMLINK linked and accessed the PRODB 193 minidisk with virtual address 501 and file mode F.
 - c. The **EXITCTL**, which is the exit EXEC defined in the VMLINK CONTROL control file, executed.
 - d. The **EXITB**, which is the exit EXEC defined on the exit tag of the NAMES file, executed, overriding the EXITCTL exit defined in the VMLINK CONTROL control file.
4. These steps are followed by the resolution of **PRODB195** nickname entry in the NAMES file:
 - a. The **PEXITCTL**, which is the preexit EXEC defined in the VMLINK CONTROL control file, executed.
 - b. VMLINK linked and accessed the PRODB 195 minidisk with virtual address 502 and file mode G.
 - c. The **EXITCTL**, which is the exit EXEC defined in the VMLINK CONTROL control file, executed.
 - d. The **EXITB**, which is the exit EXEC defined in the exit tag of the NAMES file, executed, overriding the EXITCTL exit defined in the VMLINK CONTROL control file.
5. The **INVOKEB**, which is the invoke exit defined on the invoke tag of the NAMES file, executed for **PRODB** nickname entry resolution as the last step in the VMLINK cycle.
6. All minidisks were detached because of the INVOKE tag. To keep the minidisks linked and accessed, use the KEEP option of the VMLINK command.

Example 3

Figure 57 on page 151 shows the execution of the following command:

```
VMLINK PRODA ( KEEP INVOKE EXEC SAYHI
```

The characteristics are:

- The entries declared for PRODA are in effect, as are the entries for PRODC.
- This is an example of how the corequisite invocation EXEC defined in a NAMES entry is treated. **No** VMLINK CONTROL control file was activated in order to keep the output of the VMLINK command as clear as possible.
- The **SAYHI EXEC** is a simple EXEC intended just to demonstrate the activation of the VMLINK INVOKE exit through the VMLINK command.

```
vmLink proda ( keep invoke exec sayhi
```

```
DMSVML2060I Linking Product A Mdisk 191 ... linked MR as 800 file mode G
DMSVML2060I Linking Product C Mdisk 193 ... linked RR as 200 file mode X
```

```
Hi there... Everything ok?
```

```
Ready; T=0.09/0.11 12:59:25
```

Figure 57. VMLINK PRODA with Invoke EXEC

The following occurred:

1. VMLINK linked and accessed the PRODA 191 minidisk with virtual address 800 and file mode G, as declared in its NAMES entry.
2. VMLINK linked and accessed the PRODC 193 minidisk with virtual address 200 and file mode X, as declared in its NAMES entry.
3. No minidisks were detached because the option KEEP was used.
4. The SAYHI EXEC, which is shown in Figure 58 on page 152, was executed as the last task of this VMLINK command.

```

/*
/*      A simple exec to be used with the INVOKE option on the
/*      VMLINK command.
/*
*/

say
say "Hi there... Everything ok?"
say

```

Figure 58. SAYHI EXEC Example

VMLINK Linking Behaviors

These tables show how VMLINK will behave in linking and accessing minidisks depending on the status of the minidisk when the VMLINK command is issued.

Table 15. VMLINK Linking Behavior When No Link to the Disk		
No link to the disk	Action	Notes
VMLINK with no options or linking details	Link in Read mode and access	Same result with RR or READ
VMLINK with <i>vdev</i> specified (a free one)	Link at <i>vdev</i> in Read mode and access	Same result with RR or READ
VMLINK with <i>vdev</i> specified (one in use)	Error message, no link done	Same result with RR or READ
VMLINK with <i>vdev</i> specified (one in use) with FORCE option	Detach current disk at <i>vdev</i> and link this disk at that <i>vdev</i> Read mode and access disk	Same result with RR or READ
VMLINK with specified file mode range	Link in Read mode and access in range	Same result with RR or READ
VMLINK with specified single file mode	Link in Read mode and access at mode. This will release whatever was at that file mode	Same result with RR or READ
VMLINK <DET>	No R/O to detach message	Same result with RR or READ
VMLINK <REL>	No action	Same result with RR or READ
VMLINK with (WRITE option	Link in Write mode and access	Same result with M
VMLINK with <i>vdev</i> specified (a free one) and (WRITE option	Link at <i>vdev</i> in Write mode and access	Same result with M
VMLINK with <i>vdev</i> specified (one in use) and (WRITE option	Error message, no link done	Same result with M
VMLINK with <i>vdev</i> specified (one in use) with FORCE and WRITE options	Detach current disk at <i>vdev</i> and link this disk at that <i>vdev</i> Write mode and access disk	Same result with M
VMLINK with specified file mode range and (WRITE option	Link in Write mode and access in range	Same result with M
VMLINK with specified single file mode and (WRITE option	Link in Write mode and access at mode. This will release whatever was at that file mode	Same result with M
VMLINK <DET>	No R/W to detach message	Same result with M
VMLINK <REL>	No action	Same result with M

Table 16. VMLINK Linking Behavior When Already Linked in READ (Not Accessed)

Already linked in READ (not accessed)	Action	Notes
VMLINK with no options or linking details	Keep existing Link and access disk	Same result with RR or READ
VMLINK with <i>vdev</i> specified (a free one)	Detach existing link, link at requested <i>vdev</i> and access disk	Same result with RR or READ
VMLINK with <i>vdev</i> specified (one in use)	Error message, no link done, unless <i>vdev</i> requested is where it is already linked	Same result with RR or READ
VMLINK with <i>vdev</i> specified (one in use) with FORCE option	Detach current disk at <i>vdev</i> and detach current link to disk. Link disk at requested <i>vdev</i> and access disk	Same result with RR or READ
VMLINK with specified file mode range	Keep existing link and access in range	Same result with RR or READ
VMLINK with specified single file mode	Keep existing link and access at mode. This will release whatever was at that file mode	Same result with RR or READ
VMLINK <DET>	Detach current disk	Same result with RR or READ
VMLINK <REL>	No action	Same result with RR or READ
VMLINK with (WRITE option	Link in Write mode and access. Leaves Read link as is	Same result with M
VMLINK with <i>vdev</i> specified (a free one) and (WRITE option	Link at <i>vdev</i> in Write mode and access. Leaves Read link as is	Same result with M
VMLINK with <i>vdev</i> specified (one in use) and (WRITE option	Error message, no link done	Same result with M
VMLINK with <i>vdev</i> specified (one in use) with FORCE and WRITE options	Detach current disk at <i>vdev</i> and link this disk at that <i>vdev</i> Write mode and access disk. Leaves Read link as is.	Same result with M, unless <i>vdev</i> of Read link is the one forced
VMLINK with specified file mode range and (WRITE option	Link in Write mode and access in range. Leaves Read link as is	Same result with M, unless <i>vdev</i> of Read link is the one forced
VMLINK with specified single file mode and (WRITE option	Link in Write mode and access at mode. This will release whatever was at that file mode. Leaves Read link as is.	Same result with M, unless <i>vdev</i> of Read link is the one forced
VMLINK <DET>	No R/W to detach message	Same result with M, unless <i>vdev</i> of Read link is the one forced
VMLINK <REL>	No action	Same result with M, unless <i>vdev</i> of Read link is the one forced

Table 17. VMLINK Linking Behavior When Already Linked in READ (Accessed)

Already linked in READ (accessed too)	Action	Notes
VMLINK with no options or linking details	Keep existing Link and existing access	Same result with RR or READ
VMLINK with <i>vdev</i> specified (a free one)	Detach existing link, link at requested <i>vdev</i> and access disk	Same result with RR or READ
VMLINK with <i>vdev</i> specified (one in use)	Error message, no link done, unless <i>vdev</i> requested is where it is already linked	Same result with RR or READ
VMLINK with <i>vdev</i> specified (one in use) with FORCE option	Detach current disk at <i>vdev</i> and detach current link to disk. Link disk at requested <i>vdev</i> and access disk	Same result with RR or READ
VMLINK with specified file mode range	Keep existing link and existing access if in range, otherwise reaccess disk in range	Same result with RR or READ
VMLINK with specified single file mode	Keep existing link and access at mode. This will release whatever was at that file mode	Same result with RR or READ
VMLINK <DET>	Detach current disk	Same result with RR or READ
VMLINK <REL>	Release disk but keep link	Same result with RR or READ
VMLINK with (WRITE option	Link in Write mode and access. Leaves Read link as is	Same result with M
VMLINK with <i>vdev</i> specified (a free one) and (WRITE option	Link at <i>vdev</i> in Write mode and access. Leaves Read link as is	Same result with M
VMLINK with <i>vdev</i> specified (one in use) and (WRITE option	Error message, no link done	Same result with M
VMLINK with <i>vdev</i> specified (one in use) with FORCE and WRITE options	Detach current disk at <i>vdev</i> and link this disk at that <i>vdev</i> Write mode and access disk. Leaves Read link as is	Same result with M, unless <i>vdev</i> of Read link is the one forced
VMLINK with specified file mode range and (WRITE option	Link in Write mode and access in range. Leaves Read link as is	Same result with M, unless <i>vdev</i> of Read link is the one forced
VMLINK with specified single file mode and (WRITE option	Link in Write mode and access at mode. This will release whatever was at that file mode. Leaves Read link as is.	Same result with M, unless <i>vdev</i> of Read link is the one forced
VMLINK <DET>	No R/W to detach message	Same result with M, unless <i>vdev</i> of Read link is the one forced
VMLINK <REL>	No action	Same result with M, unless <i>vdev</i> of Read link is the one forced

Table 18. VMLINK Linking Behavior When Already Linked in WRITE (Not Accessed)

Already linked in WRITE (not accessed)	Action	Notes
VMLINK with no options or linking details	Link in Read mode and access. Leaves Write link as is	Same result with RR or READ
VMLINK with <i>vdev</i> specified (a free one)	Link at <i>vdev</i> in Read mode and access. Leaves Write link as is	Same result with RR or READ
VMLINK with <i>vdev</i> specified (one in use)	Error message, no link done	Same result with RR or READ
VMLINK with <i>vdev</i> specified (one in use) with FORCE option	Detach current disk at <i>vdev</i> and link this disk at that <i>vdev</i> Read mode and access disk. Leaves Write link as is.	Same result with RR or READ, unless <i>vdev</i> of Write link is the one forced
VMLINK with specified file mode range	Link in Read mode and access in range. Leaves Write link as is	Same result with RR or READ, unless <i>vdev</i> of Write link is the one forced
VMLINK with specified single file mode	Link in Read mode and access at mode. This will release whatever was at that file mode. Leaves Write link as is.	Same result with RR or READ, unless <i>vdev</i> of Write link is the one forced
VMLINK <DET>	No R/O to detach message	Same result with RR or READ, unless <i>vdev</i> of Write link is the one forced
VMLINK <REL>	No action	Same result with RR or READ, unless <i>vdev</i> of Write link is the one forced
VMLINK with (WRITE option	Keep existing Write Link and access disk	Same result with M
VMLINK with <i>vdev</i> specified (a free one) and (WRITE option	Detach existing link, link at requested <i>vdev</i> in Write mode and access disk	Same result with M
VMLINK with <i>vdev</i> specified (one in use) and (WRITE option	Error message, no link done, unless <i>vdev</i> requested is where it is already linked	Same result with M
VMLINK with <i>vdev</i> specified (one in use) with FORCE and WRITE options	Detach current disk at <i>vdev</i> and detach current link to disk. Link disk at requested <i>vdev</i> in Write mode and access disk	Same result with M
VMLINK with specified file mode range and (WRITE option	Keep existing link and access in range	Same result with M
VMLINK with specified single file mode and (WRITE option	Keep existing link and access at mode. This will release whatever was at that file mode	Same result with M
VMLINK <DET>	Detach current disk	Same result with M
VMLINK <REL>	No action	Same result with M

Table 19. VMLINK Linking Behavior When Already Linked in WRITE (Accessed)

Already linked in WRITE (accessed too)	Action	Notes
VMLINK with no options or linking details	Link in Read mode and access. Leaves Write link as is	Same result with RR or READ
VMLINK with <i>vdev</i> specified (a free one)	Link at <i>vdev</i> in Read mode and access. Leaves Write link as is	Same result with RR or READ
VMLINK with <i>vdev</i> specified (one in use)	Error message, no link done	Same result with RR or READ
VMLINK with <i>vdev</i> specified (one in use) with FORCE option	Detach current disk at <i>vdev</i> and link this disk at that <i>vdev</i> Read mode and access disk. Leaves Write link as is.	Same result with RR or READ, unless <i>vdev</i> of Write link is the one forced
VMLINK with specified file mode range	Link in Read mode and access in range. Leaves Write link as is	Same result with RR or READ, unless <i>vdev</i> of Write link is the one forced
VMLINK with specified single file mode	Link in Read mode and access at mode. This will release whatever was at that file mode. Leaves Write link as is.	Same result with RR or READ, unless <i>vdev</i> of Write link is the one forced
VMLINK <DET>	No R/O to detach message	Same result with RR or READ, unless <i>vdev</i> of Write link is the one forced
VMLINK <REL>	No action	Same result with RR or READ, unless <i>vdev</i> of Write link is the one forced
VMLINK with (WRITE option	Keep existing Write Link and existing access	Same result with M
VMLINK with <i>vdev</i> specified (a free one) and (WRITE option	Detach existing link, link at requested <i>vdev</i> in Write mode and access disk	Same result with M
VMLINK with <i>vdev</i> specified (one in use) and (WRITE option	Error message, no link done, unless <i>vdev</i> requested is where it is already linked	Same result with M
VMLINK with <i>vdev</i> specified (one in use) with FORCE and WRITE options	Detach current disk at <i>vdev</i> and detach current link to disk. Link disk at requested <i>vdev</i> in Write mode and access disk	Same result with M
VMLINK with specified file mode range and (WRITE option	Keep existing link and existing access if in range, otherwise reaccess disk in range	Same result with M
VMLINK with specified single file mode and (WRITE option	Keep existing link and access at mode. This will release whatever was at that file mode	Same result with M
VMLINK <DET>	Detach current disk	Same result with M
VMLINK <REL>	Release disk but keep link	Same result with M

Table 20. VMLINK Linking Behavior When Already Linked Multiple Times (READ and WRITE)

Already linked multiple times (both Read and Write)	Action	Notes
VMLINK with no options or linking details	Use one of the already existing Read links and access. Detaches all other Read links. Leaves Write links as is.	Same result with RR or READ
VMLINK with <i>vdev</i> specified (a free one)	Link at <i>vdev</i> in Read mode and access. Detaches all other Read links. Leaves Write links as is	Same result with RR or READ
VMLINK with <i>vdev</i> specified (one in use)	Error message, no link done. Unless <i>vdev</i> requested is where a Read link already is.	Same result with RR or READ
VMLINK with <i>vdev</i> specified (one in use) with FORCE option	Detach current disk at <i>vdev</i> . Link disk at requested <i>vdev</i> and access disk. Detach all other Read links. Leaves Write links as is.	Same result with RR or READ, unless <i>vdev</i> of Write link is the one forced
VMLINK with specified file mode range	Use existing Read link and access in range. Detach all other Read links. Leaves Write links as is	Same result with RR or READ, unless <i>vdev</i> of Write link is the one forced
VMLINK with specified single file mode	Use existing Read link and access at mode. This releases whatever was a that mode. Detach all other Read links. Leaves Write links as is	Same result with RR or READ, unless <i>vdev</i> of Write link is the one forced
VMLINK <DET>	Detaches all Read links	Same result with RR or READ, unless <i>vdev</i> of Write link is the one forced
VMLINK <REL>	Releases any access Read links	Same result with RR or READ, unless <i>vdev</i> of Write link is the one forced
VMLINK with (WRITE option	Use existing Write Link and access the disk. Detach all other Write links. Leave Read links as is.	Same result with M
VMLINK with <i>vdev</i> specified (a free one) and (WRITE option	Link is Write mode at <i>vdev</i> requested and access disk. Detach all other Write links. Leaves Read links as is.	Same result with M
VMLINK with <i>vdev</i> specified (one in use) and (WRITE option	Error message, no link done, unless <i>vdev</i> requested is where it is already linked in either Read/Write mode.	Same result with M

Table 20. VMLINK Linking Behavior When Already Linked Multiple Times (READ and WRITE) (continued)

Already linked multiple times (both Read and Write)	Action	Notes
VMLINK with <i>vdev</i> specified (one in use) with FORCE and WRITE options	Detach current disk at <i>vdev</i> and detach current link to disk. Link disk at requested <i>vdev</i> in Write mode and access disk. Detach all other Write links. Leaves Read links as is.	Same result with M, unless <i>vdev</i> of Write link is the one forced
VMLINK with specified file mode range and (WRITE option	Keep existing link and existing access if in range, otherwise reaccess disk in range. Detach all other Write links. Leaves Read links as is.	Same result with M, unless <i>vdev</i> of Write link is the one forced
VMLINK with specified single file mode and (WRITE option	Keep existing link and access at mode. This will release whatever was at that file mode. Detach all other Write links. Leaves Read links as is.	Same result with M, unless <i>vdev</i> of Write link is the one forced
VMLINK <DET>	Detaches all Write links	Same result with M, unless <i>vdev</i> of Write link is the one forced
VMLINK <REL>	Releases any accessed Write links	Same result with M, unless <i>vdev</i> of Write link is the one forced

Table 21. VMLINK Linking Behavior When Already Linked But Directory Has Moved

Already linked but directory has moved	Action	Notes
VMLINK with no options or linking details	Link in Read mode and access	Same result with RR or READ
VMLINK with <i>vdev</i> specified (a free one)	Link at <i>vdev</i> in Read mode and access.	Same result with RR or READ
VMLINK with <i>vdev</i> specified (one in use)	Error message, no link done.	Same result with RR or READ
VMLINK with <i>vdev</i> specified (one in use) with FORCE option	Detach current disk at <i>vdev</i> . Link disk at requested <i>vdev</i> Read mode and access disk.	Same result with RR or READ
VMLINK with specified file mode range	Link in Read mode access in range.	Same result with RR or READ
VMLINK with specified single file mode	Link in Read mode and access at mode. This releases whatever was at that mode.	Same result with RR or READ
VMLINK <DET>	No R/O link to detach	Same result with RR or READ
VMLINK <REL>	No action	Same result with RR or READ
VMLINK with (WRITE option	Link in Write mode and access	Same result with M
VMLINK with <i>vdev</i> specified (a free one) and (WRITE option	Link at <i>vdev</i> in Write mode and access	Same result with M

Table 21. VMLINK Linking Behavior When Already Linked But Directory Has Moved (continued)

Already linked but directory has moved	Action	Notes
VMLINK with <i>vdev</i> specified (one in use) and (WRITE option)	Error message, no link done	Same result with M
VMLINK with <i>vdev</i> specified (one in use) with FORCE and WRITE options	Detach current disk at <i>vdev</i> and link this disk at that <i>vdev</i> Write mode and access disk	Same result with M
VMLINK with specified file mode range and (WRITE option)	Link in Write mode and access in range	Same result with M
VMLINK with specified single file mode and (WRITE option)	Link in Write mode and access at mode. This will release whatever was at that file mode	Same result with M
VMLINK <DET>	No R/W to detach message	Same result with M
VMLINK <REL>	No action	Same result with M

Table 22. VMLINK Linking Behavior When Already Linked But No Longer Have Authority

Already linked but no longer have authority to it	Action	Notes
VMLINK with no options or linking details	Keep existing Link and access disk	Same result with RR or READ
VMLINK with <i>vdev</i> specified (a free one)	Detach existing link, link at requested <i>vdev</i> FAILS because no longer have authority. Old link is now gone.	Same result with RR or READ
VMLINK with <i>vdev</i> specified (one in use)	Error message, no link done, unless <i>vdev</i> requested is where it is already linked	Same result with RR or READ
VMLINK with <i>vdev</i> specified (one in use) with FORCE option	Detach current disk at <i>vdev</i> and detach current link to disk. Link disk at requested <i>vdev</i> FAILS because no longer have authority.	Same result with RR or READ
VMLINK with specified file mode range	Keep existing link and access in range	Same result with RR or READ
VMLINK with specified single file mode	Keep existing link and access at mode. This will release whatever was at that file mode	Same result with RR or READ
VMLINK <DET>	Detach current disk	Same result with RR or READ
VMLINK <REL>	Release disk if accessed otherwise no action	Same result with RR or READ

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. You can use PRT, LETGO, and FL in place of the corresponding CMS command names after you enter the command:

```
synonym mysyn
```

The ABBREV function allows you to use the information in the third column of the record. The ABBREV function is in effect by default, but you may insure it is on by entering the SET ABBREV ON command. This function allows you to truncate any of your synonyms to the minimum number of characters specified in the third or last column. For example, you could enter P for PRINT or LETG for RELEASE. To load 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.

Note:

1. An exec procedure having a synonym defined for it can be loaded 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. For more information about DLCS, see [z/VM: CMS Commands and Utilities Reference](#) and [z/VM: CMS Application Development Guide](#).
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 TRANSLAT command to control whether they are recognized by CMS. The SET ABBREV command controls whether 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* means 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 to allow translations. CMS only recognizes translations for commands entered from the command line or those called with the REXX/VM 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 TRANSLAT OFF. Also, a command will not be translated if it is issued from another program using the search hierarchy for SVC 202, or using the REXX/VM interpreter SVC 202 search hierarchy (ADDRESS COMMAND), or the equivalent search function in EXEC 2 (&PRESUME &COMMAND CMS). For more information on the CMS command search function for translations, see *z/VM: XEDIT Commands and Macros Reference*, and for more information on how CMS uses the translation and synonym tables to find commands, see *z/VM: CMS Commands and Utilities Reference*.

CMS Command Search Order

When you enter a command in the CMS environment, CMS uses the following search order to locate the command. When found, CMS stops the search and processes the command.

Note: 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.

1. Search for an exec with the specified command name:
 - a. Search for an exec in storage. If an exec with this name is found, CMS determines whether the exec has a USER, SYSTEM, or SHARED attribute. If the exec has the USER or SYSTEM attribute, it is executed.

If the exec has the SHARED attribute, the INSTSEG setting of the SET command is checked. When INSTSEG is ON, all accessed directories and minidisks are searched for an exec with that name. (To find a file in a directory, read authority is required on both the file and the directory.) If an exec is found, the file mode is compared to the file mode of the CMS installation saved segment. If the file mode of the saved segment is equal to or higher (closer to A) than the file mode of the directory or minidisk, then the exec on the saved segment is executed. Otherwise, the exec in the directory or on the minidisk is executed. However, if the exec is in a directory and the file is locked, the execution will fail with an error message.
 - b. Search the table of active (open) files for a file with the specified command name and a file type of EXEC. If more than one open file is found, the one opened first is used.
 - c. Search for a file with the specified command name and a file type of EXEC on any currently accessed disk or directory, using the standard CMS search order (A through Z).
2. Search for a translation or synonym of the specified command name. If found, search for an exec with the valid translation or synonym by repeating Step 1.
3. Search for a module with the specified command name:
 - a. Search for a nucleus extension module.
 - b. Search for a module in the transient area.
 - c. Search for a nucleus-resident module.
 - d. Search the table of active (open) files for a file with the specified command name and a file type of MODULE. If more than one open file is found, the one opened first is used.
 - e. Search for a file with the specified command name and a file type of MODULE on any currently accessed disk or directory, using the standard CMS search order (A through Z).
4. Search for a translation or synonym of the specified command name. If found, search for a module with the valid translation or synonym by repeating Step 3.

If the command is not known to CMS (that is, all of the above fails), it is passed to CP.

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 processed. As a result, you would be able to XEDIT the file SAUCES COOKBOOK.

For more information on the CMS command search order, see [z/VM: CMS Commands and Utilities Reference](#). For more information on searching for a translation or synonym, see the SET TRANSLATE or SYNONYM command in [z/VM: CMS Application Development Guide](#).

CMS Command Execution Characteristics

For a list of the CMS commands that require special consideration when called from a user program, see [z/VM: CMS Commands and Utilities Reference](#). For example, a program running in free storage can cause part of its own code to be overwritten if it calls a nonrelocatable CMS command, which also runs in free storage. To avoid conflicts with nonrelocatable CMS commands, you should ensure that your user programs are relocatable.

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. For example, the following command would convert the file, DATA FILE A1, to fixed-length 130-character records:

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

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 command with TRUNC option to ensure that all trailing blanks are truncated:

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

If you specify the COPYFILE command with 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. For example, suppose SHORT RECS is a variable-length file with its longest existing record being of length, *L*. If you wish to pad each record in SHORT RECS with asterisks to make each record in the file of length *L*, you would enter the following command:

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

Note: If SHORT RECS was already a fixed-length file, it would not be 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.

Chapter 6. Using Real Printers, Punches, and Readers

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 CP. Each output unit is known to CP as a *unit record spool file* or *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 SPOOL command. To find out the current characteristics of your devices, enter the command:

```
query ur
```

Following is an example of the response you receive:

```
RDR 000C CL A NOCONT HOLD EOF READY
    000C 2540 CLOSED NOKEEP NORESCAN
PUN 000D CL A NOCONT NOHOLD COPY 001 READY FORM STANDARD
    000D TO MSGDE DIST 2G47-706 DEST OFF
    000D FLASH 000 CHAR MDFY 0 FCB
    000D 2540 NOEOF CLOSED NOKEEP NOMSG NONAME
    000D SUBCHANNEL = 0009
PRT 000E CL A NOCONT NOHOLD COPY 001 READY FORM STANDARD
    000E TO MSGDE DIST 2G47-706 FLASHC 000 DEST OFF
    000E FLASH CHAR MDFY FCB
    000E 1403 NOEOF CLOSED NOKEEP NOMSG NONAME
    000E SUBCHANNEL = 000A
```

Note: Spool files are managed by CP, not CMS. Therefore they cannot reside in an SFS file pool.

Some Options Available on the CP SPOOL Command

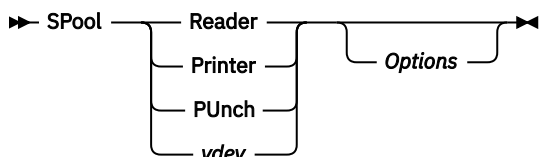
As mentioned in the previous section, you can use the CP SPOOL command to:

- Set control options for one or more of your virtual unit record devices.

Note: When you use the CP SPOOL command to do this, you do not change the status of spool files that already exist, but rather set the characteristics for subsequent output.

- Modify the disposition of a file after it has been processed. (Examples of using CP SPOOL for this purpose are given in [“Altering Spool Files”](#) on page 165.)

A simplified format for the CP SPOOL command is:



When you enter a CP 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). Listed below are descriptions of some commonly used options (the detailed command format and information on all options can be found in *z/VM: CP Commands and Utilities Reference*):

- **CLASS c**

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 z/VM 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 CP SET MSG OFF command, you would 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
```

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, suppose you entered the following sequence of commands:

```
spool punch cont to brown
punch asm1 assemble
```

```
punch asm2 assemble
punch asm3 assemble
spool punch nocont
close punch
```

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 SFS directory, however, CMS creates separate files.

Note: Separate files are created in this case using the READCARD command. For more information, see [z/VM: CMS Commands and Utilities Reference](#).

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:

- Always use SENDFILE, which resets any continuous spooling options in effect.
- 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 CP SPOOL command to specify the destination of the punch file.

The * operand places output in your own virtual reader:

```
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 z/VM 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 z/VM 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 refer to 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 following attributes using the CP CHANGE command:

- Spool file name and spool file type (this information appears on the first page or first card of output)
- Distribution code
- Number of copies
- Class
- Hold status.

For example, to change all printer files that are in a hold status to a nohold status, you could enter the following command:

```
change printer all nohold
```

If you decide that you would like to delete a particular spool file or files, you could delete a file or files with the CP PURGE command. For example, to purge printer file 0142 from previous printer display, you would enter:

```
purge printer 0142
```

For another example, you can purge the entire contents of your reader with the command:

```
purge reader all
```

After you have punched a file to some other user, or alternatively, sent a file using the SENDFILE command, you cannot change its characteristics or delete it, unless you restore it to your own virtual reader. You can accomplish this with the TRANSFER command. For example:

```
transfer all from usera
```

This command returns to your virtual reader all punch files that you spooled to and that still exist in USERA's virtual reader.

You can determine, for your reader or printer files, in what order they should be read or printed. If you decide you want printer file 0180 printed before 0142, you would enter:

```
order printer 0189 0142
```

The CP spooling system is very flexible, and can be a useful tool, if you understand and use it properly. *z/VM: CP Commands and Utilities Reference* contains complete format and operand descriptions for the CP commands used to modify spool files.

For SFS, there is an alternate way to send files to other users. Use the COPYFILE command 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. For more information, see [Chapter 3, "Using the Shared File System," on page 37](#).

Using the RECEIVE Command to Receive a File

You can use the RDRLIST command to display information about the files in your virtual reader. From RDRLIST you can receive files from your reader to an accessed minidisk or SFS directory. To receive a file, you can enter the RECEIVE command directly from the command line, or from the RDRLIST environment.

For example, you could enter the following RECEIVE command from the RDRLIST environment to receive a file to your minidisk or directory accessed as file mode B:

yourid	RDRLIST	A0	V	164	Trunc=164	Size=3	Line=1	Co			
Cmd	Filename	Filetype	Class	User	at	Node	Hold	Records	Date	Time	
	VACATION	NOTE	PUN	A	KATHY	DEPT	NONE	5	07/07	10:3:11	
	MEMO	SCRIPT	PUN	A	MGR	DEPT	NONE	22	07/07	10:3:11	
receive	/	=	bSCRIPT	PUN	A	MGR	DEPT	NONE	17	07/07	10:2:50

For additional examples of using the RDRLIST or RECEIVE command, see *z/VM: CMS Primer*. For more information on command format and options, see *z/VM: CMS Commands and Utilities Reference*.

Sending Files or Notes

You can use the SENDFILE command to send a copy of a file to the virtual card reader of another user or to your own virtual reader. To use the SENDFILE command, you could specify the SENDFILE command from the CMS command line, including the name of the file and the user ID or nickname. For example:

```
sendfile prog6 assemble a jones
```

The file PROG6 ASSEMBLE A would be sent to user ID JONES. Or, you can enter the SENDFILE command with no parameters which would cause a data entry screen to appear. You would then indicate the file or files you wish to send and the user or users you wish to send to.

You can also use the NOTE command to send a note to the virtual reader of one user or a group of users. Many options are available with the NOTE command. For examples of using SENDFILE and NOTE commands, see [z/VM: CMS Primer](#). For command formats and options, see [z/VM: CMS Commands and Utilities Reference](#).

Once you have sent the file, the RECEIVE command can be used to receive the contents of the file.

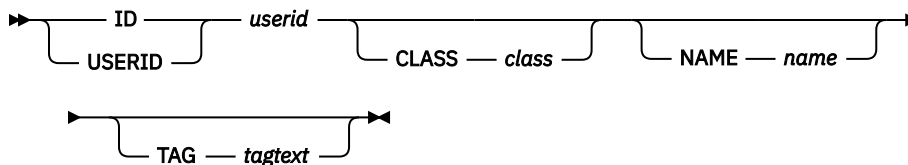
Using Your Card Punch and Card Reader in CMS

The following sections discuss the use of real cards and card punch under CMS.

Using Real Cards

To direct a real card deck to your own or another user's virtual machine reader, punch a CP ID card to precede the deck and read the cards into the system card reader.

The format of a CP ID card is:



The ID or USERID keyword must begin in column 1, and each keyword and operand must be separated by at least one blank. *userid* is the user to receive the spool file containing the card deck, *class* and *name* are optional values that can be assigned to the spool file, and *tagtext* is optional data that will be associated with the spool file. If *tagtext* is specified, it must be the last operand on the card.

Use the **RECEIVE** command to receive the file onto one of your accessed directories or minidisks.

Using Your Card Punch

Use the PUNCH command to create a punched copy of a CMS file. Once you use the 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. An example of the PUNCH command is:

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. For example:

```
receive = prog6 assemble
```

For more information on using the PUNCH and DISK DUMP commands, see [z/VM: CMS Commands and Utilities Reference](#).

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 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 REXX, EXEC 2, or CMS EXEC, use the following CMS commands to punch CMS files:

- EXECIO
- PUNCH
- DISK DUMP

Using the MOVEFILE Command

Use the MOVEFILE command, 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.

For more information on using the FILEDEF command, see [z/VM: CMS Commands and Utilities Reference](#).

Chapter 7. Using Tapes

You can use CMS to read from or write to any tape that an attached tape device is capable of reading or writing. You can control tape devices at a variety of levels.

There are two main reasons for having data on tape:

- Data can be stored less expensively on tape than on DASD
- You can move data from one system to another using a tape volume.

You can use CMS to:

- Read tapes that were created by other operating systems
- Create tapes for other operating systems to read
- Read and create standard labeled tapes
- Read and create DDR tapes, which may be done by the DDR program running stand-alone (in a virtual machine or a real machine)
- Create tapes in a few formats that are intended to be read back only by CMS.

CMS provides 3 ways of using tapes:

- CMS native tape commands and execs
- A General Programming Interface for assembler language programs based on the *native tape macros*
- OS Simulation.

CMS works with 16 virtual tape devices, named TAP0, TAP1, TAP2, ...TAPF. Each tape device must be attached with a specific virtual device number, as given in this table:

Device Name	Device Number
TAP0	180
TAP1	181
TAP2	182
TAP3	183
TAP4	184
TAP5	185
TAP6	186
TAP7	187
TAP8	288
TAP9	289
TAPA	28A
TAPB	28B
TAPC	28C
TAPD	28D
TAPE	28E
TAPF	28F

The most common device to use is TAP1, and CMS generally uses it by default.

Note: A noteworthy exception to this is the FILEDEF command, which uses TAP2 by default.

CMS uses virtual tape devices, because it runs in a virtual machine. A *virtual tape device* is a simulation of a tape device created by CP. CP always associates one of its real tape devices with each virtual tape device, allowing you to read and write real physical tapes. CP simulates the interpretation and recording of the virtual tape by reading and writing the real tape. To use a tape with CMS, you must create a virtual tape device. Real tapes have to be interchangeable on the real tape device. Procedures to do this vary from one installation to another. Sometimes they are partially automated, but you may just need to simply talk to an operator informally. The most basic means of creating a virtual tape device is to issue the CP system operator command, ATTACH.

CMS is not capable of sharing tape devices. When CMS is using a tape device, the virtual machine in which it is running must be the only host accessing the device. Therefore, the virtual tape device must *not* be created as a shareable device. A *shareable virtual device* is one that can be accessed by other hosts. Such a virtual tape device is created when the NOASSIGN option is used with the ATTACH command.

CP guarantees that a nonshareable device will be accessible by the CMS virtual machine only, no other host. A nonshareable device is created by the ATTACH command *without* the NOASSIGN option. Shareable virtual tape devices are mainly used with virtual machines running MVS.

When a virtual tape device has been successfully created, which is often referred to as *attached*, you get a CP message similar to this:

```
TAPE 181 ATTACHED
```

Using the TAPE Command

The TAPE command is perhaps the most basic way to use tapes in CMS.

The TAPE command performs some basic tape functions:

- Dumps CMS files to tape
- Restores previously dumped CMS files
- Displays contents of previously dumped tape
- Positions tape
- Writes standard labels on a tape volume to initialize it
- Displays the VOL1 label from a tape
- Selects a recording format for future tape writes
- Displays characteristics of a virtual tape device
- Writes tapemarks
- Unloads a tape
- Erases a section of tape (the *erase gap* function).

When TAPE dumps CMS files, it dumps them in a special format designed to be read back only by the TAPE command.

Examples of the TAPE Command

The following examples show how to create a CMS tape with three tape files on it, each containing one or more CMS files, and then how you, or another user, might use the tape at a later time.

The examples are in the form of a terminal session and show, 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
tell operator Please attach 500 to smith as 181 and mount a scratch tape	Send a TELL message to the system operator asking to have virtual tape device 181 created (attached) for you and associated with the real tape device 500. Ask the system administrator scratch tape on real device 500.
Tape 181 attached	Message indicates that the operator did it. Virtual device 181 (CMS device TAP1) exists now.
tape dump * assemble a	The TAPE command writes to the tape, all files on your A mode minidisk, and its extensions, that have the file type ASSEMBLE. Note: TAP1 is the default device for the TAPE command.
Dumping ... PRG1 ASSEMBLE A1 PRG2 ASSEMBLE A1 :	The TAPE command responds by printing the file identification of each file written to tape.
PRG9 ASSEMBLE A1 Ready;	The last file, PRG9 ASSEMBLE, is dumped.
tape wtm Ready;	The TAPE WTM command writes a tape mark that will separate this group of files from subsequent files.
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 rew Ready;	The tape is rewound. Note: Although there was no command to specifically write them, there are two tape marks after the text library on this tape, because of the TAPE command defaults. For more information, see “Tape Marks on TAPE DUMP tapes” on page 181 .

Terminal Display	Comments
<pre>tape scan (eof 4 eot Scanning ... PRG1 ASSEMBLE A1 PRG2 ASSEMBLE A1 PRG3 ASSEMBLE A1 PRG4 ASSEMBLE A1 PRG5 ASSEMBLE A1 PRG6 ASSEMBLE A1 PRG7 ASSEMBLE A1 PRG8 ASSEMBLE A1 PRG9 ASSEMBLE A1</pre>	The tape is scanned to verify that all of the files are on it. The EOT option ensures that the scanning does not continue past the logical end of the tape.
<pre>End-of-file or end-of-tape MYLIB MACLIB A1 CMSLIB MACLIB S2 End-of-file or end-of-tape MYLIB TXTLIB A1</pre>	Tape mark indication.
<pre>End-of-file or end-of-tape End-of-file or end-of-tape Ready;</pre>	Two tape marks indicate the end of the tape. See 6 in Figure 59 on page 173 .
<pre>detach 181 Tape 181 Detached</pre>	The CP DETACH command deletes (<i>detaches</i>) the virtual tape device and thus frees up the associated real device, after unloading the tape. The real tape unloaded at this time contains the data written virtually with the preceding commands.

The following table shows the commands to be entered after building a tape.

Terminal Display	Comments
<pre>Tape 181 Attached</pre>	Message indicating a tape device has been attached.
<pre>tape load prg4 assemble</pre>	One file is to be read onto a minidisk.
<pre>Loading ... PRG4 ASSEMBLE A1 Ready;</pre>	The TAPE command displays the name of the file loaded. Any existing file with the same file name and file type is erased. See 1 in Figure 59 on page 173 .
<pre>tape scan Scanning ... PRG5 ASSEMBLE A1 PRG6 ASSEMBLE A1 PRG7 ASSEMBLE A1 PRG8 ASSEMBLE A1 PRG9 ASSEMBLE A1</pre>	The remainder of the first tape file is scanned.
<pre>End-of-file or end-of-tape Ready;</pre>	Indication of end of first tape file. See 3 in Figure 59 on page 173 .

Terminal Display	Comments
<pre>tape scan Scanning ... MYLIB MACLIB A1 CMSLIB MACLIB S2 End-of-file or end-of-tape Ready;</pre>	The second tape file is scanned. See 4 in Figure 59 on page 173 .
<pre>tape bsf 2 Ready;</pre>	The tape is backed up and in front of the first tape file. See 2 in Figure 59 on page 173 .
<pre>tape fsf Ready;</pre>	The tape is forward spaced past the tape mark. See 3 in Figure 59 on page 173 .
<pre>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;</pre>	The next two tape files (groups of CMS files separated by tape marks) are going to be read. See 5 in Figure 59 on page 173 .
<pre>detach 181 Tape 181 detached</pre>	The real tape is unloaded and the virtual device detached.

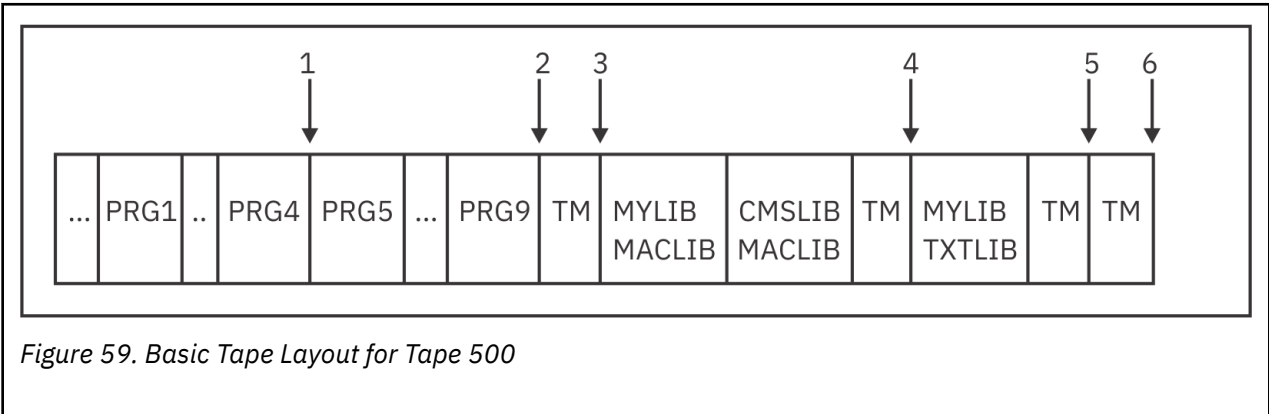


Figure 59. Basic Tape Layout for Tape 500

Using the VMFPLC2 Command

The VMFPLC2 command is almost identical to the TAPE command, but it dumps and restores CMS files in a slightly different format.

The VMFPLC2 format is superior to the TAPE format, because information about a file is recorded prior to the contents of the file instead of at the end. This lets you locate a particular file on the tape faster than with the TAPE format.

However, there are many existing tapes recorded in the TAPE format that cannot be loaded with VMFPLC2. Also, some earlier VM systems do not have the VMFPLC2 command. VMFPLC2 format tapes cannot be read on them. Finally, in previous levels of VM (before z/VM), VMFPLC2 was documented only as a system administration command. Therefore, many users are not aware of it.

Note: VM product and service distribution tapes are in VMFPLC2 format.

Using the VMFPLC2 Command

The following example shows you how to create a backup of your private minidisk or directory, and later restore a particular file from the backup. The VMFPLC2 command is used because it provides better performance than the TAPE command, when loading a single file or a small number of files from a large group of files.

Terminal Display	Comments
<pre>q tape TAPE 181 ON TAPE 0EAA Ready;</pre>	<p>Check to see if a tape is attached.</p> <p>Ensure there are no file modes that are extensions of the file mode to be dumped.</p> <p>Extensions of the file mode to be dumped are always checked for files that match the selection criteria, and any file that matches is also dumped.</p>
<pre>vmfplc2 dump * * a Dumping ... ALL NOTEBOOK A0 LASTING GLOBALV A1 PROFILE EXEC A1 : PRG9 ASSEMBLE A1 Ready;</pre>	<p>The VMFPLC2 command is used because it provides better performance when specific files are loaded from tape. To use VMFPLC2 to read the tape, the tape must be written with VMFPLC2. All the files on the A file mode are written to tape.</p> <p>The tape can now be removed and stored for future use.</p> <p>If you later decide to restore your old copy of the XEDIT profile, you must first get your backup tape mounted and ready for use. Then, use the VMFPLC2 command.</p>
<pre>vmfplc2 load profile xedit a Loading ... PROFILE XEDIT A1 Ready;</pre>	<p>The PROFILE XEDIT file is located on the tape and written to file mode A. Because a number of files have to be examined to find the requested file, this may take a noticeable amount of time. The more files that were on your file mode A that were before (alphabetically) the file you requested, the longer it will take.</p> <p>The tape can now be removed and stored away again.</p>

For more information about the VMFPLC2 command, see [z/VM: CMS Commands and Utilities Reference](#).

Using the TAPPDS Command

The TAPPDS command loads data from a tape created by the MVS IEHMOVE or IEBTPCH utility into CMS files. It also reads tapes formatted for input to the MVS IEBUPDTE utility into CMS files.

If the tape contains an unloaded Partitioned Data Set (PDS) created by the IEHMOVE utility, TAPPDS creates a single CMS file corresponding to each former PDS member.

For more information about the TAPPDS command, see [z/VM: CMS Commands and Utilities Reference](#).

Using the TAPEMAC Command

The TAPEMAC command loads data from a tape that contains an unloaded PDS created by the MVS IEHMOVE utility. It loads it into one or more CMS maclibs, with maclib members corresponding to the former PDS members.

For more information about the TAPEMAC command, see [z/VM: CMS Commands and Utilities Reference](#).

Using the MOVEFILE Command

In Chapter 6, “Using Real Printers, Punches, and Readers,” on page 163, the MOVEFILE command was used along with the FILEDEF command to place a file in your virtual reader. MOVEFILE is also the easiest CMS method for reading and writing labeled tapes. It reads and writes the most basic data formats.

MOVEFILE copies data from one place to another using the OS simulation function. You use it with the FILEDEF and LABELDEF commands, and either the source, destination, or both can be a tape.

Because of the basic functions of OS simulation, MOVEFILE can transfer data to tape from various sources:

- Card reader
- CMS file
- Another tape
- OS-formatted minidisk.

Similarly, it can transfer data from tape to various destinations:

- Card punch
- Printer
- CMS file
- Another tape
- OS-formatted minidisk.

For more information about the MOVEFILE command, see [z/VM: CMS Commands and Utilities Reference](#), and [z/VM: CMS Application Development Guide for Assembler](#).

User Programs

You can run your own program under CMS to access tapes in the following ways.

CMS Native Tape Macros

CMS provides the following four macros for use as programming interfaces. They can be used in an assembler language program to access tape data:

RDTAPE

Reads a record from a tape

WRTAPE

Writes a record on a tape

TAPECTL

Performs a variety of tape control functions

TAPESL

Writes and processes tape labels.

There are also other general-use programming interfaces that apply to CMS devices in general that are applicable to tape. For example, the CMSDEV macro gets you information about a tape device.

OS Simulation

You can design a tape-processing program to run on both MVS and VM. The program would run on VM using CMS OS simulation. Such a program can process both tapes created by CMS, and standard label tapes created by an MVS system. Also, standard label tapes created by CMS OS simulation can be processed by applications running on an MVS system. CMS simulates MVS system calls, such as CLOSE, GET, OPEN, PUT, READ, and WRITE. OS simulation, like MVS, treats all data sets as generic regardless of what devices they are on. All the general rules of using OS simulation apply to tape data.

The FILEDEF and LABELDEF commands are critical parts of OS simulation.

Note: The MOVEFILE command is essentially a program which runs under CMS and uses the OS simulation services. For more information about the MOVEFILE command and CMS user programs used for accessing, see [z/VM: CMS Commands and Utilities Reference](#), [z/VM: CMS Application Development Guide for Assembler](#), and [z/VM: CMS Macros and Functions Reference](#).

Also see [“Using Tape Library Dataservers under OS Simulation”](#) on page 182.

VSE Simulation

Using tapes with VSE simulation is similar to using tapes with OS simulation, except it is more restricted. CMS does not fully simulate some of the internal VSE control blocks that some programs (designed to run under VSE) access.

For more information about CMS user programs used for accessing tapes, see [z/VM: CMS Application Development Guide for Assembler](#) and [z/VM: CMS Macros and Functions Reference](#).

VSE/VSAM (AMSERV Command)

You may use the IMPORT and EXPORT functions of VSE/VSAM to transfer data between a tape and VSAM data sets. Use the AMSERV command to do this. The tape format created by EXPORT to be received by IMPORT is a special format used just for them.

You may use the REPRO function of VSE/VSAM to transfer data between one tape and another, or between a tape and a VSE-formatted non-VSAM disk. Use the AMSERV command to do this. The tape format, written by REPRO and expected by REPRO, is a standard VSE format, which is almost identical with the standard MVS tape formats. Considering VSE simulation only reads and writes the VSE/AF Version 1 file system, and the VSE/AF Version 1 is obsolete, this function does not have much use.

For more information about the VSE/VSAM (AMSERV command), see [z/VM: CMS Application Development Guide for Assembler](#).

Using the DDR Command (DASD Dump Restore)

The DDR command:

- Dumps data from a virtual disk to a tape
- Restores previously dumped data from a tape to a virtual disk
- Copies previously dumped tapes to other tapes.

The DDR tape data format is special to DDR. It includes raw images of disk tracks.

DDR has these advantages over other methods of getting data from disk to tape and back:

- It works with any virtual disk, regardless of the format of the data on it. All of the other methods of copying data from disk to tape require a certain disk format, for example, CMS file system format.
- Because it is dumping whole disk tracks, DDR is faster than the other methods.
- The DDR tape format can be created and read with the stand-alone DDR program.
- The DDR tape data format can use a compaction algorithm to fit more data on a tape than the other methods in some cases.

DDR also does other things that are not related to tape. For more information about the DDR command, see [z/VM: CP Commands and Utilities Reference](#).

Tape Recording Formats

One of the choices you sometimes have to make when creating tapes is the recording format. A tape recording format is defined as the relationship between data which is sent to or received from a tape device and the physical representation of that data on the recording medium. The recording format consists of factors like:

- Recording density (Bits per inch, BPI)
- The number of tracks used
- Type of medium (3480 cartridge, open reel, and so on)
- Whether data is compacted or uncompact by the device

- The way blocks are separated from one another on the medium
- Error correction information.

The following facts about recording formats will affect your choice of one:

- A specific tape device can only write certain recording formats.
- A specific recording format can only be read by certain tape devices.
- Some recording formats pack data into less space than others.
- Some recording formats allow you to move the data on and off the tape faster than others.
- Some recording formats have more integrity than others. That is, you have a smaller chance of not being able to read the data back or, worse, reading it back incorrectly.

The recording formats relevant to CMS are:

Recording Format	Description
NRZI	The Non-Return to Zero Inverted (NRZI) recording format is recorded on a standard open reel in 9 tracks with an effective data density within a block of 800 bytes per inch. It is often called <i>800 BPI format</i> .
PE	The Phase Encoding (PE) recording format is recorded on a standard open reel in 9 tracks with an effective data density within a block of 1,600 bytes per inch. It is often called <i>1600 BPI format</i> . PE is an improvement over NRZI in tape capacity and also integrity. In the PE format, for example, preambles and postambles on the data blocks allow the device to distinguish between blocks of data and noise. PE format is probably the most standard interchange format.
GCR	The Group Coded Recording (GCR) format is recorded on a standard open reel in 9 tracks with an effective data density within a block of 6,250 bytes per inch. It is often called <i>6250 BPI format</i> . GCR is an improvement over PE in tape capacity and also integrity, since GCR has more error detection and correction capability built in than PE. GCR is almost as standard as PE for interchange.
3480 Basic	The 3480 Basic format is recorded on a <i>3480 cartridge</i> . It fits considerably more data into a particular storage space than the open reel formats. It records on 18 tracks with an effective recording density of about 38,000 bits per inch. This format is often called <i>18 track format</i> or <i>38K BPI format</i> . The IBM 3480 was the first device that could read or write this format and while the format cannot be read by the traditional open reel devices, it is pretty common now and getting more so for installations to have 3480-type devices that can read it. 3480 Basic is a better interchange format than 3480 Compacted because all devices that can handle 3480 Compacted can handle 3480 Basic, but not all devices that can handle 3480 Basic can handle 3480 Compacted.
3480 Compacted	The 3480 Compacted format is just like 3480 Basic, except that it is a compacted format. For more information, see “Compacted and Noncompacted Recording Format” on page 178 .
3490 Basic	The 3490 Basic format is recorded on a <i>3480 cartridge</i> . It fits twice as much data on a volume as 3480 Basic Format by recording 36 tracks. It can only be read by fairly new devices, so that limits its value for interchange. The format records data from the beginning of the tape to the end, and then back down the tape to be beginning. This means rewinding after processing a whole tape is faster than with 3480 Basic format.

Recording Format	Description
3490 Compacted	The 3490 Compacted format is just like 3490 Basic except that it is a compacted format. For more information, see “Compacted and Noncompacted Recording Format” on page 178.
3590 Basic	The 3590 Basic format is recorded on a 3590 cartridge. The basic recording data format is the same for all 3590 drives, but is offset by the tracking density factor or the drive model; such as, Model B has 64 tracks, Model E has 128 tracks, Model H has 256 tracks. The greater the number of tracks, the greater capacity the drive has to store data.
3590 Compacted	The 3590 Compacted format is just like 3590 Basic except that the data is written to the tape in a compacted format, which saves space on the cartridge. For more information, see “Compacted and Noncompacted Recording Format” on page 178.
QIC	There are actually a variety of QIC tape recording formats, but VM doesn't have much of a need to distinguish between them. These formats are recorded on standard 1/4 inch cartridges. They are read and written by small, low performance tape devices which are usually attached to small personal systems. <i>There are no devices which read or write this format that can attach to z/VM systems.</i> We mention the format because older VM systems work with this format and you should know that it is not one that z/VM can use.

Compacted and Noncompacted Recording Format

A compacted recording format uses a data compaction algorithm to store data in less space than traditional uncompact formats. For example, data which is EBCDIC coding of text has lots of blank characters in it and very few Q characters, but both blanks and Q's take up 8 bits' worth of space on the tape. In a compacted format, blanks can be stored in less than 8 bits' worth of space, while Q's are stored in more than 8 bits' worth of space, and the data takes up less space on the tape overall. This is just an example of the concept. The actual algorithm is quite complex.

Some data compacts better than other data, so with a compacted format it is difficult to predict how much space a particular set of data will take up on the tape. In fact, some data may take up more space on the tape in a compacted format than in the corresponding uncompact format. As a worst case, the 3480 Compacted, 3490 Compacted, and 3590 Compacted recording formats should take about 5% more space than 3480 Basic, 3490 Basic, or 3590 Basic.

Most data compacts quite well. Two examples of data that does not compact well and may actually expand, are:

- Data which is already the output of a compaction algorithm (for example, a DDR dump with the COMPACT option).
- Enciphered data.

In both of these cases, the data contains very little pattern and redundancy, which is what makes data compaction algorithms work.

EBCDIC and ASCII text compacts very well. Binary measurement data does not compact well.

The facility on IBM tape devices that allows them to read and write compacted recording formats is called Improved Data Recording Capability (IDRC). On some devices, it is a feature (separately orderable), while on others it is standard. On some devices, the facility is not available at all.

Do not confuse compacted recording formats with simply writing compacted data in a noncompact recording format. Anybody can compact some data and then send it to the tape device. DDR, for example, has an option for this. But the recording format is the relationship between the data sent to the device and how the device puts it on the tape. It is defined within the device.

Compacting data and then writing it in a noncompacted recording format has essentially the same effect on storage capacity as writing uncompact data in a compacted recording format. But there are other differences:

- It takes time to compact data before sending it to the device, but when you use a compacted recording format, you can skip that step. The same applies to reading it back and decompacting it.
- The choice of recording format affects what devices you can use to write the data and to read it back. Whether you compact the data before sending it does not affect what devices you must use to read and write it.
- On the other hand, whether you compact the data beforehand affects what software you need to read and write the tape, whereas the choice of recording format has little effect on the software doing the reading and writing.

Device Recording Format Capabilities

This table shows all IBM devices which are capable of reading or writing any of the formats described in this section. Note that not all of them are supported for use by a z/VM system and are listed only because you may want to interchange tapes with non-z/VM systems.

Device	Recording Formats
2401, 2402, 2403, 2404, 2415, 2420	NRZI PE
2440	PE GCR
3410/11	PE
3410/11 with Dual Density Feature	NRZI PE
3420 Models 3, 5, 7	PE
3420 Models 3, 5, 7 with Dual Density Feature	NRZI PE
3420 Models 4, 6, 8	PE
3420 Models 4, 6, 8 with Dual Density Feature	PE GCR
3422	PE GCR
3424	PE GCR
3430	PE GCR

Device	Recording Formats
3480 without IDRC facility (see note 1)	3480 Basic
3480 with IDRC facility (see note 1)	3480 Basic 3480 Compacted
3490E (see note 1)	3480 Basic (see note 2) 3480 Compacted (see note 2) 3490 Basic 3490 Compacted
3590 (see note 3)	3590 Basic (see note 3) 3590 Compacted (see note 3)
8809	PE
9346	QIC
9347	PE
9348	PE GCR

Notes:

1. VM considers certain models of 3490 to be 3480 instead. For VM's purposes, these models are identical with 3480. All VM documentation, command syntax, and messages refer to these as 3480. All 3490s except the E models (called *3490E*) are in this category.
2. The 3490 can read the 3480 Basic and 3480 Compacted formats, but cannot write them.
3. 3590 tape drives are not compatible with prior tape cartridges, such as the 3480 or 3490 types. 3590 drives use a unique tape cartridge shared among the 3590 models. The 3590 tape drive models offer increasing capacity (number or tracks) throughout the model lines. It should also be noted that data which was written with a higher-capacity 3590 model cannot be read on a lower-capacity unit, because of tracking differences. However, a lower-capacity 3590 tape can always be read on a higher-capacity 3590 model unit. High-capacity 3590 models cannot write at a lower track capacity, and therefore cannot add data to a tape already written at lower capacity.

Selecting the Recording Format with CMS

You never have to select a recording format in order to *read* a tape. The device will sense the recording format and read the data properly (if the device is capable of reading that format at all).

When you *create* a tape, CMS will always select a recording format for you, if you do not want to select a specific one. But for all the reasons mentioned in the previous sections, you may want to select a specific one. If you select a recording format, and the device you are using cannot write in that format, CMS will not write anything and will give you an error message. Thus, even if the device can only write one recording format, it may be helpful to request that specifically. This protects you from various mistakes and misunderstandings in selecting a device.

Do not use a default recording format if for some reason you need a particular one. Defaults may change as the world of recording formats changes and as new devices are introduced.

To select a recording format, you use recording format options on the TAPE, VMFPLC2, FILEDEF, or ASSGN command, or on the native tape macros, or on DDR's OUTPUT control statement. For the exact syntax, see *z/VM: CMS Commands and Utilities Reference* and *z/VM: CMS Macros and Functions Reference*. Here are some examples:

```
TAPE DUMP * * A (3490B
TAPE DUMP * * A (DEN 6250
FILEDEF OUTMOVE TAP1 (18TRACK
WRTAPE MYBUF,L'MYBUF,MODE=3490B
WRTAPE MYBUF,L'MYBUF,MODE=(,6250)
```

You specify a recording format differently with CMS than with CP tape commands for historical reasons. In CP, a command has a MODE option to select a recording format. Also for historical reasons, the DDR command uses the CP style of recording format selection (with a MODE option on the OUTPUT control statement).

You can select a recording format even with an operation which is not a write operation. For example:

```
TAPE LOAD * * A (18TRACK
FILEDEF INMOVE TAP1 (18TRACK
FILEDEF OUTMOVE DISK PROFILE EXEC A
MOVEFILE
```

This is for historical reasons having to do with old 7 track tape.

Unless you are writing, your selection of recording format has no effect. Nonetheless, if you select a recording format, it must be one the device is capable of *writing* or CMS will recognize an error and not perform the operation.

Even if you are writing, the recording format options may be accepted and yet have no effect. Here's why: You cannot generally change the recording format of a tape in the middle. For example, you cannot record one file on a tape in PE format and the next one in GCR format; the device doesn't allow this. So if the tape is not positioned to the beginning of the volume, your selection of a recording format may be ignored. Still, though, CMS validates it, making sure that the device is capable of writing that format in general.

The one way that it is allowed to change recording formats in the middle of a tape is to go from the compacted to the uncompact version of a format, or uncompact to compacted. Sometimes, in fact, all the labels on a tape are recorded in 3480 Basic format, while the data blocks are recorded in 3480 Compacted format. The reason for this is that more devices can read 3480 Basic format, so even though a device cannot read the data on a tape, it may be able at least to read the labels. MVS always creates labeled tapes this way (assuming operating system services write the labels). CMS never does (with the same assumption).

The VMFPLC2 command uses a superior format for the data on the tape. Unless there is some reason the VMFPLC2 command will not be available to read the data back, you should use VMFPLC2 instead of TAPE. For more information, see [“Using the VMFPLC2 Command” on page 174](#).

Tape Marks on TAPE DUMP tapes

A tape should always have two or more tape marks at the end of the data, which is the conventional signal (to whatever tries to read the data) that this is the end. If a tape does not have these, a program may assume there is more data on the tape and keep reading past the end of the data. This causes unpredictable results and, with open reel tapes, may cause the device to run off the end of the tape and pull the tape from the supply reel. (This is not disastrous, but is rather inconvenient, since the tape must be manually retreaded.)

Additionally, you may want a single tape mark in the middle of your data to separate groups of files.

Tape marks take up space on the tape, though, and on a buffered tape device (like most newer tape devices, including the IBM 3480), writing tape marks severely slows down tape operations.

You can place tape marks on the tape explicitly with commands like TAPE WTM, but the TAPE DUMP command itself helps you with them. By default, TAPE DUMP will dump all the files you tell it to without any intervening tape marks, but with two tape marks after the last (or only) file. It leaves the tape positioned before the two tape marks. This means the next time you write to the tape, you write over those two tape marks. This is useful if you are going to do multiple successive TAPE DUMP commands, because it means no matter when you stop doing TAPE DUMPs, your tape ends up with two tape marks at the end and none in the middle.

But you may wish to have tape marks between all CMS files on the tape. For this, there is the WTM option. WTM causes TAPE DUMP to place tape marks between all files it dumps. The command still places two tape marks at the end of the data, but it leaves the tape positioned *between* the two. This means that if you do successive TAPE DUMP (WTM) commands, you end up with a tape that has one tape mark between every two CMS files, and two tape marks to mark the end of the data.

The only problem with the above process is that it may waste dumping time and tape space, because it writes tape marks only to write over them later. Writing tape marks causes a severe performance degradation on a buffered tape device (because it causes a buffer synchronization) and physically writing a tape mark and then backing over it takes time on any device. Furthermore, the process that CMS and the device use to accomplish the writing over of tape marks leaves, with some devices, a large section of unused space on the tape. This obviously cuts into your tape capacity.

To solve that problem, there is the NOEODTM option. This tells CMS not to worry about placing two tape marks at the end of the data. With this, you can do multiple successive TAPE DUMP (NOEODTM) commands without any costly buffer synchronizations or empty spaces on the tape. But remember that you still want two tape marks at the very end of the data, so after your last TAPE DUMP (NOEODTM), always do a TAPE WTM 2.

One other thing to be aware of, while taking advantage of the high speed of TAPE DUMP (NOEODTM), is that the device's buffer does not get synchronized before the TAPE command completes. This means that the TAPE command may complete with no error indication, but a later error may prevent the data from actually getting onto the tape. This is not generally a problem, because you eventually have to get those end of data tape marks written anyway, and if the tape marks get written successfully, you know all previously dumped data also got written successfully.

If you use both NOEODTM and WTM, the command puts a single tape mark after your last (or only) CMS file, and leaves the tape positioned after it. In this case, NOEODTM only gets you a slight performance improvement, but you still have a capacity improvement. You still have to add one tape mark at the very end, too, to make the two end of data tape marks.

Attention: The NOEODTM option defeats CMS's usual guarantee of end of data tape marks and buffer synchronization. You are responsible for ensuring the tape marks and synchronization are taken care of.

Using Tape Library Dataservers under OS Simulation

Tape Library Dataserver machines (such as the 3494, 3495, and 3595) use a robotic tape operator to automatically select, mount, and demount tapes in a mechanically controlled tape library. This library and its attached tape cartridge drives (such as the 3480, 3490, and 3590) are under the control of the Removable Media Services (RMS) system. z/VM provides an interface to the RMS system through DFSMS/VM.

If tapes are to be used under OS simulation on a Tape Library Dataserver, z/VM requires that:

- DFSMS/VM and RMS must be installed.
- The RMS CSL library (FSMPPSI CSLLIB) must be accessed. In addition, the following command must be issued to make the RMS CSL routines available to CMS:

```
rtload * (from fsmppsi
```

- The RMS system administrator must select one of the 16 SCRATCHx (where x is 0-F) library categories as the default SCRATCH processing category pool.

- The user should make sure that the default SCRATCH pool of tapes has enough physical tapes currently assigned to it to meet any application program needs.
- If unique VOLIDs are specified on a LABELDEF or FILEDEF statement, these VOLIDs must reside within the tape library to enable the Dataserver to mount the tapes.
- If the user premounts a tape on a Dataserver device with the DFSMS/VM MOUNT command, and the tape is to be used for any type of output, the Target Category should be set to VOLspecific in the MOUNT command parameters. This corresponds to the automatic system Target Category setting for output tape mounts.

CMS provides the following interfaces:

- If the LIBSRV option is specified on the FILEDEF command to indicate that tape mounts should be done on a Tape Library Dataserver machine, OS simulation calls the RMS interface to mount the tapes automatically for the user. It is suggested that this option be used whenever a Tape Library Dataserver is being used, to allow the system to mount all the tapes automatically for the user.
- If the tape drive currently in use is found to be under the control of a Tape Library Dataserver, or the FILEDEF command has been issued with the LIBSRV option, OS Simulation attempts to get subsequent multivolume tapes mounted automatically for the user through the native DMSTVS mounting service and the CMS native rewind and unload tape processing functions by calling the RMS FSMRMDMT (Demount) and FSMRMMNT (Mount) CSL routines.
- 'RUN' (rewind and unload) function processing for the CMS TAPE or VMFPLC2 command or the TAPECTL macro calls the RMS FSMRMDMT (Demount) CSL routine if a Tape Library Dataserver is found to be controlling the tape drive that the 'RUN' was issued against.
- Input tapes remain in the same category on demount, but output tapes are moved to the VOLspecific category to prevent accidental tape overwrite on the next scratch tape mount from the default SCRATCH tape pool.

OS Utility Programs

The TAPPDS command 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 SFS directory, or in the case of partitioned data sets, into multiple CMS files.

IEBTPCH

Sequential or partitioned data sets created by the IEBTPCH utility program must be deblocked 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 *
```

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 deblocked, 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 col1
```

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.

Part 3. z/VM HELP Facility

In addition to z/VM reference manuals, there is an online z/VM HELP facility that you can refer to while you are logged onto the system. The z/VM HELP facility is part of CMS.

Chapter 8, “Using the HELP facility,” on page 187 describes the z/VM HELP facility. [Chapter 8, “Using the HELP facility,” on page 187](#) describes the HELP facility and shows you how to use the HELP command to assist you in completing z/VM tasks.

Chapter 9, “Tailoring the HELP Facility,” on page 199 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 8. Using the HELP facility

An explanation of the structure and function of the HELP facility and examples are provided.

The z/VM HELP facility provides information about z/VM components, commands and subcommands, macros, statement, routines, and messages.

The HELP facility is accessed by using the CMS HELP command. For information about the syntax, operands, and options of the HELP command, see [HELP in z/VM: CMS Commands and Utilities Reference](#).

The HELP facility displays information that is in different kinds of HELP files. The different kinds of HELP files display different information and provide different interaction behaviors.

Menu files

Menu files display menus from which you can select command files and other menu files. Menu files are of two types:

1. Component menu files.
2. Task menu files.

For more information about menu files, see [“HELP Menus” on page 192](#).

Information files

Information files provide information about z/VM components, commands and subcommands, and messages. Information files are message files and command files.

Message files

Message files contain information about messages that are issued by z/VM components and facilities. For more information about message files, see [“Getting HELP on Messages” on page 188](#)

Command files

Command files contain information about any function that is invoked by specifying a name, such as a command, a subcommand, a macro, a statement, or a routine. For more information about command files, see [“Getting HELP on Commands” on page 188](#).

The HELP facility is used on 3270-type terminals in display mode. It can also be used on line-oriented terminals. In display mode, the HELP facility uses XEDIT to display the HELP files. In line-mode, the HELP facility types the contents of the HELP file to your screen one line at a time.

When you access the HELP facility, HELP uses EXECLOAD to load the HELPXED XEDIT macro, if the macro is not loaded. If you only occasionally use HELP, you might want to EXECDROP the HELPXED XEDIT macro to release the storage. For more information, see [z/VM: CMS Commands and Utilities Reference](#).

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.

If the PF key listing at the bottom of your screen is covered by a message, press Enter to refresh the screen.

You can quickly move between screens of HELP information. When you view the HELP information for one command, you can get help for another command by entering the HELP command in the current HELP screen. For example, if you view the HELP file for the CMS PRINT command, you can display the HELP file for the CMS COPYFILE command by entering the following HELP command on the command line of your current screen:

```
help cms copyfile
```

The HELP facility displays the HELP information for the COPYFILE command. If you press PF3 to quit the information about the COPYFILE command, you are returned to the information about the PRINT command.

Getting HELP on Messages

HELP information is available for messages that the system displays when you undertake a z/VM task. The HELP information explains why the message was issued and what action is required.

The HELP files for messages display the message text, an explanation of why the message was issued, the system action, and possible user responses.

You can display information about a message by entering one of these commands:

```
help msg msgid
help msgid
```

The module identifier (characters 4 - 6 of the message identifier) is ignored by HELP, so you do not need to enter it. For example, to display information about message DMSHLP002E, you can enter any of these commands:

```
help msg dmshlp002e
help msg dms002e
help dmshlp002e
help dms002e
```

If you receive a message without a message ID, it could be because you (or an application program that runs in your virtual machine) issued the CP SET EMSG TEXT command to display only message text. To get information about a message with no message ID:

- Search for a portion of the message text in the PDF version of the appropriate messages document.
- If you do not know which z/VM function issued the message (and therefore do not know in which document to search), you can search across the entire z/VM library in the z/VM information center.

Hint: In your search string, do not include any text from substitution variables. Message documentation uses variables to represent fields in which the system returns event-specific data. If you use actual values, your search string does not match.

Getting HELP on Commands

HELP command files contain information about any function that is invoked by specifying a name, such as a command, a subcommand, a macro, a statement, or a routine. Menus help you access the right HELP files, and information layers provide different kinds of information from within a file.

For more information about the information layers and sublayers that are in HELP command files, see [“Information Layers” on page 188](#). For more information about menu files, see [“HELP Menus” on page 192](#).

Information Layers

HELP command files have layers of information so that you can get an appropriate level of information for your task. By using HELP command options, you can specify three layers of information: BRIEF, DETAIL, or RELATED

The HELP facility displays only one layering option at a time. If you specify more than one layering option on the HELP command, the HELP facility uses only the last layering option on the command line. You can toggle (switch) between the other layers available for that command. For more information on how to toggle between layers of HELP, see [“Toggling” on page 194](#).

BRIEF is the default option. If you do not specify a layering option and if the brief information layer exists, the brief information layer of HELP is displayed. If the brief information layer is not available for a command, the detail layer is displayed. The following sections provide more information on the three layers of command HELP.

Note: For information about the syntax, operands, and options of the HELP command, see [HELP in z/VM: CMS Commands and Utilities Reference](#).

For information about how to create or modify the brief, detail, or related information sections of HELP files, see Chapter 9, “Tailoring the HELP Facility,” on page 199.

Brief HELP

The brief information layer might display 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.

Note: The HELP facility displays the brief information layer for HELP files that contain a section of brief information. However, z/VM HELP files do not contain a section of brief information. If you customize the z/VM HELP files, you can add a brief information section.

If you are in full-screen CMS and request HELP with the BRIEF option, your screen shows the HELP command that you entered and just below it, displays the brief information layer in a window that is displayed on your screen. If you are not in full-screen CMS, your entire screen displays the brief information layer. For more information about full-screen CMS, see Chapter 10, “Introducing Full-Screen CMS,” on page 237.

Detail HELP

The detail layer of z/VM HELP files contains a description of the command, the command format, an explanation of its parameters and options, usage notes, and error information. The detail layer provides information that is similar to the commands reference information that is in [z/VM: CMS Commands and Utilities Reference](#).

Subsetting Options

The detail layer of information can contain several sublayers. The *subsetting* options of the HELP command specify which sublayers are displayed. The HELP facility can display several sublayers at a time. The following subsetting options are valid: DESCRIPT, FORMAT, PARMS, OPTIONS, NOTES, ERRORS, and ALL. ALL is the default option, and displays all sublayers that are available for the specified command. z/VM HELP files typically contain all sublayers.

Note: It is possible to change the default options. For more information, see [DEFAULTS command](#) in [z/VM: CMS Commands and Utilities Reference](#).

For example, to display all sublayers of the DETAIL layer of the SENDFILE command in the CMS environment, enter the following HELP command:

```
help cms sendfile (detail
```

To display only the NOTES sublayer of the SENDFILE command, enter the following HELP command:

```
help cms sendfile (notes
```

z/VM HELP files often contain usage notes in the sublayer that is specified by the NOTES option.

The HELP facility displays usage notes that are similar to the following screen example.

```

CMS SENDFILE          Detail Help Information          line 1 of 281
Usage Notes

1. Tailoring the SENDFILE Command Options

You can use the DEFAULTS command to set up options and 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
customize the defaults of the SENDFILE command, yet override them
when you desire. For more information, see the DEFAULTS command.

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=             11= Related    12= Cursor

====> _
Macro-read 1 File

```

Figure 60. Sample of DETAIL HELP for the SENDFILE Command

Related HELP

The related layer of HELP provides links to similar HELP command files.

For example, when you request the related layer for the SET or QUERY commands, the screen lists and briefly describes all the SET or QUERY operands that are available for the system component. You can directly access HELP information for any of the displayed operands from the related section.

As another example, imagine that you want to remove a file from your z/VM reader. You try the ERASE command, but get an error message. You display the related information layer for the CMS ERASE command:

```
help cms erase (related
```

The related layer provides links to related commands that you can investigate. You discover that the appropriate command is DISCARD.

The following screen is an example of information that is similar to what you see when you display related information for the ERASE command.

```

CMS ERASE          Related Help Information          line 1 of 20
Related Information

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 "list-type"
            CMS command environments, such
            as FILELIST.

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=          12= Cursor

====> _
Macro-read 1 File

```

Figure 61. Sample of RELATED HELP for the ERASE Command

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 display the HELP information.

Reserved file types in the HELP Facility let you access specific HELP files. To access HELP information for any CMS SET option, enter enter HELP, followed by the component name, CMSSET, and the option name. For example, to get HELP on the CMS command SET RDYMSG, enter the following command:

```
help cmsset rdymsg
```

This command displays the HELP file for the CMS SET RDYMSG command.

To request HELP for a CMS QUERY option, enter HELP, followed by the component name, CMSQUERY, and the option name.

The same is true for CP SET and QUERY options. To get HELP on a CP SET option, enter HELP, followed by CPSET, and the option name. For HELP on a CP QUERY option, enter HELP, CPQUERY, and the option name.

To request HELP for XEDIT SET and QUERY options, you can enter HELP, followed by SET or QUERY, and the name of the option. For example, to get HELP on the XEDIT command SET TRANSLATE, enter the following command:

```
help set translate
```

Even though there is also a CMS command SET TRANSLATE, the HELP facility displays HELP information on the XEDIT SET TRANSLATE command. Enter the following command:

```
help cmsset translate
```

to request information on the CMS SET TRANSLATE command.

To display a HELP panel that directs you to the menus for the QUERY and SET options in CP, CMS, and XEDIT, enter the following command:

```
help queryset tasks
```

For more information on HELP for SET and QUERY options, see [z/VM: CMS Commands and Utilities Reference](#).

HELP Menus

The HELP facility provides menus for navigation among HELP files. To display information about a component, command, subcommand, macro, statement, or routine you can select choices from component menus. If you want information to accomplish a task, you can select choices from task menus.

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:

- Use the key marked --->|, which functions as a tab key, causing the cursor to move to the first character of the next entry.
- Use another cursor-movement key.
- 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 HELP menus:

1. HELP task menus.
2. HELP component menus.

HELP Task Menus

Task menus are organized around user tasks. The task menus are organized in a branching structure of general to specific tasks.

TASK menus might be helpful for new users because task menus guide you to the appropriate information to complete a task.

A task menu file name is often the name of a HELP component. A task menu file type is always HELPTASK.

To display a menu of high-level tasks, enter the following command:

```
help task
```

Figure 62 on page 192 displays an example menu of high-level tasks.

```

HELP TASK                      Task Help Information                line 1 of 40
(c) Copyright IBM Corporation 1990, 2020

z/VM Help, main panel

This panel lists other Help panels that provide information about
various z/VM functions, topics, and tasks.
To view a Help panel, move the cursor to any character of the name
and press the ENTER key or the PF1 key.

HELP      - z/VM HELP Facility topics
MENUS     - z/VM Help menus
TASKS     - Basic z/VM tasks
AVS       - AVS commands
CMS       - CMS commands
CP        - CP commands
DIRECTORY - Directory statements
DIRMAINT  - DirMaint commands
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 62. Sample HELP TASK Menu

HELP Component Menus

Component menus are organized around HELP components, which are organized around product components, commands, subcommands, macros, statements, and routines. The component menus are organized in a branching structure of general to specific menus.

A component menu file name is very often the name of a HELP component.

Note: If a HELP component name is more than eight characters, then the file name of the menu file for the component is the first eight characters of the HELP component name.

A component menu file type is always HELPMENU.

For a list of z/VM HELP components, see [“z/VM HELP Components” on page 202](#).

A component menu indicates the file type of the menu choices:

- An asterisk (*) indicates that the menu choice is a HELPMENU file.
- A colon (:) indicates that the menu choice is a HELPTASK file.
- If a menu choice doesn't have an asterisk or a colon, then the menu choice is an information HELP file (a command file or message file).

All command files in a given component menu are in the same HELP component.

Figure 63 on page 193 displays the CMS component MENU.

```

CMS MENU                      Menu Help Information                      line 1 of 43
(c) Copyright IBM Corporation 1990, 2003

Help for CMS commands

To view a Help panel, move the cursor to any character of the name
and press the ENTER key or the PF1 key.
An asterisk (*) preceding the name indicates a MENU panel.
A colon (:) preceding the name indicates a TASK panel.

*BUFFER  CMSBATCH  DROPBUF  Global  NAMES  RO  SVCTrace
*CMSQUERY CMSSERV  DSERV    GLOBALV  NETDATA  RSERV  SYNMSGs
*CMSSET   COMPare  Edit     GRANT   NETLCNVT RT     SYNonym
*CMSUTIL  CONV2WD  ERASE    HB      NOTE    RTNDrop SYSWATCH
*EDIT     CONWAIT  ESERV    Help   NUCXDROP RTNLoad TAPE
*FILESERV COPYfile  ESTATE   HELPCONV NUCXLOAD RTNMap  TAPEMAC
*OPENVM   CP       ESTATEW  HI      NUCXMAP  RTNState TAPPDS
*OSHELL   CREate   ETRACE   HO      OPNMSGs  RUN     TE
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 63. Sample Component MENU for CMS

To display the component menu of CMS HELP information on your VM system, enter the following command:

```
help cms menu
```

To display a high-level component menu, enter the following command:

```
help menu
```

Display and Search Options for HELP files

The format of the information that is displayed by the HELP facility can be controlled by using the SCREEN/NOSCREEN and TYPE/NOTYPE options of the HELP command.

The EXTEND option extends the search scope for a HELP file beyond the HELP component that you specify on the HELP command. If you are not sure of the appropriate HELP component, the EXTEND option might help you find the information that you are looking for.

For information about the syntax, operands, and all options of the HELP command, see [HELP](#) in *z/VM: CMS Commands and Utilities Reference*.

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 pressed is highlighted in the settings at the bottom of your HELP screen.

Toggling

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 change to reflect the kinds of HELP available to you and the options entered on the HELP command or by the DEFAULTS command. 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.

Note: If no subsetting options were specified, PF10 is not set to MOREHELP.

Table 23 on page 194 lists the settings for PF1, PF10, and PF11 when all HELP layers are available for the displayed file.

Table 23. Toggling between layers of HELP

Screen display	PF key settings
BRIEF	PF1 = All PF10 = Morehelp PF11 = Related

Table 23. Toggling between layers of HELP (continued)

Screen display	PF key settings
DETAIL	PF1 = All PF10 = Brief PF11 = Related
ALL	PF1 = Brief PF10 = Morehelp PF11 = Related
RELATED	PF1 = Help PF10 = Morehelp PF11 = Brief

Table 24 on page 195 lists 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 24. 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.

Table 24. PA and PF keys in the HELP facility (continued)

Key	Meaning	Usage
PF10	Morehelp	Displays the HELP information from a command file as if the DETAIL option was specified. Note: If no subsetting options were specified, PF10 is not set to MOREHELP.
	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.
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.
PF12	Cursor 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 [z/VM: CMS Commands and Utilities Reference](#).

Displaying HELP Files Using XEDIT

If you are using HELP in display mode on a 3270-type terminal, the HELP Facility uses XEDIT 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.

For more information on these features, see [z/VM: XEDIT Commands and Macros Reference](#).

Not all XEDIT subcommands are used on the displayed HELP files. The excluded subcommands are listed below:

ALL	READ
FILE	REPLACE
INPUT	SET
MACRO	POWERINP

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 displayed.

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 z/VM HELP files. For more information, see [Chapter 9, “Tailoring the HELP Facility,” on page 199](#).

Chapter 9. Tailoring the HELP Facility

One of the most useful features of the z/VM HELP Facility is its versatility. You can tailor the displayed HELP to suit your needs by creating or changing CMS files.

You might want to generate new HELP files for any new commands, execs, or error messages you create. You might also want to make updates to the z/VM HELP files to fulfill your individual needs.

The key role of HELP components is explained. The z/VM HELP components, file types, and high-level menus are listed. Details are provided for creating and changing files for the HELP facility.

HELP Components Definition and Purpose

HELP components organize HELP files into groups that contain information about a z/VM component, command, subcommand, macro, statement, or routine. The HELP facility uses HELP components to retrieve appropriate information, including when you specify a HELP component as an operand on the HELP command.

HELP component definition

A HELP component is a group of HELP files that share the same file type. Three special HELP file types are excluded from the definition:

- Files with type HELPMENU are HELP menu selection files.
- Files with type HELPTASK are HELP task selection files.
- Files with type HELPABBR are HELP abbreviations files.

HELP components coordinate other HELP elements

HELP file types, HELP component names, abbreviations file names, and HELPMENU file names have dependencies on each other. HELP components are the foundation.

- A HELP component name must "match" a file type. For details of the matching criteria, see [“HELP component name conventions” on page 199](#).
- An abbreviations file name must "match" a file type. For details of the matching criteria, see [“Abbreviations file name convention” on page 221](#).
- By default, a HELPMENU file name must "match" a file type. (In the non-default case, a HELPMENU file can specify a HELP component by using the .MT control word.) For details of the matching criteria, see item [“1” on page 208](#) in [“Menu choice specifications: File type” on page 208](#).
- For optimal HELP file retrieval, an abbreviations file name must "match" a HELP component name. For details of the matching criteria, see item [“2” on page 222](#) in [“Requirements for a search that uses an abbreviations file” on page 222](#).

HELP component name conventions

The following conventions guarantee that a HELP component name accurately identifies a HELP component. The HELP component name must accurately identify a HELP component in the following cases:

- The HELP component name is used as the *component_name* operand on the HELP command.
- The HELP component name is used in a HELPTASK file to specify a menu choice.
- The HELP component name is used in a RELATED HELP section of a command file to specify a menu choice.
- The HELP component name is used as the file name of a HELPABBR file.

- The HELP component name is used as the file name of a HELPMENU file that does not specify a HELP component by using an .MT control word.

All z/VM HELP file types begin with "HELP" and contain a suffix of 2-4 additional characters. Custom HELP component must begin with "HELP" and contain a suffix of 1-4 additional characters. The additional characters indicate the HELP component name.

- Short z/VM file type suffixes (2-3 characters) define their component names exactly. For example:

Table 25. Short suffix HELP components

File type	HELP component name	Abbreviations file
HELPCP	CP	CP HELPABBR
HELPCFX	FCX	FCX HELPABBR

Note that files with short suffixes can be associated with only one HELP component and only one abbreviations file. The file type suffix exactly matches the HELP component name and the abbreviations file name.

- Long file type suffixes (4 characters) define the first four characters of a component name, but do not determine additional characters of the component name. The HELP component name is typically the name of a z/VM component (REXX, XEDIT) or a command that has subcommands (CP FLASHCOPY command, CP QUERY command, CMS QUERY command). The following table shows examples of z/VM HELP file types that have a four-character file type suffix:

Table 26. Four-character suffix HELP components

File type	HELP component name	Abbreviations file
HELPREXX	REXX	REXX HELPABBR
HELXPEDI	XEDIT	XEDIT HELPABBR
HELPFLAS	FLASHCOPY	FLASHCOP HELPABBR
HELPCPQU	CPQUERY	CPQUERYHELPABBR
HELPRSCS	RSCS	RSCS HELPABBR
HELPRSCS	RSCSAUTH	RSCSAUTH HELPABBR

Notes:

- File types with long suffixes can be associated with more than one HELP component, as is the case with the HELPRSCS file type. You can create an abbreviations file (HELPABBR file type) for each HELP component name.
- File types with long suffixes can yield a HELP component name that contains more than eight characters, as is the case for the HELPFLAS file type.

A file name is limited to eight characters. The limit restricts the HELP file name to the first eight characters of the HELP component name.

The HELP command processes only the first eight characters of a HELP component name that is used as a *component_name* operand.

For a complete list of z/VM HELP components, see [Table 27 on page 202](#).

Purpose of HELP components

The HELP facility uses HELP components to retrieve appropriate HELP files. The HELP facility uses the HELP component that you specify by using the *component_name* operand or uses a list of default components. The HELP facility uses HELP components to retrieve and display appropriate HELP files in the following cases:

Synonyms and abbreviations

An abbreviations file contains synonyms and abbreviations, which the HELP facility can use to retrieve and display HELP files. If the *cmd_name* operand yields no matches, the HELP facility searches for an abbreviations file whose file name matches the *component_name* operand or the default components.

For more information, see [“Using Command Abbreviations”](#) on page 221.

Tip: In many cases the HELP facility can use the minimum component-name abbreviation (the suffix of the HELP file type) to find and display a HELP file. However, you leverage the synonyms and abbreviations in abbreviations files only when the *component_name* operand matches the file name of an abbreviations file. Abbreviations file names are typically longer than the minimum component-name abbreviation (the suffix of the HELP file type). Hence, your chances of retrieving a HELP topic increase if you use a complete HELP component name as the *component_name* operand.

Default file types when a *component_name* operand is not specified on the HELP command

If you do not specify a *component_name* operand when you specify a *cmd_name* operand, the HELP facility searches several default components. The HELP facility searches among files of the following types, in the following order:

1. HELPCMS
2. HELPCP
3. HELPMENU
4. HELPTASK
5. HELPMMSG

Direct retrieval of HELP files that are not in the default components list

If you do not specify a *component_name* operand and you specify a *cmd_name* operand that is not in the default components list, the HELP facility cannot find the requested information. To display the requested information you must specify the HELP component name.

For example, to directly retrieve the HELP topic for the WITHDRAW subcommand of the CP FLASHCOPY command, specify the HELP component name (FLASHCOPY) on the HELP command:

```
help flashcopy withdraw
```

Tip: In the case of direct retrieval of HELP for files that are not in the default components list, you don't have to use the complete HELP component name. You can use only the last, component-defining characters of the file type. Hence, the following example is also valid:

```
help flas withdraw
```

Disambiguation of HELP for same-name commands

The HELP component name can be used as an operand on the HELP command to specify an otherwise ambiguous command name. For example, CPXLOAD is a CP command, a system configuration statement, and a subcommand of the CP QUERY command. You can use the HELP component name to specify which HELP topic to retrieve:

Specify HELP for the CPXLOAD CP command:

```
help cp cpxload
```

Note: Because the CPXLOAD CP command topic is in the list of default components, you don't have to use the *component_name* operand. The following command yields the same result as the previous command:

```
help cpxload
```

Specify HELP for the CPXLOAD system configuration statement:

```
help sysconfig cpxload
```

Specify HELP for the CPXLOAD subcommand of the CP QUERY command:

```
help cpquery cpxload
```

Tip: In the case of disambiguation, you don't have to use the complete HELP component name. You can use only the last, component-defining characters of the file type. Hence, the following examples are also valid:

Specify HELP for the CPXLOAD system configuration statement:

```
help sysc cpxload
```

Specify HELP for the CPXLOAD subcommand of the CP QUERY command:

```
help cpqu cpxload
```

The HELP facility also uses HELP components when linking from component menu files. In the default case, the file name of a HELPMENU file indicates the HELP component of the menu choices. When you select a menu choice, the HELP facility searches for a file with the name of the menu choice. The HELP facility searches in the HELP component that is indicated by the name of the HELPMENU file

The non-default case is when the HELPMENU file contains an .MT control word, which specifies a HELP component. In the non-default case, the HELP facility searches in the HELP component that is indicated by the .MT control word.

Note: Contrast the difference between a task menu file (file type TASKMENU) and a component menu file (file type HELPMENU). A task menu contains menu choices whose HELP components are specified in the file and for each menu choice. Files in different HELP components can be menu choices in one task menu file. The file name of a task menu file does not determine the HELP component of the menu choices.

z/VM HELP Components

Table 27 on page 202 lists the z/VM HELP components. The associated file type and some associated high-level menus are listed.

File type	HELP component name	High-level menus	Description
HELPABBR	n/a	n/a	Although HELPABBR files have a unique file type, HELPABBR files are not a HELP component. HELPABBR files are reserved to define abbreviations and synonyms for file names in a HELP component.
HELPADVH	ADVH	ADVH	Directory Maintenance Facility commands
HELPA PPC	APPCVM	APPCVM	APPC/VM macro functions
HELPPASSO	ASSOCIATE	ASSOCIAT	CP ASSOCIATE command operands
HELPAVS	AVS	AVS	APPC/VM VTAM Support commands
HELPBORD	BORDER	BORDER	CMS window border commands
HELPCERT	CERT	CERTMGR	CERTMGR command operands
HELPCMS	CMS	CMS, CMSUTIL, FUNCTION	CMS commands, utilities, and functions
HELPCMSQ	CMSQUERY	CMSQUERY	CMS QUERY command operands
HELPCMSS	CMSSET	CMSSET	CMS SET command operands

Table 27. File types, HELP components, and high-level HELP menus (continued)

File type	HELP component name	High-level menus	Description
HELPCP	CP	CP, CPUTIL, MESSAGE	CP commands and utilities
HELPCPQU	CPQUERY	CPQUERY	CP QUERY command operands
HELPCPSE	CPSET	CPSET	CP SET command operands
HELPCREA	CREATE	CREATE	CMS CREATE command operands
HELPCRR	CRR	CRR	CRR command operands
HELPDEAC	DEACTIVE	DEACTIVE	CP DEACTIVE command operands
HELPDEFI	DEFINE	DEFINE	CP DEFINE command operands
HELPDELE	DELETE	DELETE	Operands of the CMS DELETE and CP DELETE commands
HELPDETA	DETACH	DETACH	CP DETACH command operands
HELPDFSM	DFSMS	DFSMS	DFSMS commands
HELPDIRE	DIRECTORY	DIRECTOR	Directory statements
HELPDISA	DISABLE	DISABLE	CP DISABLE command operands
HELDPDISP	DISPLAY	DISPLAY	CP DISPLAY command operands
HELPDRAI	DRAIN	DRAIN	CP DRAIN command operands
HELPDUMP	Several (See note 1): DUMP DUMPSCAN DUMPCVIEW	Several: DUMP DUMPSCAN DUMPCVIEW	Several: DUMP command operands DUMPSCAN subcommands Dump Viewing Facility commands
HELPEEDIT	EDIT	EDIT	EDIT subcommands
HELPEENAB	ENABLE	ENABLE	CP ENABLE command operands
HELPEXC2	EXC2	EXEC2	EXEC 2 statements
HELPEEXEC	EXEC	EXEC	EXEC statements
HELPEFCX	FCX	Several: BASIC FCONTROL FCX REDISPLAY PERFORM	PerfKit for VM subcommands: BASIC mode FCONTROL General panels REDISPLAY mode Performance monitor mode
HELPEFILE	FILESERV	FILESERV	File pool server commands (FILESERV command operands)
HELPEFLAS	FLASHCOPY	FLASHCOP	CP FLASHCOPY command operands
HELPEFREE	FREE	FREE	CP FREE command operands
HELPEFTP	FTP	FTP	TCP/IP FTP client (FTP command) subcommands

Table 27. File types, HELP components, and high-level HELP menus (continued)

File type	HELP component name	High-level menus	Description
HELPFTPS	FTPMSG	FTPMSG	TCP/IP FTP server administrative subcommands
HELPGIVE	GIVE	GIVE	CP GIVE command operands
HELPGROU	GROUP	GROUP	Group Control System GROUP command and panels
HELPHelp	HELP	HELP	HELP Facility topics
HELPHOLD	HOLD	HOLD	CP HOLD command operands
HELPINCH	INCH	INCH	RSCS Data Interchange Manager commands
HELPINDI	INDICATE	INDICATE	CP INDICATE command operands
HELPIPFO	IPFORMAT	IPFORMAT	TCP/IP IPFORMAT subcommands
HELPIUCV	IUCV	IUCV	IUCV macro functions used with APPC/VM
HELPLDAP	LDAP	LDAP	TCP/IP Lightweight Directory Access Protocol server administrative subcommands
HELPLE	LE	LE	Language Environment commands
HELPLOCA	LOCATE	LOCATE	CP LOCATE command operands
HELPLOGO	LOGO	LOGONID	VM/Pass-Through Facility LOGONID function
HELPMACR	MACROS	MACROS	Preferred CMS macros
HELPMENU	n/a	n/a	Although HELPMENU files have a unique file type, HELPMENU files are not a HELP component. HELPMENU files are reserved as selection files.
HELPMODI	MODIFY	MODIFY	CP MODIFY command operands
HELPMONI	MONITOR	MONITOR	CP MONITOR command operands
HELMPRO	MPROUTE	MPROUTE	TCP/IP MPROUTE server administrative commands
HELPMMSG	MSG	n/a	z/VM messages
HELPMSOC	MSOCKETS	SMAPI (Task menu)	Systems management sockets
HELPNFS	NFS	NFS, REFRESH	TCP/IP NFS server administrative commands, TCP/IP Network File System Refresh commands
HELPN SIN	NSINTER	NSINTER	TCP/IP NSLOOKUP interactive subcommands
HELPN SLO	NSLOOKUP	NSLOOKUP	TCP/IP NSLOOKUP set subcommands
HELPOMAC	OMACRO	OMACRO	OpenExtensions assembler language macros
HELPOPEN	OPENVM	OPENVM	CMS OPENVM commands
HELPOROU	OROUTINE	OROUTINE	OpenExtensions callable services
HELPOSHE	OSHELL	OSHELL	OpenExtensions shell commands
HELPPPIPE	PIPE	PIPE	CMS Pipelines stages and pipeline subcommands
HELPPREF	PREFIX	PREFIX	XEDIT prefix subcommands and macros

Table 27. File types, HELP components, and high-level HELP menus (continued)

File type	HELP component name	High-level menus	Description
HELPPSCR	PSCREEN	PSCREEN	CMS physical screen commands
HELPPURG	PURGE	PURGE	CP PURGE command operands
HELPPVM	PVM	PVM	VM/Pass-Through Facility
HELPQUER	QUERY	QUERY	XEDIT QUERY subcommand options
HELPRDEV	RDEVICE	RDEVICE	CP SET RDEVICE command operands
HELPRDVF	RDVF	RDVF	Dump Viewing Facility DUMPSCAN subcommands for RSCS
HELPREXX	REXX	REXX	REXX/VM statements
HELPROUT	ROUTINES	Several: CPIC CPIRR CRRPART CRRR ERRORCHK FILESYSR FPADMINR MULTITSK OTHERR REXXR ROUTINES RSKROUT VMDSR WORKUNIT	Routines: CPI communications CPI resource recovery CRR participation Coord. Resource Recovery CMS error checking and debug CMS file system mgmt (I/O) CMS file pool administration CMS application multitasking Other CMS routines REXX-related CSL routines CMS routines and services CMS Reusable Server Kernel VM data space CMS work unit management
HELPRQUE	RQUERY	RQUERY	RSCS QUERY command operands
HELPRSCS	Several (See note 1): RSCS RSCSAUTH	Several: RSCS RSCSAUTH	Several: RSCS commands RSCSAUTH commands
HELPRSKC	RSKCMDS	RSKCMDS	Reusable Server Kernel commands
HELPRXSO	RXSOCKET	RXSOCKET	REXX Sockets functions
HELPSEGM	SEGMENT	SEGMENT	CMS SEGMENT command operands
HELPSET	SET	SET	XEDIT SET subcommand options
HELPSFSA	SFSADMIN	SFSADMIN	SFS/CRR administrator and operator commands
HELPSFSQ	SFSQUERY	SFSQUERY	SFS/CRR administrator and operator QUERY commands
HELPSMTP	SMTP	SMTP	TCP/IP SMTP server administrative commands
HELPSNMP	SNMP	SNMP	TCP/IP SNMP administrative commands
HELPSPXT	SPXTAPE	SPXTAPE	CP SPXTAPE command operands

Table 27. File types, HELP components, and high-level HELP menus (continued)

File type	HELP component name	High-level menus	Description
HELPSQLD	n/a	n/a	The file type is reserved for DB2® Server for VM
HELPSSLA	SSLADMIN	SSLADMIN	TCP/IP SSL server administrative commands
HELPSTAR	START	START	CP START command operands
HELPSTOR	STORE	STORE	CP STORE command operands
HELPSYSC	SYSCONFIG	SYSCONFI	System configuration statements
HELPTASK	n/a	n/a	Although HELPTASK files have a unique file type, HELPTASK files are not a HELP component. HELPTASK files are reserved as selection files.
HELPTCPI	TCPIP	TCPIP	TCP/IP commands and related functions
HELPTELN	TELNET	TELNET	TCP/IP Telnet protocol client subcommands
HELPTERM	TERMINAL	TERMINAL	CP TERMINAL command operands
HELPTFTP	TFTP	TFTP	TFTP command operands
HELPTRAC	TRACE	TRACE	CP TRACE command operands
HELPtrSO	TRSOURCE	TRSOURCE	CP TRSOURCE command operands
HELPTSAF	TSAF	TSAF	Transparent Services Access Facility commands
HELPUDVH	UDVH	UDVH	Directory Maintenance Facility commands in upper case English
HELPUFTD	UFTD	UFTD	TCP/IP UFT server administrative (UFTD command) subcommands
HELPVARY	VARY	VARY	CP VARY command operands
HELPVMDS	VMDS	VMDS	VM data spaces CP macros
HELPVMDT	VMDT	VMDT	VM Dump Tool toolkit macros and stages (unsupported samples and examples)
HELPVMDU	VMDUMPTL	VMDUMPTL	VM Dump Tool subcommands and macros
HELPVMFI	VMFINS	VMFINS	VMFINS command operands
HELPVMFS	VMFSIM	VMFSIM	VMFSIM command operands
HELPVMSE	VMSESE	VMSES, SERV MGR	VMSES/E commands, SERV MGR command operands
HELPVSCR	VSCREEN	VSCREEN	CMS virtual screen commands
HELPWIND	WINDOW	WINDOW	CMS window commands
HELXPEDI	XEDIT	XEDIT	XEDIT subcommands
HELPXLIN	XLINK	XLINK	CP XLINK command operands

1. Multiple HELP components are associated with this file type. Each component is associated with a unique abbreviations file.

Restrictions for all HELP files

The following restrictions apply when you create new files or modify existing files for the HELP facility.

- Name conventions:
 - The combination of file name and file type must be unique for each HELP file.
 - You must respect the role of file type when creating a new file type or changing an existing file's type. For more information, see [“HELP Components Definition and Purpose” on page 199](#).
 - Command and message HELP files have specific name conventions. See [“Command HELP name conventions” on page 215](#) and [“Message HELP name conventions” on page 220](#).

- If your installation uses 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.

You can create your own HELP files, store them in an SFS directory, and HELP displays the files if the directory is accessed. For HELP files that are sent with z/VM, installations continue to use a minidisk.

- HELPPABBR, HELPMENU, and HELPTASK files must have a file mode number of 2.
- 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 users, you will have to get authority from the system administrator to alter the HELP facility.
- The length of a record in a HELP file must not exceed 133 characters. Long lines might require scrolling right and left to see the entire line.

Tip: z/VM HELP files contain lines of 80 characters or less.

Creating HELP Files

Instructions are provided for creating new files for the z/VM HELP facility.

Creating Menus for HELP Files

The HELP facility has two types of menus:

- Component menus have a file type of HELPMENU.
- Task menus have a file type of HELPTASK.

Creating HELPMENU Files

Component menu files have file type HELPMENU. A HELPMENU file can provide menu choices that link to HELP information files, HELPTASK files, and other HELPMENU files. The menu choices in a given file are typically for a single z/VM component or subcomponent.

A HELPMENU file is typically created for a HELP component or a group of related files in a HELP component. Except for menu choices that specify HELPMENU or HELPTASK files, all menu choices in a given HELPMENU file must belong to the same HELP component.

HELMENU name conventions

- If a HELPMENU file contains other than menu files (other than type HELPTASK and HELPMENU) and does not contain a *menu type* (.MT) control statement, then the HELPMENU file name must identify a HELP component. If a HELPMENU file contains only menu files (type HELPTASK and HELPMENU) or contains a menu type control statement, then the file name does not have to identify a HELP component. See [“Menu choice specifications: File type” on page 208](#).

Note: A *menu type* control statement is composed of an .MT control word and a HELP component name. For more information, see [Table 38 on page 226](#).

A HELP component with a 4-character file type suffix can be specified by using only the last four characters of the component's file type suffix. However, if the HELP component has a longer component

name, the longer component name is typically used as the file name of the HELPMENU file. For a complete list of z/VM HELP components and many associated, high-level HELPMENU file names, see [Table 27 on page 202](#).

Note: z/VM HELP file names are limited to eight characters. Hence, the last character of a 9-character HELP component name must be dropped to yield an 8-character file name for the HELPMENU file. For example, the FLASHCOPY component's HELPMENU is named FLASHCOP HELPMENU.

- The file type of a HELPMENU file is HELPMENU.

Menu choice specifications: File type

The file type of a menu choice can be identified in several ways:

- A menu choice that begins with an asterisk (*) specifies a HELPMENU file.
- A menu choice that begins with a colon (:) specifies a HELPTASK file.
- The HELP component of a menu choice that does not begin with an asterisk or a colon is identified by the file name of the HELPMENU file or by a menu type control statement.

The HELP component of a menu choice that does not begin with an asterisk or a colon is identified in one of two ways:

1. The HELPMENU file name "matches" a HELP component name. The criterion for matching depends on the length of the HELP component file type suffix:
 - If the file type suffix that defines a component is fewer than four characters, the HELPMENU name must match the file type suffix exactly. For example, informational files that are named in the CP HELPMENU file must have a file type of HELPCP.
 - If the file type suffix that defines a component is four characters, the first four characters of the HELPMENU name must match the four characters of the file type suffix.

[Table 28 on page 208](#) contains examples of z/VM HELPMENU file names for components with 4-character file type suffixes.

Table 28. Examples of menu files of components with 4-character file type suffixes

File name	HELP component	Description
FLASHCOP HELPMENU	FLASHCOPY	CP FLASHCOPY command operands. The file contains no .MT control statement.
REXX HELPMENU	REXX	REXX statements. The file contains no .MT control statement.

For a list of more high-level z/VM menu files and the corresponding HELP components, see [“z/VM HELP Components” on page 202](#).

For more information about HELP component names and file type suffixes, see [“HELP Components Definition and Purpose” on page 199](#).

2. The HELPMENU file contains a menu type (.MT) control statement. The .MT control statement specifies a HELP component that overrides the HELP component that is specified by the HELPMENU file name.

One .MT control statement is permitted per HELPMENU file. The .MT control statement can be on any line but must be on its own line and must begin in column 1. An .MT control statement is permitted in a file that has a default or a preformatted pattern. For information about patterns, see [“Pattern for formatting” on page 209](#).

[Table 29 on page 209](#) contains examples of file names of z/VM HELPMENU files that contain an .MT control statement.

Table 29. Examples of menu files that contain an .MT control statement

File name	HELP component	Description
BASIC HELPMENU	FCX	Performance Toolkit for VM BASIC Mode subcommand. The file contains the following .MT control statement: <pre>.mt fcx</pre>
ERRORCHK HELPMENU	ROUTINES	CMS error checking and debugging routines. The file contains the following .MT control statement: <pre>.mt rout</pre>

Menu choice specifications: File name

A menu choice identifies the file name of the linked HELP topic. The menu choices in a HELPMENU file must conform to the following conventions:

- One menu choice per line.
- The menu choice begins in column 1.
- A menu choice is the file name of a HELP file. An asterisk (*) or colon (:) can be added at the beginning of the HELP file name.
- A menu choice that specifies a HELPMENU file includes an asterisk (*) that is added to the beginning of the file name. For example, *DEFINE.
- A menu choice that specifies a HELPTASK file includes a colon (:) that is added to the beginning of the file name. For example, :TAPELOAD.
- A menu choice that does not specify a HELPMENU or HELPTASK file must be in the HELP component that is identified by the HELPMENU file name or by a menu type (.MT) control statement. See [“Menu choice specifications: File type”](#) on page 208.

Pattern for formatting

The HELP facility recognizes two patterns among HELPMENU files. The pattern determines how the HELP facility displays the file contents.

Default pattern

If the HELPMENU file contains two consecutive blank lines, then the HELP facility recognizes the default pattern.

In the section before the two consecutive blank lines, the HELP facility applies no extra highlighting and recognizes no menu choice specifications.

In the section after the two consecutive blank lines, the HELP facility applies default highlighting, expects menu choice specifications, and orders and arranges the menu choices as follows:

- Menu choices are arranged in several columns across the screen. If there are more menu choices than columns, the menu choices are stacked in the columns.
- Menu choices are highlighted.
- Menu choices are ordered by the type of the linked file. Within each type, linked file names are ordered alphabetically. Hence, blank lines in the link definitions section yield blank lines at the top of a column of the formatted display.

The z/VM HELPMENU files are an example of the default pattern.

Preformatted pattern

If the HELPMENU file does not contain two consecutive blank lines, the HELP facility does not order or arrange the links or apply default highlighting when the file is displayed.

Note: The HELP facility highlights text that is marked with highlighting control characters in both the default and the preformatted patterns. For more information, see [“Highlighting Words within a File”](#) on page 220.

Example

The HELPMENU file in [Figure 64 on page 210](#) provides an example that conforms to the conventions.

- The two blank lines (lines 10,11) indicate the default organization for formatting.
- The top of the file (before line 10) contains no menu choice specifications.
- The bottom of the file (after line 11) contains menu choice specifications:
 - The menu type (.mt) control statement indicates the HELP component of the menu choices (FCX).
 - Each line contains one menu choice specification, which begins in column 1.
 - Four HELPMENU file names are listed. An asterisk (*) is added at the beginning of each name.
 - One HELPTASK file name is listed. A colon (:) is added at the beginning of the name.
 - 16 file names are listed without asterisk or colon. The files are in the FCX HELP component.

Figure 64. Example of a HELPMENU file

```

BASIC    HELPMENU H2  V 80  Trunc=80 Size=33 Line=6 Col=1 Alt=0
0 * * * Top of File * * *
1 (c) Copyright IBM Corporation 2003, 2020
2 .cm (adapted from IBM Form SC24-6303)
3
4 Help for Performance Toolkit for VM BASIC Mode sub-commands
5
6 To view a Help panel, move the cursor to any character of the name
7 and press the ENTER key or the PF1 key.
8 An asterisk (*) preceding the name indicates a MENU panel.
9 A colon (:) preceding the name indicates a TASK panel.
10
11
12 .mt FCX
13 :PERFKIT
14 *FCX
15 *PERFORM
16 *REDIS
17 MODLEVEL
18 #CP
19 CLear
20 CMS
21 CP
22 Delete
23 ENTER
24 FCONAppc
25 FCONRmt
26 *FCONTROL
27 MONitor
28 MONScan
29 QUIT
30 REDisp
31 Reply
32 RETurn
33 TRNDScan
34 * * * End of File * * *
```

The HELP facility displays the HELPMENU file as in [Figure 65 on page 211](#).

The top of the file is displayed with no highlighting.

In the bottom of the file, which is the menu choices section, the following default formatting is applied:

- Menu choices are arranged in columns across the screen. Menu choices are stacked in the columns.
- Menu choices are highlighted.
- Regardless of the order in the file, menu choices are displayed in alphabetic order.
- If you select any of the menu choices, the HELP facility displays the specified HELP file.

Figure 65. A HELPMENU file for PerfKit BASIC mode subcommands

```

BASIC MENU                      Menu Help Information
(c) Copyright IBM Corporation 2003, 2020

Help for Performance Toolkit for VM BASIC Mode sub-commands

To view a Help panel, move the cursor to any character of the name
and press the ENTER key or the PF1 key.
An asterisk (*) preceding the name indicates a MENU panel.
A colon (:) preceding the name indicates a TASK panel.

*FCONTROL *REDISP  CLear    Delete    FCONRmt  MONScan  Reply
*FCX      :PERFKIT CMS      ENTER    MODLEVEL QUIT     RETurn
*PERFORM  #CP      CP       FCONAppc  MONitor  REDisp   TRNDScan

```

You can see the example if you enter the following command on a z/VM system:

```
help basic menu
```

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 [“Creating Menus for HELP Files”](#) on page 207, 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.

When viewed by using an editor like XEDIT, your HELPMENU file should look similar to the example in [Figure 66 on page 211](#).

To display a file, place the cursor under any character of the file wanted and pressing the Enter key or the PF1 key.

```

CLASS8
CLASS7
CLASS0
CLASSC
MOUNT
DEMOUNT

```

Figure 66. Sample of HELPMENU File Creation

When you specify HELP SYS5 MENU, the HELP facility orders and stacks in columns the file names. See [Figure 67 on page 212](#). You can then choose menu choices from this menu as you would with any other HELPMENU file.

```

SYS5 MENU                               Menu Help Information                line 1 of 5
To display a file, place the cursor under any character
of the file wanted and pressing the Enter key or the PF1 key.

CLASSC  CLASS0  CLASS7  CLASS8  DEMOUNT  MOUNT
* * * End of File * * *

```

Figure 67. Sample HELPMENU File Displayed by the HELP Facility

Creating HELPTASK Files

A HELPTASK file can provide menu choices that link to HELP information files, HELPMENU files, and other HELPTASK files. The menu choices are typically related to a user task.

HELPTASK name conventions

- A file with type HELPTASK can have any file name. The file name is not restricted by HELP components, file types, abbreviations files, or any other HELP element.

Tip: z/VM HELPTASK files typically have a file name that indicates the task to which the menu choices correspond.

- The file type of a HELPTASK file is HELPTASK.

Menu choice specifications

The file name and the HELP component of each menu choice is specified in the HELPTASK file. Menu choice specifications in HELPTASK files must conform to the following conventions:

- One menu choice can be specified per line.
- If a menu choice is specified, it must be specified in columns 1-24. No extraneous text is allowed in columns 1-24 on that line.
- Columns 25 and greater can contain extra text. The extra text does not affect the menu choice specification.
- The menu choice specification is composed of the operands and options of a HELP command. The menu choice specification must follow the syntax of the HELP command except that the HELP command name is omitted.

Pattern for formatting

The HELP facility recognizes two patterns among HELPTASK files. The pattern determines how the HELP facility displays the file contents.

Default pattern

If the HELPTASK file contains two consecutive blank lines, then the HELP facility recognizes the default pattern.

In the section before the two consecutive blank lines, the HELP facility applies no extra highlighting and recognizes no menu choice specifications.

In the section after the two consecutive blank lines, the HELP facility applies default highlighting, hides some text, and expects menu choice specifications:

- The HELP facility does not display columns 1-24. Text that is in columns 25 and greater is displayed beginning at column 1.
- The HELP facility applies highlighting to the displayed text.
- If columns 1-24 contain a valid menu choice specification, the HELP facility displays the specified HELP file when you select the line.
- If columns 1-24 do not contain a valid menu choice specification, the HELP facility displays an error message if you select the line.

The z/VM HELPTASK files are examples of the default pattern.

Preformatted pattern

If the HELPTASK file does not contain two consecutive blank lines, the HELP facility displays all contents of the file and does not apply default highlighting.

Note: The HELP facility highlights text that is marked with highlighting control characters in both the default and the preformatted patterns. For more information, see [“Highlighting Words within a File”](#) on page 220.

Example

The following HELPTASK file provides an example that conforms to the conventions.

- The two blank lines (lines 6,7) indicate the default organization for formatting.
- The top of the file (before line 6) contains no menu choice specifications.
- The bottom of the file (after line 7) contains menu choice specifications:
 - The menu choice specifications, in columns 1-24, contain only the file name and component of the file. No extra operands or options are included. There is no extraneous text in columns 1-24.
 - Because of default formatting, the HELP facility displays text that begins in line 25.
 - Because of default formatting, the HELP facility highlights the displayed text.
 - The `¢%` control character turns highlighting off for the explanatory text after the subcommand word.
 - The menu choice specifications are repeated on three consecutive lines so that the target file is clear to the editor. However, the file specifications in columns 1-24 after the initial file specification could be eliminated and the task menu would provide the same interaction. For example, in lines 9 and 10 PIPE DUPLICATE in columns 1-24 could be eliminated.

```

DUPLIC  HELPTASK H2  V 80  Trunc=80 Size=16 Line=5 Col=1 Alt=1

0 * * * Top of File * * *
1 (c) Copyright IBM Corporation 1992, 2020
2 .cm (adapted from IBM Form SC24-6252)
3
4 Move the cursor to the task that you want, then press the ENTER key
5 or the PF1 key.
6
7 |...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...>
8 PIPE DUPLICATE          DUPLICATE¢%- Copies each input record a specified
9 PIPE DUPLICATE          ¢% number of times
10 PIPE DUPLICATE
11 PIPE FANOUT             FANOUT ¢%- Copies primary input stream records to
12 PIPE FANOUT             ¢% multiple output streams
13 PIPE FANOUT
14 PIPE BUFFER             BUFFER ¢%- Accumulates all records in a single stage
15 PIPE BUFFER             ¢% not passing any on until all have been
16 PIPE BUFFER             ¢% received
17 * * * End of File * * *
```

The HELP facility displays the HELPTASK file as follows. The words in upper casing are highlighted.

```

DUPLIC TASK                      Task Help Information
(c) Copyright IBM Corporation 1992, 2020
  
```

Move the cursor to the task that you want, then press the ENTER key or the PF1 key.

```

DUPLICATE - Copies each input record a specified
            number of times
  
```

```

FANOUT    - Copies primary input stream records to
            multiple output streams
  
```

```

BUFFER    - Accumulates all records in a single stage
            not passing any on until all have been
            received
  
```

Example of HELPTASK File Creation

Assume you want to add HELP files for your internal system tasks to the HELP Facility. One HELP file documents information about using phones and one file documents information about using printers. You would follow the procedure outlined in the following steps.

1. Choose the task name of INTERNAL (Internal Procedures).
2. Create a HELP file for phones information and one file for printers information.
3. Following the rules given in “Creating Menus for HELP Files” on page 207, give each task file a file name and file type. The files might be named as follows:
 - For phones information: PHONE HELPINTE
 - For printers information: PRINT HELPINTE
4. Create a new task menu file called INTERNAL HELPTASK.
5. In the first few lines of the task menu, type information that displays above the menu items. Add two blank lines. After the two blank lines, 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 file should look similar to the example in [Figure 68 on page 214](#).

To select a task, move the cursor to the a task and press the Enter key or the PF1 key.

```

INTERNAL PRINT          Print a file on a local printer.
INTERNAL PHONE          Display a phone list for my department.
:
  
```

Figure 68. Sample of HELPTASK File Creation

When you specify HELP INTERNAL TASK, the HELP facility displays the user information and task descriptions sections of this file. See [Figure 69 on page 214](#). You can then work with this task menu as you would with any other HELP menu.

```

TASK INTERNAL          Task Help Information                      line 1 of 6
To select a task, move the cursor to the a task and press the Enter key
or the PF1 key.

      Print a file on a local printer.
      Display a phone list for my department.
* * * End of File * * *
  
```

Figure 69. Sample HELPTASK File Displayed by the HELP Facility

Creating Command HELP Files

The name conventions and structure of command HELP files is explained.

Command HELP name conventions

When you change the existing command HELP files or write your own command HELP files, you must adhere to the following name conventions:

- The combination of file name and file type must be unique.
- The file type of a command HELP file is HELPXXX, where XXXX is 1-4 characters that identify the HELP component. For more information, see [“HELP Components Definition and Purpose” on page 199](#).
- File names must represent command special characters by using reserved equivalents text.

z/VM restricts the characters that can be used in a file name. If you create a HELP file for a command that contains a special character, you must represent that special character with reserved equivalent text in the HELP file name. If the command name is a single special character, then the HELP file name must be the text equivalent word. If the command name contains more than one special character a special character and other, non-special characters, then in the file name the special character is replaced with a single character text equivalent. [Table 30 on page 215](#) lists the special characters and their reserved text equivalents when used in a file name.

Table 30. Text equivalents for special characters

Special character	Text equivalent word	Text equivalent single character
?	QUESMARK	Q
=	EQUAL	E
/	SLASH	S
"	DBLQUOTE	D
&	AMPRSAND	A
*	ASTERISK	R
.	PERIOD	P
<	LESSTHAN	L
>	MORETHAN	M

The HELP facility translates a special character in the HELP *cmd_name* (command name) operand to a text equivalent, and searches for a file with a name that matches the text equivalent. When you specify the HELP file name as an operand of the HELP command you can use the special character or the reserved text equivalent. [Table 31 on page 215](#) provides some examples of special characters in command names, HELP file names, and the corresponding HELP commands.

Table 31. Example file names and HELP commands

Command name	HELP file name	HELP command
?	QUESMARK HELPXEDI	HELP XEDIT ?
		or
		HELP XEDIT QUESMARK

Table 31. Example file names and HELP commands (continued)

Command name	HELP file name	HELP command
&STACK	ASTACK HELPEXC2	HELP EXC2 &STACK
		or
		HELP EXC2 ASTACK
**	RR HELPTTEST	HELP TEST RR
		or
		HELP TEST **

Note: Synonyms are another solution for HELP file names about commands that contain special characters. For more more information, see [“Using Command Abbreviations”](#) on page 221.

The following suggestions are guidelines:

- Use a HELP file name that matches the command name. A HELP file name that matches the command name might be easier to find than a HELP file name that does not match the command name.
 - Use a HELP file name that matches a popular abbreviation or synonym of the command file name.
- Tip:** The HELP facility can retrieve a file whose name matches a synonym or abbreviation that is defined in an appropriate HELPABBR file. For more information, see [“Using Command Abbreviations”](#) on page 221.
- Use the same file type for similar HELP topics. For more information, see [“HELP Components Definition and Purpose”](#) on page 199.

Examples of Name Conventions

Table 32 on page 216 provides examples of HELP files that conform to the conventions.

Table 32. Examples of command HELP file names

File Name	File Type	Description
ACCESS	HELPCMS	A CMS command description
BEGIN	HELPCP	A CP command description
ADD	HELPEXDI	An XEDIT subcommand description
DMS186W	HELPMMSG	A CMS message description
CMS	HELPMENU	A list of the CMS command and 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.

Structure of Command HELP files

Command HELP files are composed of eight types of sections, which correspond to the eight layering and subsetting options of the HELP command. The sections in the HELP file are delimited by a .CS control word that contains a keyword or numeric label. If the label is a keyword, the keyword must match a layering or subsetting option of the HELP command. If the label is a number, the number must be 0-7. [Table 33 on page 217](#) shows how numeric labels correspond to layering and subsetting option keywords.

The layering and subsetting options that are associated with an invocation of the HELP command determine which sections of a command HELP file are displayed.

For more information about the layering and subsetting options of the CMS HELP command, see [HELP in z/VM: CMS Commands and Utilities Reference](#).

The layering or subsetting option keyword that corresponds to a section type is not printed in the section. If a section contains a title, for example, highlighted text at the beginning of the section, it is only because the title is coded in the section. For example, the Options sections of z/VM HELP begin with highlighted text that matches the subsetting option keyword ("Options"), but other sections begin with highlighted text that does not match the corresponding layering or subsetting option keyword. See [Table 33 on page 217](#).

Table 33. Layering options, HELP file sections, and z/VM HELP section titles

Layering or subsetting option	HELP file section (.CS control word number)	z/VM Help section titles
BRIEF	0	Not applicable. z/VM HELP does not contain BRIEF information.
DESCRIPT	1	"Authorization", "Purpose"
FORMAT	2	The name of the command.
PARMS	3	"Operands"
OPTIONS	4	"Options"
NOTES	5	"Usage Notes", "Examples"
ERRORS	6	"Responses", "Messages"
RELATED	7	"Related Information"

The beginning of a section is delimited by the `.CS label ON` control statement. The end of a section is delimited by `.CS label OFF`. The sections can be in any order. A command HELP file can contain any number of sections of a given type.

Figure 70 on page 218 provides a sample of the structure of a HELP file. The organization of the sections and the section titles are typical of z/VM command HELP files, except that the sample adds a section for the brief layer of information. After a section's beginning delimiter (`.CS number ON`), a comment (`.cm`) indicates the layering or subsetting option that corresponds to the section. After the comment, a section title is coded by using highlighting control characters. The sample uses numeric labels in the `.CS` control statements, but could use the appropriate layering and subsetting-option keywords labels that correspond to those number labels.

```

.CS 0 on
.cm Displayed by the BRIEF option
¢|Command X¢%
Brief information ...
.CS 0 off
.CS 2 on
¢|Command X¢%
.cm Displayed by the FORMAT option
Syntax information ...
.CS 2 off
.CS 1 on
.cm Displayed by the DESCRIPT option
¢|Authorization¢%
Authorization information ...
.CS 1 off
.CS 1 on
.cm Displayed by the DESCRIPT option
¢|Purpose¢%
Purpose information ...
.CS 1 off
.CS 3 on
.cm Displayed by the PARMS option
¢|Operands¢%
Operands information ...
.CS 3 off
.CS 4 on
.cm Displayed by the OPTIONS option
¢|Options¢%
Options information ...
.CS 4 off
.CS 5 on
.cm Displayed by the NOTES option
¢|Usage Notes¢%
Usage Notes information ...
.CS 5 off
.CS 5 on
.cm Displayed by the NOTES option
¢|Examples¢%
Examples information ...
.CS 5 off
.CS 6 on
.cm Displayed by the ERRORS option
¢|Responses¢%
Responses information ...
.CS 6 off
.CS 6 on
.cm Displayed by the ERRORS option
¢|Messages¢%
Messages information ...
.CS 6 off
.CS 7 on
.cm Displayed by the RELATED option
¢|Related Information¢%
Related information ...
.CS 7 off

```

Figure 70. Sample HELP file structure

In command HELP files, two or more consecutive blank lines are displayed by the HELP facility as only one blank line. Vertical space is conserved.

For more information about the .CS control word, see [“Using HELPCONV to Create HELP Files”](#) on page 225.

BRIEF HELP

BRIEF HELP is intended to provide a short description of a command, its basic syntax (without options), and an example.

Note: The HELP facility supports the BRIEF option for HELP, but z/VM HELP files do not contain a BRIEF HELP section. If you customize the z/VM HELP files, you can add a BRIEF HELP section.

DETAIL HELP

DETAIL HELP is intended to present a complete description of a command, the command format, an explanation of its parameters and options, usage notes, and error information. The z/VM HELP files contain the following types of information in the sections.

DESCRIPT

Contains a description of the command.

FORMAT

Includes a syntax diagram. For a complete description of the notational conventions used in z/VM syntax diagrams, see [“Syntax, Message, and Response Conventions”](#) on page 20.

PARMS

Includes descriptions of the parameters of the command.

OPTIONS

Include descriptions of the options that are available for the command.

NOTES

Include usage notes and examples.

ERRORS

Includes responses that are related to the command. Can include a sentence on how to use HELP to get information on a specific message.

RELATED HELP

RELATED HELP provides links to related command HELP files. RELATED HELP 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. If the RELATED section conforms to the default pattern for processing, columns 1-24 on all lines in the RELATED section are not displayed by the HELP facility. For more information, see [“Creating HELPTASK Files”](#) on page 212.

The following screen sample shows the format of the RELATED section for the CMS ERASE command:

```
.CS 7 ON

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 (ALL      DELETE¢% - Removes one or more lines from
XEDIT DELETE (ALL      ¢%      a file while using XEDIT.
XEDIT DELETE (ALL
CMS DISCARD (ALL      DISCARD¢% - Removes files from "list-type"
CMS DISCARD (ALL      ¢%      CMS command environments, such
CMS DISCARD (ALL      ¢%      as FILELIST.
CMS DISCARD (ALL
CP PURGE (ALL      PURGE¢% - Removes spool files from your
CP PURGE (ALL      ¢%      reader, printer, or punch.

.CS 7 OFF
```

Figure 71. Format of the RELATED Section of the CMS ERASE Command

Note: The ¢% characters are highlighting characters, which are discussed on [“Highlighting Words within a File”](#) on page 220.

Tip: A RELATED section is typically in a file that also contains other, non-linking sections.

If a RELATED section is in a file that contains no other, non-linking sections, the HELP facility yields DMSHEL244W if the file is displayed without specifying the (RELATED option.

Creating HELP Files for Messages

The conventions for message HELP files is explained in the following sections.

You might want to create your own HELP files for messages. The file names for your HELP files must conform to the file name conventions that are described in [“Message HELP name conventions”](#) on page 220. The file type for HELP files for messages is HELPMMSG.

The z/VM HELPMMSG files provide a template for the kind of information that you might want to include in HELPMMSG files that you create. The z/VM HELPMMSG files contain the message code and message text, an explanation of why you received the message, the system action, and the user response.

The structure of HELPMMSG files is simpler than the structure of command HELP files. The .CS control words that identify information layers and sublayers in command HELP files do not apply to HELPMMSG files. The HELP command ignores the layering options and subsetting options when you specify a HELPMMSG file.

Message HELP name conventions

When you change the existing message HELP files or write your own message HELP files, you must adhere to the following name conventions:

- The combination of file name and file type must be unique.
- The file type of a message HELP file is HELPMMSG.
- The file name for messages has the form *xxxnnn(n)s*,

Where:

xxx

specifies the component code prefix (for example, DMS for CMS messages). For a list of the component code prefixes, see [z/VM: CMS and REXX/VM Messages and Codes](#).

nnn(n)

specifies the message number.

s

specifies the message type code. For example, E specifies error messages in CMS.

For example, consider CMS message DMSABC001E, which might appear in a log as follows:

```
DMSABC001E No filename specified
```

The HELP file name is DMS001E HELPMMSG.

Highlighting Words within a File

When you create and format your own HELP files, you might want 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. For example, if the screen width is 80 characters wide and a line is 90 characters long and contains 15 control character pairs, the line will take up two display lines (because the line length is greater than the display width). However, when formatted, the text will appear on one display line, because 15 characters are hidden.

Note: With TASK menus or RELATED sections that conform to the default formatting pattern, the selection entries are automatically displayed by HELP as highlighted lines. If you want to highlight only a portion of these lines, use the control characters `¢%` to turn highlighting off. 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 `¢%` turns highlighting off after selection entries.

Using Command Abbreviations

An abbreviations file contains command synonyms and abbreviations, which the HELP facility uses to search for HELP files.

Benefits of abbreviations files

Abbreviations and synonyms that are defined in an abbreviations file provide more opportunities to locate a HELP file. The role of an abbreviations file is similar to a synonym file discussed in [“Using Synonyms” on page 159](#).

An abbreviations file can be used to provide help for command names that contain special characters other than those special characters that are supported by the HELP facility. For more information about translation of special characters in command names, see [“Command HELP name conventions” on page 215](#).

Abbreviations files increase the responsiveness of the HELP facility. Even if an abbreviations file contains no abbreviation or synonym definition, it can decrease the time that the HELP facility requires to search for a HELP file. If no abbreviations are appropriate for any files in the HELP component, you can create an abbreviations file that contains only a comment line, which is indicated by an asterisk (*) in column 1.

Abbreviations file name convention

An abbreviations file name must identify a HELP component. For more information, see item [“4” on page 223](#).

An abbreviations file type must be HELPABBR.

Abbreviations file structure

Each line in an abbreviations file contains 2-3 words:

File name

The first word is the file name of the command file. The file type is not included. The file name is required.

Synonym

The second word is the file name of the command file or a synonym. The second word is required.

Minimum abbreviation

The third word is a number, 1-8, that specifies the minimum abbreviation length that is allowed when the second word is specified as a *cmd_name* operand on the HELP command. If the number is omitted, the default value is the length of the second word or 8, whichever is less. If the number is 9 or more, the definition is not valid.

The order of the entries in the HELPABBR file can affect which command file is matched by the filename operand of the HELP command. The first entry that matches the filename operand on a HELP command determines the help topic that is retrieved. Other matches after the first match are ignored. The following table shows an example of which command file is retrieved given a set of HELPABBR file entries and a HELP command with an abbreviated filename operand. Assume the abbreviations file is ASSOCIAT HELPABBR and two files with similar names are EXIT_A HELPASSO and EXIT_B HELPASSO.

Table 34. The order of entries in a HELPABBR file and the retrieved command file

ASSOCIAT HELPABBR entries	HELP commands	Filename of the retrieved help file
EXIT_A exit_a 2 EXIT_B exit_b 1	help associat e	EXIT_B
	help associat ex help associat exi help associat exit help associat exit_	EXIT_A
	help associat exit_b	EXIT_B
EXIT_B exit_b 1 EXIT_A exit_a 2	help associat e help associat ex help associat exi help associat exit help associat exit_	EXIT_B
	help associat exit_a	EXIT_A

Requirements for a search that uses an abbreviations file

The HELP facility always searches for a command file whose file name is specified by the command name (*cmd_name*) operand of the HELP command but can also search for synonyms and abbreviations of the command name. The synonyms and abbreviations must be defined in an abbreviations file (file type HELPABBR). There are matching requirements among the operands of the HELP command, the file name and contents of the HELPABBR file, and the file name and file type of the target HELP file. Conditions “2” on page 222 and “4” on page 223 indicate that one set of matching requirements is satisfied if the *component_name* operand of the HELP command and the file name of the abbreviations file are a HELP component name that accurately identifies the appropriate HELP component.

The HELP facility retrieves and displays a HELP file by using an abbreviations file when all the following conditions are true:

1. The HELP facility does not find a HELP file whose file name matches the *cmd_name* and that belongs to the HELP component that is specified by the *component_name* operand.
2. The *component_name* operand of the HELP command matches an abbreviations file name. The match criterion depends on the length of the abbreviations file name:
 - When the abbreviations file name is 8 characters, the first 8 characters of *component_name* must match the abbreviations file name.
 - When the abbreviations file name is less than 8 characters, *component_name* must match the abbreviations file name exactly.

The following table provides examples where the HELP command *component_name* operand matches an abbreviations file name.

Table 35. HELP command *component_name* operands that match an abbreviations file name

HELP command <i>component_name</i> operand	Abbreviations file
cp	CP HELPABBR
display	DISPLAY HELPABBR
directory	DIRECTOR HELPABBR
rscs	RSCS HELPABBR
rscsauth	RSCSAUTH HELPABBR
rscsauthorization	RSCSAUTH HELPABBR

3. The `cmd_name` operand of the HELP command matches an abbreviation or synonym that is defined in the abbreviations file that is identified in item “2” on page 222. The match criterion depends on the length of the command file name:
- When the `cmd_name` operand of the HELP command contains 8 or more characters, the first 8 characters of the `cmd_name` operand must match the first 8 characters of the abbreviation or synonym.
 - When the `cmd_name` operand is less than 8 characters, the `cmd_name` operand must match the abbreviation or synonym exactly.

The following table provides examples where the `cmd_name` operand of the HELP command matches an abbreviation or synonym. The abbreviation and synonym definition examples are from the DISPLAY HELPABBR abbreviations file. The abbreviations and synonyms are defined for the GUESTZ HELPDISP command file.

Table 36. HELP command `cmd_name` operands that match an abbreviation or synonym definition

HELP command <code>cmd_name</code> operand	Abbreviation or synonym definition		
gueststg	GUESTZ	GUESTSTG	8
z	GUESTZ	Zarchitecture	1
zarch	GUESTZ	Zarchitecture	1
zarchitecture	GUESTZ	Zarchitecture	1
zarchite9012345	GUESTZ	Zarchitecture	1

4. The HELP file whose name matches the synonym or abbreviation is in the identified HELP component. In this case, "HELP component" is defined strictly as the last 5-8 characters (suffix) of a HELP file type. The HELP component is identified by the file name of the HELPABBR file. The identification criterion depends on the length of the file name of the HELPABBR file:
- When the length of the file name of the HELPABBR file is four or more characters, the first four characters of the file name of the HELPABBR file must match the last four characters of a HELP file type.
 - When the length of the file name of the HELPABBR file is three or fewer characters, the file name of the HELPABBR file must match a HELP file type exactly.

The following table provides examples of abbreviations file names that identify HELP components.

Table 37. Abbreviations file names that identify HELP components

File type	File type suffix	Abbreviations file
HELPCP	CP	CP HELPABBR
HELPDISP	DISP	DISPLAY HELPABBR
HELPDIRE	DIRE	DIRECTOR HELPABBR
HELPRSCS	RSCS	RSCS HELPABBR
HELPRSCS	RSCS	RSCSAUTH HELPABBR

In the following example, the conditions are satisfied for the HELP facility to retrieve and display a HELP file by using an abbreviations file. The target file is the GUESTZ HELPDISP file, which contains information about the CP DISPLAY GUESTZ command. The CP DISPLAY GUESTZ command displays the storage

contents of a virtual machine in z/Architecture mode. The DISPLAY HELPABBR abbreviations file contains the following synonyms and abbreviations definitions for the GUESTZ HELPDISP file name:

GUESTZ	Zarchitecture	1
GUESTZ	GUESTSTG	8

The HELP facility retrieves and displays the GUESTZ HELPDISP file when you enter any of the following HELP commands:

```
help display gueststg
```

```
help display z
```

```
help display zarch
```

```
help display zarchitecture
```

```
help display zarchite901234
```

The example that contains zarchite901234 shows that the matching criterion is met when the first eight characters of the *cmd_name* operand match the first eight characters of the synonym definition. The matching term is zarchite.

Of course, the HELP facility also retrieves and displays the GUESTZ HELPDISP file when you use the file name of the command file as the *cmd_name* operand:

```
help display guestz
```

Abbreviations for HELPMENU and HELPTASK files

Abbreviations files typically define synonyms and abbreviations for command HELP files. However, the HELP facility can use abbreviations files to match the names of HELPMENU and HELPTASK files. In this case, the menu file type suffixes ("MENU" and "TASK") play the role of HELP component names and the menu file name plays the role of the command file name.

For example, create a MENU HELPABBR file with the following synonym definition for the PIPE HELPMENU file:

pipe	cmspipe	4
------	---------	---

After you create the MENU HELPABBR file, the following command displays the PIPE HELPMENU file:

```
help menu cmsp
```

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 command_name
```

where *command_name* is the name of one of your commands. This will call 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. For more information, see [“Creating HELP Files” on page 207](#).

If you have used the parsing facility and defined your command syntax using the Definition Language for Command Syntax (DLCS), you must also create or 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 HELPABBR 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 and called it 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.

For more information on using the parsing facility and defining command syntax using DLCS, see [z/VM: CMS Application Development Guide](#).

Using HELPCONV to Create HELP Files

HELPCONV is an additional text processing tool that helps you format your HELP files. HELPCONV control words can:

- 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.

Note: HELPCONV control words are not all recognized by the commonly used SCRIPT command structure. In addition, the format of the operands may differ between HELPCONV and SCRIPT. Care must be taken to use the correct control words and formats for the target processor.

These control words are summarized in [Table 38 on page 226](#). The following format words remain in the file:

```
.CS
.CM
.MT
```

The output file has the file type \$HLPxxxx.

Where:

XXXX

Identifies the last four characters of the file type of the input file.

Table 38. *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)	<i>n</i> ON/OFF or <i>keyword</i> 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)	<i>n</i> + <i>n</i> - <i>n</i>	Indents only the next line the specified number of spaces.	Yes	0
.IN (INDENT)	<i>n</i> + <i>n</i> - <i>n</i>	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)	<i>n</i> + <i>n</i> - <i>n</i>	Provides a technique for indenting all but the first line of a section.	Yes	0
.SP (SPACE)	<i>n</i>	Specifies the number of blank lines to be inserted before the next output line.	Yes	1
.TR (TRANSLATE)	<i>s t</i>	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 used by others 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.

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 this 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 215. 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, DESCRIPT, FORMAT, PARMS, OPTIONS, NOTES, ERRORS, and RELATED. (See [z/VM: CMS Commands and Utilities 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, 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.

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).

Use .FO ON to restore default HELP formatting, including both justification and concatenation of lines.

Use .FO OFF to cancel concatenation of input lines and justification of output lines. Subsequent text is printed as is. If you use the .FO control word with no operands, ON is assumed.

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.

Note: You may not want this type of formatting in all cases; you may want certain output to appear exactly as it appears in your file or when presented on the prior releases of the HELP Facility. 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 without intervening text, the indent amount is modified to reflect all the .IL values encountered; that is, the increments are added together. For example, the following two lines result in the next line being indented 10 spaces:

```
.il 4
.il +6
```

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.
This line is indented.
```

results in:

```
This line is not indented.
  This line is 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 (unlike the .IL control word which indents only one line of text). .IN 0 cancels the indentation, and output continues at the original left margin setting.

.IN cancels any .OF (Offset) setting (discussed on “.OF (Offset)” on page 231). 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, *nnnnn*, from the XEDIT menu, it is equivalent to entering, HELP XEDIT *nnnnn*. If the line .MT *xxxxx* was included in the file, selecting a command from the menu would be equivalent to entering HELP *XXXXX nnnnn*.

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 that can be 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.

For example, entering the following text with .OF control words at the beginning and end of material you want to 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.
```

results in:

```
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.
```

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.
```

Note: For HELP files other than HELPMENU and HELPTASK, two or more consecutive blank lines will be displayed as only one blank line so that the user will see as much useful information in as little space as possible. Therefore, when creating these HELP files, multiple blank lines may be added for ease of editing without concern for how their display might affect the user.

.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 changed. 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 (X'40' is the EBCDIC representation of a blank).

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 207.

If you add HELP files to an existing component, you should follow the naming conventions for HELP files given in this section. If you update a component, you should also update its menu. For information on menus see [“Creating Menus for HELP Files”](#) on page 207.

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 for SFS Users: 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 filename filetype`. If you delete a file, you should delete all links to the file.

Altering Existing HELP Files

To alter a HELP file:

1. Edit the file with a text processing editor (XEDIT).
2. Add or delete as you wish, making sure that you follow the instructions given in [“Creating HELP Files”](#) on page 207.
3. When using HELPCONV on a file that contains a RELATED section, formatting should be turned off for the entire RELATED section.

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 the 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.

Part 4. Using Full-Screen CMS

You can alter the display of your CMS session by using windowing commands to display information in boxes on top of the information currently on your screen. The following topics show you how to use windowing commands and full-screen CMS to customize your CMS session.

Chapter 10, “Introducing Full-Screen CMS,” on page 237 describes how to call full-screen CMS and use windowing functions to view information in windows on your physical screen.

Chapter 11, “Customizing Full-Screen CMS,” on page 259 describes ways in which you can change the attributes of your windows and virtual screens.

Chapter 10. Introducing Full-Screen CMS

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:

- Windows
- Virtual Screens

You can use windowing for your own purposes by using the windowing commands provided with CMS. If you wish, you may also ask CMS to use windowing for command input and output, messages, and certain other types of output. This facility is called *full-screen CMS*. 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. Remember that you can always use windowing for your own purposes even if you choose not to use full-screen CMS.

This section explains windowing support. It is divided into two sections: [“What Are Windows and Virtual Screens?”](#) on page 237 and [“Using Full-Screen CMS”](#) on page 238.

[“What Are Windows and Virtual Screens?”](#) on page 237 defines windows and virtual screens. [“Using Full-Screen CMS”](#) on page 238 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 VSCREEN DEFINE command in [z/VM: CMS Commands and Utilities Reference](#).

Because a window shows you a portion of a virtual screen, you can perform several operations on 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.

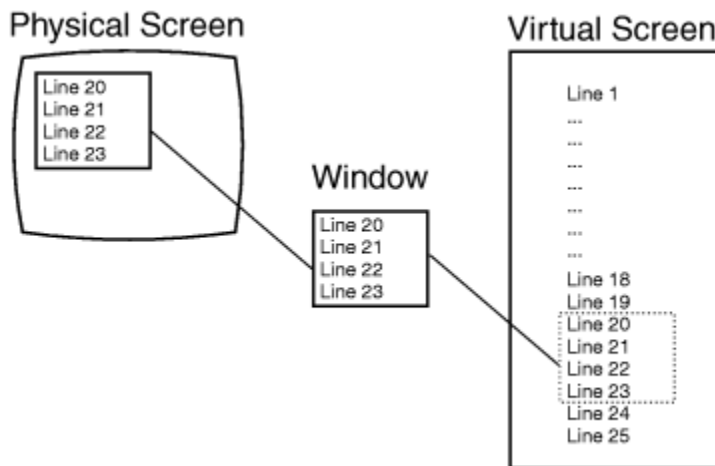


Figure 72. A Window into a Virtual Screen

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 [z/VM: CMS Commands and Utilities 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 WINDOW POP and WINDOW DROP.

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. Your PF keys are referred to as CMSPF keys. Two of these can be used to 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-run.

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 section 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 the example in [Figure 73](#) on page 239.

```

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
  
```

The screenshot shows a terminal window titled 'Fullscreen CMS'. At the top right, 'Columns 1 - 79 of 81' is displayed. Below the title, it says 'Ready;' followed by a hyphen. A large, empty rectangular area in the center is the data area. At the bottom, there is a row of PF key definitions: 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. Below this is a line with '====>' and a timestamp '15:15:08'. At the very bottom, it says 'Enter a command or press a PF or PA key'.

Figure 73. 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 241.

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 backward and forward 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 *default* 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 next section gives a description of each section of the status area.

Status Area Information

The status area, located on the bottom line of the physical screen (line **6** in Figure [Figure 73 on page 239](#)), contains the following three indicators:

Clock

Indicates the current time in hours, minutes, and seconds.

Message Class

Blank in the cited example, it 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:

- Message
- Warning
- Network.

Execution State

Reflects the status of the session. During a full-screen CMS session, one of the following execution state responses will appear in the status area.

Enter a command or press a PF or PA key

The system is waiting to process your next input.

Note: If you enter 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.

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 to 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: WINDOW CLEAR, VSCREEN CLEAR, WINDOW SHOW, or WINDOW HIDE. 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.

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 (see **2** in Figure [Figure 73 on page 239](#)), 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 [Table 39 on page 241](#).

Table 39. 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 section 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 11, “Customizing Full-Screen CMS,” on page 259](#).

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 anytime a command or response is requested in the CMS virtual screen. 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 window or windows showing the active virtual screen are not visible on the screen. (The active virtual screen is the one in which a command or response is requested.) For example, you may maximize a window that is not showing the active virtual screen so that it covers the window or windows showing the active virtual screen. Since you cannot enter commands in the active virtual screen, the WM environment is automatically entered; the WMPF keys and WM command line are available to manipulate all windows. You could then press WMPF 12 to restore the window and WMPF 3 to exit the WM environment.

When the WM window appears because the active window or windows are 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 following applies:
 - the CMS virtual screen is updated, or
 - any virtual screen (other than CMS) is updated and a pop-type window is visible, showing the update,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 anytime a command or response is requested in the CMS virtual screen. To do this, you would simply enter the command WINDOW POP WM or press the PA1 key (which in full-screen CMS defaults to the command WINDOW POP 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.

The commands you can enter from the WM window are:

CP	WINDOW DOWN
HELP	WINDOW DROP
PSCREEN PUT	WINDOW FORWARD
QUERY BORDER	WINDOW HIDE
QUERY HIDE	WINDOW LEFT
QUERY LOCATION	WINDOW MAXIMIZE
QUERY RESERVED	WINDOW MINIMIZE
QUERY SHOW	WINDOW NEXT
QUERY WINDOW	WINDOW POP
QUERY WMPF	WINDOW POSITION
SET BORDER	WINDOW RESTORE
SET LOCATION	WINDOW RIGHT
SET RESERVED	WINDOW SET
SET WMPF	WINDOW SHOW

WINDOW BACKWARD

WINDOW SIZE

WINDOW BOTTOM

WINDOW TOP

WINDOW CLEAR

WINDOW UP

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 254](#).

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. These pseudonyms are listed in [Table 40 on page 243](#). In addition to pseudonyms, optional keywords indicating when the command is processed, in relation to other commands entered at the terminal, and what is displayed on the virtual screen are also listed. This optional keyword can be one of these:

ECHO

The command is processed immediately when the function key is pressed. The key definition is echoed on the CMS virtual screen.

NOECHO

The command is processed immediately when the function key is pressed. The key definition is not echoed on the CMS virtual screen.

DELAYED

Delays the execution of the command string. When the key is pressed, the command is displayed in the input area and is not processed until you press Enter. If anything is currently in the input area, it is overlaid and no commands entered on the physical screen are processed. DELAYED is the default setting if no keyword is specified on the SET CMSPF command.

Table 40. CMSPF Key Settings

CMSPF Key	Pseudonym	Optional Keyword	Command
CMSPF 1	Help	ECHO	HELP
CMSPF 2	Pop_Msg	NOECHO	WM WINDOW POP MESSAGE *
CMSPF 3	Quit	NOECHO	SET FULLSCREEN SUSPEND
CMSPF 4	Clear_Top	NOECHO	#WM WINDOW CLEAR =
CMSPF 5	Filelist	ECHO	EXEC FILELIST

Table 40. CMSPF Key Settings (continued)

CMSPF Key	Pseudonym	Optional Keyword	Command
CMSPF 6	Retrieve		RETRIEVE
CMSPF 7	Backward	NOECHO	#WM WINDOW BACKWARD CMS 1
CMSPF 8	Forward	NOECHO	#WM WINDOW FORWARD CMS 1
CMSPF 9	Rdrlst	ECHO	EXEC RDRLIST
CMSPF 10	Left	NOECHO	#WM WINDOW LEFT CMS 10
CMSPF 11	Right	NOECHO	#WM WINDOW RIGHT CMS 10
CMSPF 12	Cmdline	NOECHO	VSCREEN CURSOR 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 appear as in the example in [Figure 74 on page 244](#).

```

                                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 WINDOW POP MESSAGE *
CMSPF 03 Quit      NOECHO     SET FULLSCREEN SUSPEND
CMSPF 04 Clear_Top NOECHO     #WM WINDOW CLEAR =
CMSPF 05 Filelist  ECHO      EXEC FILELIST
CMSPF 06 Retrieve  RETRIEVE
CMSPF 07 Backward  NOECHO     #WM WINDOW BACKWARD CMS 1
CMSPF 08 Forward   NOECHO     #WM WINDOW FORWARD CMS 1
CMSPF 09 Rdrlst    ECHO      EXEC RDRLIST
CMSPF 10 Left      NOECHO     #WM WINDOW LEFT CMS 10
CMSPF 11 Right     NOECHO     #WM WINDOW RIGHT CMS 10
CMSPF 12 Cmdline   NOECHO     VSCREEN CURSOR CMS -2 8 (RES
CMSPF 13 Help      ECHO      HELP
CMSPF 14 Pop_Msg   NOECHO     #WM WINDOW POP 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=Rdrlst    10=Left     11=Right     12=Cmdline
====>
15:32:11                                Enter a command or press a PF or PA key

```

Figure 74. 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, 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 processed in the CMS window. You can set your CMSPF keys without using #WM; however, your PF key commands can be processed 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:

CP

WINDOW DOWN

PSCREEN PUT	WINDOW FORWARD
QUERY BORDER	WINDOW HIDE
QUERY HIDE	WINDOW LEFT
QUERY LOCATION	WINDOW MAXIMIZE
QUERY RESERVED	WINDOW MINIMIZE
QUERY SHOW	WINDOW NEXT
QUERY WINDOW	WINDOW POP
QUERY WMPF	WINDOW POSITION
SET BORDER	WINDOW RESTORE
SET LOCATION	WINDOW RIGHT
SET RESERVED	WINDOW SET
SET WMPF	WINDOW SHOW
WINDOW BACKWARD	WINDOW SIZE
WINDOW BOTTOM	WINDOW TOP
WINDOW CLEAR	WINDOW UP
WINDOW DROP	

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, the pseudonym, optional keyword, and associated command.

For example, 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. Your screen should resemble the example shown in [Figure 75 on page 246](#).

```

                                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 WINDOW CLEAR =
CMSPF 17 Filelist      ECHO    EXEC FILELIST
CMSPF 18 Retrieve      NOECHO  RETRIEVE
CMSPF 19 Backward      NOECHO  #WM WINDOW BACKWARD CMS 1
CMSPF 20 Forward       NOECHO  #WM WINDOW FORWARD CMS 1
CMSPF 21 Rdrlist       ECHO    EXEC RDRLIST
CMSPF 22 Left          NOECHO  #WM WINDOW LEFT CMS 10
CMSPF 23 Right         NOECHO  #WM WINDOW RIGHT CMS 10
CMSPF 24 Cmdline       NOECHO  VSCREEN CURSOR 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 75. Setting CMSPF 9 to TELL

For more information on the SET CMSPF command, see [z/VM: CMS Commands and Utilities Reference](#).

PA1 Key

In addition to PF keys, you have a PA1 key on your keyboard that is assigned to WINDOW POP 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.

When CMS is the active virtual screen, PA1 pops the WM window. As discussed in “The WM Window” on page 241, the WM window provides you with a command line and a set of PF keys for manipulating other 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 CMSPF key settings. [Table 41 on page 247](#) lists the WM settings.

Table 41. WMPF Key Settings

WMPF Key	Pseudonym	Optional Keyword	Command
WMPF 1	Help	NOECHO	HELP
WMPF 2	Top	NOECHO	WINDOW TOP =
WMPF 3	Quit	NOECHO	WINDOW DROP WM
WMPF 4	Clear	NOECHO	WINDOW CLEAR =
WMPF 5	Copy	NOECHO	PSCREEN PUT COPY SCREEN
WMPF 6	Retrieve		RETRIEVE
WMPF 7	Backward	NOECHO	WINDOW BACKWARD = 1
WMPF 8	Forward	NOECHO	WINDOW FORWARD = 1
WMPF 9	Maximize	NOECHO	WINDOW MAXIMIZE =
WMPF 10	Left	NOECHO	WINDOW LEFT = 10
WMPF 11	Right	NOECHO	WINDOW RIGHT = 10
WMPF 12	Restore	NOECHO	WINDOW RESTORE =

Note: 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.)

Now, send a message to yourself. If you completed the previous exercise, you can press PF9 to display TELL at the command line. Complete a message to yourself, where *myuserid* is your user ID.

```
TELL myuserid This message is for myself.
```

When you receive your message, you will be notified as the:

- Terminal alarm sounds
- Status area message class indicator is updated to show that you have received a message, and the message window with the MESSAGE pops up.

Notes:

1. The MESSAGE window must contain at least one message before the window is displayed.
2. By using the CMS 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. For more information on the SET WINDOW command, see [z/VM: CMS Commands and Utilities Reference](#).

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 WINDOW POP and WINDOW DROP 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:

```
window drop message
```

If you want to redisplay the MESSAGE window, press CMSPF 2 that is assigned to pop the MESSAGE window or enter WINDOW POP MESSAGE.

Using the Message Window

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 shown in [Figure 76 on page 248](#).

```
====> NAMES (Mail panel)      File: VMUSER  NAMES  A0      <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: Rori                Notebook:
Userid:
Node: Sky

                Name: Aurora Borealis
                Phone:
                Address:
                :
                :
                :
List of Names:
                :
                :
                :
Tag:                Value:
Tag:                Value:

1= Help      2= Add      3= Quit      4= Clear      5= Find      6= Change
7= PrevNick  8= NextNick 9= Delete   10= PrevScrn 11= NextScrn 12= Cursor
=====> Screen 1 of 1 <=====

====>

Macro-read 1 File
```

Figure 76. 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, and the MESSAGE window appears on top of the names file. Your screen should resemble the example shown in [Figure 77 on page 249](#).

```

====> NAMES (Mail panel)      File: VMUSER  NAMES  A0      <====
Fill in the fields and press a PFkey to display and/or change your names file
Nickname: RORI                Notebook:
Userid:
Node: SKY

                                Name: Aurora Borealis
                                Phone:
                                Address:
                                :
                                :
                                :
+-----+-----+
|                                     Messages                                     |
| 15:35:16 MSG FROM VMUSER1 : Hi there!                                         |
| 15:48:19 MSG FROM VMUSER2 : Rori's user id is BOREAL.                         |
+-----+-----+

Tag:          Value:
Tag:          Value:

1= Help      2= Add      3= Quit      4= Clear      5= Find      6= Change
7= PrevNick  8= NextNick 9= Delete   10= PrevScrn 11= NextScrn 12= Cursor
=====> Screen 1 of 1 <=====

====> _
Macro-read 1 File

```

Figure 77. 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 WINDOW CLEAR MESSAGE. If you have received several messages, you may need to enter the command WINDOW 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 WINDOW DROP MESSAGE command. You can also use the WINDOW CLEAR MESSAGE command that scrolls the window forward past the current messages and removes the window from your screen. The WINDOW CLEAR 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 CMSPF 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 last command you entered is displayed on the command line. If you press it again, the next to the last line is displayed. 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 myuserid This message is for myself.  
Ready;  
window drop 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 WINDOW DROP MESSAGE that currently appears on your screen. Type POP over DROP and press Enter to process the WINDOW POP MESSAGE command.

The MESSAGE window reappears on your screen as shown in [Figure 78 on page 250](#).

```
Fullscren CMS                               Lines 33 - 38 of 38  
                                           Columns 1 - 79 of 81  
  
window drop message  
Ready;  
names  
Ready;  
window pop message  
Ready;  
  
+-----+  
| 15:35:16 MSG FROM VMUSER1 : Hi there! |  
| 15:48:19 MSG FROM VMUSER2 : Rori's user id 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=Cmdline  
====>  
15:45:02                      Enter a command or press a PF or PA key
```

Figure 78. Popping the MESSAGE Window

To drop the MESSAGE, move your cursor under the command WINDOW DROP 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. If you don't want your messages or warnings to be logged, use the SET LOGFILE OFF command. For details on how to log information, see the SET LOGFILE command in [z/VM: CMS Commands and Utilities 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.

For more information on the borders which are optional and are set on or off see the, WINDOW DEFINE and SET BORDER commands in *z/VM: CMS Commands and Utilities Reference*. 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:

```
vscreen clear 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!
```

Your messages now appear in the MESSAGE window. The corners of the window border are represented by a plus (+) sign.

```

                                Fullscreen CMS                                Lines 33 - 45 of 45
                                Columns 1 - 79 of 81

window drop message
Ready;
names
Ready;
window pop message
Ready;
window drop 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 79. 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.

```

                                Fullscreen CMS                                Lines 33 - 45 of 45
                                Columns 1 - 79 of 81

window drop message
Ready;
names
Ready;
window pop message
Ready;
window drop 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
====>
15:52:20 Message                                Enter a command or press a PF or PA key

```

Figure 80. 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.

Figure 81 on page 252 shows the result of scrolling forward.

```

                                Fullscreen CMS                                Lines 33 - 45 of 45
                                Columns 1 - 79 of 81

window drop message
Ready;
names
Ready;
window pop message
Ready;
window drop 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 81. 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.

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 the example in [Figure 82 on page 253](#).

```

                                Fullscreen CMS                               Lines 33 - 45 of 45
                                Columns 1 - 79 of 81
window drop message
Ready;
names
Ready;
window pop message
Ready;
window drop 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
====>
15:53:54                Enter a command or press a PF or PA key

```

Figure 82. Scrolling Backward through a Window Border

Scrolling Right and Left

Next, try scrolling the window to the right and left. Enter an `x` 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 now should look like the example shown in Figure 83 on page 253.

```

Fullscreen CMS                               Lines 33 - 45 of 45
Columns 1 - 79 of 81

window drop message
Ready;
names
Ready;
window pop wm
Ready;
window drop 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 83. Scrolling to the Right through a Window Border

Now, return the window to its previous position. Enter an L in a corner. The resulting window should now look like the example shown in Figure 84 on page 254.

```

Fullscreen CMS                               Lines 33 - 45 of 45
                                           Columns 1 - 79 of 81

window drop message
Ready;
names
Ready;
window pop message
Ready;
window drop 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 84. Scrolling to the Left through a Window Border

If you need to move a window somewhere else on your screen, use the *M* border command. Type an *m* in any corner of the window and then, before pressing Enter, move the cursor to the location on the screen where you want that corner. The entire window must fit on the screen or you will get an error message.

If you move a window and partially cover another window, you can use the *P* border command to pop the window on the bottom. Just enter a *p* in any corner of the bottom window, and it will put the partially-covered window on top.

For more information on using border commands, see [z/VM: CMS Commands and Utilities 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 should now look like the example shown in [Figure 85 on page 255](#).

```

Fullscreen CMS                               Lines 33 - 45 of 45
                                           Columns 1 - 79 of 81

window drop message
Ready;
names
Ready;
window pop message
Ready;
window drop 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 85. 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 should look like the example shown in [Figure 86 on page 255](#).

```

Fullscreen CMS                               Lines 33 - 45 of 45
                                           Columns 1 - 79 of 81

window drop message
Ready;
names
Ready;
window pop message
Ready;
window drop message
Ready;
vscreen clear 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 86. 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 should now look like the example shown in [Figure 87 on page 256](#).

```

                                Fullscreen CMS                                Lines 33 - 46 of 46
                                Columns 1 - 79 of 81

window drop message
Ready;
names
Ready;
window pop message
Ready;
window drop 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 87. 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 should now look like the example shown in [Figure 88 on page 256](#).

```

                                Fullscreen CMS                                Lines 33 - 48 of 48
                                Columns 1 - 79 of 81

window drop message
Ready;
names
Ready;
window pop message
Ready;
window drop 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 88. 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 should now look like the example shown in [Figure 89 on page 257](#).

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 covers up the entire CMS virtual screen. The WM window provides you with an area to enter commands to manipulate the window that is covering up your screen.

The MESSAGE window should now look like the example shown in Figure 90 on page 257.

[illegible]

One way to exit from the WM environment is to press WMPF 3. This removes both the WM and MESSAGE windows from your screen.

```
set window message variable
```

Introducing Full-Screen CMS

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 read [Chapter 11, “Customizing Full-Screen CMS,” on page 259](#), for information on tailoring full-screen CMS and windowing support to your own particular needs.

Chapter 11. 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 WINDOW POP WM, when you press the PA1 key when CMS is the active virtual screen, or when the window is automatically displayed on your screen. (For more information on when the WM window is displayed, see [Chapter 10, “Introducing Full-Screen CMS,”](#) on page 237.)
- 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 on the SET FULLSCREEN and SET CMSPF commands, see [z/VM: CMS Commands and Utilities 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 section 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 [z/VM: CMS Commands and Utilities Reference](#).

Tailoring System Defaults

By defining windows and virtual screens before entering full-screen CMS, you can override full-screen CMS default definitions. 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 CMS 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. Enter the command as follows:

```
vscreen define 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:

```
window define 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:

```
vscreen write message 1 1 70 (reserved blank
```

Finally, change the title of the MESSAGE virtual screen. With the VSCREEN WRITE 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.

```
vscreen write 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 */  
'vscreen define message 35 70 2 0 (system protect white'  
'window define message 10 71 10 5 (system variable pop'  
'set border message on (all $'  
'set fullscreen on'  
'vscreen write message 1 1 70 (reserved blank'  
'vscreen write 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:

Remember, because you defined the window as variable in size, the size of the window varies depending on how many messages you receive.

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:

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.

popdrop

Chapter 11. Customizing Full-Screen CMS **261**

```

                                Fullscreen CMS                                Columns 1 - 79 of 81

Ready;
vscreen write message 1 1 70 (reserved blank
Ready;
vscreen write 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=Cmndline
====>
15:20:03                                Enter a command or press a PF or PA key

```

Figure 92. 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.

WINDOW POSITION

With the WINDOW POSITION command, you can move a window anywhere on the physical screen. Let's move our new MESSAGE window.

If you use an

```
=
```

in the command syntax instead of the window name, the command moves the topmost window that was defined with the TOP option. Because the MESSAGE window is currently the topmost window, you can enter the following command to move it:

```
window position = 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 in [Figure 93 on page 263](#).

WINDOW SIZE

```
window size = 8 70
```

WINDOW MAXIMIZE and WINDOW RESTORE

If you are following the exercises, the MESSAGE window is still on your screen. It has the title: *My Personal Message*. It also has one message in it: *Let's see what happened*. To see what happens with the WINDOW MAXIMIZE 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.
```

Chapter 11. Customizing Full-Screen CMS **263**

```

                                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 94. Popping the MESSAGE Window

Notice that the location indicator in the upper right corner of the window shows Lines 1 - 6 of 11. To see the remaining messages, you can maximize the MESSAGE window. Enter:

```
window maximize message
```

The MESSAGE window now looks like the example shown in [Figure 95 on page 264](#).

```

                                My Personal Messages

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.
15:33:14 MSG FROM VMUSER : This is message 6.
15:33:20 MSG FROM VMUSER : This is message 7.
15:33:27 MSG FROM VMUSER : This is message 8.
15:33:33 MSG FROM VMUSER : This is message 9.
15:33:41 MSG FROM VMUSER : This is message 10.
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

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
====>
15:43:45                               Enter a command or press a PF or PA key

```

Figure 95. Maximizing a Window

Even though the window is maximized, it does not fill the entire screen because the MESSAGE window is variable in size. It expands depending on how much data there is to display. You will notice that the window moved to the location of line 1 and column 1 on the screen and you can see all the messages.

If you maximized a window so that the window filled the entire screen, you might not be able to enter commands if the window is protected. In this instance, the WM window would be automatically displayed, and the WMPF keys and command line would be available to manipulate the window. For more information on the WM window, see [Chapter 10, "Introducing Full-Screen CMS," on page 237](#).

To return the window to its previous size and location on the screen, you can enter the WINDOW RESTORE command:

```
window restore message
```

Here is how your screen looks (the same as it was before using the WINDOW MAXIMIZE).

```

                                Fullscreen CMS                                Lines 33 - 41 of 41
                                                                Columns 1 - 79 of 81

Ready;
tell * This is message 9.
Ready;
tell * This is message 10.
Ready;
window maximize 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 96. The Window after WINDOW RESTORE

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 the example shown in [Figure 97 on page 266](#).

```

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;
window maximize 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 97. 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.

```

Fullcreen CMS
Lines 33 - 45 of 45
Columns 1 - 79 of 81

Ready;
tell * This is message 9.
Ready;
tell * This is message 10.
Ready;
window maximize message
Ready;
window restore 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 98. 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.


```

                                Fullscreen CMS                                Lines 33 - 47 of 47
                                Columns 1 - 79 of 81

Ready;
tell * This is message 9.
Ready;
tell * This is message 10.
Ready;
window maximize message
Ready;
window restore 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 99. Deleting a Window Title

Now, set the number of reserved lines at the top of the window to 1. Enter:

```
set reserved message 1 0
```

Your screen now looks like the example shown in [Figure 100 on page 267](#).

```

                                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;
window maximize message
Ready;
window restore 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=Rdrlist  10=Left     11=Right   12=Cmdline
====>
16:13:25                                Enter a command or press a PF or PA key

```

Figure 100. Deleting a Blank Reserved Line

Now, drop the MESSAGE window by typing a D in any corner. Then, enter the following command to clear the MESSAGE virtual screen of all old messages:

```
vscreen clear message
```

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 will not 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 WINDOW POP MESSAGE to pop the window.

```

                                     Fullscreen CMS                      Lines 49 - 55 of 55
                                     Columns 1 - 79 of 81
Ready;
vscreen clear 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=Cmdline
====>
16:15:51 Message                               Enter a command or press a PF or PA key
```

Figure 101. 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 [Figure 102 on page 269](#).

```

Fullscreen CMS                               Lines 49 - 59 of 59
                                           Columns 1 - 79 of 81

Ready;
vscreen clear 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=Rdrlst 10=Left    11=Right   12=Cmdline
====>
16:18:15 Message Enter a command or press a PF or PA key

```

Figure 102. 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 are lost.

Window and Virtual Screen Tables

If you are interested in tailoring windows or virtual screens, you may find it useful to refer to [Table 42 on page 269](#) for the full-screen CMS default characteristics for windows and [Table 43 on page 271](#) for virtual screens.

Table 42. Default Windows					
Window	Lines	Cols	Ppline	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

Table 42. Default Windows (continued)

Window	Lines	Cols	Psline	Pscol	Options
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

BORDER

The window borders are displayed when possible.

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.

FIXED

The number of lines in the window is always constant.

NOBORDER

The window borders are not displayed.

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.

NOTOP

The window cannot qualify as the topmost window.

POP

The window is displayed on top of all other windows when the virtual screen that the window is showing is updated.

Pscol

The column on the physical screen where the upper left corner of the window is placed.

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.

TOP

The window may qualify as the topmost window.

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.

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 WINDOW POP WM, when you press the PA1 key when CMS is the active virtual screen, or when the window is automatically displayed on your screen. All default windows are SYSTEM windows; they are not lost when the system abends, or when the HX (halt execution) command is entered.

Table 43. Default Virtual Screens

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

Dcolor

Data color (if your terminal is equipped for color)

NOPROTECT

You can type into the window(s) connected to virtual screen; the data is not protected.

PROTECT

You cannot type into the window(s) connected to the virtual screen because the data is protected.

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.

Rbot

Bottom reserved lines

Rtop

Top reserved lines

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 WINDOW POP WM, when you press the PA1 key when CMS is the active virtual screen, or when the window is automatically displayed on your screen.

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 abends, or when the HX (halt execution) command is entered.

These CMS commands are unusable for the STATUS default virtual screen:

- SET LOGFILE
- SET VSCREEN
- VSCREEN CLEAR
- VSCREEN CURSOR
- VSCREEN GET
- VSCREEN PUT
- VSCREEN ROUTE
- VSCREEN WAITT
- VSCREEN WAITREAD
- VSCREEN WRITE

However, the CMS commands VSCREEN DELETE and VSCREEN DEFINE 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.

This 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 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) do 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

You may also find it useful to refer to [Table 44 on page 272](#) which contains information regarding the default settings for message routing.

Table 44. 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

Table 44. Default Settings for Message Routing (continued)

Message Class	Virtual Screen	Options
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.

Migration Considerations

The following section provides specific information for users of CP, CMS, XEDIT, and applications, when using full-screen CMS. 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 fully use full-screen CMS capabilities.

CMS Considerations

- If the default CMS vscreen or the default WM vscreen are to be customized, be aware that the reserved lines cannot be altered until after the first time the vscreen is displayed. Any changes made through a VSCREEN WRITE command to the reserved area will have to be made after the screen has been displayed at least once. The reserved area includes the PF key definitions (CMSPF or WMPF) and the command line.
- 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 *z/VM: CMS Commands and Utilities Reference*. The CP SPOOL CONSOLE START command does not log full-screen CMS interactions.
- To copy a full-screen CMS screen image, use the PSCREEN PUT 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 VSCREEN DEFINE, SET VSCREEN, VSCREEN WRITE, 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 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. For more information on the WINDOW DROP and WINDOW HIDE commands, see [z/VM: CMS Commands and Utilities Reference](#). 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 remains unchanged.
- PF keys set to process Immediate commands (the key definition is preceded by #WM or #CP) are immediately processed. 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. For more information on the VSCREEN WAITREAD command, see [z/VM: CMS Commands and Utilities Reference](#). 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).
- 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 called. 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 called. If XEDIT has not been called and the Session Services commands 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 called, you must call 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. For more information on the CP SLEEP command, see [z/VM: CP Commands and Utilities Reference](#).
- 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 VSCREEN DEFINE, SET VSCREEN, VSCREEN WRITE, 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. For more information on the ROUTE command, see [z/VM: CMS Commands and Utilities Reference](#). This includes messages sent using 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).

XEDIT Considerations

- XEDIT does not carry out its own I/O, but rather windowing functions are responsible for XEDIT I/O. As a result, certain CMS settings affect the XEDIT environment, especially these:
 - SET APL and SET TEXT
 - SET FULLREAD
 - SET NONDISP
 - SET REMOTE
 - SET LANGUAGE (affects Double-Byte Character Set (DBCS) display and the nondisplayable character set)
- If the BRKKEY is set to ON, and you enter full-screen CMS, the BRKKEY setting is changed to OFF. For more information, see the SET BRKKEY command in [z/VM: XEDIT Commands and Macros Reference](#) and the CP TERMINAL BRKKEY command in [z/VM: CP Commands and Utilities Reference](#).
- In full-screen CMS, the default PA1 key for XEDIT (and the NAMES and SENDFILE commands) is COMMAND CMS WINDOW POP WM, if BRKKEY is not assigned to the PA1 key.

- 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

There are special considerations that apply if you are writing an application that will process when full-screen is active. For more information, see [*z/VM: CMS Application Development Guide for Assembler*](#).

Part 5. Using Execs and Programs in CMS

The CMS facilities known as the REXX/VM 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 the following topics, the exec facilities are described in general terms. You will be acquainted with the expressions used in exec files, and how execs function in CMS.

Also in these topics, you will learn how to create and run programs in the CMS environment.

Chapter 12, “Introduction to the Exec Processors,” on page 279 presents a survey of the basic characteristics and functions of exec facilities available to you.

Chapter 13, “Creating REXX Execs,” on page 283 describes how to create and use REXX execs. Sample execs are provided for you to try.

Chapter 14, “Creating a PROFILE EXEC,” on page 287 describes how you can create your own PROFILE EXEC.

Chapter 15, “Commands Used with REXX Execs,” on page 289 provides examples of using some CMS commands with REXX execs.

Chapter 16, “Developing Programs in CMS,” on page 299 shows you how to create, compile and run your programs in the CMS environment, using XEDIT as your tool.

Chapter 12. Introduction to the Exec Processors

The following EXEC processors and interpreters are available:

- The REXX/VM interpreter handles 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.

Note: EXEC 2 programs and processing are similar to those of the CMS EXEC. REXX programs are *not* similar to those of EXEC 2 or CMS EXEC.

REXX/VM Interpreter

The REXX/VM interpreter is an interpretive command and macro processor. It coexists with the CMS EXEC and EXEC 2 processors. REXX is functionally a superset of CMS EXEC and EXEC 2, but it uses a completely different language and syntax. There is no compatibility between REXX programs and those of CMS EXEC or EXEC 2.

z/VM differentiates REXX programs from CMS EXEC or EXEC 2 programs by their first statement. The first statement of every REXX 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 REXX/VM 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 REXX/VM 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 lesser of 16MB or the user's virtual storage size) to the length of manipulated data.
- It is easy to find syntax errors in a program. The REXX/VM interpreter processes 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 REXX/VM interpreter is interpreting a particular instruction. This should help you in debugging.

These books tell you how to use the REXX/VM interpreter and the REXX language:

- *z/VM: REXX/VM User's Guide* is a step-by-step, tutorial-like, guide to writing REXX programs. It is intended for new users.
- For more information on the REXX language and the REXX/VM interpreter, see *z/VM: REXX/VM Reference*. It is intended for all users.

As a CMS user, you should become familiar with the REXX/VM interpreter and use it often to tailor CMS commands to your own needs, as well as to create your own commands. The REXX/VM interpreter and the EXEC 2 interpreter can be used by the editor for XEDIT macro processing support.

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 these 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, XEDIT).

The exec control statements for EXEC 2 are part of the CMS General-Use programming interface. To obtain a list of all the control statements, enter:

```
help exec2 menu
```

To obtain information on a particular control statement, enter `help exec2` followed by the name of the statement. For example, to get information on the &TRACE control statement, enter:

```
help exec2 &trace
```

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 more information on the LISTIO command and the \$LISTIO EXEC, see [z/VM: CMS Commands and Utilities Reference](#).

The exec control statements for EXEC are part of the CMS General-Use programming interface. To obtain a list of all the control statements, enter:

```
help exec menu
```

To obtain information on a particular control statement, enter `help exec` followed by the name of the statement. For example, to get information on the &STACK control statement, enter:

```
help exec &stack
```

Note: The module that contains the CMS EXEC processor (DMSEXT) is loaded as a nucleus extension with the PERMANENT attribute, so it cannot be inadvertently dropped by a CMS exec.

Relationship of the Exec Interpreters

The REXX/VM, EXEC 2, and CMS EXEC interpreters have their own distinct keywords and syntax. So for example, you cannot place EXEC 2 statements within a REXX program.

The three interpreters coexist, so exec programs will continue to correctly process 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 REXX/VM 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 Procedures Language REXX/VM Interpreter, automatic cleanup of an active OS or VSAM environment is not performed at command completion, as it is in the CMS EXEC processor. It is your responsibility to ensure that OS and VSAM cleanup functions are performed when needed. You can accomplish this by using the EXECOS command. For more information on the EXECOS command, see [z/VM: CMS Commands and Utilities Reference](#).

The CMS EXEC processor invokes OS and VSAM cleanup after the execution of any CMS command. Consequently, any CMS EXEC used resets the OS and VSAM environments if it contains a CMS command that is processed.

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 through the EXEC command.

When an exec file is used, CMS examines the first statement of the exec file to determine which of the following exec processors must be called to handle it:

- REXX/VM interpreter
- EXEC 2 processor
- Compiled REXX
- EXEC processor.

For more information on how CMS handles running execs, see [z/VM: REXX/VM Reference](#).

Attributes of Exec Files

Exec files can have any file name that is valid for a CMS file name. EXEC 2 and REXX files have file type exec for files that are run from the CMS environment, and the file type XEDIT for files used as XEDIT macros.

REXX 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 13. Creating REXX Execs

A REXX file, like a CMS EXEC or EXEC 2 file, has a file type of EXEC. 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 REXX/VM interpreter processes files of any logical record length. 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

specifies the file name of the exec file.

For example, an exec named THANKYOU would be processed when you entered either:

```
exec thankyou or thankyou
```

You must precede the exec file name with the exec command when you:

- Run an exec from CMS EXEC 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 REXX 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:

```
query 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 REXX program.

One exec file that you never have to specifically process is a PROFILE EXEC. It automatically runs after you IPL CMS, when your directory or minidisk with a mode of A is accessed. PROFILE EXECs are discussed in [Chapter 14, “Creating a PROFILE EXEC,”](#) on page 287.

Sample REXX Execs

Here are two sample REXX execs to give you some flavor of the language.

The sample exec in [Figure 103 on page 284](#) copies 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
```

Figure 103. Sample REXX Exec - Copy a File

The sample exec in [Figure 104 on page 285](#) sends the file to the user ID that you specify. Note that in REXX 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

```

Figure 104. Sample REXX Exec - Send a File

For more information on REXX language, see [z/VM: REXX/VM User's Guide](#) and [z/VM: REXX/VM Reference](#).
 For more information on CMS functions that you can use from execs, see [z/VM: CMS Application Development Guide for Assembler](#).

Chapter 14. Creating a PROFILE EXEC

A PROFILE EXEC is different from other execs. It has the special file name, PROFILE, and it is automatically processed 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 REXX 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 */
'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 (SYSTEM'   /* and EXECUTE storage */
'EXECLOAD EXECUTE XEDIT (SYSTEM'   /* resident execs. */
```

Figure 105. Sample PROFILE EXEC

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.

If you used the AUTOLOG or XAUTOLOG command to log on and IPL a CMS virtual machine, and variable AUTOLOG data was passed, the SYSPROF EXEC or the PROFILE EXEC (or any other exec or module called by it) may destroy the AUTOLOG data (for example, by using DESBUF). See *z/VM: CMS Planning and Administration* for more information about the SYSPROF EXEC and about AUTOLOG data being destroyed in this situation.

You can enter the following command at any time to run your PROFILE EXEC:

```
profile
```

If you make changes to your PROFILE EXEC during your terminal session, the changes will not be in effect until you run your profile again.

Should you want to suppress the processing 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:

```
z/VM Vn.n.n    yyyy-mm-dd hh:mm
```

To suppress the processing 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 [z/VM: CMS Commands and Utilities 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	PROFDLST	XEDIT	RECEIVE	XEDIT
AUTHLIST	EXEC	PROFFDAT	XEDIT	SENDFILE	EXEC
DIRLIST	EXEC	PROFFLST	XEDIT	TELL	EXEC
DISCARD	EXEC	PROFFSEA	XEDIT	X\$DLST\$X	XEDIT
EXECUTE	XEDIT	PROFFSHR	XEDIT	X\$FLST\$X	XEDIT
FILELIST	EXEC	PROFMLST	XEDIT	X\$MLST\$X	XEDIT
MACLIST	EXEC	PROFNOTE	XEDIT	X\$NAME\$X	XEDIT
NAMES	EXEC	PROFPEEK	XEDIT	X\$NDIR\$X	XEDIT
NOTE	EXEC	PROFRLST	XEDIT	X\$PEEK\$X	XEDIT
PEEK	EXEC	PROFSEND	XEDIT	X\$SEND\$X	XEDIT
PROFALIA	XEDIT	RDRLIST	EXEC		
PROFAUTH	XEDIT	RECEIVE	EXEC		

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 running 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 [z/VM: CP Commands and Utilities Reference](#) and the SET INSTSEG, QUERY INSTSEG, and EXECMAP commands in [z/VM: CMS Commands and Utilities Reference](#).

Chapter 15. Commands Used with REXX Execs

These are some of the commands used along with REXX execs. Command formats, descriptions, and usage notes for these commands can be found in [z/VM: CMS Commands and Utilities 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 the processing 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 and VSAM 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 *userid* NAMES).

PIPE

Invokes CMS Pipelines to process a series of programs or stages. A series of stages is called a pipeline. Each stage manipulates or handles data by:

- Using the stage commands and pipeline subcommands provided by CMS Pipelines. Several of the stage commands allow you to get and set REXX variables.
- Extending the set of CMS Pipelines stage commands and allowing you to write your own stage commands in the REXX language.

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 EXECTRACE

Sets tracing ON or OFF for your REXX exec 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.

VSCREEN WAITREAD

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 XEDIT to create or modify a file.

These Immediate commands can be used along with REXX 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 REXX execs. You can use CMS commands in your execs to operate on your SFS files. Callable Services Library (CSL) routines are also available for use with REXX execs to operate on SFS files and minidisk files. For more information, see [z/VM: CMS Application Development Guide for Assembler](#).

Using EXECIO

The EXECIO command manages movement of lines between files or virtual devices and the program stack or a variable. It also causes the processing of CP commands and recovers resulting output.

Note: For manipulating CMS files, consider using Callable Services Library (CSL) routines instead of EXECIO. CMS file subsystem routines provide more functions than EXECIO. For more information on using CSL routines, see [z/VM: CMS Callable Services Reference](#) and [z/VM: CMS Application Development Guide](#). You may also want to consider using the PIPE command instead of EXECIO. For more information on the PIPE command, see [z/VM: CMS Pipelines User's Guide and Reference](#).

This explanation is not intended to teach you all you need to know to write REXX programs. For more information on REXX language, see [z/VM: REXX/VM User's Guide](#).

The example in [Figure 106 on page 291](#) illustrates how you might use EXECIO commands in a REXX 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 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 REXX 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).


```

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

```

Figure 106. The RDPRIND EXEC, an Example of the EXECIO Command Used in a REXX Exec

Following are numbered explanations of each statement in the RDPRIND EXEC:

1. The first statement in a REXX 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 place holder, 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* equals any multiple of 60.
11. If the previous condition checked (in line 10) is true, then this line is processed. If it is processed, 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 the regular 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 processing. The following command loads TPHONE EXEC into user free storage:

```
execload tphone exec a
```

When the exec is subsequently run, the storage resident exec is processed. 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 processed, you will have to do one of these:

- 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 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 these:

- Specify the PUSH option on the EXECLOAD command.
- 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.

Entering 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, the following command deletes the TPHONE EXEC from storage:

```
execdrop tphone exec
```

Note: When an EXEC or XEDIT macro has been EXECLOADED into storage and the EXEC or XEDIT macro is invoked through a CMSCALL on which a FBLOCK is supplied, the high-order bit of the FBLOCK address must be set on in order for the usage count reported by the EXECMAP to be incremented. For more information on File Block, see [z/VM: REXX/VM Reference](#).

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 running 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.

To change the file mode, use the SET INSTSEG command. For example, to access the saved segment at file mode B, enter:

```
set instseg on 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.

Using EXECOS

The EXECOS command resets the OS and VSAM environments under CMS without returning to the interactive environment. If you request a reset of the OS or VSAM environment, after the processing 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.

```
/* first exec */          /* second exec */
.                          .
.                          .
.                          .
second                    .
.                          globalv put 'RUMORS' /* assign variables */
globalv get 'RUMORS'      .
.                          .
.                          .
.                          exit
exit
```

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
```

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 PIPE

The PIPE command can help you solve a complex problem by breaking it up into a series of smaller, less complex programs. If you did not use the PIPE command, you might need several CMS or CP commands, or a program to solve the problem.

For example, to display a list of all the users that are logged on and the real address of the line to which each is connected, you can issue the CP QUERY NAMES command. The following is an example of the results of a CP QUERY NAMES command:

FURMAN	- 1F83, PUTTEST	- DSC , SHERMAN	- 01C3, BERT	- L0258
BARB	- 1FEB, SHERRI	- 0D7D, GROVER	- 0DD1, DEVPVM	- DSC
BATCHIT	- DSC , OSCAR	- DSC , SERVER3	- DSC , STONE	- 1F62
SMITH	- DSC , PIPER	- 01CE, LOGN03BE	- 03BE, PVM	- DSC
TAMMIE	- DSC , MACK	- DSC , ERNIE	- 0168, PETE	- DSC

Now suppose you wanted a list, in single column format, of only the users that are currently connected and the real address of the line to which each is connected. You could either write a program to do this or use the PIPE command. The following exec, called CONNECT EXEC, uses a single PIPE command to display a list of the connected users:

```
/* Displays a list of connected users */
'pipe',
'cp q n',
'| split ,',
'| strip',
'| nlocate /- DSC/',
'| console'
exit rc
```

CONNECT EXEC displays the following:

```
FURMAN   - 1F83
SHERMAN  - 01C3
BERT     - L0258
BARB     - 1FEB
SHERRI   - 0D7D
GROVER   - 0DD1
STONE    - 1F62
PIPER    - 01CE
LOGN03BE - 03BE
ERNIE    - 0168
```

For more information on the PIPE command, see [z/VM: CMS Pipelines User's Guide and Reference](#).

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
   REXX/VM interpreter sets the variable
   RC to this returned value. Next, depending on the
   returned value, this exec selectively runs 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 EXECTRACE

You can trace your REXX exec or EXEC 2 exec by specifying SET EXECTRACE ON prior to exec invocation. To turn tracing off, specify SET EXECTRACE OFF.

Using Windows and Virtual Screens

VSCREEN WAITREAD 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 (WINDOW DEFINE and VSCREEN DEFINE commands)
- Connect the window to the virtual screen (WINDOW SHOW command)
- Write data to the virtual screen (VSCREEN WRITE command)
- Enter the VSCREEN WAITREAD command.

When an attention interrupt is received, the exec can process the WAITREAD.*n* variables and update the virtual screen using the VSCREEN WRITE command. For more information on these commands, see [z/VM: CMS Commands and Utilities 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 REXX or EXEC 2 file invoked from the XEDIT environment.

For more information on writing XEDIT macros using the REXX language, see [z/VM: XEDIT User's Guide](#). For more information about XEDIT macros written in EXEC 2 language, see the Help Facility by entering:

```
help exec2
```

Exchanging Data Between Programs Through the Stack

For more information on the REXX instructions that place data into the program stack, see [z/VM: REXX/VM User's Guide](#).

Chapter 16. Developing Programs in CMS

Some of the programming languages used with the z/VM system are COBOL, C, FORTRAN, and PL/I. In this section, you will learn how to create and run programs, regardless of the programming language you use. You don't even have to know a programming language to complete this section. Just follow the examples. You do need to know how to use the XEDIT command to create a file.

For more information on the procedures and commands discussed in this section, see [z/VM: CMS Application Development Guide](#) and [z/VM: CMS Commands and Utilities Reference](#). For information on how to write programs in the various languages, ask your administrator what IBM manuals to use.

Creating a Program

A program is a file that contains instructions for the computer. You create a program file with an XEDIT command, the same way you create any other file. A program file can have any file name that is no longer than eight characters, but its file type indicates the name of the programming language you are using. Some of the special programming language file types used in z/VM are:

```
COBOL
C
FORTRAN
PLI
```

Based on the file type, the editor gives values to various settings, like tabs. For example, suppose a programming language requires you to enter data in certain columns. The tabs are set according to that language, which makes it easy to line up the data using PF4 (the tab key). For more information on these settings, see [z/VM: XEDIT Commands and Macros Reference](#).

Because many students, scientists, engineers, and other professionals use FORTRAN for problem solving, we have chosen a simple FORTRAN program to use for our example. Your system must have VS FORTRAN for the example to work. To find out if you have VS FORTRAN, either ask your system administrator, or enter:

```
file1 fortvs2 * *
```

If the FILELIST display contains FORTVS2 MODULE, you can do the example. If not, you can still use the procedures explained in this section, but you have to use a programming language that is in your system. If you are in the FILELIST environment, press PF3 to leave it.

Our program converts a Fahrenheit temperature to Celsius. The program asks you to enter a Fahrenheit temperature with a decimal point (for example, 32.0). It then displays that temperature in Celsius.

Now enter:

```
x degrees fortran
```

Enter the following command to enter input mode:

```
====> i
```

Now type these lines, exactly as shown.

```
00010 WRITE (6,15)
00015 FORMAT (' Enter FAHRENHEIT VALUE WITH DECIMAL POINT')
00020 READ (5,25)VALUE
00025 FORMAT (F4.1)
00030 OUTPUT = (VALUE - 32 ) / 1.8
00040 WRITE (6,45)OUTPUT
00045 FORMAT (F6.1, ' DEGREES CELSIUS')
00050 END
```

You are finished entering your program instructions. Press the Enter key twice to enter edit mode. Then enter:

```
====> file
```

Now you have created a *source program*.

Compiling a Program

If you know FORTRAN, your source program is meaningful to you, but it means nothing to the computer. The computer only understands machine language. You must now *compile* your program. That means you ask a compiler to translate the program into machine language.

Before you compile your program, you must make sure you have enough virtual storage for the compiler. You probably have enough storage for all compilers except VS FORTRAN, which is used in our example. To find out how much virtual storage you have, enter:

```
query storage
```

You will see a message similar to the following (which indicates you have 1024K or 1 Megabyte of storage):

```
STORAGE = 1024K
```

If the number is less than 4096K (4MB) and you are using VS FORTRAN, you need to get more storage. Here's how:

1. Enter this command: `define storage 4m`

If you see the following message, you must ask your system administrator to increase your storage limit to 4MB:

```
Storage exceeds allowed maximum
```

After your system administrator grants your storage request, reenter the `DEFINE STORAGE 4M` command.

In addition to the message you get that verifies your new storage, you may get a message similar to the following:

```
DMKDSP450W CP entered; disabled wait PSW '00020000 00000000'
```

This means you must re-IPL CMS to load CMS again.

2. Enter IPL CMS to load CMS. When the status message, VM READ appears, press Enter again.
3. If you have not added the `SET FULLSCREEN ON` command to your `PROFILE EXEC`, type `SET FULLSCREEN ON` and press the Enter key.

Note: You need to follow the previous procedure to get more storage only one time during a terminal session. The next time you log on, you have to use it again (if you need more storage).

It is easy to compile a program with z/VM. You use a command that calls the compiler, followed by the file name of your program. The format is:

```
compiler      filename
|             |
|             |> file name of your program
|             |
|> compiler command
```

To find what compiler command to use, see the IBM manual for the programming language you are using. The `FORTVS2` command calls the VS FORTRAN compiler, so enter:

```
fortvs2 degrees
```

A message from the compiler tells you if you made any errors while typing your program. To correct any errors, you:

- Edit the source program. For our example, you would enter:

```
x degrees fortran
```

- Correct the errors and enter a FILE command.
- Compile the program again. For our example, you would enter:

```
fortvs2 degrees
```

Before you try out the program, check the list of files that have the file name DEGREES. Enter:

```
filel degrees
```

Your list should contain these files:

```
DEGREES FORTRAN
DEGREES LISTING
DEGREES TEXT
```

DEGREES FORTRAN is the file you created. When your program is compiled, two new files are created. Both have the file name of your program. The file DEGREES LISTING contains your source program and a list of any errors that were found. You can look at this file by using an XEDIT command. You can also get a printed copy by using the PRINT command.

The file DEGREES TEXT contains the machine language version of your program. This file is loaded into storage when you run the program.

Both the LISTING and TEXT files are replaced if you change the program and compile it again.

Now you can press PF3 to leave the FILELIST environment.

Running a Program

Before you can run your program, you must tell z/VM the names of certain libraries that the system needs to run your program. You do that with the GLOBAL command. It has the following formats:

```
Global MACLIB library-name...
Global TXTLIB library-name...
Global LOADLIB library-name...
```

The exact GLOBAL command you need depends on the language you are using and your installation. Ask your z/VM administrator to tell you which GLOBAL command(s) to use.

Enter the global commands necessary to identify your libraries, substituting your system's library names for those given in this example:

```
global txtlib vsf2fort cmslib
global loadlib vsf2load
```

When you find out which GLOBAL commands to use, it is a good idea to include them in your PROFILE EXEC, which is discussed in [Chapter 14, “Creating a PROFILE EXEC,” on page 287](#). That way, you do not have to enter GLOBAL commands in each terminal session.

To run your program, you use a command that loads your program into storage and runs it. It has this format:

```
LOAD filename (START
```

For example, to load and run your program, enter:

```
load degrees (start
```

You see:

```
EXECUTION BEGINS...  
Enter FAHRENHEIT VALUE WITH DECIMAL POINT
```

Now enter a value with a decimal point. For example, enter:

```
32.0
```

You see:

```
0.0 DEGREES CELSIUS  
Ready; T=0.1/0.1 11:12:09
```

The ready message (Ready;) means that your program has completed.

Suppose you want to run the DEGREES program again. Do you have to compile it again? No, all you have to do is enter the LOAD command, because the machine language version of your program (the DEGREES TEXT file) is in permanent storage.

Note: Another way to run a program is to create a *module* using either the GENMOD (generate module) or the BIND command. You may choose this method when your program is free of errors, and you want to run it often or share it with other people. For information on creating a module, see [z/VM: CMS Application Development Guide](#) and [z/VM: Program Management Binder for CMS](#).

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., in the United States and/or other countries. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on [IBM Copyright and trademark information](http://www.ibm.com/legal/copytrade) (<https://www.ibm.com/legal/copytrade>).

Adobe is either a registered trademark or a trademark of Adobe Systems Incorporated in the United States, and/or other countries.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Terms and Conditions for Product Documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal Use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see:

- The section entitled **IBM Websites** at [IBM Privacy Statement](https://www.ibm.com/privacy) (<https://www.ibm.com/privacy>)
- [Cookies and Similar Technologies](https://www.ibm.com/privacy#Cookies_and_Similar_Technologies) (https://www.ibm.com/privacy#Cookies_and_Similar_Technologies)

Bibliography

This topic lists the publications in the z/VM library. For abstracts of the z/VM publications, see [z/VM: General Information](#).

Where to Get z/VM Information

The current z/VM product documentation is available in [IBM Documentation - z/VM \(https://www.ibm.com/docs/en/zvm\)](https://www.ibm.com/docs/en/zvm).

z/VM Base Library

Overview

- [z/VM: License Information](#), GI13-4377
- [z/VM: General Information](#), GC24-6286

Installation, Migration, and Service

- [z/VM: Installation Guide](#), GC24-6292
- [z/VM: Migration Guide](#), GC24-6294
- [z/VM: Service Guide](#), GC24-6325
- [z/VM: VMSES/E Introduction and Reference](#), GC24-6336

Planning and Administration

- [z/VM: CMS File Pool Planning, Administration, and Operation](#), SC24-6261
- [z/VM: CMS Planning and Administration](#), SC24-6264
- [z/VM: Connectivity](#), SC24-6267
- [z/VM: CP Planning and Administration](#), SC24-6271
- [z/VM: Getting Started with Linux on IBM Z](#), SC24-6287
- [z/VM: Group Control System](#), SC24-6289
- [z/VM: I/O Configuration](#), SC24-6291
- [z/VM: Running Guest Operating Systems](#), SC24-6321
- [z/VM: Saved Segments Planning and Administration](#), SC24-6322
- [z/VM: Secure Configuration Guide](#), SC24-6323

Customization and Tuning

- [z/VM: CP Exit Customization](#), SC24-6269
- [z/VM: Performance](#), SC24-6301

Operation and Use

- [z/VM: CMS Commands and Utilities Reference](#), SC24-6260
- [z/VM: CMS Primer](#), SC24-6265
- [z/VM: CMS User's Guide](#), SC24-6266
- [z/VM: CP Commands and Utilities Reference](#), SC24-6268

- [z/VM: System Operation](#), SC24-6326
- [z/VM: Virtual Machine Operation](#), SC24-6334
- [z/VM: XEDIT Commands and Macros Reference](#), SC24-6337
- [z/VM: XEDIT User's Guide](#), SC24-6338

Application Programming

- [z/VM: CMS Application Development Guide](#), SC24-6256
- [z/VM: CMS Application Development Guide for Assembler](#), SC24-6257
- [z/VM: CMS Application Multitasking](#), SC24-6258
- [z/VM: CMS Callable Services Reference](#), SC24-6259
- [z/VM: CMS Macros and Functions Reference](#), SC24-6262
- [z/VM: CMS Pipelines User's Guide and Reference](#), SC24-6252
- [z/VM: CP Programming Services](#), SC24-6272
- [z/VM: CPI Communications User's Guide](#), SC24-6273
- [z/VM: ESA/XC Principles of Operation](#), SC24-6285
- [z/VM: Language Environment User's Guide](#), SC24-6293
- [z/VM: OpenExtensions Advanced Application Programming Tools](#), SC24-6295
- [z/VM: OpenExtensions Callable Services Reference](#), SC24-6296
- [z/VM: OpenExtensions Commands Reference](#), SC24-6297
- [z/VM: OpenExtensions POSIX Conformance Document](#), GC24-6298
- [z/VM: OpenExtensions User's Guide](#), SC24-6299
- [z/VM: Program Management Binder for CMS](#), SC24-6304
- [z/VM: Reusable Server Kernel Programmer's Guide and Reference](#), SC24-6313
- [z/VM: REXX/VM Reference](#), SC24-6314
- [z/VM: REXX/VM User's Guide](#), SC24-6315
- [z/VM: Systems Management Application Programming](#), SC24-6327
- [z/VM: z/Architecture Extended Configuration \(z/XC\) Principles of Operation](#), SC27-4940

Diagnosis

- [z/VM: CMS and REXX/VM Messages and Codes](#), GC24-6255
- [z/VM: CP Messages and Codes](#), GC24-6270
- [z/VM: Diagnosis Guide](#), GC24-6280
- [z/VM: Dump Viewing Facility](#), GC24-6284
- [z/VM: Other Components Messages and Codes](#), GC24-6300
- [z/VM: VM Dump Tool](#), GC24-6335

z/VM Facilities and Features

Data Facility Storage Management Subsystem for z/VM

- [z/VM: DFSMS/VM Customization](#), SC24-6274
- [z/VM: DFSMS/VM Diagnosis Guide](#), GC24-6275
- [z/VM: DFSMS/VM Messages and Codes](#), GC24-6276
- [z/VM: DFSMS/VM Planning Guide](#), SC24-6277

- *z/VM: DFSMS/VM Removable Media Services*, SC24-6278
- *z/VM: DFSMS/VM Storage Administration*, SC24-6279

Directory Maintenance Facility for z/VM

- *z/VM: Directory Maintenance Facility Commands Reference*, SC24-6281
- *z/VM: Directory Maintenance Facility Messages*, GC24-6282
- *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*, SC24-6283

Open Systems Adapter

- Open Systems Adapter/Support Facility on the Hardware Management Console (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC14-7580-02.pdf), SC14-7580
- Open Systems Adapter-Express ICC 3215 Support (<https://www.ibm.com/docs/en/zos/2.3.0?topic=osa-icc-3215-support>), SA23-2247
- Open Systems Adapter Integrated Console Controller User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC27-9003-02.pdf), SC27-9003
- Open Systems Adapter-Express Customer's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/iaa2z1f0.pdf), SA22-7935

Performance Toolkit for z/VM

- *z/VM: Performance Toolkit Guide*, SC24-6302
- *z/VM: Performance Toolkit Reference*, SC24-6303

The following publications contain sections that provide information about z/VM Performance Data Pump, which is licensed with Performance Toolkit for z/VM.

- *z/VM: Performance*, SC24-6301. See *z/VM Performance Data Pump*.
- *z/VM: Other Components Messages and Codes*, GC24-6300. See *Data Pump Messages*.

RACF Security Server for z/VM

- *z/VM: RACF Security Server Auditor's Guide*, SC24-6305
- *z/VM: RACF Security Server Command Language Reference*, SC24-6306
- *z/VM: RACF Security Server Diagnosis Guide*, GC24-6307
- *z/VM: RACF Security Server General User's Guide*, SC24-6308
- *z/VM: RACF Security Server Macros and Interfaces*, SC24-6309
- *z/VM: RACF Security Server Messages and Codes*, GC24-6310
- *z/VM: RACF Security Server Security Administrator's Guide*, SC24-6311
- *z/VM: RACF Security Server System Programmer's Guide*, SC24-6312
- *z/VM: Security Server RACROUTE Macro Reference*, SC24-6324

Remote Spooling Communications Subsystem Networking for z/VM

- *z/VM: RSCS Networking Diagnosis*, GC24-6316
- *z/VM: RSCS Networking Exit Customization*, SC24-6317
- *z/VM: RSCS Networking Messages and Codes*, GC24-6318
- *z/VM: RSCS Networking Operation and Use*, SC24-6319
- *z/VM: RSCS Networking Planning and Configuration*, SC24-6320

TCP/IP for z/VM

- [*z/VM: TCP/IP Diagnosis Guide*](#), GC24-6328
- [*z/VM: TCP/IP LDAP Administration Guide*](#), SC24-6329
- [*z/VM: TCP/IP Messages and Codes*](#), GC24-6330
- [*z/VM: TCP/IP Planning and Customization*](#), SC24-6331
- [*z/VM: TCP/IP Programmer's Reference*](#), SC24-6332
- [*z/VM: TCP/IP User's Guide*](#), SC24-6333

Prerequisite Products

Device Support Facilities

- Device Support Facilities (ICKDSF): User's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/ickug00_v2r5.pdf), GC35-0033

Environmental Record Editing and Printing Program

- Environmental Record Editing and Printing Program (EREP): Reference (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/ifc2000_v2r5.pdf), GC35-0152
- Environmental Record Editing and Printing Program (EREP): User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/ifc1000_v2r5.pdf), GC35-0151

Related Products

XL C++ for z/VM

- [*XL C/C++ for z/VM: Runtime Library Reference*](#), SC09-7624
- [*XL C/C++ for z/VM: User's Guide*](#), SC09-7625

z/OS

IBM Documentation - z/OS (<https://www.ibm.com/docs/en/zos>)

Index

Special Characters

.BX (BOX) format word [226](#)
.CM (COMMENT) format word [228](#)
.CS (CONDITIONAL SECTION) format word [228](#)
.FO (FORMAT MODE) format word [228](#)
.IL (INDENT LINE) format word [229](#)
.IN (INDENT) format word [229](#)
.MT (MENU TYPE) format word [230](#)
.OF (OFFSET) format word [231](#)
.SP (SPACE LINES) format word [231](#)
.TR (TRANSLATE CHARACTER) format word [232](#)
" (logical escape character) used in setting PF (program function) keys [12](#)
@ (logical character delete symbol) [13](#)
* (asterisk)
 file ID on command line [35](#)
 file mode field [121](#)
(logical line end character), using to stack commands [11](#)
#CP function, using on display terminals [12](#)
%, used as file ID on command line [35](#)
= (equal sign) entered as equivalent output file IDs [36](#)
\$HLPxxxx reserved file type [112](#)

Numerics

190 minidisk address accessed with file mode S [118](#)
192 minidisk address accessed as file mode D [118](#)
19E minidisk address accessed with file mode Y [118](#)
3270 screen display, example [17](#)

A

A-disk [12](#), [118](#)
access
 directories [39](#), [40](#), [44](#), [126](#)
 examples of [135](#)
 file modes
 as read-only extensions [120](#)
 master file directories for minidisks [105](#)
 minidisks [104](#), [126](#)
 with VMLINK [127](#)
ACCESS command [44](#), [104](#)
adding help sections to HELP files [218](#)
address
 virtual, for unit record devices [163](#)
administrator authority [78](#)
ALIAList command
 PF keys [73](#)
alias
 creating
 definition [65](#)
 to another user's file [69](#)
 to your own file [67](#)
 locking [87](#)
 querying
 ALIAList [72](#)

alias (*continued*)
 querying (*continued*)
 ALIAS [71](#)
alter
 characteristics of spool files [165](#)
 existing HELP files [233](#)
AMSERV command [113](#), [176](#)
AMSERV reserved file type [109](#)
application in full-screen CMS [276](#)
ASM3705 reserved file type [109](#)
ASSEMBLE reserved file type [109](#)
assign file mode letters [118](#)
ATTACH command [170](#)
attaching a tape [170](#)
attention interrupt, causing [17](#)
attribute of directories [14](#)
AUTHLIST command [81](#)
authorizing users to access your files and directories [75](#)
AUTOSAVE reserved file type [109](#)
AUXxxxx reserved file type [110](#)

B

backspace a file on tape [173](#)
BASDATA reserved file type [110](#)
base file
 definition [65](#)
 erasing [74](#)
BASIC program language reserved file type [110](#)
BEGIN command [7](#)
beta test with VMLINK [145](#)
BFS (Byte File System)
 files [27](#)
 hierarchy [27](#)
block size of SFS files [42](#)
Border command
 moving a window [254](#)
 popping a window [254](#)
 scrolling forward and backward [251](#)
 scrolling right and left [253](#)
BRIEF HELP [188](#), [189](#)
bulletin board, creating [84](#)

C

C program language reserved file type [110](#)
causing breaks in text [231](#)
change
 characteristics of spool files [163](#)
 characteristics of unit record devices [163](#)
 existing HELP files [233](#)
 file mode numbers [126](#)
 output representation of a character [232](#)
 the HELP Facility [199](#)
CHANGE command [166](#)
cleanup functions for VSAM [281](#)
clock indicator in full-screen CMS [240](#)

- CMS (Conversational Monitor System)
 - considerations for migrating to full-screen CMS [273](#)
 - editor environment [9](#)
 - environment [8](#)
 - file system [27](#)
 - installation saved segment
 - discontinuing [293](#)
 - use of [288](#)
 - using IPL command to link [293](#)
 - PIPE [5](#)
 - subset environment [9](#)
- CMS command
 - general information [4](#)
 - used with execs written in REXX language [289](#)
- CMS/DOS (Conversational Monitor System/Disk Operating System) [10](#)
- CMSUT1 reserved file type [110](#)
- CNTRL reserved file type [110](#)
- COBOL reserved file type [110](#)
- command
 - abbreviations used in HELP [221](#)
 - entering [11](#)
 - environment
 - determining [7](#)
 - moving one to another [6](#)
 - execution characteristics [162](#)
 - file mode numbers, used to assign [126](#)
 - file types they use [109](#)
 - language
 - CMS [4](#)
 - CP [3](#)
 - line in full-screen CMS [239](#)
 - search order [161](#)
 - translation [160](#)
- command HELP file
 - name convention [215](#)
- comment in HELP text files [228](#)
- communicate user with z/VM [11](#)
- COMPARE command [34](#)
- compare two CMS files [34](#)
- compile a source program [300](#)
- compilers supported in CMS [4](#)
- component MENU in HELP [193](#)
- components of z/VM [3](#)
- conditionally displaying text [228](#)
- console
 - log
 - creating file from [15](#)
 - printing [15](#)
 - output spooling for display terminal [15](#)
- continuous spooling [164](#)
- control words, HELPCONV [225](#)
- conventions for notational HELP [219](#)
- convert
 - file to a HELP file [225](#)
 - fixed-length file to variable-length format [162](#)
 - variable-length file to fixed-length format [162](#)
- copy
 - contents of a display screen [16](#)
 - files, from one device to another [168](#)
 - files, to a directory [52](#)
 - files, with COPYFILE command [33](#)
 - from tape to minidisk or SFS directory [168](#)
 - spool files [164](#)

- COPY command
 - function on display terminals [16](#)
- COPY reserved file type [111](#)
- COPYFILE command
 - changing file mode numbers [122](#)
 - copying files to other users [166](#)
 - description [33](#)
 - used to change record formats of files [162](#)
- CP (Control Program)
 - command information [3](#)
 - commands used in REXX execs [284](#)
 - considerations for migrating to full-screen CMS [275](#)
 - environment, entering [7](#)
 - spooling examples [163](#)
- CP SPOOL command
 - consoles [15](#)
- CP TRACE command [19](#)
- create
 - aliases
 - to an alias [66](#)
 - to another user's file [69](#)
 - to files [65](#)
 - to your own file [67](#)
 - command HELP files [215](#), [216](#)
 - files [29](#), [53](#)
 - HELP message files [220](#)
 - HELP text files [232](#)
 - HELPMENU files [207](#)
 - HELPTASK files [212](#)
 - menus, HELP file [207](#)
 - one spool file from many files being printed or punched [164](#)
 - PROFILE EXEC [287](#)
 - programs [299](#)
 - REXX execs [283](#)
- CREATE FILE command [53](#)
- CREATE LOCK command [86](#)
- CSLCNTRL reserved file type [111](#)
- CSLLIB reserved file type [111](#)
- CSLSEG reserved file type [111](#)

D

- data area in full-screen CMS [239](#)
- data compression [107](#)
- DATASPACE command [97](#)
- DCSSMAP reserved file type [111](#)
- DDR command [176](#)
- default disk [127](#)
- default windows in full-screen CMS [241](#)
- DEFINE command [101](#)
- DEFINE STORAGE command [300](#)
- DELETE LOCK command [88](#)
- deleting HELP files [233](#)
- DETACH command, CP [172](#)
- detaching a tape [172](#)
- DETAIL HELP [189](#)
- DFSMS/VM
 - management class [64](#)
 - migrated status [42](#), [63](#), [65](#)
 - RECALL command [65](#)
- DIRCONTROL attribute [14](#)
- DIRECT reserved file type [111](#)
- directory

directory (*continued*)

- accessing another user's [44](#)
- accessing another user's in read/write mode [54](#)
- control attribute [14](#)
- copying files to [52](#)
- copying to directory from tape [168](#)
- creating [49](#)
- definition [37](#)
- determining if locked [89](#)
- determining ownership [83](#)
- directory control attribute [14](#)
- displaying status [39](#), [127](#)
- extensions [119](#)
- file control attribute of [14](#)
- hierarchical structure [42](#)
- identifier [45](#)
- level control [90](#)
- listing a subset [46](#)
- name [28](#)
- read-only extensions [119](#)
- referring to [43](#)
- releasing [41](#)
- search order [39](#), [118](#), [127](#)
- structure, viewing with DIRLIST [45](#)
- structure, viewing with LISTDIR [48](#)
- writing CMS files to [122](#)

Directory Control directory

- definition [38](#)
- DIRREAD authority for a directory [77](#)
- DIRREAD authority for a file [77](#)
- DIRWRITE authority for a file [78](#)
- erasing [58](#)
- functional restrictions [97](#)
- granting DIRREAD authority [38](#)
- granting DIRWRITE authority [38](#)
- high-performance example [98](#)
- kinds of authorities [75](#)
- relocating [55](#)
- uses [38](#)

DIRLIST command [45](#)

dirname [43](#)

DISCARD command [33](#)

discarding files, command environments [33](#)

discontinuing access to exec in saved segment [293](#)

DISK command [167](#)

display

- DASD information [101](#)
- disk information [39](#), [103](#)
- list of CMS files [29](#)
- users in your file pool [64](#)

display terminal

- entering commands [11](#)
- example of display screen [17](#)
- setting PF keys [12](#)
- signalling interruptions [17](#)

disposition of spool files [163](#)

DLBL command for assigning file mode numbers [126](#)

DLCS (Definition Language for Command Syntax)

- files [160](#)

DOSLIB reserved file type [111](#)

DOSLNK reserved file type [111](#)

drawing boxes [226](#)

duplicate file names or file types [28](#)

E

edit shared files [86](#)

ENROLL PUBLIC command [37](#)

enter

- CMS commands, in CMS subset environment [10](#)
- CMS/DOS environment [10](#)
- commands
 - from the screen [249](#)
 - on display terminal [11](#)
- CP commands
 - from CMS command environment [7](#)
 - from edit environment [10](#)
- CP environment from CMS environment [7](#)
- HELP Facility [4](#)
- Immediate commands [17](#)
- null lines [11](#)

enter key on your keyboard [11](#)

environment

- CMS editor [9](#)
- edit [10](#)

erase

- base files [74](#)
- directories [58](#)
- files to clear file space during editing session [42](#)
- HELP files [233](#)

ERASE command [33](#), [58](#)

ESA virtual machine [9](#)

ESA/390 architecture [9](#)

ESA/XC architecture [9](#)

ESERV program reserved file type [111](#)

ESM (external security manager) [80](#)

EXEC

- loading into saved segment [288](#)
- loading into storage [292](#)
- reserved file type [111](#)

EXEC 2

- comparison to EXEC [279](#)
- comparison to REXX [279](#)
- files
 - attributes [283](#)
 - format [283](#)
 - running [283](#)

EXECDROP command [292](#)

EXECIO command [290](#)

EXECLOAD command [292](#)

EXECMAP command [292](#)

EXECSTAT command [292](#)

execute

- exec procedures [283](#)
- Immediate commands in execs [289](#)
- PROFILE EXEC [287](#)

execution

- characteristics of CMS commands [162](#)
- state indicator in full-screen CMS [240](#)

exiting from execution of a program [18](#)

EXPAND reserved file type [111](#)

explicit lock [86](#), [89](#)

extensions, read-only, using [118](#)

F

file

- Byte File System (BFS) [27](#)

file (continued)

CMS

- creating files that are erased after being read [124](#), [125](#)
- format [27](#)
- identifiers [28](#)
- renaming [35](#)

control attribute [14](#)

converting to a HELP file [225](#)

determining the owner of a [83](#)

identifier

CMS, rules for assigning [28](#)

coded as equal sign (=) [36](#)

level control [90](#)

master file directories [105](#)

sharing

across file pools [70](#)

creating aliases to files [65](#)

how to [64](#)

system [27](#)

file control directory

definition [38](#)

erasing [58](#)

granting authority [38](#)

kinds of authorities [75](#)

file mode

displaying status [39](#), [104](#), [127](#)

extensions [119](#)

in file identifier [27](#)

letters

assigning [28](#)

when to specify for reading files [120](#)

when to specify for writing files [122](#)

numbers

changing [122](#)

commands used to assign [126](#)

default [122](#)

descriptions [117](#)

used to manipulate subsets of files [122](#)

when to specify [122](#)

read-only extensions [119](#)

search order [39](#), [104](#), [127](#)

file mode A

definition [12](#)

status [118](#)

storing files [118](#)

file mode determination

default for reading files

commands that search all accessed file modes [120](#)

commands that search only file mode A [120](#)

commands that search only file mode A and its

extensions [121](#)

default for writing files

commands for which you must specify file mode

[122](#)

commands that write output to file mode A [122](#)

commands that write output to read/write file mode

[122](#)

file mode letter change during a terminal session [28](#)

file mode number

in SFS

file mode number 0 [123](#)

file mode number 1 [124](#)

file mode number 2 [124](#)

file mode number (continued)

in SFS (continued)

file mode number 3 [124](#)

file mode number 4 [124](#)

file mode number 5 [124](#)

file mode number 6 [124](#)

file mode numbers 7 through 9 [124](#)

on minidisks

file mode number 0 [124](#)

file mode number 1 [125](#)

file mode number 2 [125](#)

file mode number 3 [125](#)

file mode number 4 [125](#)

file mode number 5 [125](#)

file mode number 6 [125](#)

file mode number 7 through 9 [126](#)

file name [27](#)

file pool

default [40](#), [99](#)

definition [37](#)

enrolling in [39](#)

IBM-supplied (VMSYSU) [39](#)

primary [100](#)

using more than one [99](#)

file pool identifier [39](#), [43](#)

file space [41](#)

file type

created by assembler and language processors [109](#)

for documentation [117](#)

for reports [117](#)

in file identifier [27](#)

reserved for language processors [109](#)

temporary work files [117](#)

used by CMS commands [109](#)

file types, reserved, CMS [109](#)

FILECONTROL attribute [14](#)

FILEDEF command [126](#)

FILELIST command [30](#)

FORMAT command [102](#)

format words

.BX (BOX) [226](#)

.CM (COMMENT) [228](#)

.CS (CONDITIONAL SECTION) [228](#)

.FO (FORMAT MODE) [228](#)

.IL (INDENT LINE) [229](#)

.IN (INDENT) [229](#)

.MT (MENU TYPE) [230](#)

.OF (OFFSET) [231](#)

.SP (SPACE LINES) [231](#)

.TR (TRANSLATE CHARACTER) [232](#)

format-mode, processing [233](#)

formatting CMS minidisks, example of [102](#)

FORTRAN program

compiling [300](#)

creating [299](#)

reserved file types [112](#)

forward space file on tape [173](#)

FREEFORT reserved file type [112](#)

full-screen CMS

border commands [250](#)

changing window borders [266](#)

CMSPF keys

definition [238](#)

display [244](#)

- full-screen CMS (*continued*)
 - CMSPF keys (*continued*)
 - list of [243](#)
 - default windows [241](#)
 - deleting a blank reserved window line [267](#)
 - deleting a window title [267](#)
 - entering commands from the screen [249](#)
 - location information [241](#)
 - maximizing windows [263](#)
 - message routing [272](#)
 - messages [247](#)
 - PA and PF keys
 - Backward [243](#)
 - Clear Top [243](#)
 - Cursor [243](#)
 - Filelist [243](#)
 - Forward [243](#)
 - Help [243](#)
 - Left [243](#)
 - PA1 key [246](#)
 - Pop Msg [243](#)
 - Quit [243](#)
 - Rdrlist [243](#)
 - Retrieve [243](#)
 - Right [243](#)
 - physical screen characteristics
 - command line [239](#)
 - data area [239](#)
 - location information [239](#)
 - PF key definition area [239](#)
 - status area [239](#)
 - title line [239](#)
 - positioning windows [262](#)
 - restoring windows [265](#)
 - setting the reserved line [266](#)
 - setting window borders [265](#)
 - sizing windows [263](#)
 - status information [240](#)
 - system-defined windows and virtual screens [241](#)
 - using [238](#)
 - virtual screens [241](#)

G

- GCS (Group Control System)
 - reserved file type [112](#)
- GENMOD command [302](#)
- GENMSG command [113](#)
- GLOBAL command [301](#)
- GLOBALV reserved file type [112](#)
- granting authority [78](#)
- GROUP reserved file type [112](#)

H

- halt
 - program execution [18](#)
 - REXX execs [289](#)
 - terminal displays [19](#)
- header card [167](#)
- HELP command
 - layers of HELP
 - BRIEF [189](#)

- HELP command (*continued*)
 - layers of HELP (*continued*)
 - DETAIL [189](#)
 - overview [188](#)
 - RELATED [190](#)
 - menus
 - component [193](#)
 - task [192](#)
 - message [188](#)
 - notational convention [219](#)
 - summary of format words [226](#)
- HELP facility
 - components [187](#)
 - using [187](#)
- HELP Facility
 - components [224](#)
 - definition of components [199](#)
 - file names, special character [215](#)
 - keys, PF and PA2 [195](#)
 - list of components [202](#)
 - list of file types [202](#)
 - purpose of components [199](#)
 - special character file names [215](#)
 - tailoring [199](#)
 - using XEDIT [196](#)
- HELP file
 - name convention for command files [215](#)
 - name convention for message files [220](#)
- HELPCONV command
 - definition [225](#)
 - how it treats comments [228](#)
 - summary of control words [226](#)
 - using to create additional HELP files [225](#)
- HELPMENU files, creating [207](#)
- HELPTASK files, creating [212](#)
- HELPxxxx reserved file type [112](#)
- highlighting words in a file using HELP [220](#)
- holding spool files to keep them from being processed [164](#)
- HT (Halt Type) Immediate command [17](#)
- HX (Halt Execution), effect in CMS subset of [10](#)

I

- IBM-supplied VMSYSU file pool [39](#)
- IDRC (Improved Data Recording Capability) [178](#)
- IEBTPCH utility program
 - creating CMS files from tapes [183](#)
 - MVS utility [174](#)
- IEBUPDTE utility program
 - creating CMS files from tapes [184](#)
 - MVS utility [174](#)
- IEHMOVE utility program
 - creating CMS files from tapes [184](#)
 - MVS utility [174](#)
- Immediate command
 - entering on display terminal [17](#)
 - HX (Halt Execution) [10](#)
 - using with REXX programs [289](#)
- IMPCP operand of CMS SET command, setting [8](#)
- IMPEX, SET command [283](#)
- implicit locks [86](#), [89](#)
- implied
 - CP function, SET IMPCP, usage [8](#)
 - Exec function, SET IMPEX, usage [283](#)

- indenting text [229](#)
- INPLACE file attribute [124](#), [125](#)
- introducing new products with VMLINK [145](#)
- invoke
 - programs [301](#)
 - REXX execs [283](#)
- IPL (Initial Program Load)
 - entering CMS from CP [8](#)
 - linking CMS installation saved segment [293](#)

K

- keywords, translations of [160](#)

L

- LANGGCTL reserved file type [112](#)
- LANGLIST reserved file type [112](#)
- LANGMAP reserved file type [113](#)
- LANGMCTL reserved file type [113](#)
- language processors, types reserved for use by [109](#)
- language statements
 - in EXEC 2 language [280](#)
 - in REXX language, for REXX/VM interpreter [279](#)
- LASTING lock [87](#)
- leave
 - CMS subset environment [10](#)
 - CMS/DOS environment [11](#)
- length of CMS ready message, changing [13](#)
- libraries needed for programs [301](#)
- link
 - CMS installation saved segment [293](#)
 - examples of [135](#)
 - with VMLINK [127](#)
- LINK command [103](#)
- linking CMS installation saved segment via IPL [293](#)
- list
 - files using asterisk or percent signs [35](#)
 - files with the same character string [35](#)
 - information about CMS files [29](#)
- LIST3800 reserved file type [113](#)
- LIST3820 reserved file type [113](#)
- LIST38PP reserved file type [113](#)
- LISTCPDS reserved file type [113](#)
- LISTDIR command [48](#)
- LISTFILE command [30](#)
- LISTING reserved file type [113](#)
- LKEDIT reserved file type [113](#)
- LOAD command [301](#)
- loading
 - execs into storage [292](#)
 - files from a tape [172](#), [173](#)
- LOADLIB reserved file type [113](#)
- local IDs [100](#)
- location information in full-screen CMS [239](#), [241](#)
- lock
 - deleting, on files and directories [88](#)
 - exclusive locks [86](#)
 - lasting [87](#)
 - session [87](#)
 - share locks [86](#)
 - terminal keyboard [11](#)
 - update locks [86](#)

- locking files (SFS)
 - explicit lock [86](#)
 - implicit locks [86](#)
- log
 - messages [250](#)
 - warnings [250](#)
- LSEG reserved file type [113](#)
- LSEGMAP reserved file type [113](#)

M

- MACLIB reserved file type [114](#)
- MACRO reserved file type [114](#)
- manage
 - your file space [41](#)
 - your minidisks [106](#)
- MAP reserved file type [114](#)
- master file directory [105](#)
- MDMPxxxx reserved file type [114](#)
- MEMO reserved file type [114](#)
- menu
 - changing [233](#)
 - component [192](#), [193](#)
 - creating [207](#)
 - example, of creation [211](#)
 - task [192](#)
 - TASK [212](#)
- message
 - class indicator in full-screen CMS [240](#)
 - controlling whether you receive them [3](#)
 - HELP [188](#)
 - in full-screen CMS [247](#)
 - logging [250](#)
 - not logging [250](#)
 - routing, default settings [272](#), [273](#)
- message examples, notation used in [23](#)
- message HELP file
 - name convention [220](#)
- MESSAGE LOGFILE [250](#)
- MESSAGE window
 - changing [256](#)
 - displaying [255](#)
 - dropping [247](#), [255](#)
 - popping [247](#), [250](#), [254](#)
 - restoring [257](#)
 - using to edit names file [248](#)
 - viewing messages [247](#)
- migrated status of a file [42](#), [65](#)
- minidisk
 - accessing [104](#)
 - copying to minidisk from tape [168](#)
 - defined in z/VM directory entry [101](#)
 - defining temporary minidisks for terminal session [101](#)
 - defining virtual disks in storage [102](#)
 - definition [101](#)
 - displaying status [104](#), [127](#)
 - extensions [119](#)
 - how much space is used [106](#)
 - linking [103](#)
 - managing [106](#)
 - master file directory [105](#)
 - query status of [106](#)
 - read-only extensions [119](#)
 - releasing [105](#)

- minidisk (*continued*)
 - search order [104, 118, 127](#)
 - sharing [103](#)
 - writing CMS files to [122](#)
- mode, setting with CP TERMINAL command [17](#)
- MODULE reserved file type [114](#)
- modules, generating [302](#)
- MOVEFILE command [174](#)

N

- name
 - CMS files [28](#)
 - user commands [161](#)
- name convention for command HELP files [215](#)
- name convention for message HELP files [220](#)
- NAMES file
 - editing with MESSAGE window [248](#)
 - message window [248](#)
- NAMES file tags, VMLINK [128](#)
- NAMES reserved file type [114](#)
- NETLOG reserved file type [114](#)
- NEWREAD authority [77](#)
- NEWWRITE authority [77](#)
- nonshareable tape device [170](#)
- notation used in message and response examples [23](#)
- notational convention for HELP [219](#)
- NOTE command [167](#)
- NOTE reserved file type [114](#)
- NOTEBOOK reserved file type [114](#)
- null
 - entering to determine environment [7](#)
 - input data from terminal [11](#)
 - lines [11](#)
 - to resume program execution after attention interruption [19](#)

O

- offsetting text [231](#)
- ORDER command [166](#)
- OS (Operating System)
 - cleanup [281](#)
 - utility programs to create CMS files from tapes created by [183](#)
- output from virtual console, spooling [15](#)

P

- PA1 key in full-screen CMS [246](#)
- PA2 key in HELP [195](#)
- parent, of read-only extension [119](#)
- PASCAL program, file type usage in CMS [114](#)
- password
 - supplying on LINK command line [103](#)
 - suppressing on command line [103](#)
- permanent
 - links to minidisks [126](#)
 - minidisk [101](#)
- PF keys on a DIRLIST display [47](#)
- PIPE command [289](#)
- Pipelines, CMS [5](#)
- PL/I program language, file type usage in CMS [114](#)

- preface [xvii](#)
- print
 - CMS files [117](#)
 - HELP files [197](#)
 - multiple copies [164](#)
- printer file
 - querying status of [166](#)
 - spooling [165](#)
- PROC reserved file type [114](#)
- product tapes [174](#)
- PROFILE EXEC
 - creating [287](#)
 - description [287](#)
 - sample, using REXX language [287](#)
 - suppressing execution [287](#)
 - use to make EXECs storage-resident [288](#)
- program function (PF) keys
 - ? in HELP [195](#)
 - Backward in HELP [195](#)
 - Brief in HELP [194, 196](#)
 - Clocate in HELP [195](#)
 - Cursor in HELP [196](#)
 - definition area in full-screen CMS [239](#)
 - Forward in HELP [195](#)
 - full-screen CMS
 - Backward [243](#)
 - Clear Top [243](#)
 - Cursor [243](#)
 - Filelist [243](#)
 - Forward [243](#)
 - Help [243](#)
 - Left [243](#)
 - Pop Msg [243](#)
 - Quit [243](#)
 - Rdrlist [243](#)
 - Right [243](#)
 - HELP [195](#)
 - Morehelp in HELP [196](#)
 - PF Keys in HELP [195](#)
 - Print in HELP [195](#)
 - Quit in HELP [195](#)
 - Related in HELP [196](#)
 - Return in HELP [195](#)
 - setting
 - CMSPF keys [245](#)
 - commands, commonly used [12](#)
 - COPY function [16](#)
 - how to [12](#)
 - in PROFILE EXEC [287](#)
 - toggling between keys [194](#)
 - Top in HELP [195](#)
 - using in FILELIST [32](#)
- WM environment
 - Backward [246](#)
 - Clear [246](#)
 - Copy [246](#)
 - Forward [246](#)
 - Help [246](#)
 - Left [246](#)
 - Maximize [246](#)
 - Quit [246](#)
 - Restore [246](#)
 - Retrieve [246](#)
 - Right [246](#)

program function (PF) keys (*continued*)

WM environment (*continued*)

[Top 246](#)

programs

C [299](#)

COBOL [299](#)

FORTRAN [299](#)

PL/I [299](#)

written in REXX language for REXX/VM interpreter [279](#)

protect

application environment [11](#)

files from being accessed [124](#)

PSEG reserved file type [115](#)

PSEGMAP reserved file type [115](#)

PUNCH command [167](#)

PURGE command [166](#)

Q

QUERY command

ACCESSED command [39](#), [44](#), [104](#), [127](#)

ACCESSORS command [97](#)

ALIAS command [71](#)

BLOCKS command [42](#)

command (CMS)

CP query for status of CP SET MSG function [3](#)

display search order [119](#)

how much space is on a minidisk [106](#)

proportion of file space used [41](#)

query CMSPF keys [243](#)

command uses [296](#)

CPLANG command [5](#)

DASD command [101](#), [103](#)

description [165](#)

DISK command [103](#)

ENROLL command [44](#), [64](#)

FILEPOOL CONFLICT command [90](#)

FILEPOOL PRIMARY command [99](#)

IMPEX command [283](#)

LANGLIST command [5](#)

LANGUAGE command [5](#)

LIMITS command [41](#)

LINKS command [105](#)

LOCK command [89](#)

R

RDRLIST command [166](#)

RDTAPE macro [175](#)

read

cards from your virtual card reader [167](#)

real card decks into your virtual machine [167](#)

read authority

directory [76](#)

file [76](#)

read-only extension

using [120](#)

read, to virtual console, definition [17](#)

read/write, displaying status (of file modes) [39](#), [104](#), [127](#)

READCARD command [165](#)

reader, holding user files in [164](#)

ready message

controlling how it is displayed [13](#)

ready message (*continued*)

not displayed after #CP function is used in CMS [9](#)

real card (CP ID card) [167](#)

RECEIVE command [166](#)

RECFM [162](#)

record length of CMS file [162](#)

RELATED HELP [190](#)

RELEASE command

description [41](#)

file modes [105](#)

minidisks [105](#)

read-only extensions [120](#)

updating master file directory [105](#)

relocating files and directories [54](#)

RENAME command

changing file mode numbers [122](#), [126](#)

description [35](#)

renaming files in another user's directory [54](#)

using aliases [83](#)

reserved file type

CMS [109](#)

REPOS [115](#)

response examples, notation used in [23](#)

responses from CMS commands [13](#)

resume

after an attention interruption [18](#)

terminal displays [19](#)

return CMS subset command to leave subset [10](#)

return code in CMS ready message [13](#)

return key on your keyboard [11](#)

REVOKE AUTHORITY command [79](#)

rewind a tape [171](#)

REXX file type Pipeline

usage in CMS Pipelines [115](#)

REXX language

commands used with execs [289](#)

creating execs [283](#)

features for REXX/VM interpreter [279](#)

running execs [283](#)

sample execs [284](#)

REXX/VM interpreter

basic description [279](#)

relationship with CMS EXEC and EXEC 2 [280](#)

REXX language, interpreted by [279](#)

writing execs for the [283](#)

RPGII program, file type usage in CMS [115](#)

RT (Resume Type) Immediate command [17](#)

RTABLE reserved file type [115](#)

running a program [301](#)

S

saved segment [288](#), [292](#)

scan a tape [172](#)

screen example of 3270 display [17](#)

SCRIPT reserved file type [115](#), [117](#)

scroll

forward and backward using border commands [251](#)

right and left using border commands [253](#)

search order

file mode extensions [119](#)

for CMS commands

summary [161](#)

for file modes [118](#)

- searching read-only extensions [119](#)
- send
 - files to other virtual machine users
 - SENDFILE command [167](#)
 - using DISK DUMP command [167](#)
 - messages [3](#), [251](#)
- SENDFILE command [167](#)
- service tapes [174](#)
- session lock [87](#)
- SESSION lock [87](#)
- set of program function (PF) keys [12](#)
- SFS (Shared File System)
 - administrator [78](#)
 - AUTHLIST command from FILELIST SHARE screen [81](#)
 - AUTHLIST display, PF keys [82](#)
 - authorizations, dynamic [77](#)
 - definition [37](#)
 - directories, sample of [38](#), [42](#)
 - directory [37](#)
 - dynamic authorizations [77](#)
 - file organization [42](#)
 - file pool [37](#)
 - file space [37](#)
 - file, determining the owner of a [83](#)
 - FILELIST SHARE screen, revoking authority [80](#)
 - NEWREAD authority [77](#)
 - NEWWRITE authority [77](#)
 - revoked authority on the FILELIST SHARE screen [80](#)
 - top directory [39](#)
- share
 - files, how-to [64](#)
 - minidisks [103](#)
- shareable tape device [170](#)
- shared disk, creating [84](#)
- shared files, editing [86](#)
- simulated data set format
 - file mode number of 4 [124](#), [125](#)
 - specifying [124](#), [125](#)
- sort
 - CMS files [14](#)
 - files in FILELIST [32](#)
- SORT command to specify file mode numbers [36](#)
- source program [300](#)
- spacing between lines of text [231](#)
- special characters
 - 3270 Text feature [20](#)
 - in file names and file types [28](#)
- specifying file mode numbers on DLBL and FILEDEF commands [126](#)
- SPOOL command
 - description [163](#)
 - spooling console output [15](#)
- stacking null lines
 - after attention interruption [19](#)
 - at your terminal [11](#)
- stage, in pipelines [5](#)
- starting programs [301](#)
- status area in full-screen CMS [239](#)
- status information in full-screen CMS [240](#)
- status message [7](#)
- storage group definition [14](#)
- storing files
 - allocation of DASD space for minidisks [14](#)
 - allocation of file space in SFS directories [14](#)
 - storing files (*continued*)
 - in SFS directories [14](#)
 - methods of [4](#)
 - on both SFS directories and minidisks [40](#)
 - on minidisks [14](#)
- subdirectories [37](#)
- subsetting options in DETAIL HELP
 - ALL [189](#)
 - DESCRIPT [189](#)
 - ERRORS [189](#)
 - FORMAT [189](#)
 - NOTES [189](#)
 - OPTIONS [189](#)
 - PARMS [189](#)
- suppressing passwords on the command line [103](#)
- SYMDMP reserved file type [115](#)
- SYNONYM
 - command to load synonym tables [159](#)
 - file type [115](#)
 - used to define synonyms for CMS and user-written commands [159](#)
 - used to sort FILELIST [32](#)
- syntax diagrams, how to read [20](#)
- system disk, files available [125](#)
- SYSTUT6 reserved file type [115](#)
- SYSUT1 reserved file type [115](#)
- SYSUT2 reserved file type [115](#)
- SYSUT3 reserved file type [115](#)
- SYSUT4 reserved file type [115](#)
- SYSUT5 reserved file type [115](#)

T

- T-disk [101](#)
- tags, VMLINK NAMES Files [128](#)
- tape
 - accessing through user programs [175](#)
 - ATTACH command [170](#)
 - compaction [176](#), [178](#)
 - copying to/from cards [174](#)
 - device
 - address [169](#)
 - name (TAPn) [169](#)
 - number [169](#)
 - Dual Density Feature [179](#)
 - enciphered data [178](#)
 - error correction [176](#)
 - EXPORT (VSE/VSAM) [176](#)
 - IDRC feature [180](#)
 - IMPORT (VSE/VSAM) [176](#)
 - loading
 - IEBTPCH tapes [174](#)
 - IEBUPDTE tapes [174](#)
 - IEHMOVE tapes [174](#)
 - MOVEFILE command [174](#)
 - MVS, interchange with [175](#)
 - native tape macros [175](#)
 - nonshareable attribute [170](#)
 - number of track [176](#)
 - OS simulation [174](#), [175](#), [182](#)
 - RDTAPE macro [175](#)
 - recording formats
 - 3480 Basic [177](#)
 - 3480 Compacted [177](#)

- tape (*continued*)
 - recording formats (*continued*)
 - 3490 Basic [177](#)
 - 3490 Compacted [178](#)
 - 3590 Basic [178](#)
 - 3590 Compacted [178](#)
 - command options [180](#)
 - compaction [178](#)
 - defaults [180](#)
 - density [176](#)
 - device capabilities [179](#)
 - format [176](#)
 - GCR [177](#)
 - list of [176](#)
 - NRZI [177](#)
 - PE [177](#)
 - QIC [178](#)
 - selecting [180](#)
 - REPRO (VSE/VSAM) [176](#)
 - shareable attribute [170](#)
 - Tape Library Dataserver [182](#)
 - tape marks [181](#)
 - TAPECTL macro [175](#)
 - TAPEMAC command [174](#), [184](#)
 - TAPESL macro [175](#)
 - TAPPDS command [174](#)
 - using with CMS [169](#)
 - virtual/real devices [170](#)
 - VMFPLC2 command [173](#)
 - VSE/VSAM
 - AMSERV [176](#)
 - EXPORT [176](#)
 - IMPORT [176](#)
 - REPRO [176](#)
 - VSE simulation [176](#)
 - WRTAPE macro [175](#)
- TAPE command [170](#)
- Tape Library Dataserver, using [182](#)
- TAPPDS command
 - copying files from tapes [183](#)
- TASK menus [192](#), [212](#)
- TE (Trace End) immediate command [289](#)
- TEMPLATE reserved file type [115](#)
- temporary minidisk [101](#)
- terminal mode, setting [17](#)
- terminate program [18](#)
- TESTCOB reserved file type [115](#)
- TESTFORT reserved file type [115](#)
- TEXT
 - file type usage in CMS [116](#), [301](#)
- text feature for 3270 terminals [20](#)
- title line in full-screen CMS [239](#)
- toggling between PF keys [194](#)
- top directory
 - abbreviated form [40](#)
 - definition [39](#)
 - name [39](#)
 - referring to [43](#)
- TRACE Command for CP [19](#)
- trademarks [304](#)
- TRANSFER command [166](#)
- transient area, CMS commands that process in [162](#)
- translating output characters [232](#)
- translation synonyms for commands [160](#)

- truncate trailing blanks [162](#)
- TS (Trace Start) immediate command [289](#)
- TXTLangid reserved file type [116](#)
- TXTLIB reserved file type [116](#)
- TYPE command [88](#), [119](#)

U

- underscore in file name and file type [28](#)
- unit records devices [163](#)
- UPDATE reserved file type [116](#)
- UPDATE SESSION lock [88](#)
- update-in-place [124–126](#)
- updating master file directories [105](#)
- UPDLOG reserved file type [116](#)
- UPDTxxxx reserved file type [116](#)
- user
 - file directory [105](#)
 - hold status of spool files [164](#)
 - program area, commands that process in [162](#)
- user ID, specifying for output spool files [165](#)
- user tags, VMLINK [130](#)
- user-written
 - commands [161](#)
 - execs, samples [284](#)
- using XEDIT subcommand in HELP [196](#)

V

- virtual addresses for unit record devices [163](#)
- virtual disk [101](#), [102](#)
- virtual disk in storage [101](#), [102](#)
- virtual screen
 - default characteristics [271](#)
 - defining [259](#)
 - full-screen CMS [241](#)
 - general description [237](#)
 - system-defined [241](#)
- virtual storage
 - amount of [300](#)
- virtual tape device [170](#)
- VMFPLC2 command [174](#)
- VMLINK command [127](#)
- VMLINK CONTROL record definitions [130](#), [133](#)
- VMLNICXT EXEC [145](#)
- VMSYSU file pool [39](#)
- VSAM cleanup [281](#)
- VS BASIC program language, file type usage in CMS [116](#)
- VSBDATA reserved file type [116](#)
- VSCREEN command [260](#)
- VSCREEN WAITREAD command [296](#)
- VSE
 - simulation [176](#)
 - using tapes with [176](#)
- VSE/VSAM (AMSERV Command) [176](#)

W

- WARNING LOGFILE [250](#)
- window
 - changing borders [266](#)
 - characteristics of [238](#)
 - default characteristics [269](#)

window (*continued*)

- default, in full-screen CMS [241](#)
- defining [260](#)
- deleting a blank reserved line [267](#)
- deleting title [267](#)
- dropping [247](#)
- general description [237](#)
- maximizing [263](#)
- moving [254](#)
- popping [247](#), [254](#)
- positioning [262](#)
- restoring [265](#)
- setting borders [265](#)
- sizing [263](#)
- system-defined [241](#)

WINDOW command

- DEFINE [260](#)
- DROP [247](#)
- MAXIMIZE [263](#)
- POP [247](#)
- POSITION [262](#)
- RESTORE [265](#)
- SIZE [263](#)

WM window

- customizing [259](#)
- messages [241](#)
- PF keys
 - Backward [246](#)
 - Clear [246](#)
 - Copy [246](#)
 - Forward [246](#)
 - Help [246](#)
 - Left [246](#)
 - Maximize [246](#)
 - Quit [246](#)
 - Restore [246](#)
 - Retrieve [246](#)
 - Right [246](#)
 - Top [246](#)
- using [241](#), [254](#)

workfile [117](#)

write

- a tape mark [171](#)
- applications in full-screen CMS [276](#)
- authority
 - for a directory [76](#)
 - on a file [76](#)
- CMS files with file mode specified [122](#)
- labels on CMS minidisks [102](#)

WRTAPE macro [175](#)

X

XA virtual machine [9](#)

XC virtual machine

- ESA/XC [9](#)
- z/XC [9](#)

XEDIT

- changing file mode number with [122](#)
- CLOCATE subcommand [196](#)
- considerations for migrating to full-screen CMS [275](#)
- subcommands, invoking [4](#)
- using to display HELP files [196](#)

XEDIT reserved file type [116](#)

Z

z/Architecture CMS [9](#)

z/CMS [9](#)

z/XC architecture [9](#)

ZAP reserved file type [116](#)



Product Number: 5741-A09

Printed in USA

SC24-6266-73

