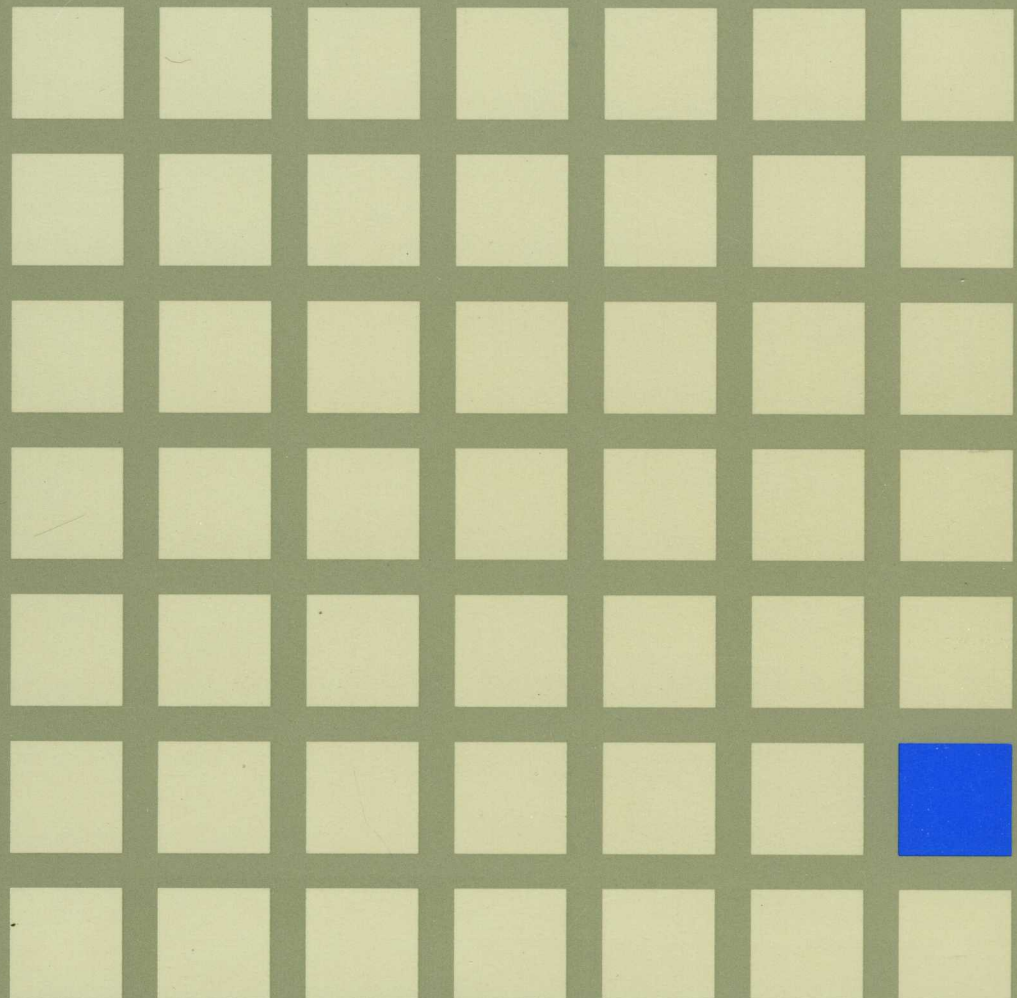




Introduction

Release 6





Virtual Machine/System Product

GC19-6200-05

Introduction

Release 6

Sixth Edition (August 1988)

This edition, GC19-6200-05, is a major revision of GC19-6200-04, and applies to Release 6 of the IBM Virtual Machine/System Product (VM/SP), program number 5664-167, and to all the next releases and modifications until otherwise indicated in new editions or Technical Newsletters. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, 4300, and 9370 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

Because readers are not expected to be familiar with past editions of the book, revision bars are not used to mark new or changed information.

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

Ordering Publications

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

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

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

© Copyright International Business Machines Corporation 1980, 1982, 1983, 1984, 1986, 1988. All rights reserved.

Preface

About This Book

This book describes, at an introductory level, what Virtual Machine/System Product (VM/SP) is and what it does for its users. It is meant for anyone who wants basic information about VM/SP and its capabilities. To learn how to use VM/SP, you should begin with the *VM/SP CMS Primer* as a guide.

If you currently have the VM/SP library, refer to the *Release 6 Guide* for new information.

Who Should Read This Book

You might use the information in this book if you:

- Plan to install VM/SP and want to get acquainted
- Know the capabilities of VM/SP, but would like to know more about VM/SP
- Need background knowledge for using licensed programs with VM/SP
- Migrate to VM/SP from another type of operating system
- Need basic knowledge about VM/SP for any reason.

None of the information in this book is meant to be used as a guide or reference for using VM/SP. Each topic discussed introduces VM/SP and gives references to more in-depth information.

What You Should Know Before Reading this Book

Before you use this book, you should know:

- Data processing basics
- Operating system fundamentals
- Data processing terminology.

This book assumes that you know the meanings of words such as *processor*, *system*, and *storage*. However, concepts such as *paging*, *spooling*, and *networking* may be new to you.

Some sources for background information are:

- *VM/SP CMS Primer*, SC24-5236
- *Dictionary of Computing*, SC20-1699.

What This Book Contains

The first page of each chapter has a brief overview of the contents of each chapter. A glossary, bibliography, and index are included at the back of the book.

You will probably find it best to read the entire book from start to finish. Most topics assume you are familiar with the ideas and terms discussed in earlier topics.

Because the information in this book reflects a system that helps people at many levels do many kinds of work, some readers elect to skip some of the topics. For example, if you are a general user, not involved in system planning, generation, administration, or maintenance, the first two chapters of this book will satisfy your needs. However, if your job relates to the system itself, rather than with just its use, you will find the entire book useful.

How to Find More Information

For many people the most difficult thing about learning a computer system, especially a large one, is simply finding the necessary information. Each topic in this book includes references that tell you where to find more information about that topic. The bibliography of this book helps you find information in other books of the VM/SP library.

Many discussions in this book include references, either direct or indirect, to specific commands related with that topic. The command names make good reference words for locating additional information in other books.

The *VM/SP General Information* book, GC20-1838, is a broader, less technical introduction to VM/SP. The *VM/SP Library Guide and Master Index*, GC19-6207, is the complete guide to finding information in the VM/SP library.

The *VM/SP Planning Guide and Reference*, SC19-6201, provides information on planning, installing, and updating a VM/SP system. The *VM/SP Release 6 Guide*, SC24-5368, describes the functional changes to VM/SP for Release 6.

Contents

Chapter 1. VM/SP Basics	1
A Short Introduction to VM/SP	2
CP Is the Resource Manager	2
CMS Manages Work Flow	3
GCS Manages Subsystems	3
AVS Enhances APPC/VM Communications	4
TSAF Enhances Virtual Machine Communication	4
IPCS Handles Software Problems and Reports	5
Licensed Programs Extend the Capabilities of VM/SP	5
Virtual Concept	6
How Virtual Storage Differs from Real Storage	7
How a Virtual Console Differs from a Real Console	9
How VM/SP Shares Processor Usage	9
What Is a Virtual Machine?	10
Requirements for Using a Virtual Machine	10
Operating Systems Run in Virtual Machines	11
Input/Output Devices	11
Other Virtual Machine Facilities	12
Creating Virtual Machines and Managing Real Resource Usage	13
System Directory Entries Define Virtual Machines	13
CP Divides Real Processor Time Among Virtual Machines	14
How CP Determines Virtual Storage Size	15
Paging Lets Many Users Share Processor Storage	16
How Virtual Storage and Real Storage Are Organized	16
Virtual Direct Access Storage: Minidisks and File Pools	18
How CP Defines Minidisks	18
The Structure of a CMS File Pool	19
Spooling	21
How CP Manages Spooling	21
How to Get Virtual Console Data through Spooling	25
Spooling Summary	25
VM/SP Is a Conversational System	27
How a General User Communicates with VM/SP	27
System Messages Communicate with Users	28
How the System Operator Communicates with VM/SP	28
A Summary of VM/SP Concepts	30
Chapter 2. Putting VM/SP to Work	31
Working in the Virtual Environment	32
Using CP Commands	32
Communicating with Other Virtual Machines	33
Communicating with the System Operator	34
Virtual Machine I/O	34
Developing Application Programs Under CMS	37
Steps in the Application Program Development Process	37
Using the System Product Editor for Program Development	37
Using the CMS Programming Interface	38
Using CMS to Assemble Programs	38
Using CMS to Compile High-level Language Programs	39
Customer Information Control System/VM	40
Debugging Application Programs	43
Checking a New Program	43

CMS Facilities for Debugging Programs	43
CP Facilities for Debugging Programs	44
Handling Problems with VM/SP Interactive Problem Control System	45
How IPCS Works	45
Running Programs under CMS	48
Preparing for Program Execution	48
Processing Programs with the RUN Command	50
Callable Services Library Routines	50
Processing Jobs in Batch Mode under CMS	52
What Is the CMS Batch Facility?	52
Using the CMS Batch Facility	52
What Is the VM Batch Subsystem?	53
Using the VM Batch Subsystem	54
Running Operating Systems in a Virtual Machine	55
Other Programs That Run Under VM/SP	57
Chapter 3. A Closer Look at CP	59
Communicating with CP	60
User Classes Determine System Use Privileges	60
System Directory Entries Help CP Produce Virtual Machines	61
Virtual Console Management	61
Defining and Changing Virtual Machine Configuration and Status	63
System Directory Entries Define Virtual Machines	63
CP Commands Help Manage the Virtual Machine	63
Inter-User Communications Vehicle	66
Basics about IUCV	66
Communicating Between Virtual Machines	67
Communicating Between a Virtual Machine and a CP System Service	67
Hardware Devices That Handle I/O	69
Defining a System Device Configuration	69
Planning Device Configurations	69
Assuring Data Integrity	71
Checking Authorization to Use the System	71
Accessing Another User's Virtual Direct Access Storage	72
Protecting Real Storage From Unauthorized Access	72
What If Abnormal Problems Interrupt a Session?	72
Performance Options Affect Execution	74
Improving System Performance	75
Diagnosing System Programming Problems	77
Chapter 4. A Closer Look at CMS	79
Communicating with CMS	80
Initializing CMS	80
Work Flow Management through Commands and Messages	82
CMS from the User's Point of View	84
How the General User Sees CMS	84
How Other Users See CMS	84
Command Language Processors under CMS	86
Exec Procedures Are Sequences of Commands	86
Exec Procedure Language and Processor in CMS	87
VM/SP System Product Editor	88
Working in the Editor Environment (XEDIT)	88
Preventing Data Loss While Using the Editor	89
CMS File System	91
How CMS Files Are Structured	91
How CMS Files Are Identified	93

How CMS Files Are Shared Among Users	94
CMS Shared File System	95
File Pools and File Space	95
Organizing Files in SFS Directories	95
Sharing Files and Directories	98
Locking Files and Directories	99
Accessing Data in File Pools in Other Systems	100
CMS Session Services	102
Window Functions	102
Full-Screen CMS	102
Macro Support	103
System Product Editor	103
CMS HELP Facility	104
Requesting HELP	105
The Console Stack Helps Keep Work Flowing	106
Exchanging Data through the Program Stack	106
Terminal Input Buffer Is Temporary Storage for Keyed Data	107
Chapter 5. VM/SP Connectivity	109
VM/SP Systems Connectivity	110
Communications	110
Connectivity Solutions	111
APPC/VM Connectivity	115
APPC/VM	115
TSAF and AVS	115
VM Resources	117
Distributed Application Development	119
APPC Programming Interfaces	119
CMS Communications Directory	120
SNA Network Products That Run on VM/SP	122
NetView	122
ACF/VTAM	122
Advanced Communications Function/System Support Programs	123
Remote Spooling Communications Subsystem Networking	124
The Network	124
VM/Pass-Through Facility	126
VM/Pass-Through from the User's Point of View	126
IBM Enhanced Connectivity Facilities	127
Why Is There a Need for Enhanced Connectivity Facilities?	127
What Is Needed to Use the Enhanced Connectivity Facilities?	127
PC/VM Bond	129
Requirements	130
Virtual Machine/Personal Computer	131
Requirements	132
Chapter 6. Applications for VM/SP	133
Application System	134
Business Planning	134
Statistics and Forecasting	134
Project Management	134
Document Preparation	135
Business Graphics	135
Professional Office System	136
Preparing Documents	136
Sending and Receiving Documents	136
Formatting and Printing Documents	137

Getting Documents	137
Handling Routine Office Tasks	137
SolutionPac	138
Document Composition Facility	139
SCRIPT/VS Is for Detailed Format Markup	139
GML Is for High-Level Format Markup Language	140
Processing DCF Formatted Documents	141
DisplayWrite/370	142
Operation	143
Compatibility with Document Composition Facility	143
Text Entry and Editing Capabilities	144
Graphical Data Display Manager	147
GDDM Utilities for Drawing Charts	147
Other GDDM Features and Facilities	148
The Information Facility	150
Table Definitions	150
Table Build, Update, and Review	150
Structured Query Language/Data System	152
The Data Base Structure	152
SQL/DS from the User's Point of View	152
ISQL Facility	153
SQL/DS with Query Management Facility	155
Advantages of Using QMF	155
A QMF Example in Two Query Languages	156
Structured Query Language/Data System for Programmers	158
SQL/DS Preprocessor	158
Using SQL Commands	160
Display Management System for CMS	161
Panel Creation	161
Writing Application Programs and Procedures for Using Panels	161
Using Applications	162
Systems Application Architecture	163
Supported Environments	163
Common Programming Interface	163
Chapter 7. Programming Language Products	165
COBOL	166
OS/VS COBOL	166
VS COBOL II	167
VM/SP Callable Services Library	168
Programming Language/One	169
Features	169
The Compiler	170
The Library	170
VM/SP Callable Services Library	171
VS Pascal	172
The Language and The Compiler	172
Features	172
VM/SP Callable Services Library	173
Assembler Language	175
Assembler Fills Specific Programming Needs	175
The Assembler H Program	175
VS FORTRAN	177
Features	177
Interactive Debug	178
VM/SP Callable Services Library	178

APL2	180
Benefits	180
Features and Facilities	180
IBM BASIC	182
Features	182
Chapter 8. Distributed Data Processing Using VM/SP	185
Why Distributed Data Processing?	186
On-Site Data Processing Has Its Advantages	186
Centralized Processing Also Has Advantages	187
A DDP Network Offers the Best of Both Methods	187
Elements of a VM/SP DDP Network	189
System Requirements for the Central Site	190
Resource Requirements for the Distributed Systems	191
Reducing Message Traffic in a Distributed Network	192
Using the Programmable Operator Facility for Message Filtering	192
Using NetView for Message Filtering	193
Automated and Remote Operations in Distributed Networks	194
NetView	194
Programmable Operator Facility	194
Samples and Examples	194
Distributing Changes to Remote Systems	195
VM/Distributed Systems Node Executive	195
NetView Distribution Manager	195
Chapter 9. Extending the Capabilities of VM	197
VM/Interactive Productivity Facility	198
VM/Directory Maintenance	200
VM/SP High Performance Option	202
VM/Extended Architecture System Product	206
Chapter 10. VM/Integrated System	207
VM/Integrated System	208
Installing VM/IS	208
Using the Base Products	209
Using the VM/IS Packaged Products	209
Chapter 11. Planning, Installing, and Servicing VM/SP	213
Planning VM/SP	214
Installing VM/SP	216
Servicing VM/SP	218
Glossary of Terms and Abbreviations	223
Bibliography	241
Index	249

Chapter 1. VM/SP Basics

This chapter discusses the Virtual Machine/System Product (VM/SP) features that:

- Help you do many data processing tasks
- Share system resources among many users
- Communicate with VM/SP users.

Many of the concepts in this chapter will not be new to you. You will simply see them from a new point of view. For example, the words *virtual* and *spool* are familiar to most people. However, from a VM/SP point of view, the definitions of these words can differ from the more familiar meanings.

A Short Introduction to VM/SP

OVERVIEW

VM/SP is an interactive, multiple-access operating system. Interactive means two-way communication between users and VM/SP. Multiple-access means many people are able to use a VM/SP system at the same time.

VM/SP consists of two major parts, Control Program (CP) and Conversational Monitor System (CMS), which help you:

- Write, test, and debug programs
- Process jobs in batch mode
- Create and edit information files
- Manage a share of system resources
- Execute many operating systems and application programs
- Share information with other users.

Other facilities of VM/SP include:

- Group Control System (GCS) - Manages subsystems that support a Systems Network Architecture (SNA) network.
- APPC/VM VTAM Support (AVS) - Connects to and communicates with an SNA network.
- Transparent Services Access Facility (TSAF) - Connects to and communicates with other APPC/VM applications.
- Interactive Problem Control System (IPCS) - An online facility that diagnoses and reports failures and manages problem information and status.

Also, many licensed programs are available for extending the capabilities of a VM/SP system.

CP Is the Resource Manager

Control Program (CP) manages system resources and gives you an individual working environment. The resources that CP manages include:

- Processor functions
- Processor storage
- Input and output devices (I/O).

The one part of VM/SP always needed for system use is CP. This is because CP creates the system work environment and controls the system resources available to you during a work session on VM/SP.

To some extent, CP lets you manage the portion of system resources it gives you. For example, you can get a resource status report from CP that shows the direct access storage given. CP also makes some changes to the resources it has given to a user.

CP lets many people share system resources. It provides resource usage in such a way that each person works independently of others on the system. In other words,

CP manages one physical system so that it seems to give each of the users a separate and independent system. In Figure 1 on page 4, each person has the impression of being the only system user. You often do not have visual contact with other users and never see the real system they use.

When you first log on to VM/SP, only CP controls the working environment. Many of the facilities of VM/SP are immediately available to the user. For example, the system operator does many system management tasks under CP control. However, most of the work done on VM/SP requires an operating system in the working environment that helps with data processing tasks and manages work flow. Many operating systems run under VM/SP.

See Chapter 3 for more information on the CP facility.

CMS Manages Work Flow

Conversational Monitor System (CMS), although a part of the VM/SP operating system, is itself an operating system. Every VM/SP system includes CMS, which runs only with CP. As the name *conversational* implies, there is two-way communication between system users and CMS. You talk to CMS with commands; CMS uses system messages to talk to you.

CMS will help you do many tasks. For example:

- Write, test, and debug application programs
- Create and edit information files
- Run application programs
- Process jobs in batch mode
- Manage the CMS working environment
- Communicate with other users
- Share information with other users.

Figure 1 on page 4, shows how each user works with a separate copy of CMS.¹

Chapter 2 discusses these and other uses for the CMS operating system.

GCS Manages Subsystems

The Group Control System (GCS) is a virtual machine supervisor that manages multiple tasks. Its main purpose is to manage subsystems that support a Systems Network Architecture (SNA) network. Like CMS, GCS is shipped as part of every VM/SP system. However, the decision to install GCS is optional.

See page 116 for more information on GCS.

¹ Each copy does not have to be a complete copy because of *segment sharing* supported by CP. This is discussed in the chapter describing CP.

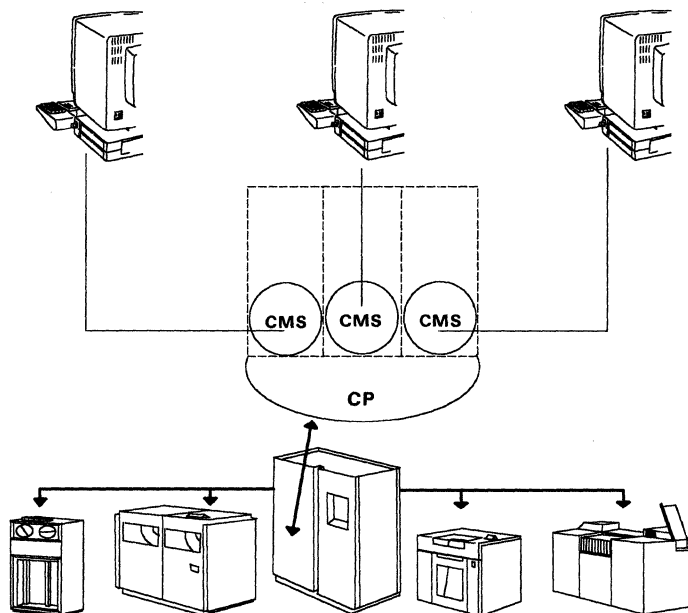


Figure 1. CP and CMS Let VM/SP Users Share One System.

AVS Enhances APPC/VM Communications

Advanced Program-to-Program Communication/VM (APPC/VM) is a programming interface that VM uses for communication within a single system or a collection of systems.

The APPC/VM VTAM Support (AVS) includes the AVS virtual machine component and lets VM users connect to and communicate with an SNA network. With AVS, an APPC/VM program in the collection can connect to APPC programs in the SNA network so you can easily access more information. Also, APPC programs in the SNA network can access global and private resources on VM.

Like CMS and GCS, AVS is shipped as part of every VM/SP system. However, installing AVS is optional. AVS runs in a GCS group and requires Advanced Communications Function for Virtual Telecommunications Access Method (ACF/VTAM) to communicate with a SNA network.

See page 116 for more information on AVS.

TSAF Enhances Virtual Machine Communication

The Transparent Services Access Facility (TSAF) virtual machine lets you connect to and communicate with other APPC/VM applications. It provides access to server resources within a VM system or across VM systems. Unlike AVS, TSAF runs in a CMS virtual machine and does not need ACF/VTAM to communicate with other VM systems in the VM collection.

TSAF is shipped with every VM/SP system, like AVS, and installation is optional.

See page 115 for more information on TSAF.

IPCS Handles Software Problems and Reports

The Interactive Problem Control System (IPCS) is an online facility that diagnoses and reports software failures and manages problem information and status.

With the problem diagnosis function, you can interactively view problem data (dumps or CPTRAP files) that resides on disks. This lets you, or a service representative, diagnose a problem from any VM/SP supported terminal without the need for hard-copy data.

The problem reporting function standardizes the problem reporting process, identifies duplicate problems on the system, and helps you quickly identify similar problems previously experienced.

With the problem and data management function, you can:

- Update problem and status reports
- Display and print reports such as Authorized Program Analysis Reports (APARs)
- Move problem data to tape for later submission to IBM.

IPCS is shipped with every VM/SP system, and it is used to service the other components of VM/SP.

See "Handling Problems with VM/SP Interactive Problem Control System" on page 45 for more information on IPCS.

Licensed Programs Extend the Capabilities of VM/SP

VM/SP facilities, together with separate licensed programs that run under VM/SP, have many additional services. These services include such things as:

- Accessing other systems for data processing
- Communicating with users on other systems
- Managing routine office tasks
- Diagnosing system program problems
- Making VM/SP easier to administer.

Reference

The *VM/SP Connectivity Planning, Administration, and Operation* book, SC24-5378, contains more information on AVS and TSAF.

The *VM/SP Interactive Problem Control System Guide and Reference*, SC24-5260, contains more information on IPCS.

Special-purpose facilities, licensed programs, and facilities provided by CP and CMS, will be discussed later in this book.

Virtual Concept

OVERVIEW

Although many people share a VM/SP system, each person seems to have exclusive use of that system's resources. This concept applies to I/O devices, processor functions, and storage. Through this concept:

- VM/SP lets direct access storage simulate actual system devices and storage.
- CP manages real system resources and gives each user separate virtual system resources.
- Total virtual storage size can be much larger than the real storage size.

This section explains how VM/SP puts the virtual concept to work.

Although many people use a VM/SP system at the same time, each of you have the impression of being the only system user. VM/SP acts as a super traffic director, making sure each user gets a fair share of system resources. VM/SP shared resources include both processor and I/O devices. The system simulates actual resources in such a way that it is like having a separate and complete system available to each of its users. Each one has a simulated (virtual) machine to use; thus the name *virtual machine*.

Why We Need Virtual Storage

Before using the idea of virtual storage, the upper limit of an address space of a computer's program was the size of processor storage. If a program was too large to run in a processor, it was written and run in pieces. This was a major problem for programmers.

In the virtual systems of today, it is possible that only a part of a program that is running can, at an instant, be in the processor's real storage. The total address space available to the program includes direct access device (virtual) storage and processor storage. Thus, for the virtual environment, storage address space extends far beyond the limiting factor of processor storage size. Now, programmers think more about the function of a program and less about storage constraints during program execution.

A VM/SP system has many people logged on at the same time. Each of you need storage space for an operating system plus a large work area. This means that each user could need one million or more storage positions (bytes). That adds up to a total of 100 million or more bytes of storage to support 100 users. This is more than the size of very large processor storage. Therefore, some of this storage must reside some place other than in the processor.

If a program needs more storage than is available in the processor, CP uses direct access storage devices (DASDs) for part of that storage. The most common kind of DASD is the disk. CP moves portions of storage content to the processor when needed and returns it to direct access devices during the time not needed for processing. Because this facility has data reside in direct access storage until needed in the processor, the size of a user's virtual storage can be very large, even larger than the entire processor storage.

How Virtual Storage Differs from Real Storage

Virtual storage can be on a DASD when the contents are not needed for processing. When virtual storage content is needed for processing, it is moved to real storage. A given area of real storage serves as virtual storage for any one of the system users, depending on the relative processing needs of the users at that time. This lets a small amount of real storage serve as processing storage for what is collectively a much larger virtual storage (see Figure 2).

A user's virtual storage size can be any multiple of 4K, with at least 8K (4K stands for 4,000; rounded from 4096 bytes). The maximum is limited only by the addressing structure of the processor on which VM/SP runs and by the availability of direct access storage. On processors using System/370 architecture, addresses are 3 bytes long. The largest hexadecimal value that can be expressed in 3 bytes is about 16 million (16 meg). Therefore, in these systems the upper limit of virtual storage is 16 megabytes (Mb).

Real storage, also called *main storage*, is a part of the system processor. Real storage size is determined by the processor model installed. Each real storage position has a constant address, ranging from 0 to the maximum processor storage size. CP adjusts (maps) virtual addresses to real addresses as needed.

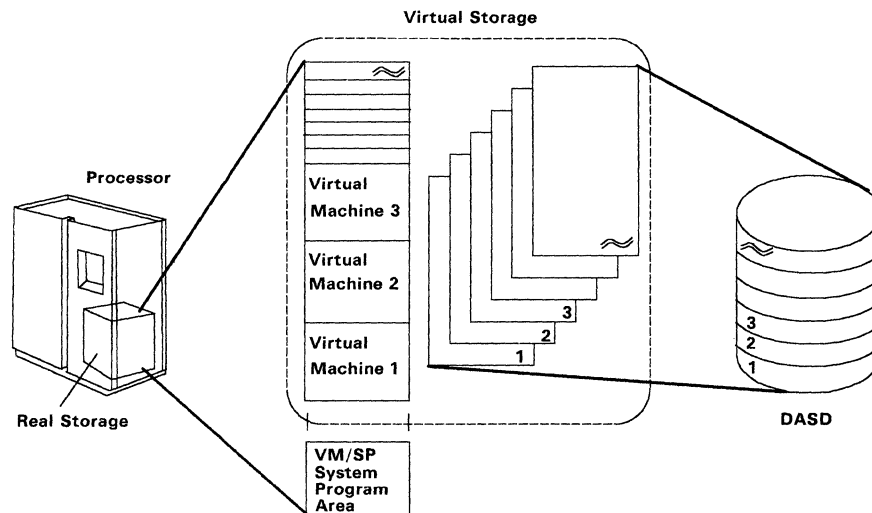


Figure 2. Virtual Storage Can Be Much Larger Than Real Storage.

Virtual Unit Record Devices

This book uses the term *unit record devices* to refer to readers, punches, and printers.² These devices, regardless of their speed, can accommodate only one user at a time. However, it is possible that many users will simultaneously need the services of the same type of I/O device. It is not practical for a system to have enough of these devices to respond immediately to all I/O requests. On the other hand, you are not expected to stop working until an I/O device becomes free.

VM/SP's answer to this problem, is to simulate device functions by spooling (storing) the data on direct access storage until the required real device becomes free to process the data. It holds the data on direct access storage until the receiving user or device is ready to retrieve it. Meanwhile, you are free to go about another task. This method of storing I/O data on direct access storage is how CP has separate virtual system devices for each user.

Magnetic Tape Devices

Because of the way tape is scanned, read, and written, having more than one person using the same tape device at the same time is not possible. If you need a magnetic tape device, you can request that it be assigned (dedicated) temporarily for your exclusive use. The dedicated usage of tape devices is controlled by the system operator.

Virtual DASDs

One DASD can meet the needs of many users. CP divides real DASD volumes into areas called *minidisks*, which function as virtual DASDs. A minidisk is defined in the CP system directory³ and assigned to a specific virtual machine. Minidisks can be defined in various sizes to accommodate system and user needs.

VM/SP uses minidisks in two ways to supply you with virtual direct access storage:

- In the CMS Shared File System (SFS), collections of minidisks called *file pools* are assigned to server virtual machines. A server is a set of programs that execute in a virtual machine to manage a resource, such as a file pool. A user who is enrolled in a file pool can be given exclusive access to an amount of *file space* in the file pool which creates and stores CMS files. Each user's file space might be distributed among several minidisks in the file pool. However, SFS users can do their work unaware of the file pool minidisks. The server virtual machine handles the management of these minidisks.
- Users can also have access to personal minidisks and shared minidisks. A personal minidisk is defined in the user's entry of the system directory. That user is considered the *owner* of the minidisk. Usually the minidisk owner is the only user who can create and store files on the minidisk. However, the owner can let other users access the minidisk, usually on a read-only basis. A minidisk accessible by non-owners is a *shared minidisk*.

² Historically, unit record devices were real card readers, card punches, and printers. However, with changes in technology systems over the years, these have become *virtual* card readers, punches, and printers (the real ones are usually in the computer room connected directly to the computer system). Today, they are still known as virtual card readers, punches, and printers and collectively as *unit record devices*.

³ The system directory is a table of information located on the direct access storage of CP. It is built as part of the VM/SP system generation process. There are entries in the system directory that tell CP how to give system resources to each user.

You can have indirect access to minidisks through the CMS Shared File System, or direct access through personal and shared minidisks, or both. At any given time, many users can be accessing the various minidisks that reside on one DASD. However, each user is usually unaware of sharing the real device with other users. There is more information about minidisks and file pools later in this chapter under "Virtual Direct Access Storage: Minidisks and File Pools."

How a Virtual Console Differs from a Real Console

The word *console*, as used in this book, means a device that you use to manage system functions. This differs from *terminal* in that a terminal can be used for any exchange between a user and the system. A console and a terminal can be the same physical device. A terminal becomes a console when it manages virtual or real resources. Broadly speaking, there are two kinds of console users:

1. System operator - manages real system resources
2. General user - manages virtual machine operation.

The main difference between the system operator's virtual console and the general user's virtual console is the authority CP gives each to enter certain commands.

The real system console is a part of the processor hardware. Only the system operator and certain maintenance people usually use the real console. They use it to take some action that affects the real system. For example, the operator uses it to start the system, or a service representative might use it to diagnose processor problems.

Most users, including the system operator, manage their view of the system through a virtual console. A virtual console is a terminal that you log on to VM/SP. Entries in the system directory for each user logging on to VM/SP determine the limits of that user's console functions.

How VM/SP Shares Processor Usage

A VM/SP system shares processing among many users through a process called *time slicing*. This system does not run many programs at the same time. Consequently, it runs part of one program for a short time, then runs part of another, then another, until it returns to the first program, and begins the cycle again. It appears to the person running each of these programs that no one else is using the processor.

Reference

The *Introduction to Virtual Storage in System/370 Student Text*, SR20-4260, describes the basics of virtual storage.

The *VM/SP Terminal Reference*, GC19-6206, lists the terminal types that will access VM/SP. It also describes virtual console options and procedures.

What Is a Virtual Machine?

OVERVIEW

A virtual machine consists of real and virtual I/O devices, a processor, virtual storage, and an operating system.

It also:

- Has functions similar to those of a real single-user system
- Is created through CP when a user logs on to VM/SP
- Lets each user access a share of each real system resource.

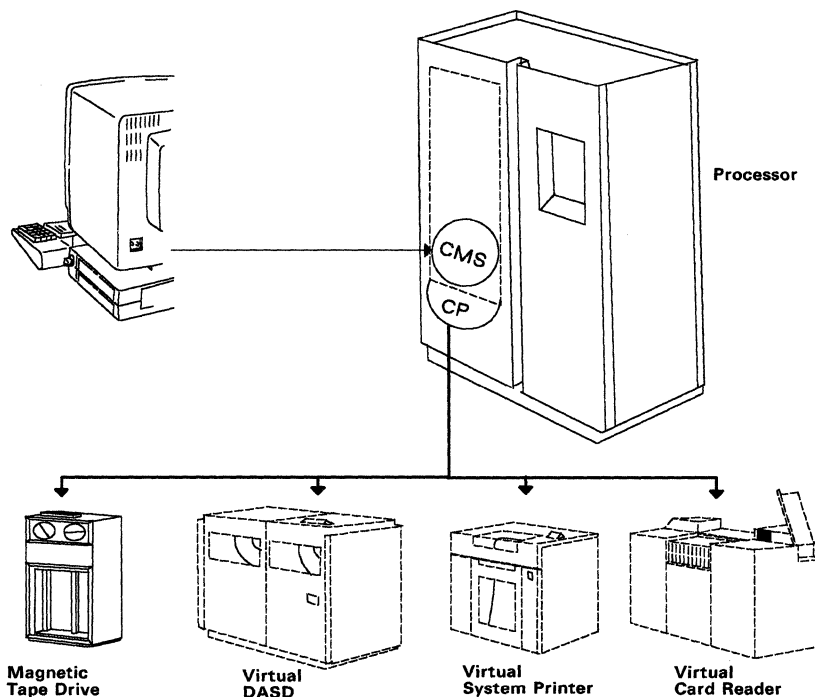


Figure 3. Using a Virtual Machine

Figure 3 shows some of the typical facilities that make up a virtual machine. Some of the devices are physically present (console, magnetic tape drive) and some are not (DASD, printer, reader).

Requirements for Using a Virtual Machine

CP produces a virtual machine for a user's exclusive use if you:

- Use a supported terminal device connected to the system
- Are entered as a user in the system directory
- Complete the system logon procedure.

After CP produces a virtual machine, work on the system as though you were the only person using the real system. You simply load any one of the many operating systems that run in a virtual machine, then go to work. A virtual machine lets you access:

- A share of processor time
- Virtual storage
- Virtual unit record devices (reader, punch, and printer)
- Virtual direct access storage (minidisks and file pools)
- Magnetic tape devices
- Other less frequently used resources. For example, communication lines to remote sites or adapters to connect processor channels to each other.

Operating Systems Run in Virtual Machines

Operating systems that run in a virtual machine are called *guest operating systems*.

CMS is the operating system most frequently run in a virtual machine. Other operating systems that run in a virtual machine include:

- VSE/SP (and earlier versions of this DOS based family of products)
- MVS/SP™ (and earlier versions of this OS based family of products)
- VM/SP itself.

For more information about running operating systems under VM/SP, see the *VM Running Guest Operating Systems* book.

Input/Output Devices

A virtual machine includes the following virtual and real I/O devices:

Unit Record Devices: These devices (reader, punch, and printer) of a virtual machine are simulated by CP direct access storage. From each user's point of view, it seems that you have exclusive use of the real system devices. In reality, you can be sharing the devices with 100 or more other users. For more information about this process, see "Spooling" on page 21.

Direct Access Storage Devices: CP shares DASD among all virtual machines by dividing the DASD surface into areas called *minidisks*. Minidisks are also called *virtual direct access storage devices*. Minidisks are discussed in a separate topic later in this book. Do not confuse the concept of sharing DASDs among virtual machines with the term *shared DASD*, which for some operating systems means sharing DASDs among multiple processors.

Magnetic Tape Devices: These devices are not usually a permanent part of a virtual machine's configuration. A system operator uses the ATTACH command to request that CP reserve (dedicate) a tape device for exclusive use by a specific virtual machine. This is usually done for tape or DASDs, but other devices are sometimes dedicated as well. Only the operating system running in a virtual machine uses a device dedicated to that virtual machine.

Other Virtual Machine Facilities

Other than those already discussed in this topic, CP produces many facilities that are available to a virtual machine. In addition to a console, basic unit record devices, virtual storage, and a share of the processor's time; CP gives to a virtual machine such facilities as a timer function or an accounting function. CP also gives more than one of each of the unit record devices, or dedicates device types not commonly included in a virtual machine.

System directory entries or user commands to CP authorize these facilities.

Communicating with Other Virtual Machines

You can exchange messages or data files with other users (virtual machines) on the same system or on other systems. SFS users can be authorized to share data stored in file pools. In addition, a virtual machine can be authorized to share data residing on a minidisk that belongs to another virtual machine. For more information on minidisks and file pools, see "Virtual Direct Access Storage: Minidisks and File Pools" on page 18.

User programs communicate with user programs on the same system or on different systems. Refer to "Advanced Program-to-Program Communication/VM (APPC/VM)" and "Transparent Services Access Facility (TSAF)" in this book for more information.

Requesting Status Information

Virtual machine users display status information about their virtual machines or the VM/SP system. The QUERY command requests CP to display at a virtual console substantial information about the status of virtual machine resources. For example, it will show if a virtual reader is currently holding any files.

The INDICATE command requests from CP a status report about current usage of major resources provided by that VM/SP system. This includes status information about all users, or information that applies only to your virtual machine.

Reference

The *VM Running Guest Operating Systems* book, GC19-6212, describes detailed information about generating and using operating systems under VM/SP for system programmers.

The *VM/SP CP General User Command Reference*, SC19-6211, describes CP commands related to virtual machine management.

The *VM/SP CP System Command Reference*, SC24-5402, describes CP commands available to the system operator.

The *VM/SP Planning Guide and Reference*, SC19-6201, describes system directory entries for giving resources and facilities to virtual machines.

Creating Virtual Machines and Managing Real Resource Usage

OVERVIEW

When you log on to VM/SP, CP supplies you with a virtual machine as defined by the system directory entries. When entering CP commands, you further define the virtual machine.

CP also:

- Lets each virtual machine use the processor resources at calculated intervals
- Makes the best use of processor storage by:
 - Giving it to users on a demand basis
 - Moving the contents of inactive portions of processor storage to direct access storage.
- Processes virtual machine requests to use real I/O devices.

System Directory Entries Define Virtual Machines

When you log on to VM/SP, CP checks the system directory for your valid user identification (user ID). If CP does not find the required user ID in the directory, it rejects your attempt to log on to VM/SP.

For successful logon attempts, CP uses system directory entries to define virtual machines. System directory information includes:

- Virtual I/O device data
- Classes of VM/SP commands a user is authorized to enter
- Minimum and maximum virtual storage sizes for each user
- Passwords that let users access the system
- Data to help account for system resources
- User priority for using system resources
- National language used in communications with the user.

Changing the Virtual Machine

Based on commands entered by the user, CP changes some of the effects of the system directory entries on a virtual machine definition. This might include how output is printed, or starting a link between a virtual machine and a minidisk in another virtual machine. These commands to CP are entered either manually or automatically from procedures. In Figure 4 on page 14, CP receives directory statements and user commands as input for creating the user's virtual machine.

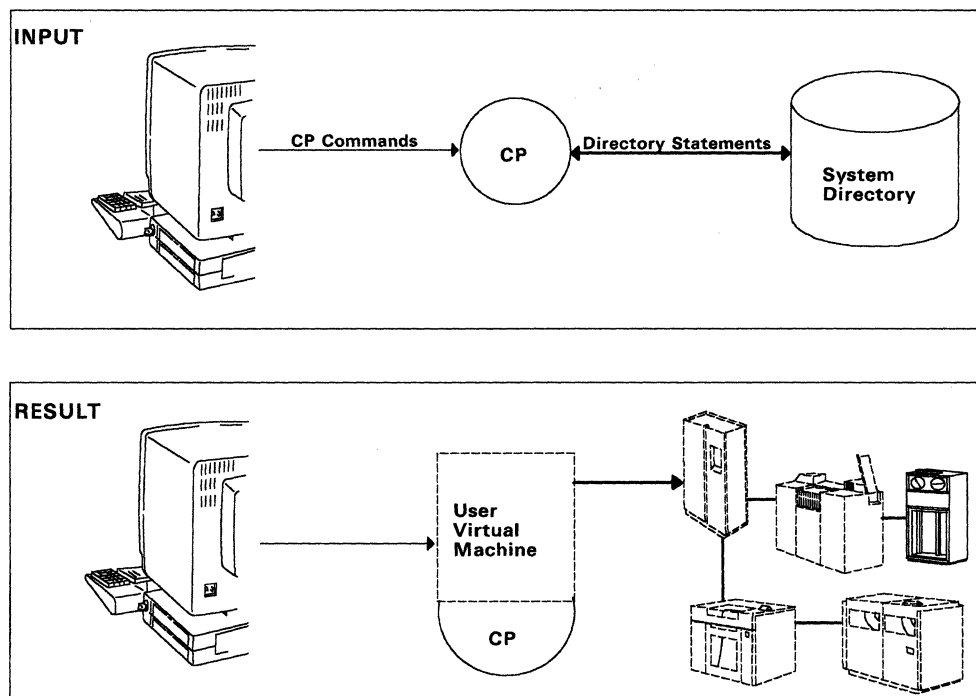


Figure 4. CP Creates a Virtual Machine.

CP Divides Real Processor Time Among Virtual Machines

VM/SP gives processor time to virtual machines on a *time slicing* basis. Time slicing means that each virtual machine receives a calculated share of the time of the real processor. This does not mean all virtual machines receive equal amounts of processor time. For example, virtual machines that run in a conversational mode receive shorter, but more frequent, slices of time than virtual machines which run application programs.

At the end of the processor time slice of a virtual machine, CP determines the type of processing that has been taking place. To do this, CP gathers data about virtual machine console request and interrupt activity. Based on this data, CP calculates when and for how long to return processor usage to each virtual machine. CP gives a virtual machine control of the processor only if that virtual machine is ready for processing. CP does not give control of the processor to a virtual machine that is waiting for certain events to occur, such as the running I/O operations.

The priority of a virtual machine for gaining control of the processor can be influenced by including a priority value in the system directory entry of the virtual machine. This value, controlled by the system administrator, is set at system generation time, but it can be changed later by the VM/SP system operator.

Real Storage Allocation Changes with User Needs

Depending on the needs of a virtual machine, CP gives more or less real storage to that virtual machine. Figure 5 on page 16 shows that during times of heavy processor activity, such as running a sort program, CP gives a virtual machine more use of real storage. The pages are moved from direct access storage to real storage as needed by CP. On the other hand, jobs like text editing will decrease the real storage given to a virtual machine. The distribution of real storage depends on the relative needs among the virtual machines at any time.

For more information on the management of CP storage transfer between real storage and direct access storage, see "Paging Lets Many Users Share Processor Storage" on page 16.

How CP Determines Virtual Storage Size

A VM/SP directory entry has the minimum and maximum virtual storage sizes for each virtual machine. When you log on to VM/SP, CP assigns the usual virtual storage size to the virtual machine it produces for you. CP will honor your request later in the session to change the virtual storage size (up to the maximum value specified in the system directory entry for that virtual machine).

Virtual Machine Requests for Real Device Usage

Each virtual machine requests output services as though it had exclusive use of system resources. A real system device can process only one of these requests at a time. Therefore, CP uses a buffering facility called *spooling* to hold output data waiting for processing on a real device. CP manages the actual processing by real system devices of spooled files.

Reference

The *VM/SP CP Diagnosis Reference* book, LY20-0892, gives more indepth information about virtual storage management.

The *VM/SP Planning Guide and Reference*, SC19-6201, describes how to change the VM/SP system directory.

Paging Lets Many Users Share Processor Storage

OVERVIEW

The basic unit of virtual storage content is called a *page* (4096 bytes). The number of virtual storage pages is usually much greater than the number of real storage page frames available. CP keeps this excess of virtual storage pages on direct access storage.

CP moves pages from direct access storage to the processor as needed (*demand paging*) and provides real storage page frames to virtual machines as needed.

How Virtual Storage and Real Storage Are Organized

Virtual storage consists of 64K-byte areas called *segments*. CP subdivides segments of virtual storage into 4K-byte blocks called *pages*. When CP moves virtual storage from a direct access device into real storage, each page fits into a 4K-byte area of real storage called a *page frame*. In Figure 5, CP sends page frames from an inactive virtual machine to an active one. To help manage page and segment activity in a system, CP maintains a separate set of page and segment data tables for each active (logged-on) virtual machine.

Virtual storage addresses have a segment number, a page number, and a displacement within that segment. During program execution, CP changes the segment number, page number, and displacement of virtual storage addresses to real (absolute) storage addresses. This process is called *dynamic address translation*. Thus, CP can place pages of any user's virtual storage into any available real storage frames.

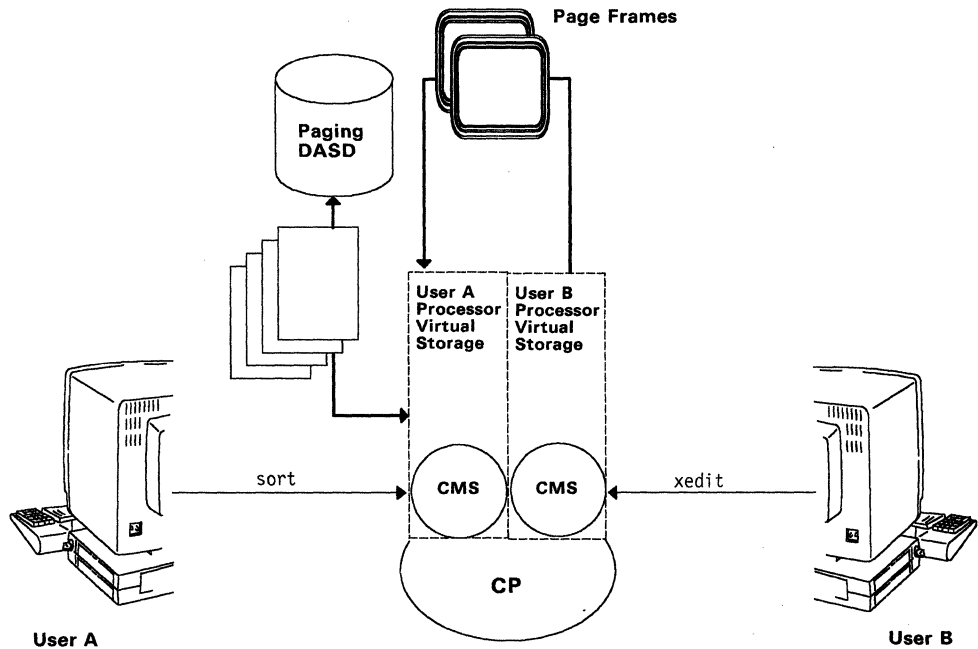


Figure 5. CP Gives Storage According to User Needs.

Paging

To make better use of real storage, CP moves inactive (or relatively inactive) pages onto direct access storage. CP moves these pages back to real storage when needed for program operation. This movement of pages between real storage and direct access devices is called *paging*.

CP manages paging operations with no action needed of the VM/SP users. The paging process is so fast that you are usually not aware that it is happening. CP reduces the effect on system performance by selectively paging out the most infrequently used storage. Pages that have been inactive are less likely to be needed by a future program event. Pages that a program refers to more often remain in real storage.

Once moved to direct access storage, a page remains there until needed by a program. The term *page fault* refers to the case where a page on direct access storage is needed by a program. A page fault results in CP's returning that page to real storage (paging in). CP does not try to predict when a particular page might be needed in real storage. This process of moving storage pages to the processor only as needed is called *demand paging*.

Because of the ability of CP to page storage to and from a DASD, a processor satisfies the needs of many active system users whose total virtual storage requirements far exceed available real storage.

Areas on a direct access device to which paged data is written or read with the least physical motion within the device are called *preferred paging areas*. Usually, pages that programs refer to more often reside in the preferred paging areas. However, if either the preferred or nonpreferred paging area begins to fill, CP shifts pages to create a more balanced page storage. If necessary, the system operator will use the MIGRATE command to ask CP to move pages between preferred and nonpreferred areas. This relocation of pages is called *page migration*.

Reference

The *VM/SP Administration* book, SC24-5285, describes in more detail paging and page migration.

The *VM/SP Operator's Guide*, SC19-6202, describes available commands to the system operator for managing page storage.

The *Introduction to Virtual Storage in System/370 Student Text*, SR20-4260, describes in detail paging concepts and the relationship of virtual storage to real storage.

Virtual Direct Access Storage: Minidisks and File Pools

OVERVIEW

A minidisk is a subdivision of consecutive cylinders on a real DASD volume. A file pool is a collection of minidisks assigned to a server virtual machine.

When you are given file space in a file pool, you can create and store CMS files in Shared File System (SFS) directories. You can also access SFS directories or minidisks and move files from one to the other.

Each directory or minidisk is accessed as a file mode and identified with a file mode letter, A through Z. This letter establishes the order that CMS searches through the file modes (the accessed directories and minidisks) when looking for data.

Each virtual machine under VM/SP uses direct access storage. Because the number of virtual machines usually exceeds the number of real DASDs available, CP divides the real DASD volumes into areas called *minidisks* which function as virtual DASDs. VM/SP users can obtain virtual direct access storage by accessing personal minidisks, shared minidisks, and temporary minidisks. In the CMS Shared File System (SFS), collections of minidisks called *file pools* are assigned to server virtual machines. SFS users who are given file space in a file pool can obtain virtual direct access storage by creating and accessing SFS directories.

How CP Defines Minidisks

A permanent minidisk is defined in the CP system directory by including a MDISK statement in the directory entry for a specific virtual machine, such as a file pool server machine or a user machine. That virtual machine is considered the minidisk *owner*. Operands on the MDISK statement specify such things as:

- Minidisk virtual address, DASD device type, and volume identifier
- Minidisk size in cylinders or blocks
- Type of primary access mode (read-only or read/write) assigned to the minidisk
- Passwords that let other virtual machine users share this minidisk.

A DASD has one real device address, but each minidisk on it has its own virtual address. The virtual address can be:

001-5FF	For a System/370 virtual machine in basic control mode.
001-FFF	For a System/370 virtual machine in extended control mode.
0001-FFFF	For a 370/XA virtual machine.

Minidisks consist of consecutive cylinders (blocks on FBA⁴ devices). Each minidisk starts at virtual cylinder/block 0. Minidisks range in size from one cylinder or block to the entire DASD volume. The storage capacity of a cylinder or block varies with the type of DASD.

⁴ FBA stands for Fixed Block Architecture. The measurement of data on FBA direct access storage devices is in a quantity of fixed-size blocks rather than cylinders, as is the case for other direct access storage devices.

The Structure of a CMS File Pool

A file pool is a collection of minidisks defined in the system directory entry for a file pool server virtual machine. A *server* is a set of programs that execute in a virtual machine to manage a resource, such as a file pool. A file pool provides virtual direct access storage that many users can share. Users enrolled in a file pool can use the file management and file sharing functions of SFS. These SFS functions are described in Chapter 4.

The minidisks in a file pool perform the following functions:

- The control minidisk contains a map of all the blocks of user data stored in the file pool.
- The catalog storage group consists of one or more minidisks that contain information about the data that exists in the file pool, who owns it, who is authorized to look at it, and so on.
- Two log minidisks maintain a record of directory changes to the file pool.
- User storage group minidisks contain all of the user data. A file pool can have many user storage groups.

A default file pool can be assigned to you by including the name of the file pool on your IPL statement in the CP system directory, or by adding a SET FILEPOOL command to your PROFILE EXEC. There is more information about the PROFILE EXEC in the “Defining and Changing Virtual Machine Configuration and Status” section.

A user enrolled in a file pool is assigned to a user storage group. Logical *file space* for creating and storing CMS files can be given to you in 4K blocks. The combined logical file space for all of the users in the storage group can exceed the actual minidisk space in the storage group. Minidisk space is given only as you *fill* the logical space with files, and this allocated space is distributed among the minidisks in the storage group.

As an SFS user, you can organize your files in directories. A *top directory* is automatically defined for each user enrolled in a file pool. Below the top directory, you can create up to eight levels of subdirectories in a hierarchy or *tree* structure. A user who is given a file space can store files at any directory level.

Accessing SFS Directories and Minidisks

You can have file space and minidisks and move files from one to the other. A CMS command (ACCESS) associates SFS directories and minidisks in CMS with the virtual machines that use them. A virtual machine accesses a directory or minidisk as a file mode, with a file mode letter (A-Z). The file mode letter determines the order that CMS uses when looking for data. Unless told to do otherwise, CMS first attempts to retrieve the data from the directory or minidisk accessed as file mode A. If unsuccessful, CMS continues the search on the next file mode, in descending alphabetical order.

SFS users can grant each other read authority or write authority on directories or the files stored in them. There is more information about SFS authorities in Chapter 4.

You can also access personal minidisks and shared minidisks. A *personal minidisk* is a permanent minidisk defined in the CP system directory entry for a user’s virtual machine. The owner has exclusive use of a personal minidisk. A *shared minidisk* is

a personal minidisk that the owner lets other users access. A minidisk is shared through use of a system directory statement (LINK) or a CP command (LINK). If a password is included on the MDISK directory statement that defines the shared minidisk, that password must be specified in the LINK statement or entered with the LINK command. The authorized access can be read/write or read-only. See the LINK statement in the *VM/SP Planning Guide and Reference* and the LINK command in the *VM/SP CP General User Command Reference*.

During a VM/SP session, you can enter a CP command (DEFINE) to request CP to give you temporary minidisk space on your virtual machine. For example:

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

requests a 10-cylinder temporary minidisk at virtual device address 291 on an IBM 3380 DASD. See the DEFINE command in the *VM/SP CP General User Command Reference*.

Initializing Minidisks

Before data is stored on any minidisk, the minidisk must be formatted to be compatible with the type of records being stored. Service programs are available to format:

- CMS minidisks
- CP minidisks
- OS, VSE, and CMS/VSAM minidisks.

Permanent minidisks (file pool minidisks, personal minidisks, and shared minidisks) retain their formatting across sessions; therefore, they need to be formatted only once. In fact, reformatting would destroy the data stored on them. However, CP destroys temporary minidisks when you log off, or when you return the temporary minidisk to CP (DETACH command). Temporary minidisks must be formatted for each session they are used; they must not be used to store permanent data.

Reference

The *VM/SP Planning Guide and Reference*, SC19-6201, describes procedures for assigning virtual machine minidisks in the CP system directory.

The *VM/SP CMS User's Guide*, SC19-6210, discusses file pools and minidisks as virtual direct access storage.

The *VM/SP CP General User Command Reference*, SC19-6211, describes the format and use of CP commands available to the general user.

The *VM/SP CMS Command Reference*, SC19-6209, describes the format and use of CMS commands.

Spooling

OVERVIEW

Spooling gives VM/SP users *apparent* immediate access to real unit record devices. The spooling process gathers a copy of virtual console data as follows:

- CP holds spooled data on direct access storage
- CP determines how to process spooled I/O files through the directory or from commands entered by the user
- CP sends spooled output to another user and to a system device.

The system (spool) operator manages spool direct access storage, real unit record devices, and the flow of spooled file traffic.

How CP Manages Spooling

Every virtual machine needs access to the real unit record devices (reader, punch, and printer) of a system. CP has orderly access to these devices by holding I/O files on direct access device queues while the files wait their turn for processing. CP also manages the flow of each of these files to the appropriate unit record device when the device becomes available. This method of giving all virtual machines a fair share of unit record device usage is called *spooling*.

You might picture spooling as files placed end to end on direct access storage, as if on a spool. However, in practice, spooled files are more likely stored in areas randomly found in spool storage.

Consider this example that points out the need for spooling. Suppose the system printer is busy printing a large file. At that time CP receives requests from three virtual machines, each to print a file on that same system printer. CP sends each user a message saying in effect "Okay, I will have it printed when the system printer is free." All three users are then free to work on other tasks. Now, it appears to the three users that their files are on the way to the system printer. Actually, CP has stored each file on direct access storage spool space while each file waits for its turn to go to the system printer. Figure 6 on page 22 shows the CP spool management facility accepting, in a parallel fashion, files to be printed and sending them, in a serial fashion, to the system printer.

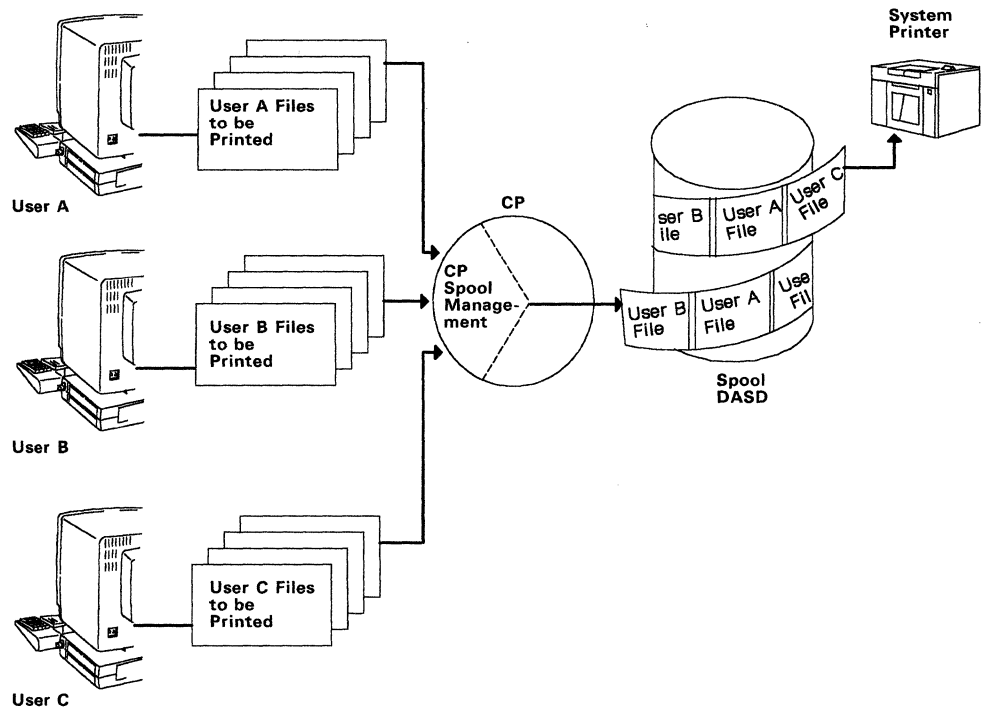


Figure 6. Shared Use of the System Printer through Spooling

The rate that a spooled file progresses from spool storage to the system printer depends a lot on the priority class given to that file by CP compared to those priorities given to other printer spool files. CP gets the needed information for this value either from a system directory or from a user request to CP to change the class of the spooled file (CHANGE command).

To process spool files, CP must receive control information from the SPOOL command, either as entered by users or from the system directory. This control information tells CP such things as:

- Which virtual devices are being spooled
- Virtual device class
- Routing data
- How output is printed.

After the spooling process of a file is done, CP sends messages to the originating virtual machine, indicating a successful completion or an error condition.

Spooled Virtual Punch and Virtual Reader Data

CP handles spooled output to a virtual punch much like spooled output to a virtual printer. CP accepts the punch request, responds to the user, and places the output file in spool storage. When it is that output file's turn for processing, CP directs the file to the system punch per your request (SPOOL command).

Receiving virtual machine input through a virtual reader is somewhat different from printer or punch spool activity. For example, you need not be logged on to VM/SP when the spooling to your virtual reader takes place. In addition, the users (not the operator) determine the loading of files from spool storage into their virtual machines.

Files spooled to the virtual reader of a user usually come from:

- Files from a user in another virtual machine
- The system operator running a card deck in the real system reader with the file directed to this user's virtual reader.

In either case, when CP sends you a message saying that a file(s) exists in your virtual reader, you can:

- Examine information about the file or the file content while the file remains in the virtual reader (RDRLIST or PEEK command)
- Load the file(s) onto your minidisk (RECEIVE command)
- Purge reader file(s) selectively with the DISCARD command or all at once with the PURGE command.

Spooling from the User's Point of View

From CMS's point of view, requesting unit record I/O operations is simple. Often it involves only a single command, such as READ, PUNCH, or PRINT. Using commands such as SPOOL or CHANGE, requests CP to change the way the system usually processes unit record data. CP also supplies information about, or changes the status of, a spooled file up to the time CP processes it on a real system device. However, CP usually completes the operation without making you aware that spooling is taking place.

Spooling results from requests to:

- Print a file
- Punch a file. Virtual punch spooling also occurs when you send a file to another virtual machine. Virtual punch spooled output is sometimes routed to the target reader of the virtual machine, rather than to the real system punch (SPOOL command).
- Read a card deck from the real card reader. Virtual reader spooling also occurs when a virtual machine receives files sent (punched) from another virtual machine.
- Collect virtual console data, such as commands entered from the console and system messages displayed at the console (SPOOL CONSOLE command)
- Send files or notes to other users on the system (SENDFILE command).

Under CMS, spooled virtual punch output or reader input is usually used for exchanging data among virtual machines, rather than for reading or punching cards on real system devices. For example, when using the SENDFILE command to send a file or note to another user, both the sending user's virtual punch and the receiving user's virtual reader get involved in the process. However, there is no involvement of the real system reader or punch device. Figure 7 on page 24 shows a typical exchange of spooled data between virtual devices without involving the real system devices.

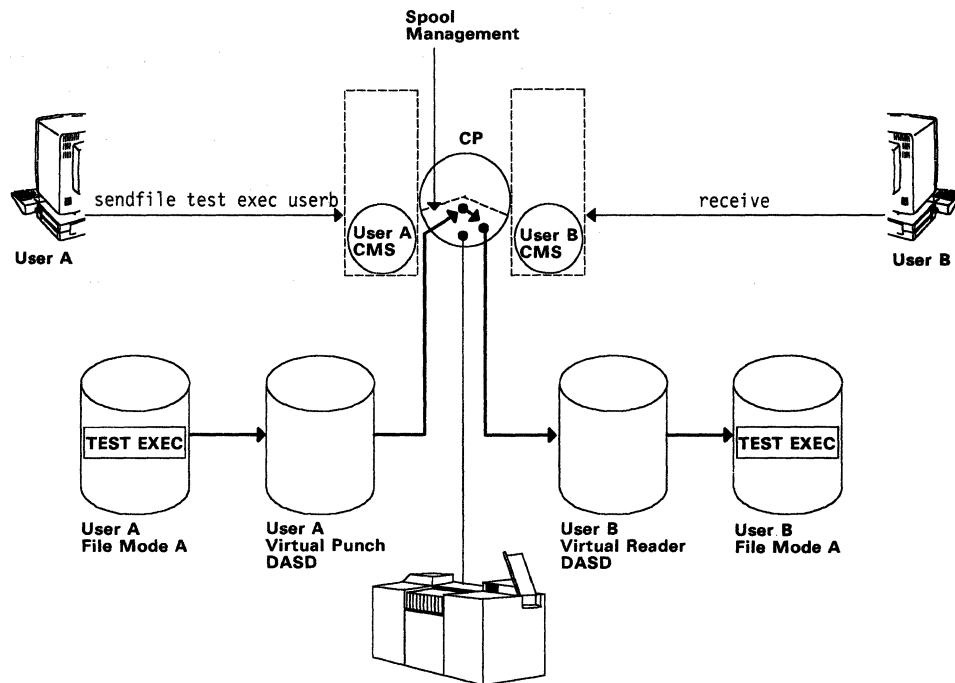


Figure 7. Spooling Virtual Punch Output to Another User

When you log on to VM/SP, you become the operator for the virtual machine that CP created for you (the user). This allows control over certain spooling functions for files associated with that virtual machine. For example, you tell CP:

- Where to send spooled output
- How to handle input from the virtual reader
- How many copies of printed output are needed.

After releasing an output file to CP for spool processing, you still have some control over that file to:

- Clear the file from spool (PURGE command)
- Change the way CP handles the file by changing the class of the file (CHANGE command)
- Rearrange the order of spooled files (ORDER command)
- Send the file to another valid queue (TRANSFER command).

CP complies with this type of request if it receives the command before actual processing by the unit record device starts.

Spooling from the Operator's Point of View

A system operator assigned to manage spool operation (a spool operator) controls movement of files between spool storage and real unit record devices. The operator alone has authority to enter commands related to spooling that affect real system devices. This authority, beyond a general user, includes issuing commands that affect spooled files. For example, the operator will:

- Restart processing of an output file from the beginning (BACKSPAC command)
- Start and stop real spool devices (START and DRAIN commands)

- Clear a file that is being processed (FLUSH command)
- Put files on, or remove files from, system hold status (HOLD and FREE commands)
- Change the type style of the system printer (LOADBUF command)
- Control the number of copies and the line spacing of printed files (REPEAT and SPACE commands)
- Send spool files to and from tape storage (SPTAPE command).

It is the spool operator's responsibility to:

- Make sure the real unit record devices are ready to satisfy user I/O requests. For example, the operator might need to put a specific print train on the printer, or load a particular type of printer paper requested by a user.
- Handle output. For example, the operator must separate printed output by user.
- Manage spooling priority. For example, although many users are assigned equal priority, one user might send a message to the operator asking that a file be printed when possible. The spool operator will cause that user's output to run ahead of the others, even though it did not arrive first on the spool. Also, the spool operator can elect to send all small files to the printer and save the large time-consuming files for when the printer is less occupied.
- Temporarily relocate some spooled files from direct access to a tape device if CP sends a message that the direct access spool area is almost full.

How to Get Virtual Console Data through Spooling

As a virtual machine operator, the user saves a copy of virtual console I/O in a spool file. There are times when this can be very useful. For example:

- When a procedure is run from which displayed data or a record of the procedure itself must be saved.
- When the user temporarily leaves the area or works from a different virtual machine, the virtual machine is run in a disconnected state. This allows the recording of messages that would be displayed at the terminal if the virtual machine had remained connected to VM/SP.

Starting, interrupting, or stopping console spooling is done by issuing appropriate options with the SPOOL CONSOLE command.

Spooling Summary

In summary, to support VM/SP spooling activity, CP:

- Provides virtual unit record devices, using DASDs as intermediate storage for data passing between virtual machines and the real unit record devices
- Manages real unit record devices of the system
- Allows for data exchange among the users' virtual machines, spool storage, and I/O devices
- Handles commands and responses related to spooling.

Reference

The *VM/SP Planning Guide and Reference*, SC19-6201, describes system directory entries that affect spooling.

The *VM/SP Operator's Guide*, SC19-6202, describes spooling in more detail.

The *VM/SP CMS User's Guide*, SC19-6210, describes spooling support for virtual machines running CMS.

The *VM Running Guest Operating Systems* book, GC19-6212, describes spooling for operating systems other than CMS running in a virtual machine.

VM/SP Is a Conversational System

OVERVIEW

Users communicate with VM/SP through commands or responses to prompting messages. However, only the system operator enters commands that affect real system resources.

VM/SP communicates with users through system messages that:

- Give a warning or show error conditions
- Give information or system status
- Ask for a response.

How a General User Communicates with VM/SP

General users communicate with VM/SP using:

- Commands to CP that relate to virtual resource management
- Commands to CMS that relate to work flow within a virtual machine.

The most common communication from a user to CP is the logon procedure to begin a VM/SP session, or the logoff procedure to end a session. Other examples include the loading of an operating system into a virtual machine, or requesting system status information. In each of these cases, the user directly communicates with CP.

So far this book has talked mostly about communication between a user and CP. This is because CP is involved in managing a user's virtual machine session right from the time you log on to VM/SP. On the other hand, communication between you and CMS takes place only after CMS is initial program loaded (IPLed) and in control of your virtual machine.

You can continue to communicate with CP after your virtual machine is under control of CMS. For example, CMS users might need to have CP assign additional direct access storage work space to their virtual machines or request the correct time. Without ever leaving the CMS environment, simply key in the appropriate CP command (see Figure 8 on page 28).

During a session when CMS controls the virtual machine, you communicate more often with CMS than CP. This is because, after CP has supplied a virtual machine for you, most session activity usually relates to CMS-controlled work within the virtual machine.

VM/SP provides a set of commands that communicate with CMS only. CMS commands are entered only when CMS is controlling a virtual machine. CP commands are entered in either the CP or the CMS environment.

For more information about communication between the user and CMS, see "Communicating with CMS" on page 80.

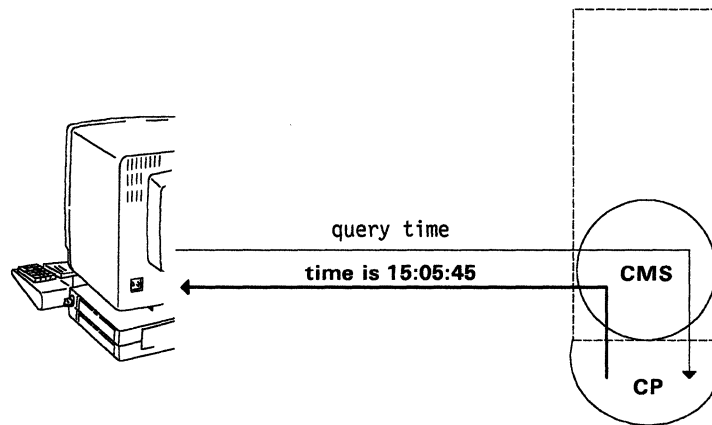


Figure 8. CP Commands Are Issued to CMS.

System Messages Communicate with Users

VM/SP has an extensive set of messages for communicating to users; CP and CMS have their own messages. The identifier for each message tells a lot about the message. For example, the message identifier DMSGEN111E tells a user that CMS entered this message (DMS), the module in CMS that generated this message (GEN), the message number (111), and that this message shows an error condition (E).

You can request, through the CP SET command, only specified portions (or none) of certain messages to actually display at your terminal. For example,

```
set emsg text
```

requests that only the text portion (no message identifiers) of error messages be displayed.

How the System Operator Communicates with VM/SP

The system operator's primary job is to manage the control of real system resources by CP and the virtual machine environment. To handle these tasks, the operator is authorized to use CP commands not available to general users. A privilege class entry in the system directory for the operator's virtual machine gives the operator authority to enter these privileged commands. Using these commands, the operator can:

- Start and stop real devices
- Control system functions
- Provide system-level management of user virtual machines.

Communication with the system operator by CP is usually as system status messages. The system operator's communication to CP might include commands that affect system resource management.

Command	Function
QUERY	Requests system status.
ORDER	Changes the relative positions of spooled files.
SHUTDOWN	Stops VM/SP system activity.

Reference

The *VM/SP Operator's Guide*, SC19-6202, describes commands whose usage is restricted primarily to the system operator.

The *VM/SP CMS Command Reference*, SC19-6209, describes commands used to communicate with CMS.

The *VM/SP CP General User Command Reference*, SC19-6211, describes general users commands used to communicate with CP.

The *VM/SP CP System Command Reference*, SC24-5402, describes system operator or programmer commands used to communicate with CP.

The *VM/SP System Messages and Codes* book, SC19-6204, describes system message format and identifier meanings, and explains the message text plus the expected response for all messages entered by VM/SP.

A Summary of VM/SP Concepts

- A single real VM/SP system gives many users their own individual virtual machines. CP manages real system resources and provides the virtual machines.
- Each virtual machine is a functional equivalent of a real system. Each virtual machine shares real processor function, storage, console, and I/O device resources.
- CMS is the conversational operating system of VM/SP for controlling work flow in a virtual machine
- When you log on to VM/SP, CP creates a virtual machine as defined in the system directory.
- Any one of many operating systems, other than CMS, controls work flow in a virtual machine.
- Paging makes real storage usage more efficient by moving inactive 4K-byte blocks of virtual storage to direct access storage.
- Spooling lets many users share unit record devices by holding I/O data on direct access storage until the needed real device is available to process it. CP carries out the spooling facility.
- VM/SP is a conversational system, because it communicates to users through system messages and prompts, and receives communication from users through responses and commands.

Chapter 2. Putting VM/SP to Work

This chapter discusses many jobs VM/SP makes easier. Because using VM/SP means different things to different people, VM/SP is used in entirely different ways by programmers, technical writers, production analysts, and so forth. Generally speaking, the most common work that people do on VM/SP is classified as:

- Running programs
- Developing programs
- Creating information files
- Getting information files
- Running other operating systems.

Remember, the purpose of this chapter is to make you aware of the kinds of things you can do on VM/SP. Do not use it as a guide for actually doing these things. Other books of the VM/SP library have guide and reference information for using VM/SP. Each topic includes references for finding additional information.

Working in the Virtual Environment

OVERVIEW

Most user sessions on VM/SP have some things in common. Among these are:

- Logon, IPL (loading an operating system), and logoff
- Other communication with CP, such as requesting system status or requesting changes to a virtual machine
- Communication with CMS (or some other operating system) to manage work in a virtual machine
- Communication with the system operator
- Communication with other users (virtual machines)
- Virtual machine I/O handling.

Using CP Commands

There are two types of commands for communicating with CP. The first type are the privileged usage (other than general user) commands which manage the real system processor or devices, or those commands that affect the overall virtual machine environment. Examples include commands such as VARY (make a device available or unavailable to CP), SHUTDOWN (stop VM/SP functions), and FORCE (force logoff of a user). A system directory entry authorizes the system operator to enter privileged commands. If a general user attempts to enter one of these commands, CP sends a response indicating rejection of the request. The *VM/SP CP System Command Reference* fully describes system operator tasks and the commands available to do these tasks.

A second type of CP command is used by general users during a VM/SP session. These commands request CP to do such things as:

- Display status information about the system or about a virtual machine. For example:
 - How busy is the system processor, or what is the current paging activity?
 - Has a file been printed yet?
 - Are there files in the virtual reader?
- Make changes to a virtual machine. For example:
 - Change the way CP handles spool files. CP usually uses internal (default) processing specifications or system directory entries to determine how spooled files are processed. For example, the number of printed output copies you are to receive can be changed.
 - Change a virtual machine configuration. For example, the size of your virtual storage can be increased or decreased.
 - Remove a device from a virtual machine configuration. For example, suppose that a virtual machine has had temporary use of a magnetic tape device. When you no longer need the device, you must detach it from the virtual machine so that it will be available to other users.

The *VM/SP CP General User Command Reference* describes CP commands that are available to most users.

CP commands are entered from either the CP or CMS environment. From the time you turn on a terminal until the CMS operating system is loaded, you work directly with CP. During this time, the system does not recognize CMS commands. When the CMS IPL process is done, the virtual machine enters the CMS environment. While a virtual machine is running under CMS, you can continue to enter CP commands.

Figure 9 shows typical communication with CP. The system operator requests CP to add a magnetic tape device to your virtual machine configuration. You then request CP to send a message confirming that the tape device is available.

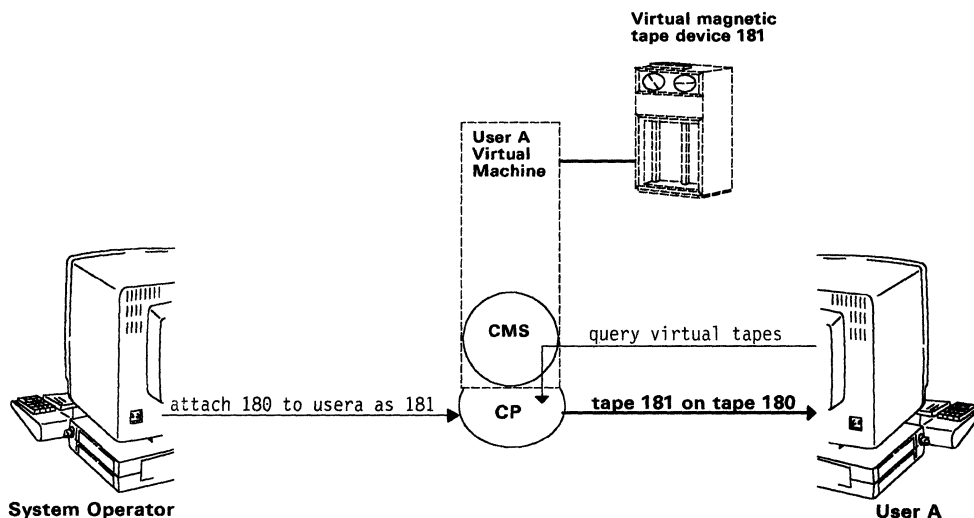


Figure 9. Communicating with CP

Communicating with Other Virtual Machines

You can exchange data files and messages with other users on the same system and with users on other systems. Within VM/SP, users of the CMS Shared File System (SFS) can share information by granting each other read authority or write authority on files or directories. The CMS Shared File System is described in Chapter 4.

VM/SP users can also send and receive files and messages. If a data file is sent to another user, VM/SP sends the file to that user's virtual reader. If a message is sent to another user, VM/SP displays that message on the receiving user's console if that user is logged on. Otherwise, VM/SP displays a message at the sender's console indicating that the receiving user is not logged on. You can request that CP not interrupt with messages (SET MESSAGE OFF). As a result, CP does not send messages to you, but tells the message sender that the receiving user has turned off message reception.

The special message facility of CP (MSG) sends messages to other virtual machines programmed to accept these messages. The message text is received by a program in the receiving virtual machine rather than being displayed on the receiving console of the virtual machine. The SET MSG and the AUTHORIZE commands let the receiving user accept or reject incoming special messages. To use this command properly, refer to the *VM/SP CP General User Command Reference* for more details.

CP also supports facilities for data exchange with users on other systems. See "Remote Spooling Communications Subsystem Networking" on page 124 for more information.

Communicating with the System Operator

There are times when program execution in a virtual machine depends on some action being taken on a real device. For example, mounting a magnetic tape on a tape device or printing output must be handled in some special way. Situations like these need communication with the system operator. Except for lengthy or complex requests, simply send the operator a message (MESSAGE command with the OPERATOR operand). The operator's response can be a request for more information or to say that the handling of the request is complete.

Virtual Machine I/O

The terms *virtual reader* and *virtual punch* do not necessarily imply any data exchange between a virtual machine and a real reader or punch device. Usually, virtual reader refers to a facility of VM/SP for supplying input to a virtual machine. Virtual punch refers to a facility of VM/SP for distributing virtual machine output. Neither of these terms refers specifically to a real system device. A virtual reader or virtual punch can be used for data exchange between a virtual machine and the real system reader or punch device. On the other hand, a virtual reader receives data sent by a user in another virtual machine. Also, a virtual punch serves as a vehicle for sending data to another virtual machine.

Some examples of virtual machine I/O data include (see Figure 10 on page 35):

- Virtual reader
 - Input card decks loaded by the system operator into a real card reader and read into a virtual reader
 - Files sent by users in other virtual machines.
- Virtual punch
 - Output files punched on a real system card punch
 - Files sent to a user on another virtual machine.
- Virtual printer
 - User-created files printed on the system printer
 - Files resulting from diagnostic dumps
 - Spooled virtual console output.

Options of the SPOOL command tell CP how to handle spooled virtual reader, punch, printer, or console data. For example, virtual printer output or spooled console data is usually sent to the real system printer for processing. However, the SPOOL command can be used to route these outputs to another user instead of to the system printer.

CP displays a system message at your terminal telling you if files exist in your virtual reader. This happens either at logon (if files were sent while you were logged off), or during a session (at the time CP puts files into the virtual reader). You can request CP to display information about the virtual reader files (QUERY READER ALL). You will be able to send the files from the virtual reader to virtual direct access

storage (RECEIVE command). Or, clear them, either selectively (DISCARD command) or all at once (PURGE command) from the virtual reader.

It is possible to destroy old files on virtual disk storage when loading new ones from the virtual reader. This happens when the file in the reader has the same identifiers as a file on the minidisk to which the reader file is being sent. The RECEIVE command displays a warning message telling you that another file with the same identifiers exists. The DISK LOAD command lets you load reader files sent by the DISK DUMP command. Using the DISK LOAD command with the default option of NOREPLACE, you will be told when a file is not received because it would overlay an existing file on the receiving disk or directory. Facilities are available for checking identifiers of files in the virtual reader and on virtual direct access storage before loading reader files.

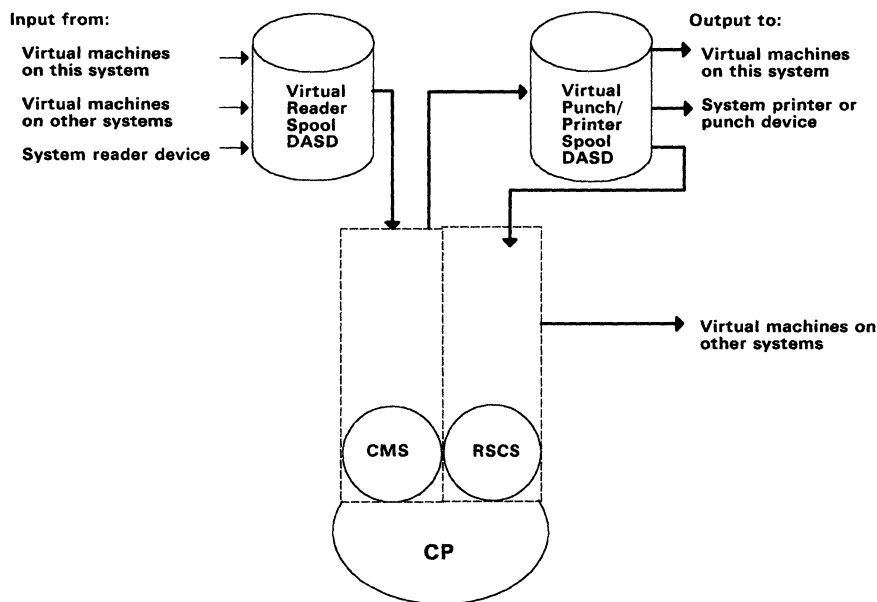


Figure 10. Virtual Machine I/O

The RDRLIST command under CMS displays information about files in a virtual reader.

The FILELIST command under CMS lets you see a list of the files that are on your accessed disks or Shared File System (SFS) directories. CMS users use the SENDFILE command to send files or notes to other users on their system or other systems. For example,

```
sendfile parts auto rocky
```

sends the file named PARTS AUTO to the virtual reader of a user identified as ROCKY in a special *names* file that the user has created using the NAMES command.

Reference

The *VM/SP CMS Primer*, SC24-5236, describes how to log on and off from VM/SP, and guides the user through many CMS functions of creating and editing CMS files.

The *VM/SP CP General User Command Reference*, SC19-6211, describes all general user commands that communicate with CP.

Putting VM/SP to Work

The *VM/SP CP System Command Reference*, SC24-5402, describes all other commands that communicate with CP.

The *VM/SP CMS Command Reference*, SC19-6209, describes all commands that are entered to CMS for managing a virtual machine.

Developing Application Programs Under CMS

OVERVIEW

To help with program development, VM/SP has:

- The VM/SP System Product Editor for writing and editing source code
- The VM/370 assembler
- Support for many program compilers, such as FORTRAN, COBOL, and PL/I
- CICS/VM for transaction processing.

A few helpful definitions in developing programs are:

Source program. A program expressed in a source language, such as Assembler, COBOL, or FORTRAN.

Assemble. The process of converting a source program written in assembler language to an object program in machine language.

Compile. The process of converting a source program written in a high-level language, such as COBOL or PL/I, to an object program in machine language.

Listing file. A CMS file to which is written a listing of source resulting from an assemble or compile process.

Object program. A program that results from using an assembler or compiler to translate source program code to machine code (code that is run on a computer).

Text file. A CMS file to which an object program is written during an assemble or compile process.

Steps in the Application Program Development Process

The steps in the application program development process are:

1. Write a source code program in assembler or high-level language
2. Assemble or compile the source code
3. Correct assembly or compile source-code errors
4. Load the object program
5. Identify files to be processed by the program
6. Run the object program
7. Correct any program errors
8. Run the object program in production mode.

Using the System Product Editor for Program Development

The System Product Editor is a VM/SP facility that creates and edits CMS files. For more information about using the System Product Editor, see "VM/SP System Product Editor" on page 88.

Using the CMS Programming Interface

CMS provides application developers with programming interfaces for assembler, REXX, and high-level language application programs. CMS provides a preferred programming interface that is bimodal. Bimodal means you can use these interfaces to code an application that is architecture-independent—it can be executed in a System/370 or 370-XA environment. You can write and assemble applications on VM/SP that are portable across architectures. When the program is executed in VM/XA SP (see “VM/Extended Architecture System Product” on page 206), it can exploit 370-XA architecture.

CMS also provides a compatibility interface that is not bimodal. This allows existing applications to execute without change in a System/370 environment.

The CMS programming interfaces consist of:

- CMS assembler macros and functions, that manage storage, perform I/O, handle interrupts, process abends, and so forth.
- Callable Services Library (CSL) routines, that call shared file system functions, access control block information, communicate with other programs, and so forth.
- OS/MVS and DOS/VSE simulation interfaces, that simulate OS/MVS and DOS/VSE macro functions in CMS.

These programming interfaces provide an application developer with stable access to a wide range of CMS functions, such as file sharing, using free storage, and various I/O operations. For more information about using the CMS programming interface, see the books listed at the end of this section.

Using CMS to Assemble Programs

The CMS assemble (ASSEMBLER XF) program converts program source files written in assembler language code into object programs in machine language. Source files to be assembled must have a file type of ASSEMBLE. The assemble process produces two output files on direct access storage. Each of these output files has the same file name as the source file. However, one has a file type of TEXT, and the other a file type of LISTING. The resulting program is ready to run in a virtual machine.

The assemble process starts by using the ASSEMBLE command under CMS. For example, entering the command:

```
assemble proga
```

processes an assembler source file named PROGA. Options can be included on the command line that affect the assemble process. The file type, although not specified, is assumed to be ASSEMBLE.

CMS finds the source file named in the command. It then causes the assembled program to run in the user's virtual machine. It writes output TEXT and LISTING files to the file mode from which it obtained the source file, if possible. Otherwise, it writes the two output files to the first available read/write disk in the user's access order.

Using CMS to Compile High-level Language Programs

Among the high-level program compilers supported by VM/SP are:

- IBM BASIC
- OS/VS COBOL and VS COBOL II
- VS FORTRAN
- APL2
- OS PL/I
- VS Pascal.

Unlike the ASSEMBLER XF program, high-level language compilers are not a part of VM/SP. Licensed programs such as COBOL, FORTRAN, PL/I, and APL can be ordered separately.

Any references to language licensed programs in this book are for information only and not intended to be used as a guide for using the programs. Each of these programs have their own information library. Anyone intending to produce or use programs using any of these high-level languages should get a copy of each of the books that support that language compiler. References to some of these books are at the end of this topic.

Compiling a program written in one of the supported high-level languages is similar to processing an assembler language program. For example:

```
pliopt mysource
```

executes the PL/I optimizing compiler to process a PL/I source file named **MYSOURCE**. Options that affect the compile process are included on the command line. Like the assemble process, the compiler writes a **TEXT** file and a **LISTING** file to your minidisk.

Figure 11 on page 40 shows the issuing of a PL/I compiler to process a source PL/I program (**MYSOURCE**). The top part of the figure shows the compiler and the source program loading into the user's virtual machine. The lower part of the figure shows the output files produced by the process.

Putting VM/SP to Work

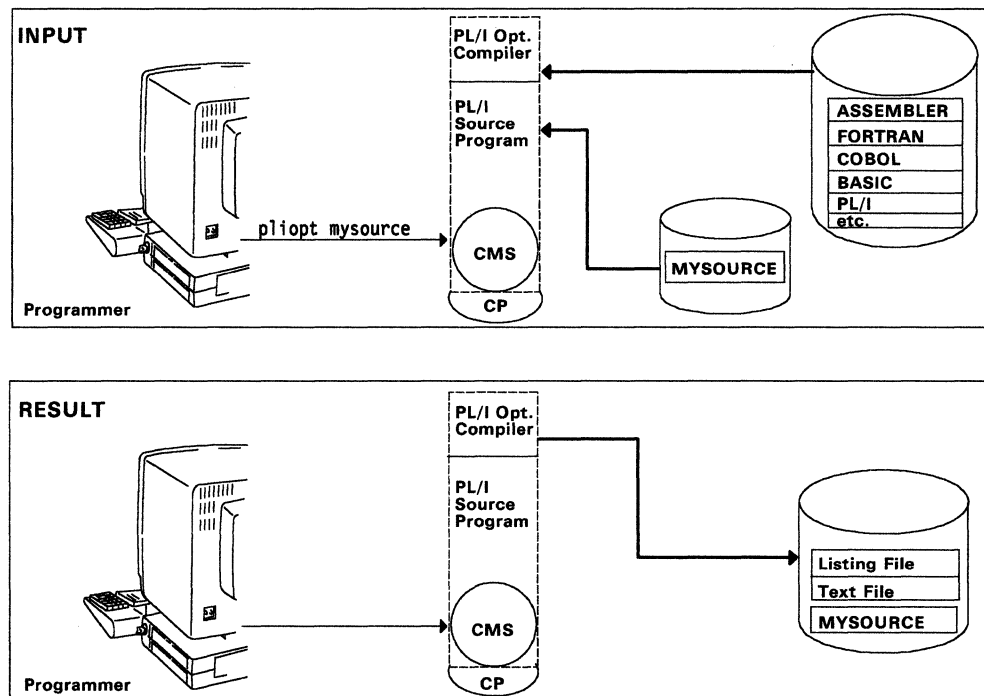


Figure 11. Compiling a High-Level Language Program

Customer Information Control System/VM

Customer Information Control System (CICS)/VM runs under CMS. It provides a transaction processing capability for use in distributed departmental VM systems. This means users in individual departments can mix transaction processing requests with a wide variety of other office and commercial applications, such as PROFS™.

With CICS/VM, you can write and execute CICS application programs to do a production transaction processing system in the VM environment. Using CICS/VM in its application mode, rather than its production mode, lets you write applications.

In a VM system with CICS/VM, end users simply choose what job they want to do, regardless of whether it requires a CICS application program. However, when jobs do require CICS applications, end users do not need any knowledge of CICS or to be aware of CICS involvement.

To familiarize yourself with application programming and CICS/VM, refer to the *CICS/VM Application Programming* book, SC33-0570.

Reference

The *VM/SP Application Development Guide for FORTRAN and COBOL*, SC24-5247, describes how to compile, link, load, run, test, and debug FORTRAN and COBOL programs using CMS.

The *VM/SP Application Development Reference for CMS*, SC24-5284, describes how to write and process program source code using CMS.

PROFS is a trademark of the International Business Machines Corporation.

The *VM/SP Application Development Guide for CMS* book, SC24-5286, describes how to develop and manage application programs, particularly assembler language application programs.

The *VM/SP CMS Primer*, SC24-5236, describes the basics of creating and editing source files.

The *VM/SP CMS Primer for Line-Oriented Terminals*, SC24-5242, describes creating and editing source files for VM/SP users of line-oriented (line mode) video display terminals.

The *VM/SP CMS Command Reference*, SC19-6209, describes the ASSEMBLE command and its options.

The *CICS/VM General Information* book, GC33-0571, introduces the CICS/VM program product and its benefits.

The *CICS/VM Application Programming* book, SC33-0570, describes application programming with CICS/VM.

The following lists some of the information available for the programming languages supported by VM/SP. See the *IBM System/370, 30xx, 4300, and 9370 Processors Bibliography*, GC20-0001, for additional information.

1. Assembler H

- *Application Programming Guide*, SC26-4036.

2. Assembler XF

- *OS/VS, VM/370 Assembler Programmer's Guide*, GC33-4021.

3. COBOL

- OS/VS COBOL Compiler and Library (program number 5740-CB1)
General Information, GC28-6470
VM/370 CMS User's Guide for COBOL, SC28-6469.
- VS COBOL II Compiler and Library (program number 5668-958)
General Information, GC26-4042.

4. FORTRAN

- VS FORTRAN Compiler and Library (program number 5748-FO3)
General Information, GC26-3983
Application Programming Guide, SC26-3985.
- FORTRAN Interactive Debug (program number 5734-FO5)
Guide and Reference, SC28-6885.

5. PL/I

- OS PL/I Optimizing Compiler and Libraries (program number 5734-PL1)
General Information, GC33-0001
CMS User's Guide, SC33-0037.
- OS PL/I Checkout Compiler (program number 5734-PL2)
General Information, GC33-0003

CMS User's Guide, SC33-0047.

- DOS PL/I Optimizing Compiler (program number 5736-PL1)

Programmer's Guide, SC33-0008

CMS User's Guide, SC33-0051.

6. RPG - DOS/VS RPG II (program number 5746-RG1)

General Information, GC33-6120

User's Guide, SC33-6074.

7. APL - APL2 (program number 5668-899)

General Information, GH20-9214

Installation and Customization under CMS, SH20-9221.

8. BASIC - IBM BASIC (program number 5668-996)

General Information, GC26-4023

Application Programming Guide, SC26-4027.

9. Pascal - VS Pascal (program number 5796-PNQ)

General Information, GC26-4318.

Debugging Application Programs

OVERVIEW

VM/SP has many facilities to help find problems in application programs. These include:

- Program listing files (file type LISTING) resulting from an assemble or compile process
- DEBUG command (CMS)
- SVCTRACE command (CMS)
- TRACE command (CP)
- PER command (CP)
- DISPLAY, ADSTOP, and STORE commands (CP).

The subject of application program debugging is far too broad and complex to be described fully in this book. This topic presents the VM/SP facilities that help debug application programs and where to find more complete information about these facilities. The *VM/SP Diagnosis Guide* has a more in-depth discussion of debugging and executing application programs under VM/SP. "Diagnosing System Programming Problems" on page 77 discusses system program debugging.

For information on tools and commands for debugging programs that run under GCS, see the *VM/SP Diagnosis Guide*.

Checking a New Program

A newly assembled or compiled program can be tested by loading it into a virtual machine and then starting it. For example, when a program named TEST1 is compiled, one of the results is an executable file named TEST1 TEXT. To run the program, enter:

```
load test1
start
```

Additional commands are needed in preparation for program execution. For example, if the program TEXT file resides in a system library, CMS needs to be told where to find it (GLOBAL command). Or, if the program refers to I/O files, they need to be defined before the program is run (FILEDEF command).

CMS Facilities for Debugging Programs

The following are some CMS facilities to help you with program debugging:

LISTING File: To help you find application program problems, CMS has a LISTING (file type LISTING) file on direct access storage when a source program is assembled or compiled. Using a printout of this file, or displaying the file at a terminal, helps you follow step-by-step program execution.

DEBUG Command: The CMS DEBUG command displays status information following ABEND processing.

SVCTRACE Command: The SVCTRACE command under CMS gives more complete data about SVC execution in a program than the CP TRACE command. It has such information as:

- Register contents before and after the SVC
- Name of the called routine
- Location from where the routine was called
- Contents of the passed parameter list.

CP Facilities for Debugging Programs

The following are some CP facilities to help you with program debugging:

TRACE Command: This command under CP monitors many kinds of program activity, including: instructions, branches, interrupts, and I/O.

Options affecting execution can be included on the TRACE command line. For example:

```
trace program branch
```

traces virtual machine program interrupts and successful branches and sends the results to the virtual console.

PER Command: The program event recording (PER) command monitors virtual machine events as they occur during program execution. It does this by suspending program execution after each event. The PER command traces:

- Successful branches
- Instruction fetching and execution
- Instructions that alter general-purpose registers
- Instructions that alter storage.

DISPLAY Command: Looks at the contents of virtual storage, registers, or system status words (CSW, CAW, PSW).

ADSTOP Command: Stops program execution at a specified instruction address before execution.

STORE Command: Changes the contents of a storage position, register, or control word.

Reference

The *VM/SP CMS Command Reference*, SC19-6209, describes in detail the DEBUG command.

The *VM/SP CP General User Command Reference*, SC19-6211, describes in detail the CP commands used for program debugging.

The *VM/SP Diagnosis Guide*, LY24-5241, describes the GCS debugging facilities.

Handling Problems with VM/SP Interactive Problem Control System

OVERVIEW

VM/SP Interactive Problem Control System (IPCS) is included as part of VM/SP and runs in a CMS virtual machine. Its facilities help you handle system software failures.

IPCS has three major functions:

- Problem reporting
- Problem diagnosis
- Problem and data management.

IPCS is an integral part of VM/SP. It runs as an application under CMS in a virtual machine. IPCS has many facilities that help IBM and customer service personnel handle system software problems in an orderly, efficient, and well-recorded manner. IPCS helps stop such time-consuming things as lengthy diagnosis of different occurrences of the same problem. It simplifies the tasks of locating existing fixes or reporting new problems to IBM.

How IPCS Works

IPCS facilities include:

- A set of commands for diagnosing and reporting problems
- Messages
- Data files for each problem, including:
 - Problem report
 - Summary of problem symptoms
 - Problem status
 - Problem data record
 - IPCS dump output
 - CPTRAP trace activity (spool file)
 - IPCS load map
 - Supplementary problem data
 - Customer profile.

IPCS uses information from CPTRAP files, CP ABEND dumps, CMS dumps, GCS dumps, TSAF dumps, PVM dumps, RSCS dumps, AVS dumps, SFS server dumps, and any other dumps produced by the CP command VMDUMP or Stand-Alone Dump, or DIAGNOSE X'94'. Using VMDUMP, the operator makes available to IPCS a dump of user-detected software problems. Figure 12 on page 46 shows how IPCS produces a dump listing or data for IBM support center problem assistance.

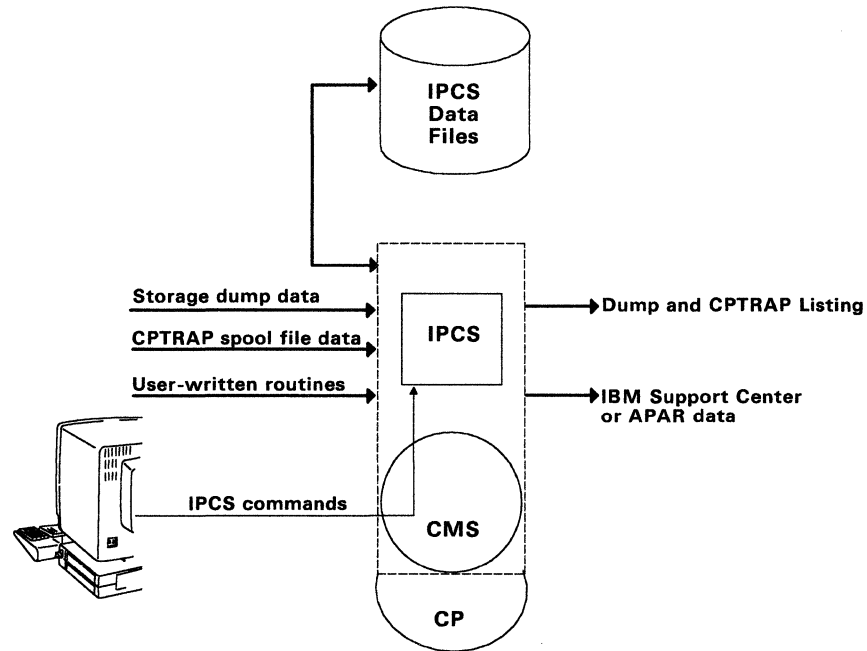


Figure 12. IPCS Is an Organized Approach to Handling Software Problems.

You can use the dump data, CPTRAP data, and IPCS commands to:

- Gather information needed when contacting an IBM support center for a possible fix
- Interactively examine the dump contents and CPTRAP files for a possible solution
- Produce an APAR,⁵ if necessary, to send to IBM for help
- Update problem files in the IPCS data base.

After a problem is fixed, a complete record of the problem becomes a part of the IPCS data base to be used for diagnosing future problems.

IPCS lets you use your own routines to:

- Scan dumps and CPTRAP files
- Format portions of dumps and CPTRAP files
- Produce your own problem reports
- Produce maps to append to dumps.

⁵ An APAR is an Authorized Program Analysis Report. It reports program problems to IBM personnel responsible for resolving the problems.

Reference

The *VM/SP Interactive Problem Control System Guide and Reference*, SC24-5260, describes all the IPCS commands and subcommands.

Running Programs under CMS

OVERVIEW

Before program execution, if the program or any data needed for execution resides in a system library, that library must be identified to CMS and any I/O files referred to by the program need to be defined.

Execution starts by either:

- **LOADing and STARTing a program TEXT file (file type of TEXT), or**
- **Executing a MODULE file (file type of MODULE) by name.**

As discussed in the previous topic, source code for debugging is run by loading and running program TEXT files that result from the assemble or compile process. However, the aim of program development is to run programs in a production environment. When TEXT files are fully debugged, one or more of them can be converted into an executable *module* file. The module, made up of tested files, is then run by entering the module name from the virtual console or as a statement in an executable procedure (exec).

Preparing for Program Execution

Starting the execution of an application program requires only that the name of the program module be entered. However, a few preliminary commands are usually entered to prepare for program execution. The most efficient way to do this is from an exec procedure, especially if the same sequence of commands is to be entered on a regular basis. That way, by entering only the name of that procedure, the complete process, from preparation to program execution, is started.

If a program refers to I/O files, the FILEDEF command can be used to override certain default definition values assigned by CMS. For example, this might apply to such things as record length or block size values for these files.

During program execution, CMS needs to know which system library a file is located in. For example, TEXT files resulting from a compile process are available only from system TXTLIB (TEXT file library) members. Before a TXTLIB member is loaded, it must be identified to CMS (GLOBAL command). Figure 13 on page 49 shows the building of an executable module for running a payroll job, then running that module to print the checks.

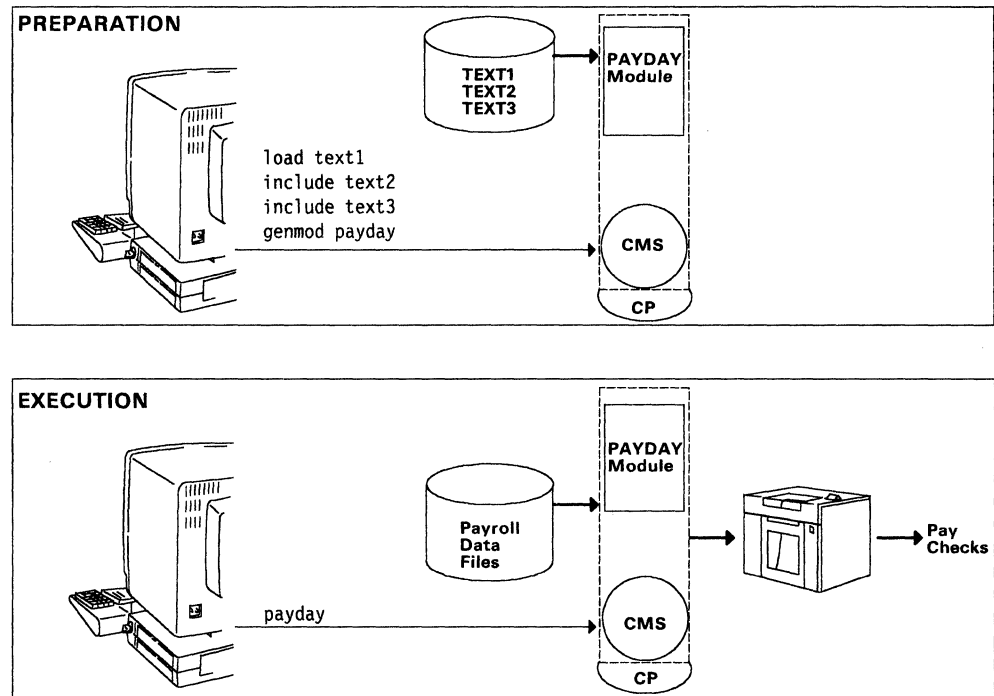


Figure 13. Generating and Running an Application Program

A Command Recap

From the time a source program is ready to be tested to the time the application program is run, some of the CMS commands used are:

Command	Function
GLOBAL	Tells CMS where to find any routines needed when processing source code during program development. Also, this command tells CMS which system library members to search for a compiled program or for other files needed during program execution.
FILEDEF	Defines each I/O file referred to by a program.
ASSEMBLE, PLIOPT, and so forth	Executes the programming language processor to convert the source code into relocatable object code in a TEXT file.
LOAD/INCLUDE	Each loads a TEXT file into a virtual machine. The LOAD command loads a single TEXT file or the first TEXT file of a group that make up a module. The INCLUDE command loads the remaining TEXT files of a group. Note that the INCLUDE command does not reset internal loader tables that establish linkage between TEXT files.
START	Begins execution of the object code after it has been loaded into virtual storage.
GENMOD	Combines one or more loaded TEXT files into a single MODULE file (file type of MODULE). The combined program is then run by entering the name of the module.

Processing Programs with the RUN Command

The complete process of program compilation and execution is done by entering the RUN command under CMS. The resulting procedure examines the file type of the file to be processed to determine what action is to be taken. Excluding an error that interrupts processing, the procedure does all steps needed for program execution.

For example:

- If the file type of the file to be processed is TEXT, the procedure simply loads the file and begins execution.
- If the file type is COBOL, FORTRAN, and so forth, the procedure first executes the proper compiler to produce an object program. The procedure then loads the object program and begins execution.

The RUN procedure passes operands to a program at execution time.

Callable Services Library Routines

VM/SP comes with a callable services library (CSL) named VMLIB. VMLIB contains a set of routines. Application programmers can call these routines, much like subroutines, to perform VM services without writing unique assembler subroutines.

The supplied VMLIB library contains routines that perform the following:

- Call shared file system functions
- Access the current generation of REXX variables
- Interface with the VM command environment through a REXX EXEC
- Execute the CMS extract/replace facility, which allows application programs to obtain or modify selected system information without release or VM system dependencies.
- Call program-to-program communication functions using the Systems Application Architecture Common Programming Interface (CPI) Communications.

The *VM/SP Application Development Reference for CMS* book contains detailed information about all the VMLIB routines except those for program-to-program communication. Those routines are documented in the *VM/SP Connectivity Programming Guide and Reference*.

CSL routines can be called from application programs written in the following languages:

- Assembler
- OS/VS COBOL and VS COBOL II
- VS FORTRAN
- VS Pascal
- PL/I
- REXX
- C.

VMLIB can be automatically loaded into storage when either the system profile or a user's PROFILE EXEC is executed.

You can find more information about callable services libraries in the *VM/SP Application Development Guide for CMS* book.

Reference

The *VM/SP Application Development Guide for CMS* book, SC24-5286, describes information about executing programs.

The *VM/SP CMS Command Reference*, SC19-6209, describes the LOAD, RUN, and START commands.

The *VM/SP Connectivity Programming Guide and Reference*, SC24-5377, assists you in writing an application program to communicate with another application program using APPC protocol.

Commands that execute compilers (for example, PLIOPT) are described in reference books for the respective compilers. Some of these compilers were listed earlier in this chapter under “Developing Application Programs Under CMS” on page 37.

Processing Jobs in Batch Mode under CMS

OVERVIEW

The system operator produces and manages the CMS batch virtual machine, which is shared by all users for program execution. The batch virtual machine runs programs one at a time, in the order it receives the programs.

VM Batch Subsystem, an IBM program offering, runs within VM/SP.

What Is the CMS Batch Facility?

The CMS batch facility is a virtual machine available to all users on that particular system. The system operator generates and controls the CMS batch facility virtual machine on a user ID dedicated to the execution of jobs in batch mode. The operator then puts the virtual machine in batch mode by entering the CMSBATCH command. The batch facility virtual machine is available to all VM/SP users for running programs that would otherwise tie up their virtual machines. System users share the facility on a first-in first-out basis. To make sure that each job starts with an error-free virtual machine, the batch machine does device housekeeping by reloading CMS between jobs.

Using the CMS Batch Facility

An application program runs in the batch virtual machine about the same as it would in your own virtual machine. The program is sent, along with some control statements, to the batch user ID (virtual machine). The batch user ID is available from the system operator who starts and manages the batch virtual machine. Either a real system card reader or a virtual punch spooled to the batch user ID runs a program.

The first records in the job stream must be:

- Job control statements (for example, the JOB statement)
- File definition (FILEDEF) statements, if the program refers to I/O files
- LINK and ACCESS statements if the batch virtual machine must refer to input files residing on your direct access storage
- SPOOL statement if the printed or punched output is sent to a destination other than the submitting user.

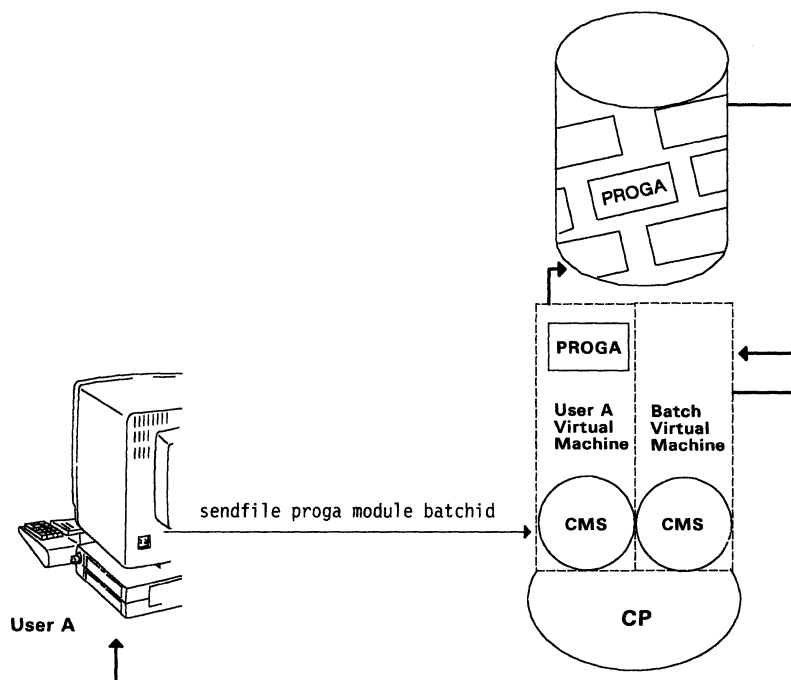


Figure 14. Running Application Programs in the Batch Virtual Machine

Some CP and CMS commands can be included in the batch job stream to control execution in the batch virtual machine. Or, the name of an exec having these commands can be passed in the job stream. The *VM/SP CMS User's Guide* discusses restrictions on using CP and CMS commands in batch jobs.

The batch facility automatically collects console data in a spool file. Output files resulting from program execution are usually routed as follows:

- Printer files and spooled console output, identified with the submitter's user ID, go to the system printer.
- Punch files, identified with the submitter's user ID, go to the system card punch.

Changing the routing of batch machine output files is done by the SPOOL command. For example, if a program's printed output and spooled console output is sent to SMITHRL (user ID) rather than the submitter, the control statements that precede the program should include:

```
cp spool prt for smithrl
```

The spool operator is able to use the PURGE or ORDER commands to control batch jobs that have not begun execution. The user, who sent the job, uses the CLOSE, PURGE, CHANGE, or ORDER commands to manage spooled batch virtual machine output files before they are processed.

What Is the VM Batch Subsystem?

The VM Batch Subsystem is an IBM program offering that customers are able to order. It schedules, starts, and monitors jobs within VM/SP. Because the subsystem controls many virtual machines that run batch jobs, many jobs can run at one time. The VM Batch Subsystem schedules each job by the job class, when it was sent, start-time range, and priority set by you or the operator. When one of the virtual machines that run batch jobs becomes free, the subsystem chooses a job and starts it in the virtual machine. The subsystem then monitors each and makes sure the job is

not stalled (incomplete, yet no longer using the processor). It also checks that the job does not exceed the maximum time limit or number of output records set by you.

Using the VM Batch Subsystem

Your job is usually a CMS EXEC file. It can also include data files that are placed on the minidisk of the batch machine.

With the VM Batch Subsystem, you will be able to easily run your job without using job control language (unless sent through remote job entry or a card reader). Jobs sent through remote job entry or a card reader need to start with a control record that states the job owner and id, job class, password, and options. Simple commands let you send and cancel a job, review a job being processed or waiting to be processed, and check the status of a job.

When you request the status of a job, the subsystem responds with the job options, the current or final status, and a summary of the job's use of system resources.

An installation can extend the VM Batch Subsystem to meet its needs. Some of the tasks that the subsystem lets user-written routines do are:

- Process user-defined commands
- Audit and change control information (for example, the account number that you assign to a job)
- Cause a job or command to be rejected and report the reason to the user who sent the job or command
- Keep statistics on subsystem use by each user
- Detect whether input through remote job entry is from an acceptable source
- Control which users can enter certain subsystem commands.

Reference

The *VM/SP CMS User's Guide*, SC19-6210, describes the CMS batch facility in detail.

The *VM System Facilities for Programming* book, SC24-5288, describes the CMS batch facility from the system programmer's point of view.

The *VM/SP Operator's Guide*, SC19-6202, describes the CMS batch facility from the system operator's point of view.

The *VM Batch Subsystem Description/Operations* book, SH20-2652, describes how to install and use the VM Batch Subsystem.

Running Operating Systems in a Virtual Machine

OVERVIEW

Logging on to VM/SP creates a virtual machine for each user. Using an operating system in a virtual machine is much like using that operating system in a real processing unit. Other than sharing system resources, operating systems run in separate virtual machines, which run independently of each other.

A new operating system can be installed and tested in a virtual machine while the old system continues to function.

CP produces a virtual machine for each user who logs on to VM/SP. This virtual machine is of limited use until you load an operating system to control work flow. There are many operating systems that run in virtual machines. CMS is the most commonly used because it can be used for many job types.

Other operating systems that run in a virtual machine include VSE/SP (and earlier versions of this DOS-based family of products) and MVS/SP (and earlier versions of this OS-based family of products). Even VM/SP runs in a VM/SP virtual machine. Figure 15 on page 56 shows how one user prepares to run MVS/SP while a second user prepares to run VSE/SP in a virtual machine.

Running an operating system in a virtual machine lets a VM/SP user work in that operating environment of the system. For example, you might load (IPL) a VS1 operating system into a virtual machine, then run in the VS1 environment while sharing the VM/SP system with other users.

The migration process under VM/SP involves:

- Moving to a new level of the base VM/SP system
- Moving to a new level of the operating system that runs in a virtual machine
- Replacing the operating system that runs in a virtual machine with a different type of operating system.

The migration process includes lengthy system testing and debugging. On some non-VM/SP systems this could mean that the system would be unavailable for productive work during installation and testing. Using VM/SP, the new operating system is installed and tested in a virtual machine while the old system continues to function.

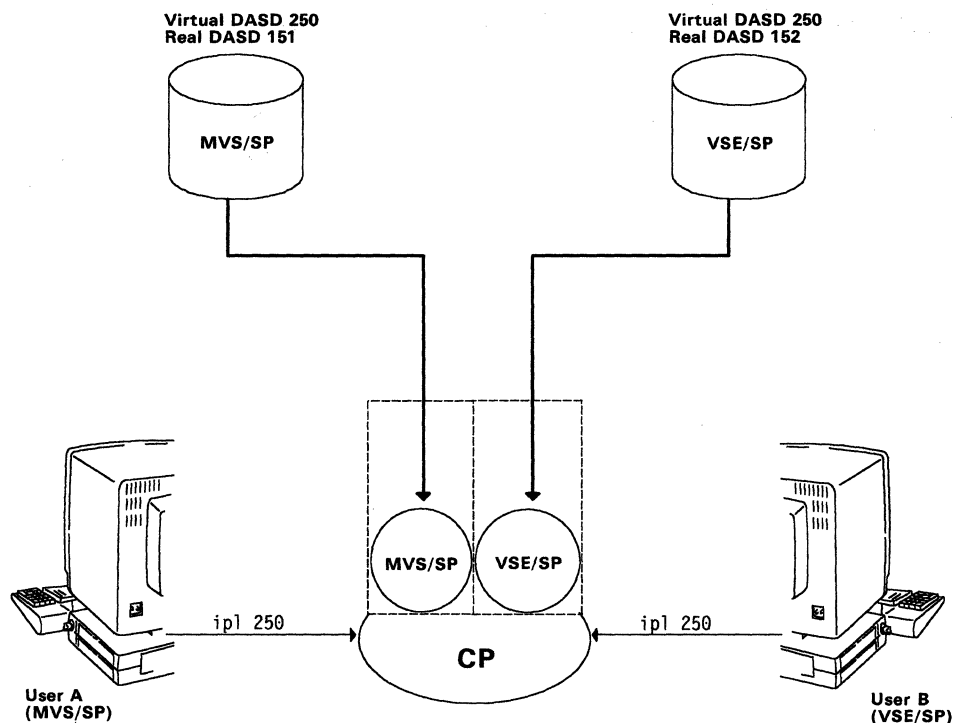


Figure 15. Loading Operating Systems into Virtual Machines

Using an operating system under VM/SP consists of:

- Logging on to VM/SP
- Loading (IPLing) an operating system into a virtual machine
- Using the operating system as if it were not running under VM/SP.

The command set of the operating system is used as though the system were running in native mode (not running under VM/SP).

Reference

The *VM Running Guest Operating Systems* book, GC19-6212, describes running operating systems other than CMS in a virtual machine.

The *VM/SP CMS User's Guide*, SC19-6210, describes CMS as an operating system in a virtual machine.

Other Programs That Run Under VM/SP

OVERVIEW

As already pointed out, many operating systems run in virtual machines under CP to manage work flow. These other programs run in virtual machines to give functions beyond those of the VM/SP program offering. You can also write your own programs.

There is a growing list of programs other than operating systems that run in a virtual machine. The primary purpose of these programs is to provide additional functions that extend the capability of VM/SP. Many of these programs are available from IBM under separate license agreements. This book refers to these programs as licensed programs. Some examples of these programs are:

Remote Spooling Communications Subsystem (RSCS) Networking. Transmits messages and data files among different systems in a network.

SQL/Data System. Manages access to structured user data in a computer.

VM/Pass-Through Facility. Lets users interactively access other systems in a network of systems.

User-written programs. Users write their own programs to provide specific functions in VM/SP. These functions are added to those provided by VM/SP or they replace portions of VM/SP programming.

Advanced Communications Function for Virtual Telecommunications Access Method (ACF/VTAM) for VM/SP. Controls communications and flow of data in Systems Network Architecture (SNA) network.

Reference

Later chapters discuss some of the licensed programs that run under VM/SP. Note that each program has its own documentation.

The *VM/SP Application Development Guide for CMS* book, SC24-5286, describes how to use CMS programming interfaces when writing applications to run under VM/SP.

Chapter 3. A Closer Look at CP

This chapter describes how specific CP facilities affect the VM/SP user. It includes information about:

- What CP facilities are available
- How these facilities will help you
- How to make these facilities available.

It does not include detailed instructions on how to use CP facilities.

After reading this chapter, you will be better prepared to use the following:

- The *VM/SP Administration* book
- The *VM/SP Operator's Guide*
- The *VM/SP CP General User Command Reference*
- The *VM/SP CP System Command Reference*.

Communicating with CP

OVERVIEW

CP receives information from:

- Entries in the system directory
- General user commands
- Commands from users with privilege other than general user, such as operator and system programmer

to help it manage real resources and the virtual machine environment.

Each CP command falls into one or more privilege classes that define who can enter it. You can enter CP commands from only the privilege class(es) that you are authorized to use.

An installation has the option of increasing the number of classes from 8 to 32 and defining which commands will be in each class. An entry in the system directory gives you the privilege to use certain CP commands.

To manage system resources and provide user virtual machines, CP must have information made available to it. For example, CP must be told how to provide resources to users, handle virtual machine I/O, and what operating systems users want loaded (IPLed) into their virtual machines. CP uses data from system directory entries or commands from users to guide it in resource management and system operation.

Usually, CP is involved in three levels of system resource management and system operation:

1. Creation of the virtual machine environment
2. Control of real system resources by the system operator
3. Control of individual virtual machines by users.

Certain CP functions are available anytime you are logged on to VM/SP. For example, entering the CP QUERY command from a virtual machine running under CMS and under CP, allows you to ask for status information about the system or about a virtual machine. However, CP does not recognize CMS commands or commands of other operating systems.

User Classes Determine System Use Privileges

The most common way to communicate with CP is through commands supplied for that purpose. CP commands are grouped according to user privilege classes (see Figure 16 on page 61). You are assigned one or more classes (A-G) in the system directory. Enter commands from these assigned privilege group(s) only. For example, general users are assigned privilege class G in the system directory. These users can enter any class G commands. On the other hand, system operators are assigned privilege class A. They can enter any class A commands, which are not available to a general user. The class ANY commands (LOGON, for example) are available to all users. An installation will extend the number of user classes from 8 to 32 by using class override control statements. These statements let the installation assign unique classes to specific commands.

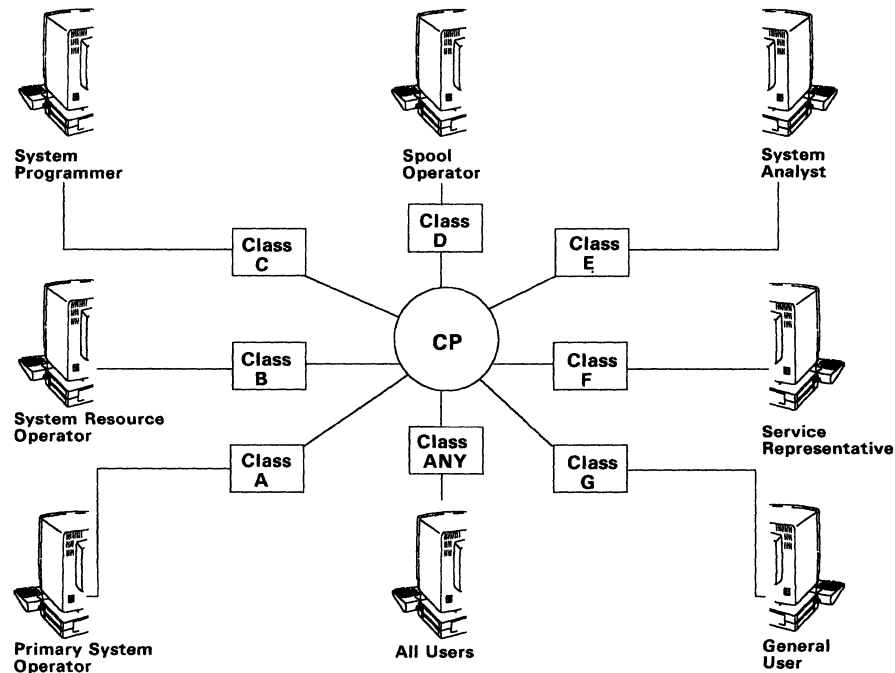


Figure 16. A Command Class for Each Type of User

System Directory Entries Help CP Produce Virtual Machines

Not all virtual machine configurations are alike. A virtual machine is the equivalent of a customized system dedicated to one user. Because each user has different needs, there are many system resources chosen by individual users.

During the VM/SP system generation procedure, control statements that define a virtual machine for each planned user are entered to the system directory.

Each time you log on to VM/SP, CP produces a virtual machine based on your user's system directory entries. In addition to common directory entries such as user IDs, passwords, and default virtual storage size; control statements often define less-common virtual machine characteristics. For example, if a device is reserved for exclusive use by one user on a regular basis, a DEDICATE control statement is included for that user. CP then automatically dedicates that device to that user's virtual machine each time the user logs on to VM/SP. This is especially helpful for virtual machine guest operating systems running applications that need specific I/O devices.

Virtual Console Management

The single console image facility of CP lets one user, running in a single virtual machine, control multiple disconnected virtual machines of other users at the same time. The CP command language lets you easily disconnect a console from one virtual machine and connect it to another in a serial fashion. A virtual machine continues to process programs when its' console is disconnected. The user of the controlling virtual machine is called the *secondary user*. The users of the controlled virtual machines are called *primary users*. The secondary user must be identified at system generation by a CONSOLE directory control statement for each of the affected primary users. When the primary user disconnects and the secondary user is logged on, the secondary user receives control of the primary user's virtual machine.

Reference

The *VM/SP Planning Guide and Reference*, SC19-6201, describes how to plan and generate control statements for the system directory, and how to plan changes to the user class structure.

The *VM/SP Administration* book, SC24-5285, describes how to make changes to the user class structure.

The *VM/SP CP General User Command Reference*, SC19-6211, describes commands available to the general user for communicating with CP.

The *VM/SP CP System Command Reference*, SC24-5402, describes the CP commands available to other than general users for communicating with CP.

Defining and Changing Virtual Machine Configuration and Status

OVERVIEW

CP determines virtual machine characteristics from:

- System directory entries
- CP commands entered by you during a VM/SP session.

System Directory Entries Define Virtual Machines

The system directory is usually produced at VM/SP system generation using the VM/SP directory service program. The directory program takes information from control statements in a user-prepared CMS file and writes it to the system directory. This file is produced using the System Product Editor, a VM/SP facility for producing and editing CMS files. The file must have a set of control statements for each authorized user. Some examples of the information written to the directory through control statements are:

Control Statement	Function
USER	User ID, password, virtual storage size, user class.
CLASS	Defines up to 32 classes for each user.
CONSOLE	Defines the virtual machine console.
IPL	Defines the system to be loaded into the user's virtual machine at logon (for example, CMS), and can define a default file pool.
LINK	Describes a shared minidisk that the virtual machine can access.
MDISK	Defines the size and location of a permanent minidisk assigned to the virtual machine.
SPOOL	Specifies the virtual machine's virtual unit record devices.

The person replacing or making changes to the system directory must have a user privilege class of A, B, or C and have access to the volume on which the directory resides. A directory maintenance facility (see Chapter 9) makes the updating of the system directory easier and lets general users make changes to information associated with their user IDs.

CP Commands Help Manage the Virtual Machine

The QUERY command with the VIRTUAL operand can display at a virtual console the status of virtual:

- Storage
- Unit record devices
- Tape and direct access storage devices
- Display devices

- Console
- Channels
- Communication lines.

The QUERY command is entered from environments other than CP. It produces different results for each environment. For example, the QUERY command under CMS causes a virtual console to display the status of many CMS functions in that virtual machine.

The DEFINE command makes changes to a virtual machine configuration. For example:

```
define storage 1m
```

sets up the size of a virtual storage at 1 megabyte (1Mb). Any changes made to a virtual machine through the DEFINE command are temporary (for the life of the session). At times, such as redefining virtual storage size, you must IPL CMS again in the virtual machine after entering the DEFINE command. Therefore, the DEFINE STORAGE command should not be entered as part of the PROFILE EXEC procedure run during the IPL of CMS.

Some other CP commands that affect virtual machine status are:

Command	Function
DETACH	Removes virtual devices from the configuration.
DISCONN	Disconnects the virtual console from VM/SP, but lets running in the virtual machine continue.
LINK	Makes minidisks in other virtual machines available to a virtual machine.
SET	Turns on or off many virtual machine functions, such as performance options, program function (PF) key functions, and message handling.
TERMINAL	Controls virtual console functions, such as character translation, command line highlighting, maximum line length, and special purpose characters (symbols).

Figure 17 on page 65 shows how two of these commands are used. In one case, a user no longer needs the magnetic tape device in the virtual machine configuration. Therefore, by using the DETACH command it will be freed. In the other case, the user is entering the DEFINE command to increase the virtual processor storage size of that machine.

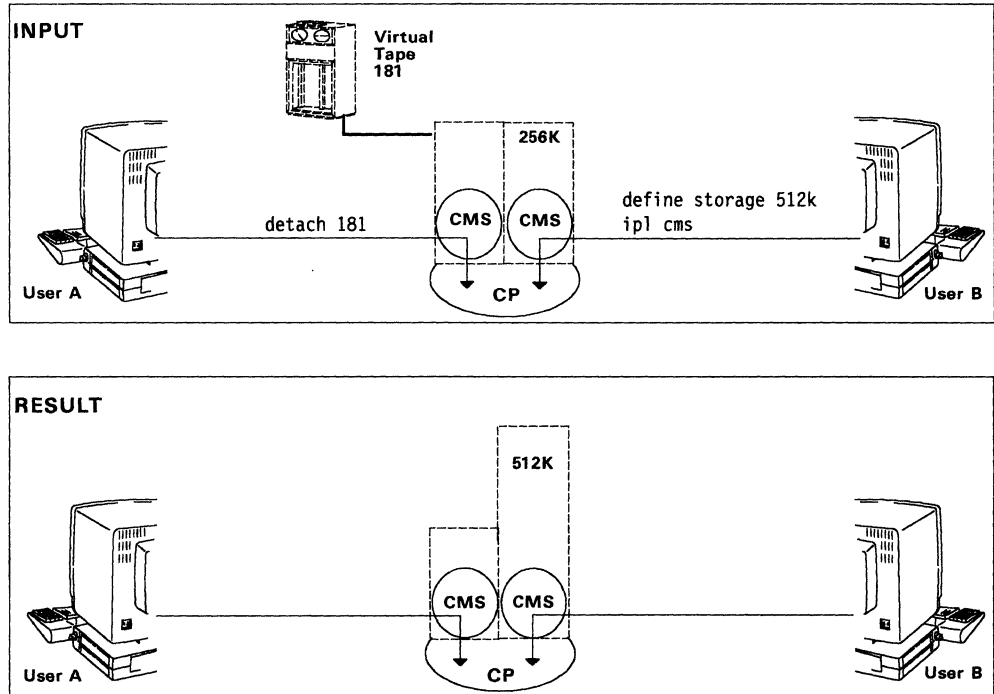


Figure 17. Changing Virtual Machines

Reference

The *VM/SP Planning Guide and Reference*, SC19-6201, describes directory control statements that define virtual machines.

The *VM/SP CP General User Command Reference*, SC19-6211, describes commands a user can enter to change the characteristics of a virtual machine.

Inter-User Communications Vehicle

OVERVIEW

This topic is primarily of interest to programmers in that:

- **Inter-User Communications Vehicle (IUCV) manages communication among programs in virtual machines on the same VM/SP system.**
- **IUCV also manages communication between a virtual machine and a CP system service.**
- **The targets that a virtual machine communicates with must be defined by an IUCV entry for that virtual machine in the system directory.**

Basics about IUCV

IUCV lets you communicate over defined paths within a VM/SP system. Communication starts with IUCV macro instructions in programs running in virtual machines. To use IUCV facilities, virtual machines must be authorized to start communication paths. Communication paths are authorized by including IUCV control statements for affected virtual machines when building the system directory at system generation. Figure 18 on page 67 shows two kinds of communication paths: virtual machine to virtual machine and virtual machine to a CP system service.

Communication over IUCV facilities takes place between a source communicator (the virtual machine program starting the communication) and a target communicator (the virtual machine receiving the communication). The information communicated is called a *message* and is of any length. A SEND/RECV protocol is set up by programmed IUCV macro functions in the source and target virtual machines.

CMS support helps IUCV functions by multiple programs running in the same CMS virtual machine. The CMS macros HNDIUCV and CMSIUCV do the following:

- Identify IUCV programs to CMS
- Begin or end communication with another virtual machine or with CP
- Specify exits for IUCV external interrupts.

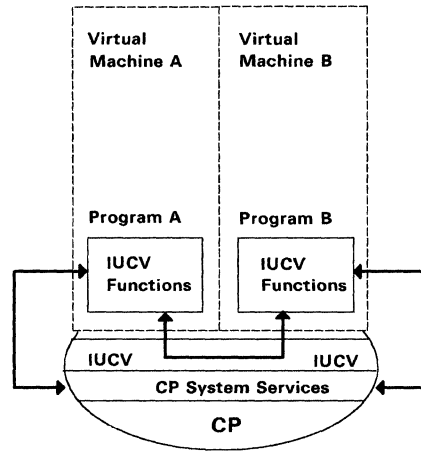


Figure 18. Communication Paths for IUCV Functions

Communicating Between Virtual Machines

When a communication path is authorized in the system directory, two virtual machines communicate through programmed sequences of IUCV functions. Typical communication tasks done by IUCV functions include:

- Each communicator generates an interrupt information buffer (DECLARE BUFFER).
- The source communicator requests a communication path (CONNECT).
- The target communicator completes the path (ACCEPT).
- The source communicator sends data (SEND).
- The target communicator receives data (RECEIVE).
- The source communicator breaks the communication path (SEVER).
- Drop interrupt information buffers (RETRIEVE BUFFER).

For more information on virtual machine communication through IUCV, see the *VM System Facilities for Programming* book.

Communicating Between a Virtual Machine and a CP System Service

IUCV supports communication between virtual machines and these CP system services:

- Console Communications Services
- Message System Service
- Message All System Service
- DASD Block I/O System Service
- Signal System Service
- Error Logging System Service (LOGREC System Service)
- Advanced Printer Subsystem System Service (SPOOL System Service)
- Access Verification System Service
- Collection Resource Management System Service

- Identify System Service.

When a virtual machine program communicates with a CP system service, IUCV routes the communication to the proper CP module for handling.

Reference

The *VM System Facilities for Programming* book, SC24-5288, describes IUCV further.

The *VM/SP Planning Guide and Reference*, SC19-6201, describes the IUCV system directory control statement.

Hardware Devices That Handle I/O

OVERVIEW

Device support, together with control units and channels, must be defined in a system file (DMKRIO) created at system generation. This includes assigning real device addresses. A VM/SP system's device requirements depend to some extent on system facilities and licensed programs available to the system users.

The device classes that VM/SP supports are:

- Processors
- Direct access storage devices (DASDs)
- Magnetic tapes
- Terminals
- Graphics
- Unit record devices
- Transmission control units and communication controllers
- Other devices (your IBM Marketing Representative can provide you with a complete list).

Defining a System Device Configuration

The VM/SP system generation process includes the preparation of a file that defines that system's real I/O device configuration. That file, DMKRIO ASSEMBLE, consists of groups of device macro instructions.

Planning Device Configurations

Setting up the device configuration for a VM/SP system, although not difficult, requires planning. A minimum configuration, listed in the *VM/SP Planning Guide and Reference*, serves as a good starting point. Also to be considered, are special device requirements for:

- CMS
- Operating systems other than CMS
- RSCS Networking
- Telecommunications
- Licensed programs running in the VM/SP system
- Special-purpose terminals.

Anyone involved in device planning for a VM/SP system should review related information in the *VM/SP Planning Guide and Reference*.

Reference

The *VM/SP Planning Guide and Reference*, SC19-6201, describes and gives examples of how to define a system device configuration. It also lists many devices supported by VM/SP.

Assuring Data Integrity

OVERVIEW

Virtual direct access storage is protected from unauthorized access. Gaining access to another user's minidisk usually requires a special password. Gaining access to files or directories in another user's file space requires read authority or write authority granted by the owner.

When logging on to VM/SP, you must:

- Be identified (have a user ID) in the system directory
- Respond correctly when CP prompts for the logon password.

Storage keys protect information in real storage from unauthorized use. Users stop a virtual machine from accessing a shared segment of storage that another virtual machine has modified.

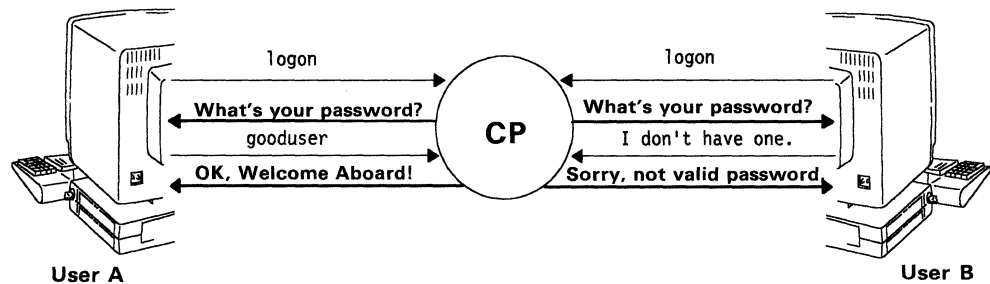


Figure 19. CP Protects against Unauthorized System Use.

Because many people can share real system resources, each user's data must be protected from unauthorized access. Data must also be protected against accidental loss caused by system failures. VM/SP has many functions that assure user data integrity. However, there is a limit to how much VM/SP can do in these areas. Each user must be aware that work environment and procedures contribute, either favorably or unfavorably, to the protection of sensitive information.

Checking Authorization to Use the System

To make sure that each person attempting to log on to VM/SP is an authorized user, CP requires that:

- The logon includes a valid user ID (included in the system directory)
- The user responds correctly to a password promptly.

At the option of an installation, passwords are invisibly masked when typed in on the screen.

An automatic deactivation feature lets the system administrator prevent certain restricted passwords. You are not allowed to set up and or use these passwords.

Accessing Another User's Virtual Direct Access Storage

To access files on a minidisk in another virtual machine, you must obtain a special *readshare* or *writeshare* password from the owner of minidisk, if the owner has defined such a password. During a VM/SP session, you must include the password with the LINK command that establishes a communication path between your virtual machine and the minidisk in the other virtual machine. Permanent linkage to other virtual machines (shared minidisks) can be established by LINK statements in the VM/SP system directory.

For virtual machines running under CMS, virtual machine linkage is started through statements in a PROFILE EXEC procedure. Also, linkages to other minidisks are started through statements in a PROFILE GCS procedure for virtual machines running under GCS. The *VM/SP Planning Guide and Reference* has information about PROFILE GCS. For more information about PROFILE EXEC, see "Communicating with CMS" on page 80.

CMS Shared File System (SFS) users can grant each other read authority or write authority on SFS directories or the files stored in them. For more information about SFS functions, see "CMS Shared File System" on page 95.

Protecting Real Storage From Unauthorized Access

CP helps protect information in real storage from unauthorized program access by storage protection keys. Each 2K or 4K area of storage has a protection key associated with it. Programs that access storage also have keys assigned. The program and storage keys must match before the program accesses a protected storage area.

Specified segments of real storage are designated as *shared* at system generation. This means that many virtual machines share a single copy of these segments of real storage. Sometimes problems can result if any virtual machine is allowed to modify a storage segment being shared by other virtual machines. The default value of a system generation option protects virtual machines from using a modified copy of a shared segment of real storage (see the NAMESYS macro instruction in the *VM/SP Planning Guide and Reference*).

What If Abnormal Problems Interrupt a Session?

Any system is subject to serious problems when a user has no control. When this happens, you might be in the middle of a session and not be able to store work files or log off VM/SP before the session ends. This could cause serious problems. The most obvious is the loss of newly generated data not yet permanently recorded on virtual direct access storage. If a serious system problem occurs, you seldom have an opportunity to log off from a VM/SP session. That could be like leaving the door to a virtual machine open to anyone having access to the system.

Fortunately, CP facilities and user procedures can effectively deal with these problems. CP assures that interrupted sessions are finished before the system is again available for use. In addition, a system generation option (VMSAVE operand of the OPTION directory control statement) lets an installation assure that, if an abnormal end occurs, the register of a virtual machine and storage contents are saved.

A CMS facility for saving working file data when a System Product Editor ends abnormally is described under "VM/SP System Product Editor" on page 88.

Reference

The *VM/SP CMS User's Guide*, SC19-6210, describes VM/SP features that help protect user data from unauthorized access.

The *VM/SP Administration* book, SC24-5285, describes shared segment protection.

The *VM/SP Planning Guide and Reference*, SC19-6201, describes system generation macro instructions that affect storage access.

The *VM/SP Installation Guide*, SC24-5237, describes the step-by-step procedures for installing VM/SP.

The *VM/SP CP Diagnosis Reference*, LY20-0892, describes the storage protection keys.

Performance Options Affect Execution

OVERVIEW

This topic is primarily of interest to system programmers in that:

- Many factors affect the performance of an operating system in a virtual machine.
- VM/SP has many options that, under certain conditions, improve system performance.

A detailed discussion of performance factors is beyond the scope of this book. The concepts and terminology associated with performance improvement are not intended for the general user and can require additional reading for any user.

Among the factors that affect system performance are:

- Processor model
- Total number of virtual machines in use
- The number of SFS server machines in use
- Type of work done in each virtual machine
- Paging device(s)
- Availability of real storage
- VM/SP performance options in effect.

Most performance options are specified at VM/SP system generation or through operands of the SET command under CP. Two CP commands for determining system performance are:

Command	Function
MONITOR	(Privileged) controls the recording of events that occur in the real VM/SP system.
INDICATE	(General use) displays at a virtual console the use of, and contention for, major system resources.

Automatic collection of data related to system performance is requested with the SYSMON macro instruction in the DMKSYS file at system generation.

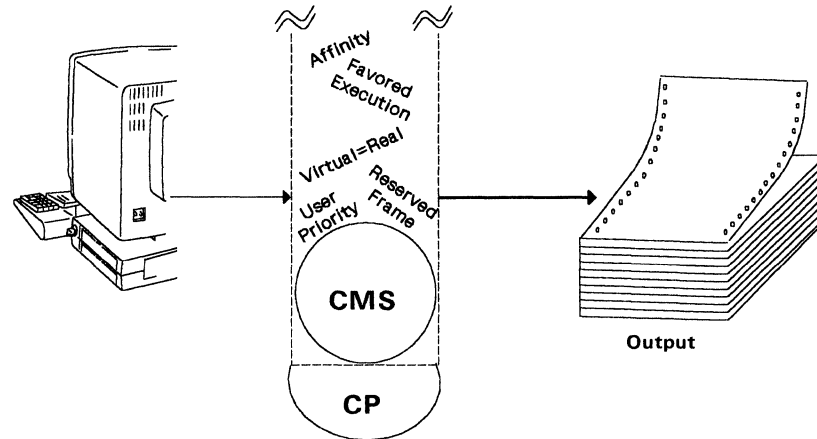


Figure 20. Performance Options Improve Productivity.

Improving System Performance

The following options improve system performance:

Favored Execution: This option gives more system resources to a given virtual machine than it would have under usual operation. The system operator specifies this option through operands of the SET command under CP.

User Priority: The system operator uses the PRIORITY operand of the SET command to give relative dispatching priorities of virtual machines on a VM/SP system. A virtual machine with a high priority (lower number) gets a larger share of system resources than one of lower priority (higher number).

Reserved Page Frames: The system operator uses the RESERVE operand of the SET command to reserve page frames for exclusive use by a given virtual machine. This will improve the performance of the affected virtual machine by reducing the contention for page frames of that machine.

Virtual = Real: This option can be specified at VM/SP system generation. It is specified on the OPTION directory control statement for any virtual machine whose virtual storage pages (other than page 0) are locked into fixed positions of real storage. The virtual addresses and the real addresses of these storage positions are equal. This eliminates CCW translation (except for page 0) and paging for the affected virtual machine. As part of its system management functions, CP relocates page 0.

Affinity: This option lets virtual machines run on attached processor (AP) or multiprocessor (MP) systems to select a processor for program execution. This option is selected at system generation time using the AFFINITY operand of the OPTION control statement. The system operator (or user authorized by an affinity directory entry) begins the option using the AFFINITY operand of the SET command.

Multiple Shadow Table Support: To reduce system overhead when a virtual machine changes control register 1 (CR1) values, VM/SP maintains a queue of segment table origins and associated shadow tables for virtual machines. Thus, each time an MVS or SVS system dispatches a new address space (changes CR1), VM/SP

dispatches the proper shadow table. General users control multiple shadow table support by the STMULTI operand on the SET command under CP.

Single Processor Mode: Running in single processor mode improves throughput for an MVS (AP or MP) system running under VM/SP. The single processor mode option lets an installation dedicate a processor to an MVS Virtual = Real (V = R) virtual machine. The system operator turns single processor mode environment on or off using the SPMODE command. The *VM Running Guest Operating Systems* book has more information about this option.

Queue Drop Elimination: VM/SP attempts to optimize throughput by dropping idle virtual machines from the active queue. Sometimes, however, virtual machines become active again sooner than expected and must be put back on the active queue. Removing virtual machines from, and restoring them to, the active queue causes considerable system overhead. The system operator uses the QDROP operand of the SET command to eliminate scanning of the page and segment activity of a virtual machine for removing inactive pages.

Virtual Machine Assist: This option, which is found on most supported VM/SP processors, handles SVC interrupts, invalid page conditions, and many privileged instructions. Although virtual machine assist improves VM/SP performance, it also results in improved virtual machine performance because it makes more resources available to virtual machine users. The system operator uses the SASSIST operand on the SET command to activate or deactivate the facility.

Extended Control Program Support:VM/370 (ECPS:VM/370): This option extends, for certain privileged instructions, the hardware support provided by virtual machine assist. The system operator uses the SASSIST and CPASSIST operands of the SET command to begin or end ECPS.

Segment Protection Facility: Segment protection support in VM/SP uses the segment protection facility, a processor enhancement that uses microcode assist, to provide segment protection. The segment protection facility prevents changing the contents of protected shared segments. This feature eliminates CP monitoring of shared segment pages.

Reference

The *VM/SP Administration* book, SC24-5285, describes VM/SP performance options.

The *VM/SP Planning Guide and Reference*, SC19-6201, describes system generation of performance options.

The *VM Running Guest Operating Systems* book, GC19-6212, describes the basic process of running your own operating system.

Diagnosing System Programming Problems

OVERVIEW

The information on this topic is primarily of interest to system programmers.

For example:

- System programming problems can usually be classified as one of the four general types.
- Problem debugging is most effective when a systematic step-by-step approach is used.
- VM/SP offers many tools to help diagnose system programming problems.

VM/SP system programming problems are generally four types: (1) program loops, (2) wait state, (3) abnormal program termination (abend), and (4) incorrect results.

Problem debugging consists of four basic steps:

1. Recognizing that the problem exists
2. Identifying the type of problem and the area of the system affected
3. Collecting and analyzing data, then sorting out the data that applies to the problem
4. Finding and fixing the problem.

VM/SP commands let the system programmer dump storage data, display data, and store data to find and fix programming problems.

Reference

The *VM/SP Diagnosis Guide*, LY24-5241, describes techniques, procedures, and VM tools for locating and fixing system programming problems.



Chapter 4. A Closer Look at CMS

As pointed out earlier in this book, any one of many operating systems controls work flow in a virtual machine under VM/SP. CMS is the most versatile and most widely used operating system. It runs only in a virtual machine under CP.

This chapter discusses some of the more commonly used CMS facilities. The information in this chapter prepares you to use some of the other books of the VM/SP library, such as:

- *The VM/SP CMS Primer*
- *The VM/SP CMS User's Guide*
- *The VM/SP CMS Command Reference*
- *The VM/SP CMS Shared File System Administration book*
- *The VM/SP System Product Editor User's Guide*
- *The VM/SP System Product Interpreter User's Guide*
- *The VM/SP Application Development Guide for CMS*
- *The VM/SP Application Development Reference for CMS*
- *The Programmer's Guide to the Server-Requester Programming Interface for VM/SP.*

This chapter is not intended to be used as a reference for *using* CMS.

Communicating with CMS

OVERVIEW

Communication between you and CMS manages work flow in your virtual machine. To handle this task CMS supports:

- A set of user commands
- Subcommands and macro instructions entered from secondary environments under CMS
- Messages that prompt, reply, convey information, or show an error condition
- A set of macro instructions and functions for programs written in assembler language
- Command language processor control statements
- Commands that run only in the CMS/DOS environment
- Programs that compile high-level language programs.

Initializing CMS

The first step of a user session is usually loading (IPLing) CMS into your virtual machine. During the IPL CMS process, commands that help define the CMS virtual machine environment are automatically passed to the system. These commands are usually included as statements in an exec procedure named PROFILE EXEC. This exec must reside on file mode A (your top directory in the default file pool, or, if a default file pool is not defined for you in the system directory, your minidisk at virtual address 191). CMS runs PROFILE EXEC, if it exists, as part of the IPL process.

This procedure automatically enters commands that would otherwise be manually entered at the start of each session. These commands do the following:

- Describe the virtual console
- Define spooling for virtual unit record devices
- Set up macro and text libraries
- Access SFS directories
- Access permanent minidisks
- Set up links to other virtual machines
- Set up program function key operations.

CMS also has a System Profile feature. The System Profile feature provides an easy method for installations to define your CMS environment. It lets system administrators tailor environments for users at initialization time. It is a versatile tool that accomplishes anything you want done from within an exec.

The environment is tailored by modifying the SYSPROF EXEC. A default SYSPROF EXEC is provided in VM/SP. It:

- Displays the CMS system ID

- Issues the initial console read
- Processes parameters passed to it through the IPL command
- Accesses either your top directory in the default file pool (if defined) or the 191 minidisk as file mode A; accesses the 192 minidisk (if defined) as file mode D
- Issues initialization-related messages
- Runs your PROFILE EXEC (if one exists).

The term *CMS virtual machine* means a virtual machine on which CMS is the operating system. Chapter 2 discussed many ways of using a CMS virtual machine. Regardless of how you use a CMS virtual machine, actions are started by entering commands to CMS. For example:

- Use the System Product Editor (XEDIT command) to create a CMS file named INCOME DIVIDEND by entering:
xedit income dividend
- Have CMS display the primary virtual direct access storage (file mode A directory or minidisk) status by issuing:
query disk a
- Read a file from the virtual reader to virtual direct access storage by entering:
receive

To get a good idea of all the things that you can ask CMS, look through the *VM/SP CMS Command Reference*. Figure 21 on page 82 shows two CMS commands. One user is beginning an editing session to make changes to a file named BIGFILE. The other user is displaying a list of the files in the directory or on the minidisk that has been accessed as file mode A. (If file mode A is a directory, the list might also contain the names of subdirectories.) The second user then uses CMS commands to begin many actions related to using and managing the displayed files.

Entering commands to CMS is much like telling another person in his or her native language what is to be done. If you make a mistake or CMS needs more information, CMS displays appropriate messages. CMS messages convey many types of information, such as user errors, completion of events, and response to user queries. CMS messages also prompt for input information needed to complete an action.

If you are not sure how to request an action, entering:

```
help
```

asks the CMS HELP facility to help. Another topic later in this chapter describes the CMS HELP facility.

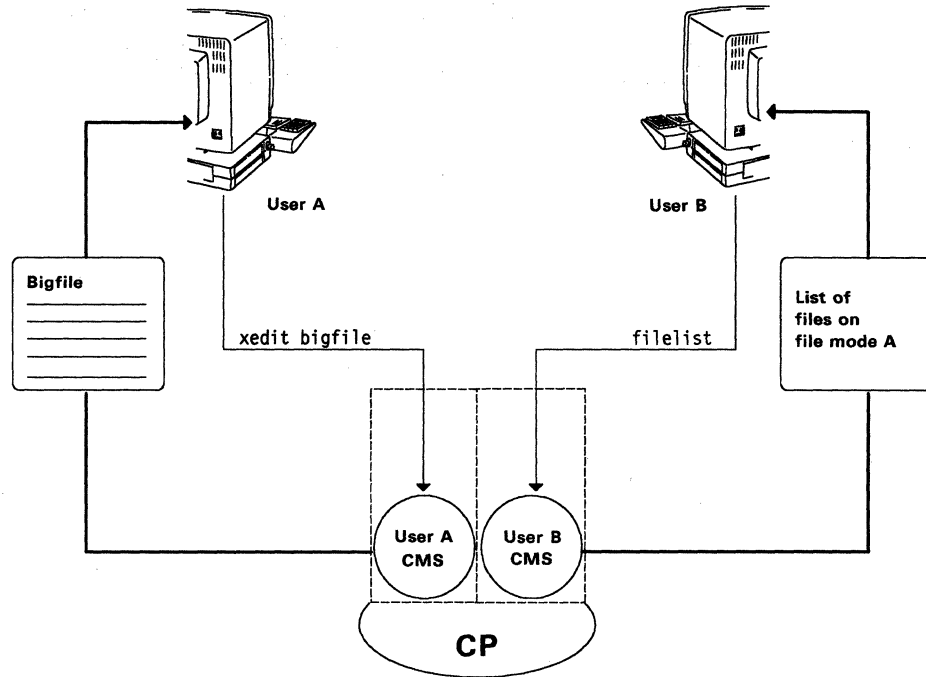


Figure 21. User and CMS Communication

Work Flow Management through Commands and Messages

Subcommand Environments under CMS

Special-purpose facilities that support their own sets of commands can be executed under CMS. The System Product Editor (XEDIT) is an example of a subcommand environment. CMS does not recognize subcommands of these environments until you enter the CMS command to cause the virtual machine to enter that environment. However, once a virtual machine is in a subcommand environment, you can still enter regular CMS, or even CP, commands. For example, before a virtual machine enters the System Product Editor environment, the UP command (an XEDIT subcommand) has no meaning to CMS. However, in the System Product Editor environment, CMS continues to respond to the QUERY READER (CMS) command and the UP (XEDIT) subcommand.

Most CMS commands are documented in the *VM/SP CMS Command Reference*. CMS subcommands are documented, by facility, in separate sections of that book, or in separate documentation describing the facility. Some special cases, such as the command to execute the programmable operator facility (PROP), are documented in the *VM System Facilities for Programming* book.

Procedures for Automated Command Execution

Topics earlier in this book mention how a CMS virtual machine environment can be governed by making a PROFILE EXEC available to CMS during the IPL process. During a session, other sequences of commands in exec procedures are communicated to CMS by simply entering the file name of the procedure. For example, an exec procedure named SHIPFILE EXEC is run by simply entering the file name:

```
shipfile
```

CMS includes the System Product Interpreter command language processor for performing automated procedures. A topic later in this chapter describes the System Product Interpreter and gives references to related information sources.

Reference

The *VM/SP CMS User's Guide*, SC19-6210, describes communication between users and CMS.

The *VM/SP CMS Command Reference*, SC19-6209, describes CMS commands and subcommands.

The *VM/SP Application Development Reference for CMS*, SC24-5284, describes macro instruction and functions.

The *VM/SP System Product Interpreter Reference*, SC24-5239, describes the command language processor for CMS.

CMS from the User's Point of View

OVERVIEW

CMS is a general purpose operating system that does many jobs. From a wide perspective these jobs are:

- Plan, install, and maintain the VM/SP system
- Write, debug, and run programs
- Produce and edit data files
- Communicate with other system users
- Convert CMS messages in the user's national language (for those languages supported by VM/SP).

CMS helps different users in different ways. Its facilities are so diverse that most users never need to use all of them. However, the more you are aware of the capabilities of CMS, the more potentially productive you become.

How the General User Sees CMS

A general user of VM/SP is a person to whom CMS is simply a tool for getting a job done. Most general users are concerned about the results, not how the system produces those results. This person's view of the system usually includes only a terminal, the data displayed at that terminal, and output such as listings that are printed and delivered as expected.

CMS is quite accommodating to the general user with limited experience. People who know little or nothing about data processing or computers quickly learn to use CMS. Most user errors in communicating with CMS cause no serious problems. Displayed messages help you get back on the right track. When not sure about the format of a command, you can request that CMS display a HELP panel to explain that command. You can communicate with CMS in your national language if VM/SP is installed with that support.

To the general user, CMS means being able to do many tedious and time-consuming tasks more easily and efficiently. These tasks include such things as:

- Writing letters and reports
- Writing and debugging application programs
- Performing application programs
- Communicating with other system users.

How Other Users See CMS

Specialized VM/SP users include the:

- System analyst and System programmer

People responsible for productivity within their organization see CMS as a tool for increasing efficiency. For example, programmers write, test, and run programs far more efficiently using CMS facilities than using traditional hands-on methods. Developing programs online makes punched card source decks unnecessary. Rather than waiting for batch processing results,

programmers test programs in their virtual machines. They correct errors interactively from a terminal and immediately retest the program. From the programmer's point of view, this makes programming more enjoyable because it means quick turnaround, better debugging facilities, and less frustration.

- **System operator and System administrator**

The system operator uses CMS commands while doing system management tasks. For example, if the operator needs to make changes to VM/SP system files, the System Product Editor will be used the same way a general user might use it for making changes to an application program. The system operator also uses CMS facilities such as those issued by the CMSBATCH, FORMAT, LISTDS, and PRINT commands. These people are responsible for providing a system that is as useful and efficient as possible. CMS is an important part in meeting those goals and responsibilities.

Reference

The *VM/SP CMS Primer*, SC24-5236, helps a new user get started using CMS.

The *VM/SP CMS User's Guide*, SC19-6210, is the most complete information source for users to learn about CMS.

Command Language Processors under CMS

OVERVIEW

System Product Interpreter is the VM/SP command language (exec) processor for procedures written in the Restructured Extended Executor (REXX) language.

Exec procedures range from a single statement to complex programmed sequences of statements. (CP or CMS commands are run from exec procedures.) These procedures will greatly simplify communication between you and VM/SP.

New users sometimes hesitate to get involved with exec procedures. Maybe it is the name *exec* that frightens them. Perhaps they have seen some of the complex procedures developed by experts. In any case, there is really nothing to be frightened of. Exec facilities are easy to use and simplify computer use. Exec procedures save time, especially for sequences of commands and other statements entered on a regular basis.

Exec Procedures Are Sequences of Commands

An exec procedure is a sequence of commands and other statements that are processed by CMS. Besides CP or CMS commands, exec procedures have statements to provide functions such as the testing of variables and conditional branching. The System Product Editor produces exec procedures. Although certain format and syntax rules apply, the process is the same as for producing other types of CMS files.

Depending on the complexity of the procedure, many control statements that affect performance can be included. For example, they:

- Perform a portion of the procedure only if certain conditions exist
- Issue informational messages or messages that request input
- Substitute variables supplied by the user into statements in the exec procedure.

An exec can be of any length, from a single command to a complex set of routines. An exec file can have any file name, but some file names (PROFILE, for example) are reserved for a special purpose. Usually, the file type of the procedure must be EXEC.

Once an exec procedure is stored on virtual direct access storage, the procedure is run by simply entering the name of the exec file, in the same manner as CP or CMS commands. Thus, each exec procedure becomes, in effect, a new command. For example, a procedure whose file name and file type identifiers are SHIP EXEC is executed by simply entering:

```
ship
```

If the name of an exec file is already the name of an existing CP or CMS command, the system usually carries out the exec procedure instead of the system command procedure having the same name. This is controlled with the IMPEX option of the SET command.

Figure 22 shows a user communicating a sequence of commands to CP by simply entering *list*, the name of the procedure having that sequence.

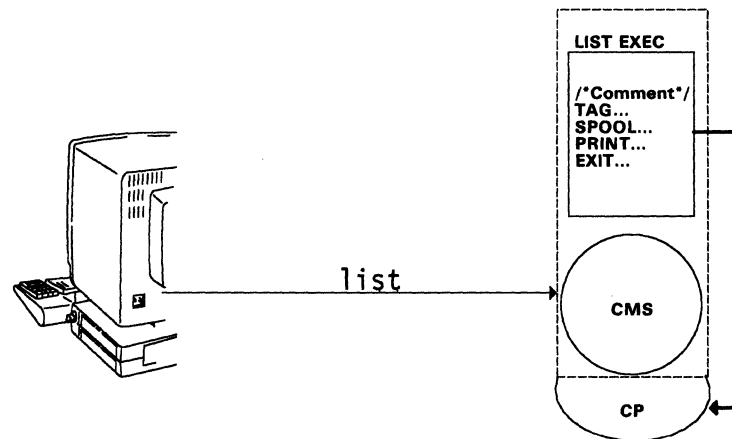


Figure 22. Exec Procedures Simplify User-to-System Communication.

Exec Procedure Language and Processor in CMS

Just as any program is written in a language supported by a processor for that language, exec procedures for CMS are written in the Restructured Extended Executor (REXX) language. REXX statements are processed by the System Product Interpreter command language processor in CMS.

Anyone can use REXX, from a novice general user to an experienced programmer. For example, REXX statements do such things as:

- Perform conditional loops
- Assign values to variables
- Do if ... then ... else conditional performance
- Do arithmetic operations
- Do character conversion and translation
- Compare values.

Usually, REXX statements are written in simple English words and phrases. For example, to display a request for a user to enter the current date, the procedure statement is:

```
say "please enter today's date."
```

Or, to give the value of 40 to a variable named *rate*, the procedure statement is:

```
rate = 40
```

Reference

The *VM/SP CMS Primer*, SC24-5236, provides information to help users learn to write exec procedures.

The *VM/SP CMS User's Guide*, SC19-6210, provides detailed information about creating exec procedures to use with CMS. System Product Interpreter, EXEC, and EXEC2 are discussed and differences are explained.

The *VM/SP System Product Interpreter User's Guide*, SC24-5238, describes and gives examples for using the REXX language.

VM/SP System Product Editor

OVERVIEW

The System Product Editor is used to create and modify CMS files. System macros and user-written procedures are performed from the System Product Editor environment.

The AUTOSAVE facility protects data from being accidentally lost when an editor session ends abnormally.

In this discussion, the System Product Editor is referred to as *the editor*. The editor is a full-screen editing facility that runs under CMS. *Full-screen* allows you to work with a full video screen of file data displayed at a terminal. This is a considerable advantage over editing programs that work with one data line at a time.

Traditionally, the only functions expected of a system editor were those of file creation and updating. The VM/SP editor has many facilities to make these tasks easy and efficient. For example, it has a power input (POWERINP subcommand) facility so you can key in text as though it were one continuous line. The editor takes care of shifting to new input lines and reformatting words that get split between the end of one line and the start of the next line. It is like typing without ever having to do a carriage return.

A summary of the tasks that the editor helps you do are:

- Create CMS files
- Make changes to existing files, including simultaneous editing of different files or multiple sections of the same file
- Combine existing files or portions of files
- Place file lines in the console stack
- Sort data in all or part of a file
- Write and use macro procedures
- Ask for help in using the editor
- Search files for specific data
- Redefine the editor environment.

Working in the Editor Environment (XEDIT)

You begin an editing session by entering the XEDIT command under CMS. While working in the editor environment, you create new CMS files or make changes to existing CMS files. For example, to create a CMS file named BICYCLES TOURING (or make changes to an existing file of the same name), you begin by entering:

```
xedit bicycles touring
```

The editor then displays a working copy of the requested file on your terminal screen. From that point, the display screen is like an electronic notebook for making additions or changes.

The editor takes advantage of CMS Session Services. (See "CMS Session Services" on page 102 for a description.) You have the option of specifying what window XEDIT should use to display a file; the default window is XEDIT.

Editor (XEDIT) subcommands have many ways to manage a file on a display screen while making changes. The SET subcommand has many options for managing the editor environment. Users perform, from the editor environment, system macro routines executed by XEDIT subcommands. For example, all or part of a file can be sorted (SORT subcommand), or file lines placed on the console stack (STACK subcommand). Procedures, written in the Restructured Extended Executor (REXX) language, run from the editor environment by entering the procedure name.

You are able to start many editor functions simply by pressing program function keys. Program function (PF) keys are automatically set to do useful functions in the editor environment. You override these settings with the SET PF subcommand, either in a file called PROFILE XEDIT, or by manually entering it during an editor session. For example, PF10 is set to display the current tab column numbers by entering:

```
set pf10 query tabs
```

Thereafter (until the end of that XEDIT session), pressing PF10 causes XEDIT to display the tab column numbers defined by the SET TABS subcommand. By pressing the CLEAR key, the editor session is then resumed.

For descriptions and examples of the kinds of things that can be done during an editing session, see "Reference" at the end of this discussion. To end an editing session and save your changes, enter the FILE subcommand. This causes the working file copy to become a permanent copy on file mode A.

Preventing Data Loss While Using the Editor

Accidentally losing large amounts of work during an editing session is prevented by periodically entering the SAVE subcommand. Entering this command protects your data against a system failure, and you can be sure your changes are not lost.

Users request the editor to automatically save a copy of the current working file at specified intervals during an editing session. This is done by using the AUTOSAVE option of the SET subcommand. For example, entering:

```
set autosave 10
```

causes your file to be written to disk automatically, after you have typed in or changed 10 lines. The editor continues to update each time a combined total of 10 lines are added, changed, or deleted in the working file. Your permanent copy of that file is unaffected by the AUTOSAVE option. In the example just given, the most you might lose are the last 10 changes made to the file.

The editor helps you do many things besides simple file creation and editing. Most VM/SP users occasionally use the editor. This book does not discuss all the editing facilities because the editor has separate and complete documentation in the VM/SP library.

Reference

The *VM/SP CMS Primer*, SC24-5236, instructs a new user on how to use the editor.

The *VM/SP System Product Editor User's Guide*, SC24-5220, describes editor facilities and gives examples for their use.

The *VM/SP System Product Editor Command and Macro Reference*, SC24-5221, describes the format and options for each editor subcommand.

The *VM/SP CMS User's Guide*, SC19-6210, describes CMS Session Services.

CMS File System

OVERVIEW

A file is the working unit of information in the CMS environment. It is made up of records of specified length and format. CMS files are:

- Stored on minidisk space (in file pools or on individual minidisks) that have been formatted specifically for CMS files
- Written or read by CMS only
- Identified by file name, file type, and file mode
- Shared with other users.

How CMS Files Are Structured

Most VM/SP users' view of the CMS file system is limited to CMS files displayed at their terminal. Because CMS handles file management, usually based on default format options, you usually do not need to be concerned about how CMS stores files on its virtual direct access storage. However, many CMS users eventually must do tasks that require some knowledge of the CMS file system.

Generally, CMS stores each displayed data line as one record in the user's virtual direct access storage. CMS does some tailoring of these lines, depending on the record specifications in effect at the time the record is stored. These specifications result from default values or from overriding your input.

CMS records are either fixed length or variable length.

- **Fixed-length records**

This means that data is stored in fixed-length units. Record length varies from file-to-file, but all records (lines) within a given file have the same length. CMS fills any positions of the record following the last significant character with blank characters. You can specify the record length, but default values cover most situations. Storing data in fixed-length records improves processing speed over variable-length records, but you cannot use direct access storage efficiently if a significant number of blank characters are needed to fill out the records.

- **Variable-length records**

Variable-length records are any length up to a maximum default value or user-specified value. CMS stores only the positions up to the last significant character. Even though overhead characters, such as line sequence numbers, take additional space, this usually results in more efficient use of direct access storage space than fixed-length records. However, variable-length record files will need more processing time.

The differences between fixed-length and variable-length records are shown in Figure 23.⁶ The characters at the end of each line in the fixed-length format file have been inserted by the system so that each record has a total of 32 characters (the specified record length).

⁶ Displayed text from *Lucy*, by William Wordsworth, 1770-1850.

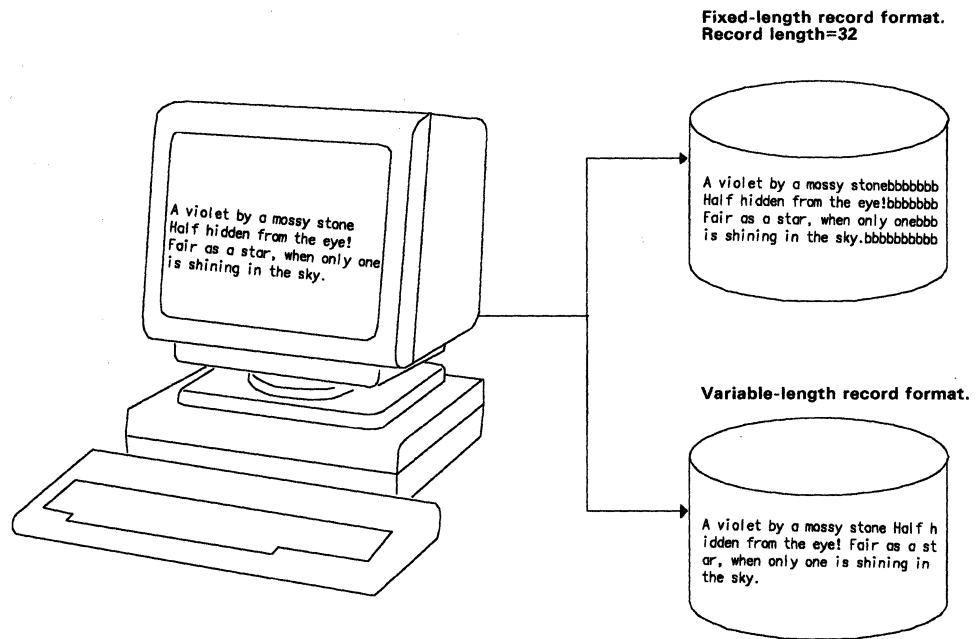


Figure 23. Fixed- and Variable-Length Records

It is possible for certain records to be stored more efficiently as fixed-length files than as variable-length files. This is the case when all positions of most input lines are filled with significant characters (assembler programs for example).

Factors such as file type (LISTING, EXEC, and so forth) affect whether a file should be fixed or variable-length, and what the length should be. The System Product Editor supplies default formatting values, depending on the type of file being processed. For example, the record format and logical record length values supplied for a COBOL file type are:

RECFM = F (fixed)

LRECL = 80

For an exec file these values are:

RECFM = V (variable)

LRECL = 130

These values for a file are changed during an editing session using the SET subcommand.

CMS files can reside in file pools or on individual minidisks. File pools are collections of minidisks which the data from a single file might be distributed among several minidisks. All minidisk space used to store CMS files must be formatted for CMS files. Minidisk space is subdivided into blocks. A block can contain a variable number of records, depending on the block size and the record length used. The block size of the minidisks in a file pool is 4K.

The maximum number of records that can reside on a minidisk depends on several factors, such as the type of direct access volume being used and the size of the records being stored. However, the number of records on a minidisk, or the number of 4K blocks in a file pool, cannot exceed $2^{31}-1$. This is the largest number that can be expressed in binary by the one-word (32-bit) value used by the CMS file system.

How CMS Files Are Identified

The name of every CMS file consists of three identifiers:

File name. Distinguishes one CMS file from another.

File type. Groups CMS files according to some common characteristic. This is very helpful when using CMS facilities, such as the LISTFILE command, which lets you selectively display only those files with common identifiers.

File mode. A letter (A-Z) that tells CMS where the file resides. This letter identifies an accessed SFS directory or minidisk.

CMS assigns file identifiers according to operands (or default values) for the command used to begin the file creation. Initial file identifiers are assigned when entering the XEDIT command to run the System Product Editor. For example, entering:

```
xedit bigfile taxes a
```

causes CMS to assign the newly created file these identifiers:

Filename = BIGFILE

Filetype = TAXES

Filemode = A

File identifiers are changed by using the RENAME command under CMS. They are changed during an editing session using the SET (FNAME, FTYPE, or FMODE) subcommand.

File names and file types are one-to-eight characters long, having the characters A-Z, 0-9, \$, #, @, +, - (hyphen), : (colon), or _ (underscore). Some reserved file types have special meanings to VM/SP and must be used accordingly. They are listed and explained in the *VM/SP CMS User's Guide*.

Typical VM/SP users often find that over time they accumulate dozens or even hundreds of files on virtual direct access storage. To help recall the purpose of these files from their names, it is a good idea to assign significant file identifiers. For example, consider a file named DEDUCT TAX A1. The file name suggests that this file has information about deductions. The file type is a way of grouping this file with others related to taxes. The two characters (A1) of the file mode show:

Character	Meaning
(A)	The file resides in the SFS directory or on the minidisk that you have accessed as A.
(1)	This file is both read from and written to. (This is the default file mode.)

Each CMS file in the same SFS directory or on the same minidisk must have a unique combination of file name and file type. Often the file type, and usually the file mode, need not be specified when accessing the file. CMS supplies default values, depending on the working environment at the time.

How CMS Files Are Shared Among Users

The CMS Shared File System (SFS) lets you share files or directories with other users. The next section of this chapter describes SFS.

Entire minidisks (as opposed to individual files) are shared with other users by LINK and ACCESS commands. For more information about sharing minidisks, see the *VM/SP CMS User's Guide*.

An IBM program offering, VM File Storage Facility, lets you share minidisk data files with other users. In this non-SFS environment, files or groups of files are given passwords that provide security ranging from *anyone can access* to *available to the owner only*. Passwords give you read access only or read/write access. Only one user has write access to a file at one time. However, more than one user can have read access at one time. Therefore, more than one user can read a file while another user has write access to that file.

The VM File Storage Facility has commands that let you:

- Store and retrieve files
- Rename and erase files
- Send files to other users
- Find out what files they own, the size and number of records in each file, the date and by whom each file was last updated, who can share each file, and other file status information
- Change passwords.

Reference

The *VM/SP CMS User's Guide*, SC19-6210, describes the CMS file system, together with special considerations for using it.

The *Virtual Machine/File Storage Facility Description/Operations* book, SB21-3085, describes how to install and use the VM File Storage Facility.

CMS Shared File System

OVERVIEW

The Shared File System (SFS) is an extension of the CMS file system that offers you additional file management and file sharing functions. In SFS:

- CMS files are stored in file pools.
- A user can be given an amount of file space in a file pool.
- The files in a file space are organized in directories.
- A file can be placed in more than one directory.
- Users can grant each other read authority or write authority on files or directories.
- Multiple users can have concurrent access to the same file or directory.
- Locks on files and directories ensure data integrity among multiple users.
- You can share files and directories with users in other systems.

File Pools and File Space

A *file pool* is a collection of minidisks assigned to a *file pool server* virtual machine. A *server* is a set of programs that execute in a virtual machine to manage a resource, such as a file pool. A file pool provides virtual direct access storage that many users can share. A single VM/SP system can have many file pools.

An SFS administrator manages the resources of a file pool and enrolls users. The administrator gives an enrolled user exclusive access to an amount of *file space* in the file pool. In this file space, you can create, manipulate, and store CMS files. You can only have one file space in a file pool, but you can be enrolled in more than one file pool. A default file pool can be defined for a user either by including the name of the file pool on the IPL statement in the user's entry in the system directory or by adding a SET FILEPOOL command to the user's PROFILE EXEC.

CMS files stored in a file space do not appear any different from CMS files stored on a minidisk. You can have access to a file space and minidisks and move files from one to the other. While doing their work, SFS users are unaware of the minidisks that form the file pool.

Organizing Files in SFS Directories

SFS lets you organize files in *directories*. When an SFS administrator enrolls a user in a file pool, SFS automatically defines a *top directory* for that user. Under the top directory you can use the CREATE DIRECTORY command to create up to eight levels of subdirectories in a hierarchy or *tree* structure. If you are given a file space by an administrator of the file pool, you can create and store CMS files at any level in the hierarchy.

The name of a user's top directory is the same as the user's user ID⁷. Each subdirectory under the top directory can have a name up to 16 characters long. A specific directory in the hierarchy can be identified by using a *dirname*. The format of the *dirname* looks like this:

filepoolid:userid.dir1.dir2.dir3.dir4.dir5.dir6.dir7.dir8

where *filepoolid* is the name of the file pool, *userid* is the name of the top directory, and *dir1* through *dir8* are the name of the subdirectories in that *branch* of the hierarchy. The name of the file pool must be followed by a colon; the name of the top directory must be followed by a period; and the names of subdirectories must be separated by periods.

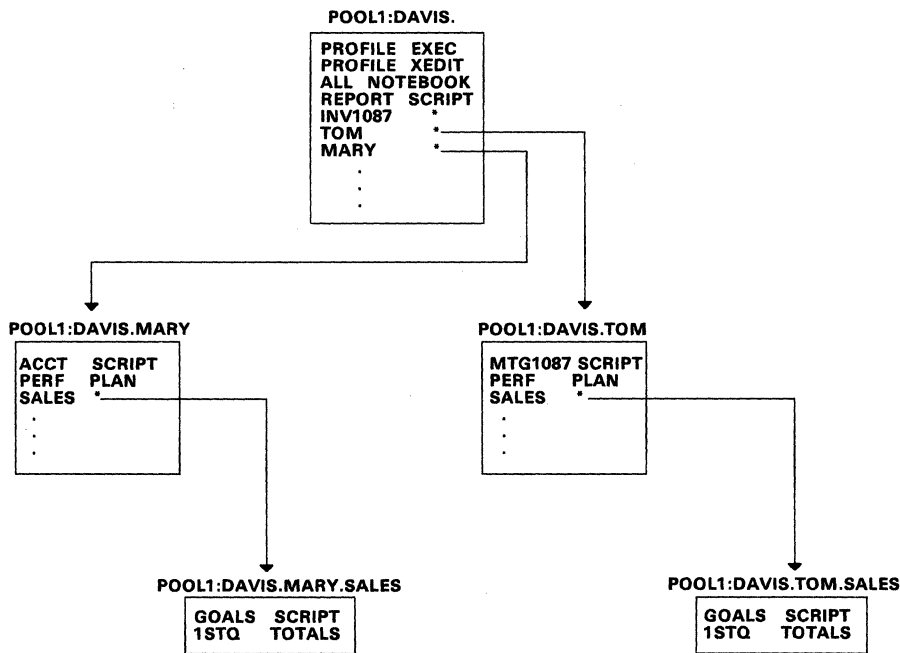


Figure 24. Example of a Directory Hierarchy

Figure 24 shows an example of a directory hierarchy. In this example, the file pool is named POOL1. The *dirname* of the top directory of user DAVIS is:

POOL1:DAVIS.

TOM and MARY are the names of subdirectories in the DAVIS top directory, and each contains a subdirectory named SALES. The following *dirname*s are used to identify them:

POOL1:DAVIS.TOM.SALES

POOL1:DAVIS.MARY.SALES

⁷ The name of your top directory in a file pool is the same as your user ID in the Transparent Services Access Facility (TSAF) collection in which the file pool is located. Each user ID in a TSAF collection must be unique. If the file pool is located in a different TSAF collection from the user's virtual machine, the user's top directory name (userid) in the file pool could be just a user identification without a corresponding virtual machine.

When a file is stored in a directory, the *dirname* becomes part of the identification of the file, as follows:

filename filetype dirname

Accessing Directories

Not all CMS commands accept *dirname*s when specifying a file. Even if they did, it would be quite tedious to have to enter the full *dirname* all the time. So SFS lets you access directories through CMS with the ACCESS command, in the same manner as minidisks. An accessed directory or minidisk is assigned a unique file mode letter (A-Z) in the user's virtual machine. The file mode letter establishes the order that CMS searches through the accessed directories and minidisks when looking for data. If a default file pool is defined on the IPL statement in a user's entry in the system directory, and if the user is enrolled in that file pool, then the user's top directory in the file pool is automatically accessed as file mode A when the user logs on. Otherwise, the user's 191 minidisk is accessed as file mode A.

You can get the *big picture* of the directory hierarchy with the DIRLIST command. This command lists the *dirname*s of all the directories, whether or not they are accessed, in a full-screen display. The DIRLIST display can serve as a starting point to browse through the contents of the listed directories. By moving the cursor and pressing a PF key, you can get a FILELIST display of any directory in the DIRLIST. The FILELIST display shows the names of all the files and directories listed in that directory. By repeating the FILELIST procedure, you can navigate through the hierarchy and display the contents at any level.

Creating Aliases

An SFS directory is simply a list of the names of CMS files and SFS subdirectories. File names and directory names are not written in the user's file space. SFS records them in a part of the file pool called the *catalog* that keeps track of the files and directories that exist in the file pool, who owns them, who is authorized to look at them, and so on.

When a user who has a file space *stores* a file in a directory, only the name of the file and certain file attributes are written in the directory. The contents of the file are written in the user's file space, which is allocated from the user storage area of the file pool. The file name and the directory name in the *catalog* *point* to the data residing in the file space.

SFS allows multiple pointers to a file. These pointers are like additional images of the file. Additional images of the file can appear in other directories, and even in the same directory. The original file is called a *base file*, whether it resides in a top directory or in a subdirectory. Additional images of a base file, called *aliases*, are created with the CREATE ALIAS command. Aliases can appear in directories owned by the user who owns the base file, and they can appear in directories owned by other users. A user, enrolled in a file pool, does not need file space to create aliases. An alias can have the same name as the base file if it appears in a different directory. An alias in the same directory as the base file must have a different name.

Aliases have complete file IDs and behave just like base files. Aliases can be entered in commands. Aliases appear in FILELIST and other file displays. Wherever the name of a file must be specified, an alias could be used. Because aliases are really pointers, they are written in the *catalog*; they do not occupy additional user storage. However, a base file and all of its aliases represent (point to) the same file data; a change to any one is reflected in the others.

Sharing Files and Directories

Unlike sharing minidisks, where one user links to another user's minidisk, sharing information in SFS does not involve linking to another user's file space. Instead, SFS users grant each other *read authority* or *write authority* on files, sets of files, and directories with the GRANT AUTHORITY command. Write authority includes read authority. Granting authority on a base file also grants that authority on all aliases of the file. Because authority is always associated with the base file data, granting authority on an alias is the same as granting that authority on the base file.

A user who is enrolled in a file pool can create a personal directory structure. The user *owns* these directories and all of the base files stored in them. You automatically have the authority to read from and write to all owned files and directories. The owner can grant read authority or write authority on these files and directories to another user, to a group of users, or to all users. The owner can also revoke this authority.

The following authorities exist in SFS:

- **Read authority on a file**

An authorized user can read, but not change, the contents of the file. You can browse through the file, print it, copy it, create aliases of it, and do anything else that does not change the file.

- **Write authority on a file**

An authorized user can read the file and modify the contents using any command or program. If you also have write authority on the directory that the file resides, you can rename or erase the file.

- **Read authority on a directory**

An authorized user can read the list of names in the directory. You cannot read the contents of any listed file or subdirectory unless also granted read or write authority on that object. You see only the names and attributes of the objects at the authorized level of the hierarchy; objects at other levels are not visible.

- **Write authority on a directory**

An authorized user can read the list of names in the directory and create base files and aliases in it. You automatically have write authority on any base file or alias that you create. However, this authority can be revoked by the owner of the directory. You cannot read or modify any file or subdirectory already listed unless also granted read or write authority on that object. You can rename an alias or erase it from the directory if you have read authority on the base file (wherever that base file is located). You can rename a base file or erase it from the directory if you have write authority on the base file. However, only the owner of the directory (or an administrator) can rename or delete the directory, or create, rename, or delete subdirectories.

Only the owner of an object can grant authority on that object⁸. A user who has been granted authority on an object cannot *pass on* that authority to another user. Authority on a directory does not imply authority on any file or alias within that

⁸ Administrator authority includes *quasi-ownership* of every object in the file pool. An administrator can do anything that an owner can do. For more information about administrator authority, see *VM/SP CMS Shared File System Administration* book.

directory. Authority on a file or alias does not imply authority on the directory that the file or alias resides.

There are three ways that SFS users can use these authorities to share file data:

- **Create an alias of a base file**

User1 grants user2 read or write authority on a base file. User2 creates an alias of the file in one of user2's own directories. User2's authority on the alias is the same as the authority that user1 granted on the base file.

This method is most appropriate when:

- Individual users want to share only a few files
- Multiple users have write authority
- Sharing users want to have different names for the files or group them differently.

- **Access a directory**

User1 grants user2 read or write authority on a base file and read or write authority on the directory in which the file resides. User2 accesses user1's directory with the ACCESS command. User2 can either manipulate the file in user1's directory or create an alias of the file in user2's directory. If user2 has write authority on the file, user2 can use the System Product Editor (XEDIT command) to edit the file in user1's directory even if user2 has only read authority on the directory.

This method is most appropriate when multiple users want to:

- Readshare an entire directory and all of its files (such as a tools library)
- Write items in a directory (such as a bulletin board)
- Edit files without creating permanent aliases.

- **Read or modify a file directly through a program**

User1 grants user2 read or write authority on a base file. User2 runs a program that uses the granted authority to read or modify the file. The directory does not have to be accessed, and authority on the directory is not required.

This method lets application programs use SFS files. Write mode requires SFS program functions, which are available as callable services library (CSL) routines. For more information about CSL routines, see the *VM/SP Application Development Reference for CMS* book.

These SFS authorities and files sharing methods allow multiple users to access files and directories simultaneously. Many users can read a file while another user is updating the file. The version of the file that the readers see is not disturbed. After the writer commits the changes, any reader who opens the file sees the updated version. If more than one user wants to update a file, SFS automatically prevents simultaneous updates through a system of locks.

Locking Files and Directories

Because many users can have read and write authority on SFS files and directories, SFS provides a way to *lock* objects to prevent simultaneous updates. When a user is actively working with SFS files and directories, SFS automatically creates and deletes locks (called implicit locks). You can also manually create a lock (called an explicit lock) on an object with the CREATE LOCK command. The object can be a

file, a group of files, or a directory. You can select the type of lock and the duration.

There are three types of explicit locks:

- share lock** All authorized users can read the contents, but no one can make any changes.
- update lock** All authorized users can read the contents, but only the user who created the lock can make changes.
- exclusive lock** Only the user who created the lock can read the contents and make changes.

There are two lock durations:

- session lock** The lock lasts until the end of the CMS session, or until the connection with the server is broken, unless the user who created the lock specifically deletes it.
- lasting lock** The lock lasts until the user who created the lock specifically deletes it.

Locking an alias is the same as locking the base file; locking a base file locks all aliases of that base file. An SFS administrator has the authority to delete locks created by other users.

Accessing Data in File Pools in Other Systems

Communication between users' virtual machines and the SFS server virtual machine is through the Advanced Program-to-Program Communication/Virtual Machine (APPC/VM) application program interface. The Transparent Services Access Facility (TSAF), which provides access to server resources within a VM processor or across VM processors, also uses APPC/VM. Therefore, file pools in different VM systems can be resources in the same TSAF collection. With APPC/VM VTAM Support (AVS), you can access file pools in other TSAF collections. A user in one system can access data in a file pool in another system under any one of the following conditions:

- The user is enrolled in the file pool in the other system and specifically authorized to access the data.
- The user is enrolled in the file pool in the other system, and authority on the data has been granted (through the PUBLIC key word) to all users who can connect to the file pool.
- The ENROLL PUBLIC command has been issued for that file pool. Then any user who can access the other system can access the data. The user does not have to be enrolled in the file pool.

There is more information about APPC/VM and TSAF later in this book under "VM/SP Connectivity".

Data Security

SFS includes the following features to protect data security:

- To access a file pool, a user must be enrolled by the SFS administrator, or PUBLIC must be enrolled.
- A user who is given file space in a file pool is the only user who can create files in that file space, unless the user specifically grants that authority to other users.

- Users control access to their own files and directories by granting read authority or write authority to other users.
- Locks on files and directories ensure data integrity among multiple users.

External security programs can also be used to supplement or replace the SFS security functions.

Reference

The *VM/SP CMS User's Guide*, SC19-6210, describes how to use the CMS Shared File System.

The *VM/SP CMS Command Reference*, SC19-6209, describes the general user CMS commands.

The *VM/SP Application Development Guide for CMS* book, SC24-5286, describes how to develop programs that use the SFS program functions.

The *VM/SP Application Development Reference for CMS*, SC24-5284, describes the SFS program functions.

The *VM/SP CMS Shared File System Administration Guide*, SC24-5367, contains information about the SFS server virtual machine and describes the commands and functions available to the SFS administrator.

CMS Session Services

OVERVIEW

CMS Session Services improve the usability of VM/SP on 3270 display terminals. It includes:

- Window functions for the end user
- Full-screen environment for CMS
- CONSOLE macro for applications using 3270 I/O
- Updates to the System Product Editor (XEDIT).

Window Functions

CMS Session Services let you handle information in much the same way that pieces of paper are rearranged on a desk top. You display information, recall previous information, and work without unexpected interruptions from the system.

In addition, you scroll through data displayed in windows. Data is also logged into a CMS file for future reference.

When using full-screen CMS, you will find it helpful to view messages through a MESSAGE window without leaving your current environment. Other windows, such as the STATUS and WARNING windows, display important system information.

More experienced users can tailor the default virtual screens and windows of the system. Data can be written to virtual screens and viewed through windows. Windows are positioned anywhere on the physical screen, displayed on top of each other, and overlapped.

Virtual Screens

In CMS Session Services, virtual screens maintain the information displayed in windows. A virtual screen is a functional simulation of a physical display screen. The characteristics of the virtual screen are defined by you or the program, regardless of the physical attributes of the hardware device. A program that displays data, such as the System Product Editor (XEDIT), does not have to know the device features or the size of the physical screen.

Full-Screen CMS

Full-screen capability for CMS is given by Session Services. It is optional for 3270-type terminals. Users can request to run in full-screen mode by entering:

```
set fullscreen on
```

or by putting this command in their PROFILE EXEC. Once in full-screen mode, you can enter data almost anywhere on the screen, even over existing data.

Full-screen mode uses CMS Session Services to define virtual screens and windows. VM output and messages are routed into windows. The system lets you determine if messages should be automatically displayed or if a notice should be issued that a message has arrived and is waiting to be viewed.

Other features of full-screen mode let you:

- Specify extended attributes for output
- Define Program Function (PF) keys.

Through CMS Session Services, interactive routines issue the output one line at a time. CMS Session Services also capture and display CP command responses and asynchronous messages. Current machine *states* such as RUNNING and HOLDING are replaced by more significant status indications such as ENTER A COMMAND OR PRESS A PF OR PA KEY and EXECUTING A COMMAND.

Macro Support

The CONSOLE macro instruction accesses full-screen console services and the following:

- Performs 3270 I/O operations
- Builds the Channel Command Word (CCW) or, for the CONSOLE EXCP function, runs the CCW built by the application
- Issues the DIAGNOSE code X'58' or SIO instruction
- Waits for the I/O to complete
- Checks any error status from the device.

CONSOLE lets programs, including CMS Session Services, open *paths* to a display device. It coordinates the screen by indicating to an application writing to the device that another path has updated the screen last and that the screen must be reformatted. Thus, full-screen applications do not have to rewrite the entire screen each time a write occurs.

Two CONSOLE path names, FSOFF and FSON, are reserved for the window functions.

When running in full-screen mode, the LINERD and LINEWRT macro should be used for doing line mode I/O and the CONSOLE macro for doing full-screen I/O. These macros are compatible with existing line mode interfaces, but also exploit CMS Session Services functions such as the specification of virtual screen names. The LINERD macro reads a line of input from the terminal and the LINEWRT macro displays a line of output at the terminal.

System Product Editor

For information on how the System Product Editor (XEDIT) uses CMS Session Services, see the "VM/SP System Product Editor" on page 88.

Reference

The *VM/SP CMS User's Guide*, SC19-6210, describes the full-screen CMS and window functions in more detail.

CMS HELP Facility

OVERVIEW	<p>HELP information is available for:</p> <ul style="list-style-type: none"> • CP, CMS, and IPCS end-user commands • CP and CMS messages • System Product Editor (XEDIT) • System Product Interpreter (REXX) • EXEC and EXEC2 statements • Transparent Services Access Facility (TSAF) • Enhanced Connectivity Facilities (SRPI) subcommands • End-user tasks.
-----------------	---

The HELP facility mainly uses a 3270-type terminal in full-screen mode, but can be used on terminals that display one line at a time. Full-screen mode refers to video display terminals that present a full screen of characters for information or editing. Line-oriented terminals display information one line at a time.

The full screens displayed by the HELP facility are called *panels*. There are three basic types of HELP panels:

Panel	Function
Menu	Prompts you to select the subject that help is needed on.
Information	Displays information about the selected subject.
Task	Prompts you to select the task that help is needed on.

The HELP facility is usually used to display reference information about using a particular command or subcommand, or for interpreting a system message.

To get the desired information panel displayed, you can use a series of menu panels. For example, to display an information panel that tells how to use the COPYFILE command while running under CMS, you would do the following:

1. Issue the HELP command
2. Follow instructions on the displayed general menu panel and select the COMMANDS panel
3. Select COPYFILE from the displayed COMMANDS panel.

Using HELP usually takes less time than finding the information in a book. You will be able to bypass menu panels by including the appropriate operand on the command line. For example, entering:

```
help copyfile
```

immediately displays a panel explaining the COPYFILE command.

The HELP command has three levels: BRIEF, DETAIL, and RELATED.

The BRIEF level, for the inexperienced VM/SP user, displays a command, its description, its basic syntax (command without options), an example, and, if applicable, a message telling you that either more detailed or related information is available. If CMS Session Services is active, the information appears in a pop-up window rather than on a full screen.

The DETAIL level, for the experienced user, displays a complete description, syntax, and usage notes.

The RELATED level gives you additional information about a command.

Requesting HELP

In full-screen mode, you ask for help by typing HELP on the CMS command line, on the command line directly from menus, from the XEDIT environment, or from within HELP itself. If there is more than one panel of information, scroll through the information using PF keys.

In line mode, you can request help by typing HELP and pressing the ENTER key.

For messages, HELP displays the message text, an explanation, a system action, and a user action.

Reference

The *VM/SP CMS User's Guide*, SC19-6210, describes the CMS HELP facility, including how to use it or tailor it for specific needs.

The Console Stack Helps Keep Work Flowing

OVERVIEW

CMS uses a two-part storage area called the *console stack* as a processing buffer. The two parts of the console stack are:

- **Program stack** — Temporary storage for lines (or files) being exchanged by programs running under CMS
- **Terminal input buffer** — Holds lines keyed in from a user's terminal until CMS processes them.

CMS commands and exec control statements are used to manage the program stack and data flow through it. CMS usually uses the following sequence to read input lines:

1. Lines from the buffer area within the program stack
2. Lines from the terminal input buffer
3. Lines keyed directly from the terminal keyboard.

The *console stack* is created in a user's virtual machine processor storage on an as-needed basis (dynamic creation). It is created to fill a specific need and varies in size according to the amount of data it must hold.

Exchanging Data through the Program Stack

The program stack is primarily for exchanging data among exec routines and application programs running under CMS. For example, data lines are put on the program stack using PUSH or QUEUE control statements in an exec routine, then obtained with PULL control statements in another exec routine.

Figure 25 on page 107 shows the terminal input buffer and the program stack.

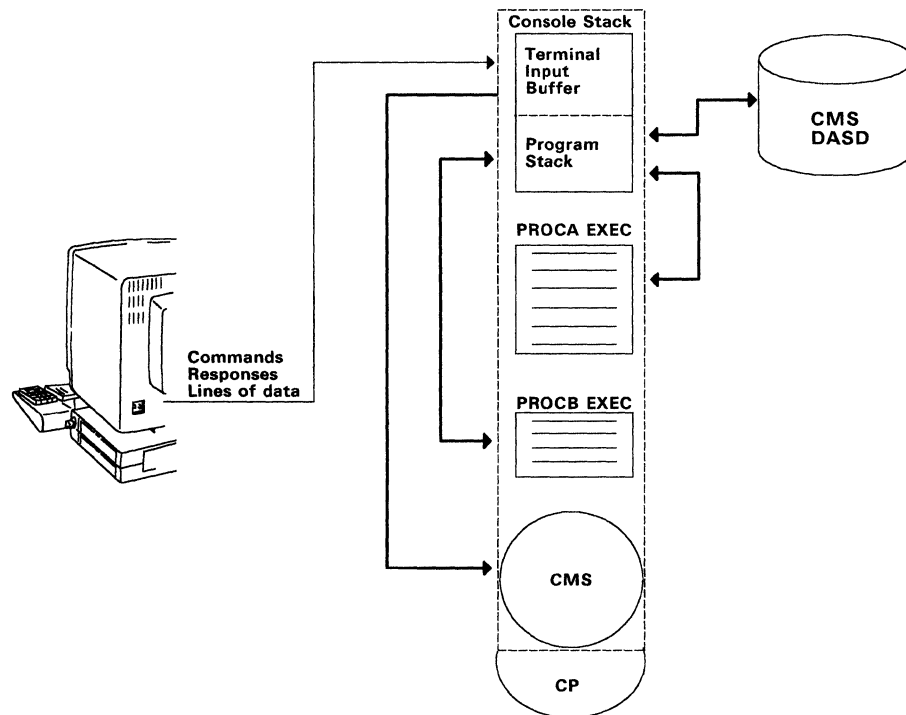


Figure 25. Using the Program Stack

Using the program stack, options of the EXECIO command under CMS let you:

- Read lines from virtual direct access storage or from a virtual reader to the program stack
- Write lines from the program stack to a CMS file or to a virtual unit record device
- Recover on the program stack the results of CP command execution.

Application programs running under CMS run the ATTN function to write a line to the program stack. To obtain a line from the program stack, application programs run the WAITRD function. CMS usually obtains program stack lines in a first-in first-out (FIFO) sequence. However, you can specify that CMS obtain lines in a last-in first-out (LIFO) sequence.

CMS expands the program stack as needed from available free storage. The maximum length for lines written to the program stack is 255 characters.

Terminal Input Buffer Is Temporary Storage for Keyed Data

The terminal input buffer receives and holds lines keyed from a terminal until CMS is ready to process them. This lets you key in commands, responses, and other data at a rate that tends to be independent of the rate of processing input by CMS. CMS is fast, but it is often busy at the instant you choose to enter data. If so, the keyed lines are placed into the next available positions of the terminal input buffer. When it is ready to process more data, CMS obtains lines from this buffer on a first-in first-out (FIFO) sequence.

The maximum line length for passing through the terminal input buffer is 130 characters.

Reference

The *VM/SP CMS Command Reference*, SC19-6209, describes and has examples of the EXECIO command.

The *VM/SP Application Development Reference for CMS* book, SC24-5284, describes the CMS application program functions available for using the program stack.

The *VM/SP EXEC 2 Reference*, SC24-5219, describes the exec control statements.

The *VM/SP System Product Interpreter User's Guide*, SC24-5238, describes and gives examples for using the REXX language.

The *VM/SP CMS User's Guide*, SC19-6210, has introductory information about writing execs.

Chapter 5. VM/SP Connectivity

Many options for teleprocessing, networking, and communications functions are supported by VM/SP and available licensed communications programs. VM/SP's connectivity ranges across network architectures, local area networks, wide area networks and branch exchanges, and includes many distributed data processing functions.

This chapter describes the major connectivity capabilities of a VM/SP system among VM/SP systems, other IBM systems, and original equipment manufacturer (OEM) systems.

VM/SP Systems Connectivity

OVERVIEW

VM/SP systems support the two standard communications network architectures, Systems Network Architecture (SNA) and Transmission Control Protocol/Internet Protocol (TCP/IP).

Communications

SNA is the IBM networking architecture. SNA provides access to a large number of IBM applications, databases, devices, and network management capabilities. Using SNA, hosts and workstations can connect to allow file transfer, data sharing, remote logon, and other functions.

TCP/IP is a public domain networking architecture that provides multivendor connectivity to hosts and workstations for file transfer, mail exchange, remote logon, and other functions of importance in the multivendor environment.

The SNA and TCP/IP network architectures support local area networks (LANs) and wide area networks (WANs). SNA provides support for the IBM Token Ring LAN, while TCP/IP for VM supports Ethernet™, Token Ring, and PC Network LANs. SNA wide area communications includes Synchronous Data Link Control (SDLC) lines, X.25 circuits, T1 lines, and Binary Synchronous Communication (BSC) lines. TCP/IP wide area communications include X.25 and the Defense Data Network.

Distributed application support provided in VM/SP includes:

- The Advanced Program-to-Program Communications/VM (APPC/VM) connection to other VM/SP systems and to the Virtual Telecommunications Access Method (VTAM) for access to SNA networks
- The Systems Application Architecture™ Common Programming Interface (CPI) Communications, a high-level Advanced Program-to-Program Communications (APPC) programming interface.

The APPC/VM connection to other VM/SP systems lets APPC/VM programs communicate with APPC/VM programs in the other systems. The APPC/VM connection to VTAM allows APPC/VM programs to communicate with other SNA LU 6.2 (APPC) programs in an SNA network, that may be non-VM programs. In addition, non-VM programs that communicate using APPC can communicate with APPC/VM programs. Thus, APPC/VM programs can communicate with other systems that support APPC. For example, a Customer Information Control System/VM (CICS/VM) server running on VM/SP can communicate with CICS running on Multiple Virtual Storage (MVS). VTAM communications support (such as X.25, SDLC, T1, and Token Ring) increases APPC/VM program communications options.

Ethernet is a trademark of the Xerox Corporation.

Systems Application Architecture is a trademark of International Business Machines Corporation.

Systems Application Architecture CPI Communications lets REXX and high-level language programs use APPC/VM to communicate. CPI Communications makes writing APPC programs much easier. In addition, Systems Application Architecture specifications enforce common syntax and a common set of verbs. This improves program portability between different operating systems that support CPI Communications.

You can make your information processing much more efficient by taking advantage of cooperative processing—the ability to connect an intelligent workstation, such as an IBM Personal Computer or Personal System/2®, to a VM/SP host system. Enhanced Connectivity Facilities (ECF) host servers are available for (PC DOS) Personal Computers attached to the VM/SP host. ECF host servers include virtual file, virtual disk, virtual print, file copy, and host data base access.

Connectivity Solutions

Figure 26 on page 112 shows the connectivity solutions for integrating VM/SP systems into existing SNA and TCP/IP networks. As shown, the following can connect to a VM/SP system (the IBM 9370 Information System) using TCP/IP or SNA networks:

- 3270 and ASCII terminals
- Token Ring and Ethernet LANs
- Packet switched (X.25) networks.

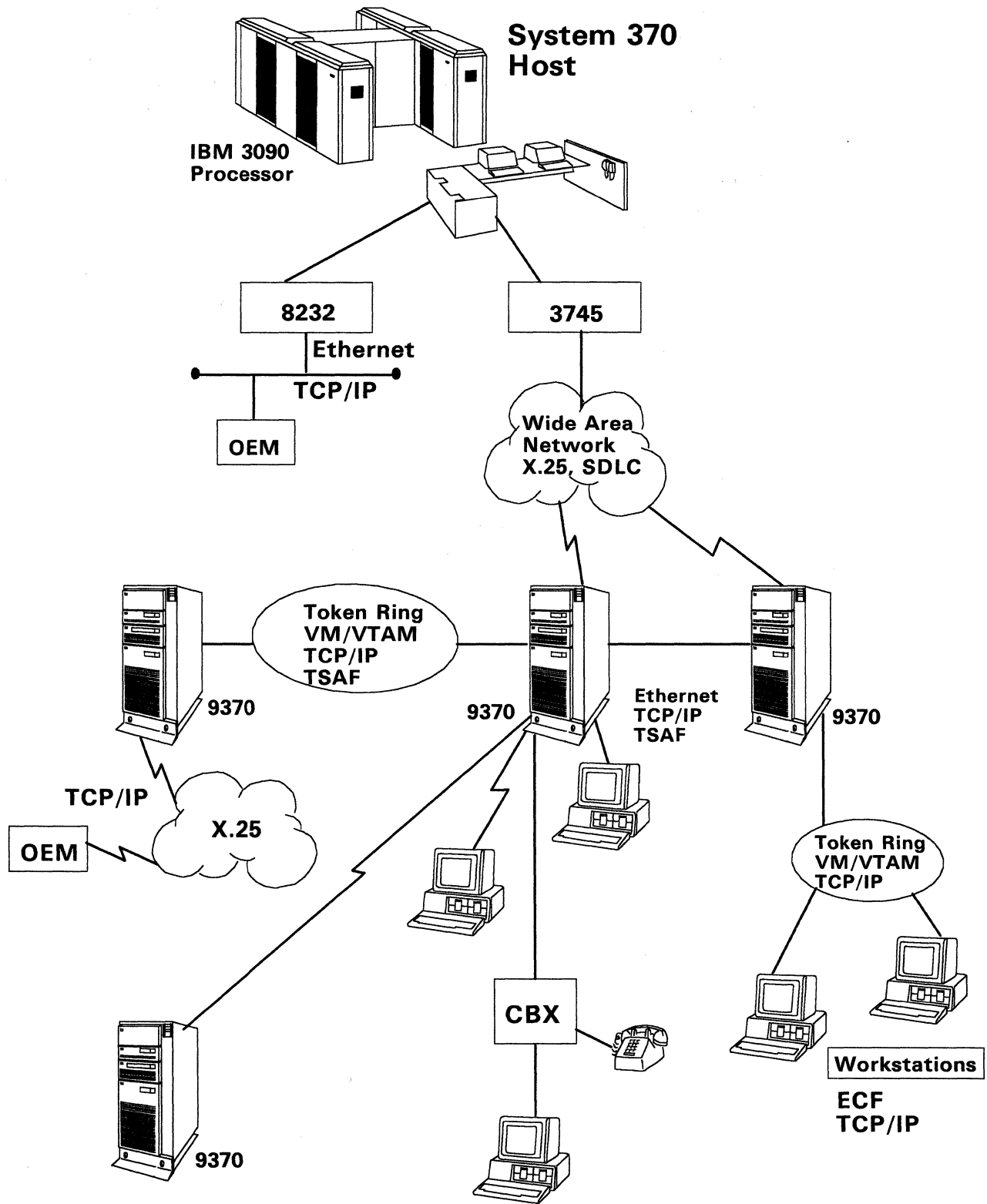


Figure 26. VM/SP Connectivity Capabilities

Remote Support: As shown in Figure 26, the 9370 Information System provides remote departments with 3270 terminal access, ASCII terminal support (3270 protocol conversion) through the ASCII subsystem, and Token Ring and Ethernet LAN support. Remote departments can access 9370 VM resources and remote host

resources. Public packet-switched networks (X.25) can directly attach to the 9370 Telecommunications subsystem, to allow VM/SP systems to communicate with each other or to any X.25 attached SNA host using the Network Control Program (NCP) Packet Switching Interface (NPSI) support. VM/SP systems can also communicate over X.25 or Defense Data Networks using TCP/IP for the multivendor environment.

VTAM: VTAM supports SDLC, X.25, Token Ring, and BSC links through the 9370 integrated communications subsystems and channel attached communications controllers. The 9370 Block Multiplexer Channel can be connected to a channel-to-channel adapter (integrated CTCA or 3088 device) for communication with a S/370 host. In addition, with a channel attached controller (3745), VTAM also supports T1 links.

Transparent Services Access Facilities: VM/SP uses the Transparent Services Access Facilities (TSAF) for peer-to-peer communications. TSAF provides VM/SP users with transparent access to server resources within a VM/SP system or across (up to eight) VM/SP systems.

APPC/VM VTAM Support: VM/SP's APPC/VM VTAM Support (AVS) allows an APPC/VM connection to VTAM. Therefore, APPC/VM applications can communicate over an SNA network to other APPC programs. This increases APPC/VM connectivity to programs on other systems (for example, MVS) and to communications media supported by VTAM (for example, SDLC, X.25, and T1).

TCP/IP: TCP/IP is available for multivendor connectivity and communication with the IBM RT PC®, PC, and Personal System/2® (PS/2) over Ethernet. The TCP/IP for VM provides the 9370 with communication support for Ethernet via its IEEE 802.3 Local Area Network subsystem and for the IBM Token Ring (IEEE 802.5) LAN subsystem. The combination of Ethernet (IEEE 802.3) and TCP/IP support provides the 9370 with the capability to coexist with multivendor systems in an Ethernet LAN environment. (VM, PC, and PS/2 versions of TCP/IP are available from IBM.)

Remote Spooling Communications Subsystem: The Remote Spooling Communications Subsystem (RSCS) networking product provides an asynchronous store and forward file transfer capability to VM/SP systems. RSCS can also be used to transfer files to MVS and VSE systems. RSCS can communicate over SNA networks using VTAM or over dedicated BSC links that RSCS controls. RSCS also supports dedicated channel communications. VM/SP Release 6 systems using VTAM can share a single VTAM controlled line for RSCS, PVM, APPC/VM, and other SNA network communications. This line sharing reduces line costs and simplifies interconnection.

VM/Pass-Through Facility: The remote logon program product, VM/Pass-Through Facility (PVM), provides virtual terminal functional capabilities for the VM/SP environment. (PVM also allows VM/SP users to log on to a MVS or Virtual Storage Extended (VSE) system via a dedicated BSC link, that emulates a 3174/3274 control unit.) PVM supports both BSC and CTC communications. When PVM

RT PC is a registered trademark of International Business Machines Corporation.

Personal System/2 is a registered trademark of International Business Machines Corporation.

VM/SP Connectivity

Release 1.4 is installed on VM/SP Release 6, PVM uses APPC/VM to communicate via VTAM Version 3 Release 2 over an SNA network.

Reference

The *VM/SP Connectivity Programming Guide and Reference*, SC24-5377, describes Common Programming Interface (CPI) Communications and APPC/VM programming interfaces for APPC and contains details on APPC/VM functions.

The *VM/SP Connectivity Planning, Administration, and Operation* book, SC24-5378, describes in detail the TSAF and AVS virtual machine components, resources, and gateways.

The *Systems Application Architecture Common Programming Interface Communications Reference*, SC26-4399, describes in detail the Systems Application Architecture common programming interface and how VM uses this interface. This book includes a description of the complete set of common programming interface routines.

APPC/VM Connectivity

OVERVIEW

The APPC/VM connectivity features of VM/SP described in this section are:

- APPC/VM
- Transparent Services Access Facility (TSAF)
- APPC/VM VTAM Support (AVS)
- VM resources
- Gateways.

APPC/VM

The VM/SP program-to-program communication protocol is called *APPC/VM*. APPC/VM is the VM/SP implementation of the SNA APPC architecture, an IBM standardized protocol for program-to-program communication. APPC/VM is used by VM/SP Shared File System (SFS), CICS/VM, the Structure Query Language/Data System (SQL/DS) Version 1.3.5 and Version 2.1, the Professional Office Systems (PROFS) Version 2.2.2 program products, and other licensed programs.

TSAF and AVS

The TSAF virtual machine is a service machine that transports APPC/VM protocols between (up to eight) VM/SP systems. This group of VM/SP systems is called a *TSAF collection*. Within a TSAF collection, any program in a VM/SP system can have an APPC/VM path to a program in the same VM/SP system or in another VM/SP system. The systems that make up the TSAF collection are connected by any of the following TSAF-controlled links:

- Channel-to-channel (CTC) links, including 3088 links
- Binary Synchronous Communications (BSC) links
- IEEE 802.3 Local Area Network (Ethernet or IEEE 802.3) subsystem on the IBM 9370 Information System processors
- IBM Token Ring Local Area Network subsystem on the IBM 9370 Information System processors.

Figure 27 on page 116 shows program-to-program communication within a TSAF collection. A user program in one VM/SP system has an APPC/VM path to another program, called a *VM resource*, residing in a virtual machine of another VM/SP system. A VM resource manager in the target virtual machine controls access to one or more VM resources.

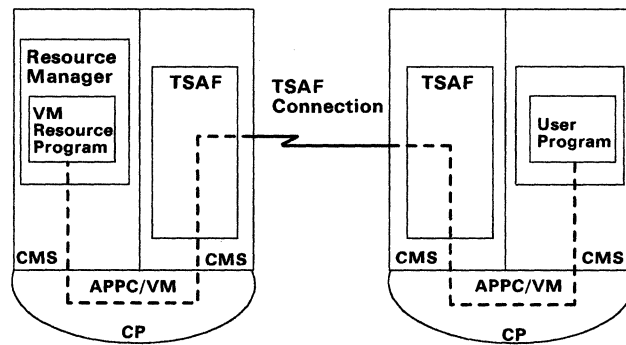


Figure 27. APPC/VM Program-to-Program Communication within a TSAF Collection

AVS, along with ACF/VTAM Version 3 Release 2, allows connections between a program in a virtual machine using APPC/VM and a program in the SNA network using SNA APPC protocol. Applications can be developed that communicate between VM/SP and other operating systems, such as MVS and Virtual Storage Extended (VSE), that are connected through the SNA network. AVS, a VTAM application that runs in the same Group Control System (GCS) group as the VTAM virtual machine, handles the translation between APPC/VM and APPC/VTAM (the VTAM implementation of APPC) letting programs in virtual machines communicate with SNA APPC programs.

GCS is a virtual machine supervisor that supports subsystems (such as AVS and VTAM). A GCS or virtual machine group consists of one or more virtual machines running under one GCS multitasking supervisor. Each virtual machine running under GCS is part of this group. Virtual machines within a GCS group communicate with each other and share common read/write storage.

An APPC/VM program can use AVS running in the same VM/SP system or, through TSAF, running in any VM/SP system in the TSAF collection. Figure 28 shows how AVS acts as a gateway between the SNA network and a TSAF collection.

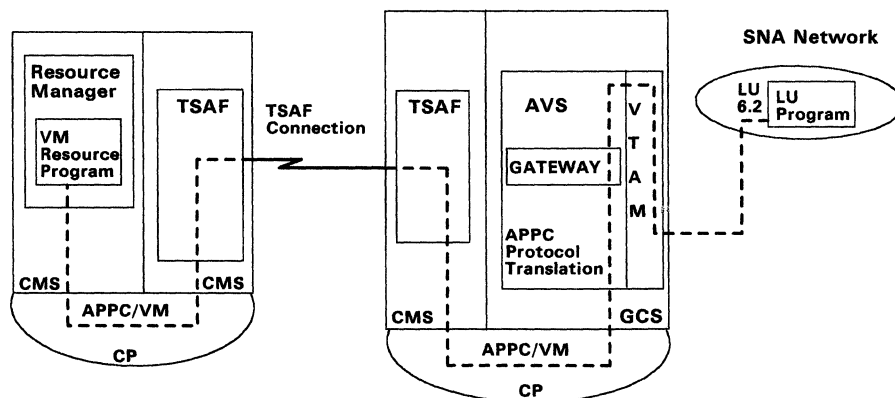


Figure 28. APPC/VM Program-to-Program Communication between a TSAF Collection and the SNA Network

AVS and TSAF are communications servers. They are authorized in their CP directory entries to establish connections to other communication servers on behalf of user programs.

VM Resources

A resource manager, running in a server virtual machine, controls a VM resource. A VM resource is a program, a data file, a specific set of files, a device, or any other entity or set of entities that you might want to identify for use in application program processing. For example, the resource can be an SQL/DS data base or a Shared File System file pool. There are three types of resources: local, global, and private.

Local and Global Resources

Resource managers identify local and global resource names to CP. The resource name is unique in the environment in which it can be accessed. Global resources can be accessed by programs located in the same VM/SP system, in other VM/SP systems in the TSAF collection, or in a system in the SNA network. The global resource name has to be unique within the TSAF collection. Local resources can be accessed only by users on the VM/SP system in which they reside. The local resource name has to be unique only within the local VM/SP system. The local or global resource server virtual machine must be logged on and the resource manager must be running when a program requests to connect to the resource.

Private Resources

Private resources are programs that execute under CMS and can be accessed by programs located in the same VM/SP system, in other VM/SP systems in the TSAF collection, or in a system in the SNA network. Private resources are not identified to CP and their names are unique only in the virtual machine in which they reside.

The private resource server virtual machine does not need to be logged on and the private resource manager does not need to be running when a program requests to connect to the private resource. The virtual machine will be autologged when needed and logged off when not needed.

In the private resource server virtual machine, there is a special file that lists the private resources and the authorized users for each resource. When a private resource connection request is made, CMS checks if the requester is authorized to access the private resource.

Gateways

AVS gateways (communications servers) are either global or private. Global gateways are used by APPC programs in the SNA network to access global resources that reside in the TSAF collection. Global resource managers use the global gateways to communicate with APPC programs in the SNA network. Private gateways are used by APPC programs in the SNA network to access private resources that reside in the TSAF collection. Private gateways can be defined as dedicated or nondedicated. If a private gateway is dedicated, it is dedicated to a single virtual machine (user ID) in the TSAF collection. All connections routed through that gateway are sent directly to that virtual machine. If a private gateway is nondedicated, it can be used to route connections to multiple virtual machines.

Reference

The *VM/SP Group Control System Command and Macro Reference*, SC24-5250, describes the function and use of GCS and GCS macros.

The *VM/SP Connectivity Programming Guide and Reference*, SC24-5377, describes Common Programming Interface (CPI) Communications and APPC/VM programming interfaces for APPC and contains details on APPC/VM functions.

The *VM/SP Connectivity Planning, Administration, and Operation* book, SC24-5378, describes in detail the TSAF and AVS virtual machine components, resources, and gateways.

The *Systems Application Architecture Common Programming Interface Communications Reference*, SC26-4399, describes in detail the Systems Application Architecture common programming interface and how VM uses this interface. This book includes a description of the complete set of common programming interface routines.

Distributed Application Development

OVERVIEW

You can write distributed programs using APPC to communicate with applications on the same VM/SP system, in other VM/SP systems, or on non-VM/SP systems.

APPC, a program-to-program communication protocol, is the part of SNA that lets APPC programs running under different operating systems communicate in a distributed application environment.

APPC Programming Interfaces

The two APPC programming interfaces that VM/SP programs use for communication are:

- Common Programming Interface (CPI) Communications

CPI Communications is a high-level programming interface for APPC, which is used by high-level programming languages, such as COBOL and FORTRAN, and procedures languages, such as REXX. CPI Communications is part of IBM's Systems Application Architecture. CPI Communications defines a common syntax for calling APPC functions and a standard base set of APPC functions. Programs using this common syntax can be more easily moved to other environments that use Systems Application Architecture.

- Advanced Program-to-Program Communications/VM (APPC/VM) Programming Interface

The APPC/VM programming interface is a VM/SP assembler language programming interface for APPC.

A CMS program can use either programming interface. The relationship of the two APPC/VM programming interfaces and APPC/VM services is shown in Figure 29 on page 120. A CMS program, using either interface, can communicate within a single VM/SP system or to other systems using either the TSAF transport or the SNA transport.

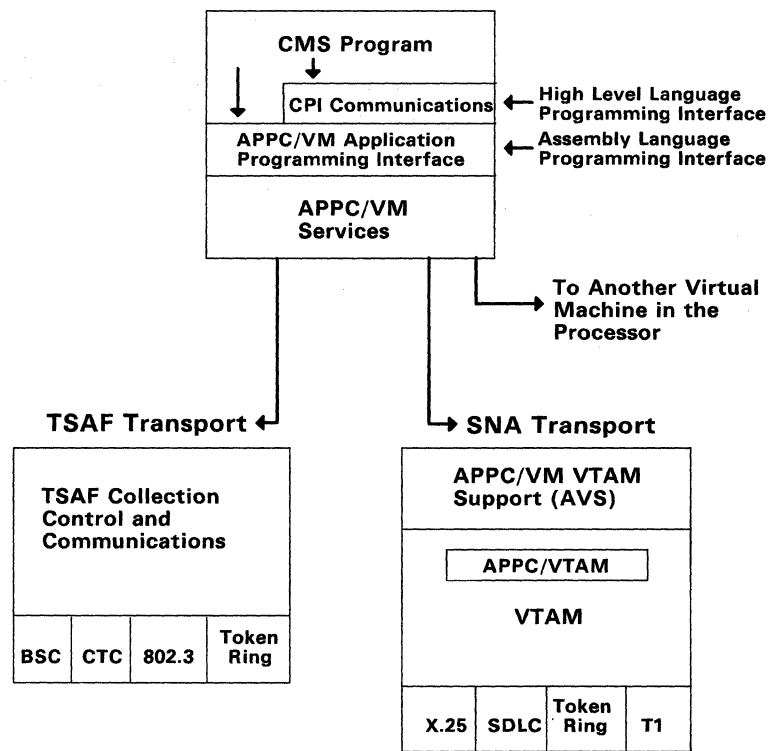


Figure 29. APPC/VM Programming Interface (Includes Assembly Language and CPI Communications)

CMS Communications Directory

The CMS communications directory allows an APPC/VM program to specify a symbolic resource name as the target of a connection request. The CMS communication directory is a file that contains the information needed to transform a symbolic resource name into an SNA network name. In addition to this information, the CMS communications directory can contain APPC access security information.

Specifying symbolic names, rather than SNA network names as the target of the connection request, lets APPC/VM programs communicate transparently within a TSAF collection or across an SNA network. Resources can be moved to other systems in the collection or in the SNA network without user programs being changed, because the location of the resource is transparent to the user programs. Only the CMS communications directory needs to be updated.

The CMS communications directory has two levels, system-level and user-level. The system administrator sets up the system-level communications directory. CMS users can set up their own user-level communications directories if their programs use resource names that are not defined in the system-level directory.

Access security user ID and password information can be specified in either communications directory. For security reasons, your installation may not want to put this information in system files or in user files. To avoid this, the information can be placed in the requester virtual machine's CP directory entry using the APPCPASS directory statement. The APPCPASS statement specifies the name of the target LU and the user ID and password authorization for the target system.

Reference

The *VM/SP Connectivity Programming Guide and Reference*, SC24-5377, describes the Common Programming Interface (CPI) and APPC/VM programming interfaces for APPC and contains details on APPC/VM functions.

The *VM/SP Connectivity Planning, Administration, and Operation* book, SC24-5378, describes the CMS communications directory.

The *Systems Application Architecture Common Programming Interface Communications Reference*, SC26-4399, describes in detail the Systems Application Architecture common programming interface and how VM uses this interface. This book includes a description of the complete set of common programming interface routines.

SNA Network Products That Run on VM/SP

OVERVIEW

NetView™ is a licensed program that enhances the usability, operability, and installability of the network management functions.

NetView

NetView is an SNA network management licensed program product. NetView automates network operations and network problem analysis.

NetView provides a comprehensive set of SNA network management services in a single product. It includes an online help desk facility, and a browse facility. The primary objective of NetView is to enhance the usability, operability, and installability of SNA network management functions.

NetView services include:

Browse function. Lets the operator browse SNA network management library files, such as VTAM major nodes, or logs, such as a log of all operator commands and messages.

Command facility. Controls, records, and automates many operator tasks. It is used as an operator's interface to VTAM. It allows optional logging of operator commands and messages on either disk or printers.

Hardware monitor. Collects and displays hardware problem determination information.

Online help. NetView includes help panels that explain the syntax and usage of each command.

Online help desk. The help desk has step-by-step instructions for operators doing network problem determination. Users can customize the help desk to conform to their own configurations and procedures.

Session monitor. Collects, correlates, and displays for selected LU sessions.

Status monitor. Has a full-screen method of controlling a network. It displays network status, accepts network operator commands, and allows for automatic reactivation of failed nodes.

ACF/VTAM

ACF/VTAM for VM/SP lets an installation use SNA in a VM/SP environment. It runs under the control of a GCS group in its own virtual machine. ACF/VTAM controls data flow between the devices defined to it and ACF/VTAM programs running in other virtual machines. All terminals, programs, lines, and devices defined to ACF/VTAM become part of the SNA network.

A common use of ACF/VTAM is to provide the SNA connection to transfer files between VM systems in an SNA network. RSCS Version 2 and the File Transfer Program (FTP) are products that can serve as two VTAM applications and can be

NetView is a trademark of the International Business Machines Corporation.

used to transfer files. FTP, which can transfer CMS files and VSAM files, can share the same SNA connection with RSCS Version 2.

ACF/VTAM also gives users in an SNA network greater access to applications. For example, if two or more VM systems participate in an SNA network, a user connected to ACF/VTAM in a local system can select an application name in a remote VM/VTAM system. The VTAM directory services locate the target application and establish the session for the user, without the user knowing where the applications reside in the SNA network.

In addition, using ACF/VTAM for all S/370 operating systems (VM, MVS, and VSE) provides efficient access to applications across systems. By connecting the various systems with SNA connections, users at any terminal in the SNA network can use VTAM to access any application owned by any VTAM system that is participating in the SNA network. It is transparent to the user if VTAM uses a CTCA, SDLC, X.25, or IBM Token Ring Network to connect the VM system to MVS or VSE.

Advanced Communications Function/System Support Programs

Advanced Communications Function/System Support Programs (ACF/SSP) has utility functions for 3705, 3725, and 3745 communication controllers, which run the Advanced Communications Function/Network Control Program (ACF/NCP). ACF/SSP is used to load the Network Control Program (NCP) into a controller, start an NCP dump of the controller, and generate an NCP for the controller.

Reference

The *Network Program Products General Information* book, GC30-3350, has general information about VTAM, NCP, SSP, and NetView.

The *Network Program Products Planning* book, SC30-3351, assists in planning a network containing VTAM, NCP, SSP, and NetView.

Remote Spooling Communications Subsystem Networking

OVERVIEW

The Remote Spooling Communications Subsystem (RSCS) Networking licensed program product manages spool file transfer among VM, MVS, and VSE systems. RSCS lets you:

- Send and receive commands and messages
- Send input and output data to other (peer) systems, and receive input and output data from other systems
- Print output (including graphics) on printers it controls
- Send and receive output for system controlled devices
- Receive input data from, and return output data to, remote job entry (RJE) work stations.

The RSCS Networking licensed program handles data exchange (mainly spool files) between virtual machines in a network of VM systems. This includes printing files on remote IBM 3270 Information Display System (or equivalent) printers in an RSCS Networking environment. RSCS can operate in an SNA and non-SNA environment.

RSCS can also be used to transfer files between VM and MVS or VSE, if those other operating systems are running under the control of a network job entry (NJE) compatible subsystem such as one of the following:

- JES3 Release 1.3.1 or later
- JES2 Release 1.3.3 or later
- VSE/POWER Version 2.

In an SNA environment, RSCS must run as a VTAM application and the SNA connection is owned by ACF/VTAM. In this environment, the RSCS driver SNANJE is used to communicate with JES2 in an MVS system, or VSE/POWER in a VSE system. To communicate with JES3 in an SNA environment, the MVS system must have the product MVS/BDT Release 2 installed. In a non-SNA environment, RSCS controls dedicated binary synchronous communications (BSC) or CTCA connections.

To send data to a user on another system, the sender must tell RSCS Networking the location and user ID of the receiving user and the file name to be sent. From the CMS user's point of view, this is easily done by the SENDFILE command.

The Network

Each RSCS Networking operating system has a directory of data that defines the network. Directory entries include such things as:

- The identification of the local node (node ID)
- The identification of each network node, with which the local node communicates, and whether the node is SNA
- Network paths (links) over which the local node communicates with other nodes in the network.

This data lets each RSCS Networking system decide how to handle data entering or leaving its node.

Data transmission is over long-distance communication facilities or, for systems at the same site, over channel-to-channel adapters. RSCS Networking has its own set of commands and messages to help the operator manage network operation.

Reference

The *RSCS Networking Version 2 General Information* book, GH24-5055, describes the program and other books in the RSCS library.

The *RSCS Planning and Installation* book, SH24-5057, describes how to prepare, install, customize, and automate RSCS.

VM/Pass-Through Facility

OVERVIEW

The VM/Pass-Through Facility (PVM) licensed program lets VM users log on to remote VM, MVS, and VSE systems. VM users access this facility either from the CP environment or when running under CMS.

PVM is used to log on to remote systems. Using PVM, you can log on and use a remote system in a network as though your terminal were directly connected to that system. PVM can run in both SNA and a non-SNA environment.

VM/Pass-Through from the User's Point of View

Accessing a remote system using PVM is done in three easy steps.

1. Enter the DIAL command without logging on from the CP environment, or log on and enter the PASSTHRU command from the CMS environment. After either command is entered, a menu of systems is displayed.
2. Select the system you want to use.
3. When the logo of the selected system is displayed on the your terminal screen, log on to the remote system.

PVM has many other facilities, such as:

- Disconnecting from a PVM session (processing in that system continues), returning to the local CMS virtual machine to do some processing, and then resuming the disconnected session.
- Storing console screen images on your virtual file mode A when your PVM session was started from the CMS environment.

Reference

The *VM/Pass-Through Facility Overview*, GC24-5373, has an overview of PVM for a potential user.

The *VM/Pass-Through Facility Managing and Using* book, SC24-5374, has an in-depth discussion of PVM.

IBM Enhanced Connectivity Facilities

OVERVIEW

The Enhanced Connectivity Facilities permit workstations (for example, Personal Computers) and hosts to communicate. It provides requesters on the workstations and servers on the host that perform the work required.

Why Is There a Need for Enhanced Connectivity Facilities?

Workstations are widely accepted in the marketplace. Workstations have brought computer power to the desk of the end user. Organizations and individuals alike appreciate the convenience, processor power, and ease of use.

However, even with these advantages, a workstation user is isolated from the rest of the organization. A workstation user also needs additional computer services that are available only on a host. Connecting the two worlds of desktop computing and traditional mainframe processing offers a solution to this problem.

What Is Needed to Use the Enhanced Connectivity Facilities?

For a complete list of the workstations, host systems, and physical connections that are supported and the licensed programs you need to use the Enhanced Connectivity Facilities, see the *Introduction to IBM System/370 to IBM Personal Computer Enhanced Connectivity Facilities* book, GC23-0957.

Running Enhanced Connectivity Facilities Applications

Some of the functions that are provided with the IBM Enhanced Connectivity Facilities include:

- **Virtual Disk**
You can use the disk space of a host system as if it were a fixed disk on the workstation. The data is actually stored at the host which means that you can access more storage than is available on the workstation.
- **Virtual File**
You can use host files at your workstations.
- **Virtual Print**
You can access host printers that let you print documents on high-speed quality printers.
- **File Transfer**
You can copy files between your workstations and the host, and format conversions will be done automatically.
- **Host Data Base Access**
You can extract data from host data bases and use that data at the workstation.

Reference

The *Introduction to IBM System/370 to IBM Personal Computer Enhanced Connectivity Facilities* book, GC23-0957, describes Enhanced Connectivity Facilities in greater detail.

The *VM/SP CMS Command Reference*, SC19-6209, has information on how to start Enhanced Connectivity Facilities using the CMSSERV command.

The *Programmer's Guide to the Server-Requester Programming Interface for VM/SP*, SC24-5291, has information on how to write and install servers.

The *Getting Started with the IBM PC Requesters and IBM PC Requesters Reference*, (package) part number 6316993, describes the Enhanced Connectivity Facilities on workstations in detail.

PC/VM Bond

OVERVIEW

Through connectivity between a Personal Computer and VM, PC/VM Bond lets you access the larger operating system with its built-in advantages of computing power and additional disk storage space. It can access the many facilities available on the VM/SP system while retaining the stand-alone advantages of the Personal Computer Disk Operating System (DOS).

With PC/VM Bond, the Personal Computer is used both as a Personal Computer and as a terminal connected to a VM System. You switch back and forth from VM (or *terminal mode*) to DOS (*Personal Computer mode*). The Personal Computer display is supported as an emulated IBM 3278 Model 2 Display Station.

In terminal mode, you can access all VM commands, macros, programs, and so forth. In Personal Computer mode, you have all the functions of DOS plus the added functions of PC/VM Bond:

- DOS/VM communications
- Extra disk drives
- Upload and download capabilities
- REXX/PC EXEC language.

DOS/VM Communications

You can communicate with VM from DOS. Using PC/VM Bond commands, you can send and receive messages with other VM users. You enter VM/SP commands from DOS and receive their results on DOS using the advanced features of PC/VM Bond. The Host Programming Interface feature of PC/VM Bond supports this function.

Extra Disk Drives

PC/VM Bond lets you use more disk drives than you physically have on your Personal Computer. The extra disk drives are actually *virtual* disk drives that reside on your VM disk. You use these virtual disk drives just as you would your real disk drives on your Personal Computer.

Upload and Download Capabilities

PC/VM Bond lets you copy files from VM to your Personal Computer or from your Personal Computer to VM. This process uses virtual disk files to accomplish the upload and download operation.

REXX/PC

The REXX/PC EXEC language lets you write programs on your Personal Computer. REXX/PC is a programming language for Personal Computer users that is similar to the Command Language Processor (REXX) available on VM/SP. It has an alternative to the batch facilities in DOS. You do not need the services of VM/SP to use REXX/PC.

Requirements

To support this connectivity, the following are needed:

- IBM Personal Computer XT™, XT™/370, AT®, or AT®/370
- IBM 3278/79 Emulation Adapter Card or the IBM 3278/79 Advanced Emulation Adapter Card
- DOS Version 2.00 or later
- PC/VM Bond consists of two parts:
 - VM Bond, which operates on the Personal Computer.
 - PC Bond, which operates on your VM/SP system. Your system administrator is responsible for installing and letting you access PC Bond.
- User ID, password, and a minidisk on your VM/SP system.

Reference

The *PC/VM Bond User's Guide*, part number 6476128, describes how to use PC/VM Bond. It has a primer that introduces you to PC/VM Bond and a reference section with all the commands.

The *PC/VM Bond Programmer's Guide*, SH24-5087, describes how to plan and install PC/VM Bond.

IBM Personal Computer XT and XT/370 are trademarks of the International Business Machines Corporation.

IBM Personal Computer AT and AT/370 are registered trademarks of the International Business Machines Corporation.

Virtual Machine/Personal Computer

OVERVIEW

Virtual Machine/Personal Computer (VM/PC) is a DOS application that runs on an IBM Personal Computer AT/370 and gives you a VM system within a Personal Computer.

VM/PC lets you:

- Use a local 370 environment in a Personal Computer AT/370. This lets VM/PC users run CMS applications in their Personal Computers.
- Emulate a 3278/79 terminal
- Send and receive files to and from DOS
- Maintain active sessions between DOS, VM/PC, and a host system
- Send data and programs between your host VM computer and your IBM Personal Computer.

Using Local VM/PC Services

You can use your IBM Personal Computer as a standalone VM system. This standalone VM system is called VM/PC or local session. You can authorize many people to use VM/PC, but because there is only one machine, only one person uses it at a time. If many users have access to the machine, you control access to VM/PC and to each user's data. The VM/PC Configurator controls who has access to what and is described in the *Installing and Configuring VM/PC* book.

Host 3270 Session

Connecting to a host computer as a 3270 terminal, you can combine your host VM session with your local session into a single environment. This lets you use some of the resources of your host system as if they were attached to your local VM/PC session. You can:

- Access CMS minidisks on your host VM system to:
 - Run programs on minidisks attached to your host system
 - Copy files from your host system to your local system and from your local system to your host system.
- Print files on the printers of your host VM system

The VM/PC Host Server lets you spool your output to your host virtual printer so any printing done on your local session will be sent to your host virtual printer. This lets you access high-speed printers.

- Access the reader, punch, and console of your host VM system

Your IBM Personal Computer does not support real or virtual readers or punches; however, the VM/PC Host Server lets you use your host VM devices as if they were local devices.

- Send DOS files to your host VM system

DOS files are sent to your local VM/PC system by using the VM/PC IMPORT command. VM files are sent to DOS by using the VM/PC EXPORT command.

Requirements

To use VM/PC, you need:

- IBM Personal Computer AT/370
- One fixed disk (minimum)
- One diskette drive (minimum)
- VM/PC 2.01
- DOS Version 3.10.

To use VM/PC and VM/PC Host Server, you need:

- IBM 3278/79 Emulation Adapter Card or IBM 3278/79 Advanced Emulation Adapter Card
- Coaxial connection for the 3278/79 emulation adapter card.

Reference

The *VM/PC User's Guide*, SC24-5254, describes how to use VM/PC.

The *VM/PC Host Server Programmer's Guide*, SH24-5125, has the information you need to plan and install the VM/PC Host Server.

The *Installing and Configuring VM/PC* book, part number 6467040, describes how to install VM/PC on your PC AT/370 and how to configure your system using the VM/PC Configurator.

Chapter 6. Applications for VM/SP

This chapter discusses licensed programs that can give you additional capabilities for VM/SP. These programs run under CMS. Although not all licensed programs that run on VM/SP are discussed, those included provide a good sample of the different program types available.

Additional sources of information about programs that run on VM/SP are:

- *The Software Directory*, GB21-9949
- *The Software Catalog*, G320-6530
- *The IBM Engineering and Scientific Application Programs Available From Non-IBM Sources*, G320-6739
- *The VM Application Programs Available from Non-IBM Sources*, GY33-6951.

Application System

OVERVIEW

Application System (AS) is an integrated system for information handling, offering comprehensive facilities for data management and display. Its functions are combined in many ways to satisfy the requirements of many applications in:

- Business planning
- Statistics and forecasting
- Project management
- Document preparation
- Business graphics.

Its tools can be used individually or combined to do complex tasks. All applications in AS use common methods and conventions for data entry and storage. This lets you use a single set of data for many different purposes without having to learn a new package.

Business Planning

AS helps you develop a model to plan and predict how different assumptions can affect your business.

Business planning involves basic data-handling processes—the creation of planning data, the calculation of results, the printing of reports, and so on. These needs are ideally met by a general information processing system such as AS. However, AS goes further by having functions that are specific to planning, such as *what if* analysis and report consolidation.

Statistics and Forecasting

AS has extensive facilities for statistical analysis and forecasting. All the analyses in AS statistics are produced by entering a single command. Also, certain analyses provide a set of basic information, which let you interactively derive further results.

To use AS statistics, you need to tell AS two things: (1) what file to use, and (2) which analysis to do. The IN command specifies which file to use. The STATISTICS command, used with keywords, tells AS what statistical or forecasting analysis you want and what data to use from the input file.

Project Management

AS analyzes arrow and precedence networks based on time or resource usage. Network drawings are produced using four scales with multiple facilities for examining all or part of the network in selected detail. Resources are given to tasks and analyzed to compare availability with usage, the results being displayed graphically. A risk analysis distribution of probable completion dates, either by activity or globally, is displayed as a graph.

Document Preparation

The document preparation function of AS contains two parts: MEMO and COMPOSE.

MEMO is the word processing facility. It fulfills the requirements for handling a small amount of text, enough for a usual business letter. It is screen-based and has a way of immediately creating formatted documents. The document is formatted whenever necessary, and the output is seen on the screen as it would appear on paper. The formatting is done according to the rules you specify.

COMPOSE is the text processing facility of AS. It lets you enter text on the terminal, with formatting statements, to produce documents in a suitable format. COMPOSE formats larger documents that need more formatting control than MEMO.

Business Graphics

AS offers a fully-interactive, syntax-free full-screen graphics interface. Data is converted into line plots, scatter diagrams, surface charts, bar charts, 2- or 3-dimensional pie charts, radar charts, real or symbolic map drawings or combinations of charts on the same axes or displayed separately. High resolution graphics are created with up to seven colors, and graphs are combined with data, text, and graphics from other files and areas of AS.

Reference

The *Application System General Information* book, GH45-5000, describes more about what AS is, what it can do, and what you need to install AS.

The *Using Application System* book, SH45-5002, offers tips and techniques on using AS to develop business applications.

The *Developing Applications with AS* book, SH45-5005, describes the tasks you can do with AS, how to work with your data, and how to produce output.

The *Managing Projects with Application System* book, SH45-5008, describes how to do project analysis with AS. It tells how to use AS to define project networks, prepare data files and reports, do time and risk analysis, allocate resources, and display results.

The *Creating Documents with Application System* book, SH45-5009, describes how AS creates, changes, formats, displays, and prints documents.

The *Business Planning with Application System* book, SH45-5010, introduces AS business planning facilities and the basic AS functions for building planning models. It describes how to add to and change plans, and design reports about them.

The *Creating Business Graphics with Application System* book, SH45-5011, describes how the Draw function of AS is used to create, modify, and format charts, graphs, and maps. Information on plotting graphics is also included.

The *Analyzing Data with Application System* book, SH45-5012, explains the statistical techniques available on AS. It also includes sections on how to prepare and analyze data with any of the statistical forecasting and testing techniques.

Professional Office System

OVERVIEW

Professional Office System (PROFS) helps you do many office tasks; such as,

- **Preparing documents**
- **Sending and receiving documents**
- **Formatting and printing documents**
- **Getting documents**
- **Handling routine office tasks.**

PROFS helps people who work in an office environment do their jobs more easily and efficiently. By automating many office tasks, PROFS promotes efficiency and reduces the adverse effects of human error and misplaced information. Most of the clutter around the office disappears because much of the traditional paperwork is replaced by electronic correspondence. When documents are printed, use of predefined formats assure a consistent, more professional document quality.

With minimum training, executives, managers, secretaries, technical personnel, and clerical personnel can learn to use PROFS productively. Displayed prompting panels and program function key facilities help simplify its use.

Preparing Documents

PROFS has facilities that simplify the creation of documents such as letters and memos. Installations can design formatted prompts for information such as addressee, subject, reference, copy list, and so on. An author profile facility lets information about you, such as name, title, and department be inserted automatically into a document. Text format control is provided by you as the text is entered, or provided automatically by the document preparation facility.

PROFS also helps you proofread completed documents. Each word is compared with entries in an internal dictionary. The system gives correct spelling for any misspelled or incorrectly keyed words. This facility also locates incorrectly used words in the context of the sentence and provides you with alternatives. Also, the facility locates and suggests alternatives to awkwardly used phrases.

By using PROFS, completed documents are handled in many ways, including printing, filing, distributing, updating, and deleting them.

Sending and Receiving Documents

PROFS lets you send documents to other users on the system on which it is installed. When used with Remote Spooling Communications Subsystem (RSCS) Networking program product, documents are sent to users on other systems as well.

The person to whom a document is sent receives a summary about the document, containing information such as date, author, and subject. By reviewing this summary, the receiver decides what action to take. For example, the receiver can print the document, view it on a display screen, file it, or pass it to another user.

Formatting and Printing Documents

Document layout is automatically controlled by installation-defined format files, or by control information entered by the user at the time the document is created. Documents are printed on VM/SP and RSCS-supported printers. An interface is provided for adapting printer output for printing on IBM 6670 and 6640 devices. Printing specifications, such as number of copies, are prespecified for each user or selected by the user from a displayed menu.

Getting Documents

PROFS documents are indexed for easy retrievability using many search arguments. Information such as who wrote the document, the date it was written, the name of the person to whom the document was sent, and the name of the subject, is extracted for indexing. Other information, in the form of searchable keywords, is included as search arguments by users.

Index records are created for documents other than PROFS documents. This lets the PROFS search facility find these documents.

Screens prompt you to fill in the necessary search arguments. These can specify author, date, and so on. You can display the results of a search at the terminal.

Handling Routine Office Tasks

Besides document preparation and handling, PROFS helps you do administrative tasks. These include:

- Routing short, informal communications among users without the need of writing memos or letters
- Handling incoming electronic mail
- Creating, interrogating, and updating conference room schedules and personal appointment schedules
- Scheduling follow-up action on documents
- Setting up an event reminder, which causes an audible tone at the terminal and displays specified information at a specified time and intervals.

Most of the displayed menus for using PROFS have associated help screens that are accessed through program function keys.

Reference

The *Planning for and Installing the Professional Office System* book, SH20-6800, describes planning, installing, preparing, and maintaining PROFS.

The *Using the Professional Office System* book, SH20-6797, describes how to use PROFS and has related examples.

SolutionPac

OVERVIEW

SolutionPac™ is a combination of hardware, software, support, and service to help you install applications more easily.
--

SolutionPacs provide resources for areas such as education, software engineering, expert systems, office support, and publishing. Some SolutionPacs are:

- VM/Software Engineering

This provides a powerful environment for the development, management, and control of software applications developed under the VM/SP operating system.

- Office Series VM Edition

For the office environment, this includes host and PC software in application areas of electronic mail, text, decision support, calendars, relational data base and query support.

- Publishing System VM Edition

For VM users, this provides a system for publishing documents with an extended life that can be produced by many authors and illustrators. The base product includes ProcessMaster, BookMaster, and BrowseMaster. Optional features include: graphics (DrawMaster), image processing (Image Handling Facility), formula formatting (SCRIPT Mathematical Formula Formatter), and printing options.

Reference

For more information, please refer to the following books:

IBM Publishing Systems ProcessMaster General Information, GC34-5031

IBM Publishing Systems BookMaster General Information, GC34-5006

IBM Publishing Systems BrowseMaster Installation and Reference, SH23-6091

IBM Publishing Systems DrawMaster User's Guide and Reference, SC34-5022

Image Handling Facility User's Guide, SB11-6328

SCRIPT Mathematical Formula Formatter Program Description/Operations Manual, SH20-0055.

SolutionPac is a trademark of the International Business Machines Corporation.

Document Composition Facility

OVERVIEW

Document Composition Facility (DCF) is an invaluable tool for producing almost any kind of formatted text document, including such things as: specifications, reports, books, manuscripts, and proposals.

Document formatting, through DCF, is obtained by SCRIPT/VS control words and Generalized Markup Language (GML) tags. A document, once processed by the DCF formatter, is printed, displayed, or used as input to other text handling programs.

With minimum training and no previous text formatting experience, you can learn to use DCF facilities to produce neat, professional looking documents that are easy to read and use. DCF has clear advantages over other methods of document formatting, which include:

- Documents, even those produced by different people, tend to be consistent in appearance.
- Formatting standards are easier to manage, especially when using a high-level formatting language such as GML.
- Document preparation is faster when irregular formatting is required. This includes such things as centered text, two-column text, variable size spaces for illustrations, and so forth.
- Document formats are changed with minimum effort.
- A given text is shared among many documents without the need of rekeying it for each one.

DCF provides many text document formatting capabilities. When used with the System Product Editor (described earlier in this book), almost any type of formatted document is produced. DCF results in consistency and accuracy that is very difficult to achieve when manually preparing a document, and it does it with far less effort.

Although its primary use is document formatting, DCF has other functions. For example, symbols embedded in documents can be used for such things as controlling conditional processing and entering characters that are not on the keyboard. It is also used for preparing documents for other text processing programs that need formatted documents as input.

SCRIPT/VS Is for Detailed Format Markup

One of the two major facilities of DCF is SCRIPT/VS. SCRIPT/VS has:

- A set of formatting control words
- A document format processor.

SCRIPT control words are made up of two alphabetic characters, preceded by a period (.). The two characters usually show the function of the control word. For example, .FN is used for inserting a footnote into text. Control words are included in the text, usually at the point that they affect the format of the document.

SCRIPT/VS control words have far more formatting functions than are summarized in this discussion. However, here are a few to give you an idea of what control words do.

Control Word	Function
. BX	Encloses text in rectangular line.
. CE	Centers text between left and right margins.
. IM	Imbeds another (specified) file into this file at this point.
. IN	Indents text a specified amount.
. PA	Ejects the form that the file is being printed to the top of the next page.
. RF	Prints specified text at the bottom of each page of the printed document (running foot).
. SK	Skips a specified number of line spaces at this location in the document.
. TC	Inserts an automatically generated table of contents at this location in the document.

SCRIPT/VS control words are used for functions other than text formatting. For example, the .DU control word updates the SCRIPT/VS dictionary of words. This dictionary is used by DCF for things such as checking the spelling of words in a file, if you request it.

GML Is for High-Level Format Markup Language

The second major facility of DCF is Generalized Markup Language (GML). Using GML is a short cut to document formatting. GML has a set of tags that define document elements rather than specific detailed format. A tag has a colon (:), followed by descriptive letters, and usually ending with a period (.). For example, the tag :p. starts a new paragraph. A tag expands into a sequence of SCRIPT/VS control words when it is processed. A programmer would think of a tag as a kind of macro statement.

Consider what goes into formatting lists of items, such as the one in this paragraph. Using SCRIPT/VS control words for formatting, each detail of the list must be dealt with separately. This includes:

- Spaces before and after the list, as well as between list items
- Indents for the second and subsequent lines of each list item
- The bullet (•) character and spaces preceding each list item.

Using GML tags, to format this list, you need:

- A :ul. tag to begin the list and a :eul. tag to end the list
- A :li. tag to begin each list item.

Another advantage to users is that format details do not have to be remembered. These include things like how many character spaces between the bullet character and the first character of the list item, or where to insert line spaces. Also, the sequence of control words (application processing function, or APF) that get

substituted for a tag are modified as necessary to change the tag's effect on document formatting. This is done without changing the source document.

A starter set of GML tags is included with DCF. Users expand this set by writing new APFs and assigning unique GML tags to them.

Processing DCF Formatted Documents

The DCF document processor program is executed using the SCRIPT command. Many options of the SCRIPT command are available to you for controlling the processor. For example, the formatted output from the processor is directed, by SCRIPT command option, to your terminal (default option), to a printer (PRINT option), or to a minidisk file (FILE option). Other options control such things as printing page numbers, checking of word spelling, selectively printing only certain pages, and suppressing messages.

Reference

The *DCF and DLF General Information* book, GH20-9158, has an overview of the product.

The *DCF: Generalized Markup Language Starter Set User's Guide*, SH20-9186, has information about using the GML tags supplied with DCF.

DisplayWrite/370

OVERVIEW

Features of DisplayWrite/370 are:

- Full-screen interactive text editor and formatter supporting the IBM 3270 Information Display System and the IBM 3270-PC display terminal
- Optional support of the 3270 extended stream
- Context dependent help and tutorial
- User's ability to set own defaults
- Command lists for special tasks
- Split-screen capability
- UNDO and REDO of system commands and local editing
- Basic and advanced text editing capabilities
- Variable specification and merging support
- Pattern letters support
- National Language Support including multilanguage linguistic aids.

DisplayWrite/370 is a System/370 host text processing application package that includes functions for document creation and revision, interactive document formatting, printing, and linguistic support. You do not need extensive end-user training to use DisplayWrite/370. This editor is easy to learn and is used by casual end users familiar with DisplayWrite/370, but unfamiliar with text processing.

Screen Characteristics

Full-Screen Support: DisplayWrite/370 has full-screen support for a minimum of 24 x 80 to a maximum of 62 x 160 screen sizes.

Extended Attributes: The editor has optional support to the 3270 extended attributes that feature seven display colors, blinking, reverse video, and underscore options.

Command List: When using the same commands in the same sequence, you can combine them into a command list (CLIST). Instead of entering all the commands, only the name of the CLIST is entered.

Document Representation: The screen will display the text in correct relative horizontal position only if the text is in 10/12/15 pitch uniformly. Mixed text in different pitches or proportional space fonts will not be properly displayed.

Cursor Movement: In edit mode, the cursor shows where the editing operation should take place. Commands are entered by a PF key or command line. A floating command line (displayed by pressing a PF key) maintains cursor position and decreases time spent using the command line.

Window Techniques: When the DisplayWrite/370 functions need interaction with the end user, the editor overlays a small display area on the screen prompting the user to key a new value. By entering the END or CANCEL command, the window disappears, and the original contents of the screen are redisplayed.

Operation

Because of the many formats available for entering commands, the DisplayWrite/370 accommodates both the infrequent and the experienced user. Entered commands are written to a log file for UNDO, REDO, and recovery purposes. An editor command is entered from any one of the following ways:

- Menu windows
- PF key
- Command line
- Floating command line
- Command list.

Error Recovery: You can revoke all changes made since the start of the editing session or since the last save command (whichever is more recent) through the UNDO command. The REDO command lets you issue again any or all of the commands revoked by the UNDO command.

Help Facility: The HELP facility presents information on a specific subject including HELP on selected menus and messages using the window display technique. When HELP alone is requested, the HELP main menu panel is displayed, letting you select the area where assistance is needed.

Tailoring DisplayWrite/370: Most features of DisplayWrite/370 are changed by you or defined in profiles. These features include PF keys, Revisable Form Text Document Content Architecture (RFTDCA) controls, and screen colors.

Split Screen: Split screen allows for the viewing of two different documents or two versions of the same document at the same time. The top half of the display is edited while the bottom half is viewed or copied. You can also control the size of the screen for each document while in split-screen mode.

Compatibility with Document Composition Facility

DisplayWrite/370 has an interactive formatting function, whereby you can directly view the formatted document. DisplayWrite/370 does not support formatting functions that need two-pass processing; these are handled by DCF.

Using DisplayWrite/370 under VM/SP, you can convert most of the RFTDCA controls into SCRIPT controls.

Document Processing Options

The DisplayWrite/370 document processing options let you:

- Create or revise an RFTDCA document
- View any RFTDCA or Final Form Text Document Content Architecture (FFTDCA) document
- Paginate a document interactively
- Transform a RFTDCA document to a print data stream.

Text Entry and Editing Capabilities

DisplayWrite/370 basic and advanced text editing capabilities include:

- Create and update footnote entries
- Arithmetic functions
- Line and page format specifications
- Insert and delete text
- Underline or emphasize text
- Block move, insert, copy, and delete
- Tabs (left, right, center, decimal, comma, and colon)
- Subscript and superscript (cannot be displayed)
- Overstrike
- Headers and footers (including alternate headers and footers)
- Automatic line adjust (without Entry Assist)
- Auto hyphenation
- Global search and replace
- Find and change text (multidirectional)
- GoTo (page number, line number)
- Change font
- Instructions (insert controls)
- Escape sequence, defining and entering
- Multiple formats (line, page, document, and alternate).

Language Support

DisplayWrite/370 includes multilanguage support as follows:

Linguistic Support: The linguistic support includes:

- Spelling verification
- Grammatical editing⁹
- Spelling correction assistance
- Automatic hyphenation
- Synonym support¹⁰
- Grade level analyzer¹¹.

DisplayWrite/370 Language Dictionaries: The linguistic supports described earlier are available in the following languages:

- Brazilian Portuguese and Portuguese
- Catalan
- Danish¹²
- Dutch
- English Dictionaries¹³
- Finnish
- French and Canadian French

⁹ Canadian French, French, Norwegian, Spanish, Swedish, and US English only

¹⁰ Canadian French, French, Norwegian, Spanish, Swedish, UK and US English only

¹¹ US English only

¹² Spelling correction assistance limited to noncompound words

¹³ UK and US English, legal and medical for US English only

- German and Swiss German
- Italian
- Norwegian
- Spanish
- Swedish.

National Language Support: National language support for all menus, HELP screens, and user messages is provided in the following languages:

Brazilian Portuguese	French	Norwegian
Danish	German	Portuguese
Dutch	Icelandic	Spanish
Finnish	Italian	Swedish
		UK and US English

Extended character set support is provided by customization functions done by each customer during installation. This support includes both workstation character set and printer character set support.

Note: The IBM 3270-PC does not provide support for all listed languages at this time. Please consult your IBM marketing representative.

Executing DisplayWrite/370

Under VM/SP and VM/SP HPO, DisplayWrite/370 is entered from the CMS command line and works with PROFS Version 2, Release 2.

Document Interchange

DisplayWrite/370 documents are interchanged in either RFTDCA or FFTDCA by all products supporting the Document Content Architecture (DCA).

Reference

The *Introducing DisplayWrite/370* book, GH12-5170, for general readers, including customer executives, data processing personnel, and professional end users. It has short descriptions and example panels showing the functions provided by DisplayWrite/370.

The *Using CLISTs with DisplayWrite/370* book, SH12-5196, describes how to create and use CLISTs, windows, and messages with DisplayWrite/370.

The *Getting Started with DisplayWrite/370* book, SH12-5171, contains mini-lessons that show you how to create and edit documents.

The *Using DisplayWrite/370* book, SH12-5172, is designed for anyone who wants to use the product to create, edit, and format documents.

The *DisplayWrite/370 Installation and Administration (VM/CMS)* book, SH12-5180, describes how to install and administer DisplayWrite/370.

The *DisplayWrite/370 Diagnosis Guide* book, SH12-5175, describes what to do if DisplayWrite/370 is not working correctly.

The *DisplayWrite/370 Reference* book, SH12-5176, gives programmers and advanced users technical details about DisplayWrite/370.

The *Using Images and Graphics With DisplayWrite/370* book, SH12-5177, describes how to use the image and graphics feature that is optionally available with DisplayWrite/370.

The *DisplayWrite/370 Quick Reference* book, SX12-5001, gives you abbreviated instructions for the tasks and commands you use in DisplayWrite/370.

Graphical Data Display Manager

OVERVIEW

Graphical Data Display Manager (GDDM™) adds a general graphics capability to many IBM licensed programs. Some application areas include:

- Adding graphics to business charts
- Interactive graphic programs that select or move displayed graphic representations
- Interface for displaying computer-designed or computer-aided drawings.

GDDM is effectively used for applications that use a mixture of graphic and alphanumeric data. Symbol editors help produce symbols such as company logos, scientific symbols, and fonts.

Because of its versatility, GDDM is used for graphic applications by many IBM licensed programs, both graphic and nongraphic. The primary task of GDDM is to put text or graphic data into suitable form for display or printing. The graphics created can be business charts and graphs, engineering drawings, symbols, or even pictures defined by colored dots or lines.

GDDM Utilities for Drawing Charts

GDDM provides three interactive utilities that let you produce attractive charts quickly and efficiently.

- Interactive Chart Utility
- Image Symbol Editor
- Vector Symbol Editor.

Interactive Chart Utility: This utility is a simple and versatile tool for drawing charts. Working together, you can display a partially completed chart for inspection and make changes while it is being developed. For example, the format is switched between line chart and bar chart. Some of the elements are deleted and others emphasized. Detail and color are changed, and the result is viewed by simply pressing a program function key.

Image Symbol Editor: This utility is typically used to add symbols such as company logos or special symbols to existing character sets. An image symbol is a rectangle of monochrome background on which colored dots define the symbol to be displayed or printed. For creating new image symbols, the Image Symbol Editor displays a matrix into which dots are keyed to define an enlarged version of the new symbol.

Vector Symbol Editor: This utility is typically used to produce symbols that need to be displayed in various sizes. This is useful when size is used for emphasis. A vector symbol is defined by line strokes, not dots as are image symbols. When creating a new vector symbol on the matrix displayed by the Vector Symbol Editor, only the points that define the start and end of a line need to be entered. Vector symbols

vary in size, and are rotated or tilted. Compared to image symbols, they are more versatile, but display less accurately at small sizes.

Other GDDM Features and Facilities

GDDM has an extensive set of two-dimensional graphic functions. These let you create pictures by describing them in terms of lines, arcs, colors, patterns, and so forth. The Interactive Map Definition feature of GDDM helps you design display screen formats. The names given to fields on the screen are the same names used in a program to refer to those fields. GDDM statements are coded into application programs written in COBOL, FORTRAN, PL/I, Assembler, BASIC, and APL2 programming languages. The *GDDM General Information* book has more information about writing applications for GDDM. It also lists other IBM programs that are available to GDDM users.

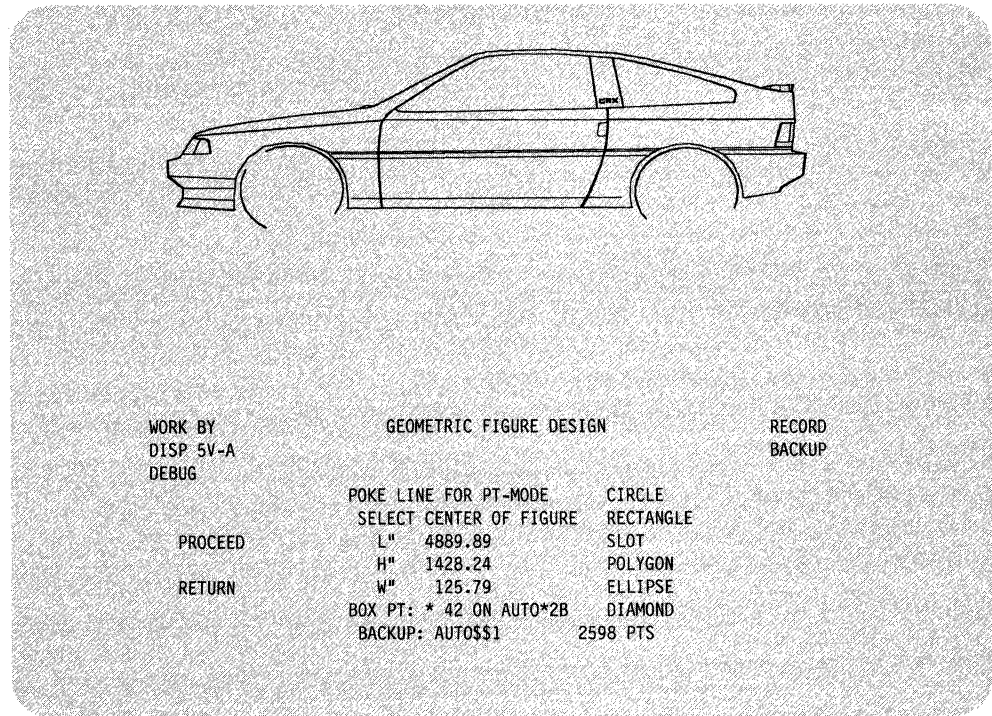


Figure 30. GDDM is Used for Displaying Many Graphics.

GDDM is used with many devices to produce both displayed and hardcopy graphics in monochrome or color. For example, using GDDM with the IBM 3279 Video Output Feature and a suitable projector is an effective way to produce high-quality visual aids for presentations. Graphics are printed in four colors on the IBM 3287 Printer. A Print Utility function has printing for both graphic and alphanumeric output.

GDDM is used in batch processing mode. This lets routine production programs or time-consuming jobs run when the processing computer load is light.

Reference

The *GDDM General Information* book, GC33-0100, gives a general description of GDDM, its use, and some IBM programs that use it.

The Information Facility

OVERVIEW

The Information Facility (TIF) is an interactive program that lets you manage information and make applications.

Some of the routine tasks that users do with TIF are:

- Store their own data or access existing data
- Review, update, delete, and add to the data
- Produce reports or charts from the data.

The options of TIF are based on the idea of tables. A table is an arrangement of data in columns and rows. Using TIF lets you:

- Define tables
- Build, update, and review tables
- Prepare and define reports and charts.

TIF also has *secondary* functions:

- Dictionary functions - Displays names of tables, screens, reports, and charts that you have defined.
- Table functions - Moves, reformats, renames, deletes, or sorts tables. You can also extract selected rows from a table.
- Conversational editor - Lets you edit table definitions, screens, reports, and charts using prompts.

Although all data used appears to you as TIF data, it can be a CMS file or a relational table managed by SQL/DS.

Table Definitions

Table definitions are the essential facts about a table. They include:

- Its name
- Maximum number of rows needed
- Column information:
 - Names
 - Widths
 - Data types
 - Which ones are keys.

Table Build, Update, and Review

TIF displays a prompt screen where you specify a table and the names of the columns you want to display. In specifying the table and columns and giving some other optional information, you are defining a screen in TIF. When you run the screen, TIF shows you the table (empty if it is new) and the columns. You can then build, update, or review on the terminal screen. These terms mean:

Build. The initial entry of rows into the table.

Update. Adding or deleting rows or changing the data in the columns.

Review. Displaying the table on the screen.

Prepare a Report or Chart from a Table

TIF lets you view the data in a table in many chart forms (bar, pie, line, and so forth). To define the report, you select the option you desire.

Reference

The *Introduction to The Information Facility* book, GC26-4217, describes how to start TIF, prepare simple reports, charts, graphs, and update and review data.

The *Information Facility: Concepts and Functions* book, SC26-4204, describes the concepts and functions of TIF and what it can do. This book is for users who want to understand TIF thoroughly and prepare themselves to become developers of TIF applications.

Structured Query Language/Data System

OVERVIEW

The Structured Query Language/Data System (SQL/DS) is a database management system that has:

- Data sharing - Users share data with each other
- Multiwrite - Users update concurrently
- Views - Many fields of data are selected for viewing
- Data security - The owner of the data controls access to the data
- Data recovery - Data is recovered up to the last update if there is a power or system failure.

The SQL/DS program product manages data stored in tables in a computer.

Data access is through a set of commands called *Structured Query Language (SQL)*. SQL commands offer you many functions to define, manipulate, and control access to data.

SQL/DS is used by people with little or no data processing experience. With it, you can easily access information stored as tables. Data is accessed by:

- Entering SQL commands from a terminal
- Running a routine containing SQL commands
- Embedding SQL commands in programs written in COBOL, PL/I, FORTRAN, or assembler language
- Using any one of many end user products that support SQL/DS data.

The Data Base Structure

Data in an SQL/DS database is organized in table (rows and columns) format. A database consists of one or more tables. Three designations help locate data within these tables:

- Each of these tables has a unique name
- Each column within a table has a unique name
- Rows within a table are identified by field content.

Selecting data from SQL tables requires knowing at least the names of the tables where the data is stored. Some databases have many tables, so remembering all their names is not practical. To help, SQL/DS maintains catalogs of information about the tables that make up the database. The catalogs have, among other things, table and column names for all tables in the database. The SELECT command for retrieving table data under SQL/DS displays information from these catalogs to help find the table and column names needed.

SQL/DS from the User's Point of View

Working with a structured database, SQL commands are available to:

- Select an item or group of items from the database
- Update an item or group of items
- Use as statements in programs that use SQL/DS

- Control authorization to use the database
- Manage the database and its use.

SQL/DS is commonly used to display selected columns (and perhaps selected rows within those columns) of data from a database table(s). The basic SQL query command:

```
select (column names) from (table name) where (row select conditions)
```

lets you retrieve data from any table in the database.

An entire table, selected columns, or rows from the table are retrieved. For example, suppose that a given table has records of sales data. Information in the table is structured so that the name of the city associated with each row of data is always entered in the column named CITY. This lets you ask that all rows of that table having BALTIMORE in the CITY column be retrieved.

Besides simple queries, a SELECT statement:

- Has logical operations such as *greater than*
- Asks for calculations, including integrated functions like maximum or average
- Selects data from more than one table at a time.

If needed, a copy of the query results is printed.

SQL/DS, as it applies specifically to the application programming environment and the business environment, is described later in this book.

ISQL Facility

SQL/DS includes a facility, Interactive SQL (ISQL), that lets you access and handle data directly from a terminal. ISQL commands are used to manage a SQL/DS session interactively to:

- Re-enter, with or without changing, the previous (current) SQL command entered from that terminal
- Format query results before they are printed
- Store frequently used SQL commands for reuse without retyping
- Specify the page size and number of copies for printed reports.

Data Security

All data stored in a SQL/DS table is owned by the SQL/DS user who created the table. The table owner controls all types of access to that table. The owner does many types of management activities on the table and grants some or all table management privileges to specific users. These privileges can also be revoked. Therefore, data security is the responsibility of the table owner.

Reference

The *SQL/Data System for VM/SP General Information* book, GH09-8043, provides an overview of the program to help you decide if installation is suitable to your needs.

The *SQL/Data System for VM/SP Concepts and Facilities* book, GH09-8044, provides a closer look at SQL principles and their implementation.

The *SQL/Data System for VM/SP Terminal User's Guide*, SH09-8047, describes how to use the program.

SQL/DS with Query Management Facility

OVERVIEW

QMF (Query Management Facility) has an easy-to-use, interactive way of accessing and controlling data in a SQL/DS database. Key features of QMF are:

- Full-screen interaction with users
- Online help facility
- Choice of two query languages
- Use of simple and familiar data model of tables
- Interactive tailoring of reports
- Preparation of data for graphics presentation.

SQL/DS is a database management system for manipulating data stored in row and column format (tables).

This topic discusses the combined facilities of SQL/DS and Query Management Facility (QMF). QMF serves as an effective communications interface between users and SQL/DS. QMF runs in a CMS virtual machine, together with SQL/DS and any other program products needed to take advantage of its full capabilities.

Besides getting data and displaying it on a screen, QMF lets you:

- Format retrieved data into a report
- Include totals, subtotals, averages, maximums, and minimums in a report
- Update, insert, and delete information in data tables
- Add new tables to a database
- Define many views of data from one or more tables.

Advantages of Using QMF

QMF is easy to learn, use, and needs no prior data processing experience. The few commands that must be learned have simple, easy-to-remember names like DISPLAY, RUN, PRINT, SAVE, HELP, and END. You can focus your attention on the job requirements rather than how to do it.

In QMF, there is interactive communication between QMF and users. Prompts are displayed to help you provide input. Full-screen communication facilities greatly simplify the process of providing information to the system.

Program function keys are used to begin many functions. For example, online help is available to you for help on operations or error messages by pressing a program function key. If a user error occurs, informative messages help you get back on track.

QMF accepts input in either of two query languages:

- SQL provides a simple English-like language. It requires you to learn only a few basic rules, such as keyword placement.

- Query by example (QBE) displays a graphic representation of data tables on which you enter an *example* of the output you want to see.

A QMF Example in Two Query Languages

The example, shown in Figure 31 on page 157, is not intended to be used as a guide or reference for using QMF. The purpose of the example is to point out:

- Ease with which SQL or QBE language is used with the QMF facility
- How to use the full-screen facility for communicating with QMF
- One type of data presentation that is requested
- Similar results from using either query language.

The top screen in Figure 31 on page 157 shows how an SQL query might be entered to request specified information from a data table named STAFF. The entire query, which is highlighted, must be entered.

The middle screen in Figure 31 on page 157 shows how a QBE query might be entered to request the same information. The grid, displaying the name of the table and the columns having the information requested and the names of the columns, resulted from a prior DRAW STAFF command. Only the characters highlighted must be entered.

The bottom screen shows the results of either the SQL or QBE query. You get a printed copy of the results by pressing the PRINT program function key (4).

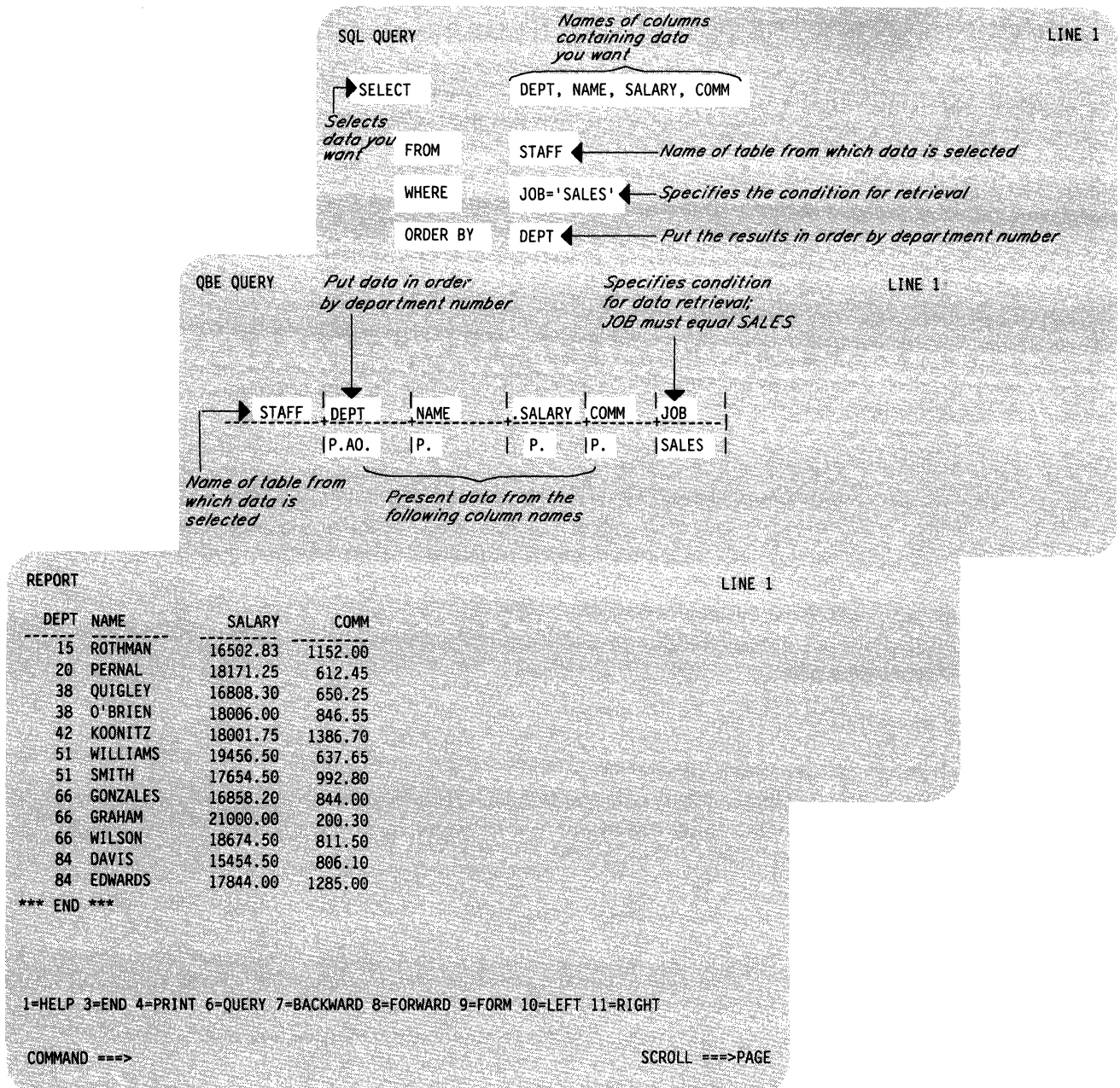


Figure 31. SQL/DS Responds to Either SQL or QBE Queries.

Reference

The *Query Management Facility Introduction* book, GC26-4101, describes the product, installation tasks, and requirements.

The *SQL/DS for VM/SP Concepts and Facilities* book, GH09-8044, describes SQL/DS for administrators, application designers and programmers, and system programmers.

Structured Query Language/Data System for Programmers

OVERVIEW

SQL commands can be embedded in application programs written in the following program languages:

- Assembler
- COBOL
- PL/I
- FORTRAN.

Embedded SQL commands are analyzed and converted into a module that is accessed for SQL/DS functions when running the host program.

Data in SQL/DS databases is accessed by programs running in CMS virtual machines and accessed interactively from a terminal. SQL/DS supports programs written in Assembler, COBOL, PL/I, or FORTRAN languages. Figure 32 on page 159 shows examples of SQL commands embedded into programs of each of the supported languages. Embedded SQL commands can reduce the programming needed to have data handling capability.

SQL/DS Preprocessor

Preprocessors do two functions:

1. They cause the SQL commands to become comment statements in the source code and add statements in the language of the host program to execute communication routines that replace the SQL commands. The changed version of the code is used as input for compiling (or assembling) the host program.
2. They convert the SQL commands into a module (access module), having the machine code equivalent of the SQL commands. They then store the access module in the SQL/DS database.

Any embedded SQL commands must be analyzed and converted to an access module by the SQL/DS preprocessor facility, PREP, before the host program is compiled (or assembled). When the application program is run, it calls the access module to handle the SQL commands via communication routines.

Using SQL/DS Extended Dynamic Statements, installations write their own preprocessors for other host programming languages.

COBOL Example:

```

DATA DIVISION.
WORKING-STORAGE SECTION.
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01  XX.
    49 XX-LENGTH PICTURE S9(4) COMPUTATIONAL.
    49 XX-VALUE  PICTURE X(24).
77  YY PICTURE S9(9) COMPUTATIONAL.
77  ZZ PICTURE S9(4) COMPUTATIONAL.
    EXEC SQL END DECLARE SECTION END-EXEC.
    •
    •
PROCEDURE DIVISION.
    EXEC SQL SELECT DESCRIPTION, QUANTITY INTO :XX, :YY
    FROM INVENTORY WHERE PARTNUMBER = :ZZ END-EXEC.

```

PL/I Example:

```

EXEC SQL BEGIN DECLARE SECTION;
    DCL XX CHAR(24) VAR;
    DCL YY BIN FIXED(31);
    DCL ZZ BIN FIXED(15);
EXEC SQL END DECLARE SECTION;
    •
    •
EXEC SQL SELECT DESCRIPTION, QUANTITY INTO :XX, :YY
    FROM INVENTORY WHERE PARTNUMBER = :ZZ;

```

Assembler Language Example:

```

    EXEC SQL BEGIN DECLARE SECTION
XX      DS   H,CL24
YY      DS   F
ZZ      DS   H
    EXEC SQL END DECLARE SECTION
    •
    •
    EXEC SQL SELECT DESCRIPTION, QUANTITY INTO :XX, :YY          C
    FROM INVENTORY WHERE PARTNUMBER = :ZZ

```

Figure 32 (Part 1 of 2). Examples of SQL Commands Embedded in Programs

FORTTRAN Example:

```

EXEC SQL BEGIN DECLARE SECTION
    CHARACTER*24  XX
    INTEGER      YY, ZZ
    •
EXEC SQL END DECLARE SECTION
    •
EXEC SQL DECLARE C1 CURSOR FOR
*   SELECT DESCRIPTION, QUANTITY
*   FROM INVENTORY
*   WHERE PARTNUMBER = :ZZ
EXEC SQL OPEN C1
EXEC SQL FETCH C1 INTO :XX, :YY
EXEC SQL CLOSE C1

```

Figure 32 (Part 2 of 2). Examples of SQL Commands Embedded in Programs

Using SQL Commands

Any SQL commands used for doing operations from a terminal can be used as embedded statements in an application program. However, there are some SQL commands that are used only as embedded statements in an application program.

SQL/DS has facilities (PREPARE and EXECUTE) for handling SQL commands entered dynamically while an application program is running; for example, entering queries and receiving results interactively while running an application. These special facilities are needed because the SQL/DS preprocessor does not process SQL statements that are entered dynamically.

Some special rules apply for coding SQL commands into an application program. For example, program variables shared by SQL/DS and the host program are usually declared, but the section of program where they are declared is preceded and followed by special commands. Also, program variables are preceded by a colon (:) when referred to from embedded SQL/DS statements.

Reference

The *SQL/Data System for VM/SP Concepts and Facilities* book, GH09-8044, describes the facilities and concepts of SQL/DS in the VM/SP environment.

The *SQL/Data System for VM/SP Application Programming* book, SH09-8019, describes the product from the application programmer's point of view.

Display Management System for CMS

OVERVIEW

Display Management System for CMS (DMS/CMS) helps you format and use displayed panels in an interactive environment. Three basic tasks are involved:

1. Design and create the panels
2. Write applications that use the panels
3. Use the applications.

Many programs use displayed panels to communicate between users and the programs. For example, the PROFS facility described in this book uses interactive panels for communicating with users. These panels are often in menu form. A menu lets you make selections by keying in words in the spaces provided, or by simply pressing a program function key. DMS/CMS helps you produce display panels and write the application programs that use them.

Panel Creation

DMS/CMS makes five display screens available to help create panels for use with application programs. These screens are, in effect, panels themselves. However, DMS/CMS documentation calls them *screens* to distinguish them from the panels being created. The five DMS/CMS screens are:

Screen	Function
Panel Name	Gives the panel a name.
Panel Size	Defines the dimensions of the panel.
Design Grid	Designs the layout and content of the panel.
Field Definition	Defines panel field characteristics, such as intensified display, blinking, or color.
Completion Options	Tells DMS/CMS whether to save the panel and if another panel is to be created.

Movement from screen to screen and cursor movement within a screen is controlled by program function keys. Most information needed to use these screens is included in the information displayed on the screens. However, HELP screens are available to help you if needed.

DMS/CMS has its own set of editing commands for creating panel designs on the Design Grid screen. The format and content of the information developed on the Design Screen become the format and content of the finished panel.

Writing Application Programs and Procedures for Using Panels

Panels created under DMS/CMS are used with either programs or exec procedures developed for that purpose. Programs are written in COBOL, PL/I, RPG II, or Assembler language. The exec procedures are written in any of the command language processors supported by VM/SP. DMS/CMS makes special commands, macro instructions, and procedures available to aid programmers in writing programs and procedures for displaying panels.

Using Applications

There are many ways to use panels created with DMS/CMS. For example, they are used to replace traditional paper forms that must be handwritten. The panel offers clear advantages over paper forms, such as:

- Ease of filling in data and correcting errors
- Ease of handling, storing, and retrieval
- Ease of changing panel layouts
- Reduced storage space requirements (for example, no filing cabinets).

Designers who layout panels, and programmers who write routines that use the panels, develop many applications for using a computer interactively.

Reference

The *VM/370 Display Management System for CMS: Guide and Reference*, SC24-5198, describes the program and how to use it, and has examples of related programs in the supported languages.

Systems Application Architecture

OVERVIEW

Systems Application Architecture is a definition—a set of software interfaces, conventions, and protocols that provide a framework for designing and developing applications with cross-system consistency.

Applications designed and developed for VM according to Systems Application Architecture specifications can be transported to other Systems Application Architecture environments. For example, an application for CMS can also run in a TSO/E environment, if it does not contain system-specific material.

Systems Application Architecture:

- Defines a *common programming interface* you can use to develop applications that can be integrated with each other and transported to run in multiple Systems Application Architecture environments.
- Defines *common communications support* that you can use to connect applications, systems, networks, and devices.
- Defines a *common user access* that you can use to achieve consistency in panel layout and user interaction techniques.
- Offers some *common applications* written by IBM using the common programming interface, common communications support, and the common user access.

Supported Environments

Systems Application Architecture provides a framework across the three major IBM computing environments:

- System/370 (CMS under VM, and TSO/E under MVS/XA)
- System/3X
- Personal Computer (Operating System/2™).

Common Programming Interface

As its name implies, the Common Programming Interface (CPI) provides languages, commands, and calls programmers can use to develop applications that take advantage of the consistency offered by Systems Application Architecture. These applications can easily be integrated and transported across the supported environments.

The components of the interface currently fall into two general categories: languages and services. Those marked with an asterisk (*) have VM support.

- Languages
 - Application Generator*
 - COBOL*
 - FORTRAN*

Procedures Language*

- Services

- Communications Interface*

- Database Interface*

- Dialog Interface*

- Presentation Interface*

- Query Interface*

The CPI is not in itself a product or a piece of code. But—as a definition—it does establish and control how IBM products are being implemented, and it establishes a common base across the Systems Application Architecture environments.

Thus, when you want to create an application that can be used in more than one environment, you can stay within the boundaries of the CPI and obtain easier portability.

Reference

More detailed information on the components of the common programming interface is available in the following Systems Application Architecture manuals:

Application Generator Reference, SC26-4355

COBOL Reference, SC26-4354

Database Reference, SC26-4348

Dialog Reference, SC26-4356

FORTTRAN Reference, SC26-4357

Presentation Reference, SC26-4359

Procedures Language Reference, SC26-4358

Query Reference, SC26-4349.

The following publications are also useful:

The *Systems Application Architecture: An Overview*, GC26-4341, introduces Systems Application Architecture concepts, and identifies the environments and elements that participate.

The *Common User Access: Panel Design and User Interaction* book, SC26-4351, defines the common user access for Personal Computers and System/370 and System/3X terminals, including panel layout and user interaction techniques.

The *Writing Applications: A Design Guide*, SC26-4362, provides guidance on developing application programs that are consistent and portable across the Systems Application Architecture environments. These applications will use the common programming interfaces and implement the common user access specification.

Chapter 7. Programming Language Products

This chapter discusses programming languages that have additional application capability for VM/SP users.

The products described in this section are intended for anyone involved in planning or writing application programs. Application programmers use most of the programming language products. Some languages, such as Assembler, are of special interest to system programmers.

Additional sources of information about programs that run on VM/SP are:

- *The Software Directory*, GB21-9949
- *The Software Catalog*, G320-6530.

COBOL

OVERVIEW

Common Business-Oriented Language (COBOL) is a programming language for commercial applications.

VM/SP supports two COBOL compilers:

- **OS/VS COBOL**
- **VS COBOL II.**

COBOL is a high-level programming language widely used for commercial data processing applications. It is a powerful language, yet its English-like format makes it easy to learn and use.

OS/VS COBOL

OS/VS COBOL has advanced programming facilities that reduce development time, resulting in increased programmer productivity. The OS/VS COBOL program product has a source language compiler and a subroutine library. The subroutine library is also available as a separately orderable program product.

The OS/VS COBOL compiler offers:

- Program development aids
- Advanced program applications
- Efficient object time performance
- Productive compile time performance.

Program Development Aids: Informal programming rules simplify program development. For example, relaxed punctuation rules allow a space before a period, semicolon, or colon.

Debugging features let you specify conditions under which data items or procedures are monitored during program execution.

Advanced Program Applications: OS/VS COBOL programming advancements available to the programmer include:

- Enhanced VSAM support
- Expanded physical sequential file capability
- Added communication support
- Expanded library facilities
- Powerful data manipulation
- Extended computational facilities
- User-defined collating sequences.

OS/VS COBOL Subroutine Library: The OS/VS COBOL subroutine library is a partitioned data set residing on a DASD. It has COBOL object-time library subroutines in load module form. This library is a separately orderable program product, and one copy is shipped with the OS/VS COBOL program product.

The library subroutines do execution-time operations that have either repetitive or extensive code. This significantly reduces the size of the program object module. Library routines needed to run the problem program are either combined with the object module at link-edit time, or dynamically fetched during program execution. Some types of library subroutines are:

- I/O (excluding VSAM)
- Conversion
- Arithmetic verbs
- Other verbs
- Sort and merge interfaces
- Checkpoint and restart
- Segmentation feature
- Communications
- Debugging
- VSAM.

VS COBOL II

The VS COBOL II program product has a source language compiler, a library, and generated object programs that run under VM/SP. The compiler and library are shipped together as a single program product or the library can be ordered as a separate product.

The VS COBOL II compiler offers:

- Extended addressing capabilities
- System management aids
- Program development aids
- Compatibility with OS/VS COBOL (object modules).

VS COBOL II enhances the benefits of OS/VS COBOL.

Extended Addressing: Extended addressing allows programs and data areas to be larger than OS/VS COBOL.

System Management Aids: The following system management aids help make VS COBOL II easy to use:

- Programs can be compiled to be reentrant
- Run time of the object programs can be reduced
- Amount of storage used by the programs can be reduced
- Storage management can be changed for applications

Programming Language Products

- Enforcement of *in-house* programming standards through reserved word control and compiler option control.

Program Development Aids: VS COBOL II provides the following program development aids to make coding and debugging easier:

- Development of program code that is easy to understand, maintain, and debug
- Assists in top-down design with a nested COPY statement
- Conveniently passes literal data to a subprogram
- Gives you access to VSAM return codes to obtain detailed information about your VSAM input or output requests
- Sets certain types of data items to predetermined values using the INITIALIZE statement. The SET condition-name to TRUE statement gives you a symbolic way of assigning a value to a conditional variable.
- Uses run-time options in environments in which dynamic specification is not possible
- Lists many diagnostic messages immediately below the statement in error
- Integrated interactive and batch mode symbolic debugging.

Compatibility Feature: The VS COBOL II compiler produces results that are compatible with OS/VS COBOL Release 2.

VM/SP Callable Services Library

OS/VS COBOL and VS COBOL II can call routines that reside in a callable services library (CSL). One such CSL library, called VMLIB, is supplied with VM/SP. Calls to CSL routines are not resolved until program execution. This lets you make changes to CSL routines without having to re-link the routine to the application program, recompile the program, or modify any of the program's source statements.

You can find more information about callable services libraries in the *VM/SP Application Development Guide for CMS* book.

Reference

The *VM/SP Application Development Guide for CMS* book, SC24-5286, has information about executing programs.

The *OS/VS COBOL Compiler and Library General Information* book, GC28-6470, has information to aid systems planners and analysts in evaluating the product. This information includes performance, compatibility, and system requirements statements, and lists restrictions for using OS/VS COBOL under CMS.

The *VM/370 User's Guide for COBOL*, SC28-6469, has information on using COBOL under CMS.

The *VS COBOL II General Information* book, GC26-4042, has information to help system programmers, application programmers, and system analysts evaluate VS COBOL II and to plan for its use.

Programming Language/One

OVERVIEW

Programming Language/One (PL/I) is a powerful, general-purpose programming language. It is implemented by the:

- OS PL/I Optimizing Compiler - Compiles the PL/I source into optimized object code.
- OS PL/I Library - Is used by the object code during linking and execution of the PL/I programs.
- OS PL/I Interactive Test Facility (PLITEST) - Provides extensive capabilities to allow the testing and debugging of PL/I programs.

These OS PL/I Version 2 products are available as:

- Compiler, Library, and PLITEST (program number 5668-909)
- Compiler and Library (program number 5668-910)
- Library (program number 5668-911).

PL/I is a powerful, comprehensive language that addresses commercial, scientific, and system programming needs. The OS PL/I Optimizing Compiler (PL/I compiler) incorporates some of the best features of many other proven compilers.

PL/I statements are written in a high-level, easy-to-learn format, much like English or elementary algebra. PL/I provides many structured programming constructs, which allow easier development and maintenance of applications.

Features

The main features include:

- **Extensive Testing and Debugging Aids:** Minimize time and effort required for program checkout by using:
 - PLITEST, an interactive testing facility that allows testing and debugging of PL/I application programs in PL/I terms and with full knowledge of the source program while requiring the least possible amount of system resources. It allows for the detection of errors and the application of fixes interactively, minimizing the need to recompile. Online help is provided, minimizing the need for manuals. Interactive line mode support and batch mode support are also provided.
 - Extensive implementation of ON-units to allow the PL/I program to handle software and hardware detected conditions
 - Clear and precise compile-time and run-time diagnostic messages
 - Statement number tracing facility
 - Statement frequency counting facility
 - Check condition support
 - Data-directed I/O.

Programming Language Products

- **Compile-time Facilities:** The PL/I Optimizing Compiler gives programmers many compiler facilities. For example, programmers can use:
 - The preprocessor for many useful tasks, including symbolic replacement, generating code, creating reusable procedures, and conditionally including code
 - Optimization options to choose the degree to which PL/I optimizes their source code into efficient object code
 - Compiler options to request information or optional compiler facilities, setting these options as defaults at installation time and setting or resetting them at compile-time
 - The %INCLUDE statement to include code from external libraries.
- **Interlanguage Communication Facilities:** Allow PL/I procedures to call FORTRAN, COBOL, and Assembler programs/subroutines and PL/I procedures to be called by FORTRAN and COBOL programs/subroutines.
- **Large System Support:** OS PL/I Version 2 supports array subscripts in the range -2^{31} through $2^{31}-1$, AREAs and aggregates to a maximum size of $2^{31}-1$.
- **Programming Support:** PL/I's system and subsystem support gives programmers a variety of useful tools, including an interface to the PL/I MAIN procedure and interfaces to other subsystems through the SYSTEM option. Using PL/I improves programmer productivity through extensive storage allocation facilities, many data types and descriptions, modular structure, machine independence, and use of expressions.

PL/I generates reentrant code permitting PL/I programs to reside in DCSS (discontiguous shared segments) or installed as nucleus extensions.
- **Run-Time Control:** PL/I contains a number of run-time options to help programmers control storage and to aid in testing and error-handling. PL/I also supports a run-time user exit.
- **Access To Operating System Facilities:** PL/I gives programmers access to sort and data base facilities of the operating system.
- **Input and Output Facilities:** PL/I's I/O facilities allow programmers to choose among such factors as simplicity, machine independence, and efficiency. For example, a programmer can choose between stream-oriented and record-oriented I/O.

The Compiler

The OS PL/I Version 2 Optimizing Compiler compiles PL/I source statements into machine instructions. In some cases it inserts references to subroutines that are stored in the library. During link-editing, PL/I includes subroutines from the library in the object module. These subroutines can contain references to other subroutines stored in the library. During program processing, PL/I loads, processes, and deletes library subroutines as required.

The Library

The PL/I library provides the following functions:

- Mathematical subroutines
- Data type conversions
- Record- and stream-oriented transmission

- Edit-, list-, and data-directed I/O (stream-oriented transmission)
- Program initialization/termination
- Storage management
- Error handling
- Run-time message handling facilities
- File opening and closing.

VM/SP Callable Services Library

PL/I can call routines that reside in a callable services library (CSL). One such CSL library, called VMLIB, is supplied with VM/SP. Calls to CSL routines are not resolved until program execution. This lets you make changes to CSL routines without having to re-link the routine to the application program, recompile the program, or modify any of the program's source statements.

You can find more information about callable services libraries in the *VM/SP Application Development Guide for CMS* book.

Reference

The *VM/SP Application Development Guide for CMS* book, SC24-5286, has information about executing programs.

The *OS PL/I Version 2 General Information* book, GC26-4313, introduces installation managers, analysts, and programmers to the facilities available with the OS PL/I.

The *OS PL/I Version 2 Programming Guide*, SC26-4307, tells programmers how to use the OS PL/I under CMS to compile and run PL/I programs.

The *OS PL/I Version 2 Using PLITEST* book, SC26-4310, tells programmers how to test and debug PL/I programs under CMS.

VS Pascal

OVERVIEW

In today's data processing environment, VS Pascal is a likable structured programming language. Using VS Pascal produces reliable code by automatically performing many error detection checks.

The Language and The Compiler

The VS Pascal compiler and run-time library are shipped together as a licensed program, or you can order the library separately.

Features

Some features of VS Pascal are:

- MVS/Extended Architecture support. VS Pascal can produce application programs that execute in the extended address space above the 16-megabyte line. Now, large applications (using large tables of data) can be constructed without using the segmentation technique to fit a large program into available address space.
- Accepts source programs coded in Pascal that conform to the ANSI/IEEE 770 X3.97-1983 Pascal standard and Federal Information Processing Standard Publication (FIPS PUB) 109. FIPS PUB 109 adopts the language specification in the 1983 ANSI/IEEE Pascal language standard.

VS Pascal also has many extensions to the 1983 ANSI/IEEE Pascal standard, such as:

- The capability to develop programs modularly with separate compilation
- Support for varying length character strings
- I/O enhancements that include:
 - A SEEK procedure to specify the number of the next file component to be operated on by a GET or PUT operation (for random access)
 - A PDSIN and a PDSOUT procedure to open a member of a partitioned data set (under MVS) or MACLIB (under VM/CMS) for I/O
 - A TERMIN and a TERMOUT procedure to open a file for terminal I/O
 - An UPDATE procedure to open a file for both I/O (for updating).
- Other enhancements that support:
 - Integer hexadecimal constants, integer binary constants, floating-point hexadecimal constants, and string hexadecimal constants. Constant expressions can be used where the ANSI 1983 standard permits only constants.
 - Structured constants. The type of the constant is determined by the identifier type. These constants can be used in constant declarations, value declarations, or in executable statements.
 - RANGE values on the CASE statement and on variant records

- Execution of alternative statements on the CASE statement (OTHERWISE keyword).
- Helps programmer productivity through its *top-down* design. In this structure, the major functions of the program are identified first, followed by the functions that are needed to perform the major functions, and so on, until the lowest-level functions are identified. This program structure is easier to understand, fix, and maintain.
- Aids in program debugging and maintenance. VS Pascal provides a tool that lets programmers quickly debug its programs, without having to write debugging statements directly into their source code. Basic functions include: tracing program execution, viewing the values of program variables, breaking at intermediate points of execution, and displaying statement frequency counting information. You can use VS Pascal source names to refer to statements and data.

Using the VS Pascal trace facility during debug, you are given a list of all the routines in the procedure chain.

- Provides interlanguage communication. VS Pascal can call or be called by FORTRAN, COBOL, Assembler, and PL/I. This is useful for obtaining services not directly available in VS Pascal. This lets you take advantage of VS Pascal in an existing application without rewriting the entire application.
- Provides code optimization. The VS Pascal compiler reduces the execution time of the object programs it produces and, in doing so, reduces the storage used by the programs.
- Provides compiler directives. VS Pascal lets you specify libraries to be searched when compiling your program by using the %INCLUDE statement.

Compiler directives also let you control printed output, enable or disable the run-time checking features of VS Pascal, redefine left and right margins of the compiler input, and write messages to the terminal at a specified location in the program during compilation.

- Supports double-byte character set (DBCS) data.

VM/SP Callable Services Library

VS Pascal calls routines that reside in a callable services library (CSL). One such CSL library, called VMLIB, is supplied with VM/SP. Calls to CSL routines are not resolved until program execution. This allows you to make changes to CSL routines without having to re-link the routine to the application program, recompile the program, or modify any of the program's source statements.

You can find more information about callable services libraries in the *VM/SP Application Development Guide for CMS* book.

Reference

The *VS Pascal General Information* book, GC26-4318, provides an overview of VS Pascal.

The *VS Pascal Language Reference*, SC26-4320, describes the syntax and rules of the VS Pascal language.

The *VS Pascal Application Programming Guide*, SC26-4319, describes how to compile, link-edit, run, and debug VS Pascal programs.

The *VM/SP Application Development Guide for CMS* book, SC24-5286, has information about executing programs.

Assembler Language

OVERVIEW

Assembler is a symbolic language that closely maps (one-to-one) to the machine language instructions that result from it. System programmers are the primary users of assembler language.

All programming languages, other than Assembler and REXX, discussed in this book are high-level languages. Usually, each statement of a program written in one of these languages results in many machine language instructions being generated when the source program is *compiled*. High-level languages generally make writing programs easier and faster.

On the other hand, each source statement of an assembler language program usually results in one machine language instruction being generated when the source program is *assembled*. Writing programs in assembler language is usually more time-consuming and is generally considered more difficult than using high-level languages.

Assembler Fills Specific Programming Needs

Assembler language gives programmers a degree of program control that is not available to them when using the high-level languages. There are times when coding in assembler is a requirement for a particular application. System programmers use assembler language for doing such things as:

- Maintaining programs written in assembler language
- Coding functions at a byte or bit level, not provided for by a high-level language
- Writing routines, such as special interfaces, that must be coded in assembler language.

Macro instructions are available to assembler language programmers. A macro instruction is a program statement that results in a sequence of machine language instructions being generated, in much the same way a statement of a high-level programming language expands when compiled. Macro routines (definitions) provide a fast and easy way to program functions such as I/O and data management operations. You can code your own macro definitions and enter them from assembler programs.

The Assembler H Program

The Assembler H program product (5668-962) is ordered separately. This assembler runs under the control of CMS. It is compatible with earlier assemblers and will accept programs that have been processed on them with minor exceptions. Since system programmers can change the assembler to tailor it to the needs of their installation, you should check to identify which valid options can be used.

The assembler is run under CMS using the HASM command. Default values are given for all options of the HASM command, but alternative values are specified. The option default values are set to suit the needs of each installation.

The input to the assembler is a program written in assembler language. The file type of the source program file must be ASSEMBLE. This input is called a *source*

module. The output of the assembler has a program listing (LISTING file) and an object module (TEXT file).

The assembler has many diagnostic features to aid in the location and analysis of program errors. These include:

- Diagnostic messages for many stages of the assembly process; Assembler H has two stages
- A macro trace facility (MHELP statement)
- A specially formatted dump occurs when Assembler H detects a severe internal problem. This dump does not always appear when an assembly cannot be completed.

Reference

The *Assembler H Version 2 General Information* book, GC26-4035, describes the Assembler program, with emphasis on the most recent enhancements.

The *Assembler H Version 2 Application Programming Language Reference*, GC26-4037, describes how to use the assembler program.

The *Assembler H Version 2 Application Programming Guide*, SC26-4036, describes how to access the assembler in different environments and its messages.

VS FORTRAN

OVERVIEW

FORTRAN is derived from FORMula TRANslator. It is a mathematically oriented high-level programming language. Applications range from simple problem solving to large scale numerical calculations. The FORTRAN programming languages are used primarily for scientific and engineering applications.

The VS FORTRAN program product has added features that result in more effective and efficient FORTRAN programming.

The VS FORTRAN compiler, an execution-time library, and a debugger are shipped together as a program product.

Two levels of FORTRAN programs processed by the VS FORTRAN compiler are:

- FORTRAN 77 standard, plus IBM extensions
- FORTRAN 66 standard, plus IBM extensions.

The level is specified to the language processor at installation or compile time. In addition, VS FORTRAN implements IBM's Systems Application Architecture FORTRAN definition.

Features

Programmers can develop applications easily and efficiently with the following features:

- The VS FORTRAN compiler is reentrant and can generate reentrant object code. In addition, many of the library routines are reentrant. Therefore, the compiler or a compiled program or program part can be made resident with one copy serving the need of concurrent users.
- VS FORTRAN programs can be compiled on one of the supported VS operating systems, then link edited and run on any other supported VS operating system.
- The VS FORTRAN compiler uses the IBM 3090 Vector Facility to process programs faster.
- Improved I/O language gives you more control when processing FORTRAN files and more information about each record processed
- Using internal files, you can move data from one internal area to another while converting it from one format to another.
- Facilities for top-down program design and structured programming
- Various mathematical, character, and bit functions are available
- Service subroutines give you control over certain mathematical exceptions and over program termination when unusual conditions occur.
- During program processing, error handling subroutines issue messages if certain errors occur.
- VS FORTRAN uses the data-in-virtual facility on MVS/XA.
- Multitasking facility lets a single VS FORTRAN application program use several processors in a multiprocessing system simultaneously.

- The character data type gives you more adjustable control over specification and processing of nonnumeric data
- Constants are defined with symbolic names.
- Programs are written in free format, to avoid many of the coding restrictions associated with writing code in fixed format.
- Flagging is provided to help you indicate source language that does not conform to the language defined by IBM's Systems Application Architecture.
- Debugging *packets* can be specified at the beginning of a source program to give you run time diagnostic information that is easy to understand and use
- Diagnostic information issued by the compiler and by the execution time library is more precise and more informative than in the past.

Interactive Debug

Interactive Debug, a component of VS FORTRAN, lets you debug FORTRAN programs conversationally. Interactive Debug subcommands let programmers control, monitor, and change running FORTRAN programs. It results in:

- Simplified interaction with running FORTRAN programs
- Increased productivity through simpler debugging procedures
- More reliable code.

Interactive Debug lets you:

- Start and restart the program at selected points
- Examine and change values of variables, arrays, and array elements
- Display all or parts of a source program
- Trace program transfers
- Ask for online HELP information
- Issue system commands while debugging
- Locate errors, repair the problem, and continue debugging
- Identify the parts of the program that use the most processor time.

VM/SP Callable Services Library

VS FORTRAN can call routines that reside in a callable services library (CSL). One such CSL library, called VMLIB, is supplied with VM/SP. Calls to CSL routines are not resolved until program execution. This lets you make changes to CSL routines without having to re-link the routine to the application program, recompile the program, or modify any of the program's source statements.

You can find more information about callable services libraries in the *VM/SP Application Development Guide for CMS* book.

Reference

The *VM/SP Application Development Guide for CMS* book, SC24-5286, has information about executing programs.

The *VS FORTRAN Version 2: General Information* book, GC26-4219, has an overview of the product. It includes compatibility and requirements information.

The *VS FORTRAN Version 2: Programming Guide*, SC26-4222, provides guidance information about designing, coding, compiling, running, and debugging VS FORTRAN programs.

The *VS FORTRAN Version 2: Language and Library Reference*, SC26-4221, describes rules for coding source programs.

The *VS FORTRAN Version 2: Interactive Debug Guide and Reference*, SC26-4223, gives guidance on invoking and executing VS FORTRAN Interactive Debug.

APL2

OVERVIEW

The APL2 program product interprets the APL programming language and runs system commands and system functions to control the APL environment.

APL2 offers both interactive and batch processing services.

APL2 is a powerful facility for using a language for both business and scientific applications. Financial planners, analysts, statisticians, engineers, and other professionals use APL2 for many different types of applications.

Benefits

There are many benefits to APL2 users. They include:

- Fast problem solving and data analysis
- Interactive program development and production computing
- Error handling facilities to detect errors as they occur and resolve errors quickly
- Interfaces to a wide range of other IBM products
- Adjustable tailoring of applications to suit the user's individual needs.

Features and Facilities

In addition to providing a full-function programming language, APL2 provides interfaces to communicate with system services and other products outside of APL2. This lets APL users request special host-dependent tasks, such as external file I/O operations, and other services such as database and graphics. Examples of the functions include:

- Calling compiled programs written in VS FORTRAN or assembler
- Letting APL users enter CP and CMS commands without leaving APL mode
- Providing access to CMS files
- Letting APL users add entries to the CMS console stack
- Passing SQL statements from APL2 to SQL/DS
- Interacting with the Graphical Data Display Manager (GDDM) program product to control certain user display or printer terminals.

APL2 workspaces can be encapsulated into packaged form and placed outside the user machine to be shared by many users. The packaged workspace can also be placed in a DCSS.

APL2 has a session manager, which provides commands for conducting and controlling an APL2 session. The session manager offers: a session log, session manager commands, PF keys, full-screen management, and highlighting. Session manager commands let you copy work to a file, scroll through the session log to find a previously entered line, and control the content and format of the output display. The APL2 session manager requires the Graphical Data Display Manager (GDDM) program product.

APL-Related Products

The following products run on APL2 and provide a wide range of end-user function.

- DrawMaster (program number 5664-388) provides the ability to create high-quality graphic images.
- InfoCenter/1 (program number 5668-897) provides an end-user information center with query and report writing based on internal files or SQL/DS.
- Application Prototype Environment (program number 5668-808) is a *toolbox* of APL2 utilities for application development.
- Print Management Facility (program number 5664-310) provides utilities for creating fonts for 3800 printers.
- NETDA (Network Design Aid) (program number 5664-202) is a network configuration tool for VM.

Reference

The *APL2 General Information* book, GH20-9214, has an introductory description of APL2 for data processing managers, system designers, and application designers.

The *APL2 Installation and Customization under CMS* book, SH20-9221, discusses how to plan an APL2 installation, install and verify the installation, customize APL2, and administer the APL2 environment.

The *APL2 Migration Guide*, SH20-9215, assists you in migrating from VS APL to APL2.

IBM BASIC

OVERVIEW

The IBM BASIC language environment supplies you with all the tools needed for easy management of programming activities. The environment shields you from the underlying operating system structure. You get a response to selected IBM BASIC statements when entering the statements into the system.

By running IBM BASIC in batch mode, you compile and combine main and subprograms together, or dynamically combine these programs at run time. The processor and the library routines are reentrant; many programs can share one copy maintained in shared virtual storage.

IBM BASIC is an easy to learn general problem-solving language. It is a new-generation of the BASIC programming language, which supplies you with many powerful development capabilities. Some are reflected in the improved BASIC language and some by the environment surrounding the language.

While remaining simple and easy to use, IBM BASIC adds power and versatility to the language. It has a complete programming environment, including its own language-sensitive command set. It lets you intermix BASIC source statements with editing, debugging, and library management commands. IBM BASIC statements are grouped into a program or, at times, run immediately just like commands. The IBM BASIC environment is complete and self-contained, so you are generally free of operating system concerns.

Features

IBM BASIC has extensive capabilities for writing source programs using English-like statements. It is used for a broad range of applications whose purpose is not limited to a specific application environment, such as scientific, or business. Some of the features that make IBM BASIC a good general-purpose programming language are:

- Statements, such as the DO loop and IF/THEN/ELSE, help structured programming design and development.
- Variables and statement labels are given significant names of up to 40 characters.
- Versatile character string handling provides for extracting, inserting, replacing, or moving character strings or substrings of variable lengths.
- Arithmetic and relational operators allow much control and comparison of numeric data.
- The IMAGE and FORM statements give you a method of representing I/O data in many formats.
- Programs can be segmented. They can also be developed in a main program and subprograms format.
- The INPUT FIELDS statement and the PRINT FIELDS statement provide for programmed control of terminal I/O.
- Arrays of up to seven dimensions are permitted.
- Programs are developed, edited, compiled, debugged, and run interactively. Debugged programs can be compiled and run in batch mode.

- A HELP command is available to display information about IBM BASIC commands, statements, and so forth.
- Easy access to system functions, such as MERGE, FETCH, or RUN.
- Some statements are run immediately when entered, for tasks such as doing calculations and program debugging.
- Many integrated (intrinsic) functions are also provided. For example, SQR(X) for calculating the square root of X, or TIME\$ for time of day (HH:MM:SS).

Reference

The *IBM BASIC General Information* book, GC26-4023, gives a general description of the product.

The *IBM BASIC Programming Guide*, SC26-4027, has information for designing, coding, running, and debugging IBM BASIC programs.

The *IBM BASIC/VM System Services* book, SC26-4028, describes the usage of the IBM BASIC processor, including related commands.

Chapter 8. Distributed Data Processing Using VM/SP

Distributed data processing (DDP) is data processing that spreads the processing facilities over many sites. Two major factors have contributed to the growth of distributed processing under VM/SP:

- The relative cost of hardware has declined. This encourages additional facilities to be located where they are most accessible by those who need them.
- VM related licensed programs are easy to use. This means that all data processing work does not need to be done by experts at the *home office*.

This chapter introduces some of the concepts, facilities, and terminology associated with DDP.

Some DDP Definitions

Terms used in this chapter that might be new to you are:

Central site. The network site responsible for providing network management and support services.

DDP network. A distributed data processing environment where the many sites are connected by telecommunications media.

Distributed data processing (DDP). Data processing where the hardware and software facilities are spread over many sites.

Distributed or remote system. The system(s) that the central site manages (operates, administrates, and maintains). Each system depends on the central site staff and systems for network and system support. Remote or distribution systems have little or no skills or resources for computer operations or software maintenance.

Why Distributed Data Processing?

OVERVIEW

Distributed data processing (DDP) is a cost-effective method of processing data for a multiple-site enterprise. Some advantages are:

- DDP puts facilities close to the people who need them
- Tasks that need special skills are done by support people at a central site
- Many processing sites enable better work flow versatility
- A DDP network has more versatile growth options than central site processing
- VM/SP supports many processors that are used in the DDP environment
- VM/SP, together with other related licensed programs, has an excellent, easy-to-use DDP environment.

On-Site Data Processing Has Its Advantages

Today it is common for companies to have operating units that send information over an entire country, or even throughout the world. In supplying for the data processing needs of such companies, there are advantages to having each site maintain its own processing facilities. For example:

- On-site processing puts facilities close to the people who are most involved with the data being processed.
- The smaller distributed systems often fit the needs of local operations better than one central system attempting to satisfy the needs of many sites.
- Distributed systems have more versatile growth patterns.
- Delays associated with shipping data to a central facility for processing are avoided by doing the processing on-site.

As shown in Figure 33 on page 187, Raleigh is the central site and Philadelphia, St. Louis, Dallas, San Diego, and Chicago are all remote systems.

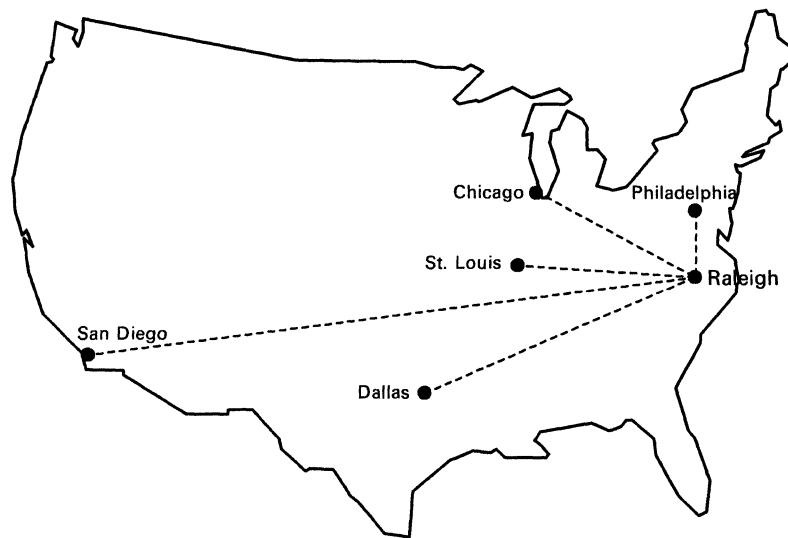


Figure 33. Sample Setup of a Distributed Environment

Centralized Processing Also Has Advantages

On the other hand, there are also advantages to having data processing for all sites work as a unit. For example:

- Fewer skilled support people, such as system programmers, serve the needs of all operating units
- Operating procedures are easier to manage
- Reduced dependence on commercial communications facilities.

A DDP Network Offers the Best of Both Methods

The DDP network is a solution that offers the advantages of distributed processing facilities working as a unit. In addition to all the advantages listed earlier for both centralized processing and on-site processing, a DDP network offers versatility of resource use. Workload is easily sent from one network site to another. In the DDP network environment, most of the network management, operation, and servicing is done efficiently by a few support people at a central site.

In a DDP environment, there are facilities available that allow for:

- Reducing message traffic through filtering, using the Programmable Operator Facility and NetView Message Automation Table
- Automating operations
 - Starting and stopping service machines
 - Autologging service machines
 - Managing logon and warning messages
 - Automatically changing service machine passwords
 - Managing spool files and spool space full conditions
 - Backing up user and system data.

- Remote operations
 - Check status of service machines
 - Monitor remote system activity from the central site by creating and viewing system files.
- Remote system administration facilities manage user IDs, minidisks, and the CP directory using VM/Interactive Productivity Facility (VM/IPF).
- Change Distribution
 - VM/Distributed Systems Node Executive (DSNX) distributes changes from the central site to any or all remote systems, installs the change(s) automatically, and builds nuclei and resaves CMS if needed.
 - NetView Distribution Manager (NetView DM) provides services for centrally controlled data distribution and the implementation of software changes in SNA networks.

Reference

The *VM/SP Distributed Data Processing Guide*, SC24-5241, describes tasks relating to the planning and use of distributed data processing.

The *VM/IS How to Support Your Distribution System*, SC24-5355, describes and illustrates procedures to help you create, install, automate operations of, manage, or maintain distributed VM systems.

Elements of a VM/SP DDP Network

OVERVIEW

The basic parts of a distributed data processing network are:

- Hardware facilities at the central site and at distributed systems (nodes)
- Programs to manage data flow among nodes
- Transmission media (data links) for carrying data between nodes.

A major advantage of the distributed data processing (DDP) network is that many of the more costly resources, including support people such as system operators and system programmers, are needed at only one of the network sites. Facilities are available to allow performance of tasks such as system start up and problem diagnosis for remote systems. The network site at which the group of network support people work is called the *central site*. Other sites in the network are called *distributed systems*.

The central site and the distributed systems both have their own hardware, software, and personnel requirements that must be considered when planning a DDP network. The extent of these resource requirements depends on what the processing capabilities of the network will be.

Processor features and VM/SP facilities, plus related programs form a foundation for a VM/SP DDP network. These include:

- Processor and VM/SP
 - IBM 4300 Remote Operator Console Facility (ROCF)
 - IBM 9370 Processor Remote Operator Facility (ROF)
 - Programmable Operator Facility
 - Inter-System Control Facility.
- Program products
 - VM/Distributed Systems Node Executive (DSNX)
 - NetView Distribution Manager (NetView DM)
 - VMBACKUP Management Subsystem (VMBACKUP-MS)
 - VM/Pass-Through
 - RSCS Networking
 - Advanced Communications Facility for Virtual Telecommunications Access Method (ACF/VTAM) for VM/SP with VM SNA Console Support (VSCS)
 - NetView Release 2
 - VM/IPF.

Programming requirements for DDP planning are release sensitive. RSCS as used here, refers to Version 2 or later.

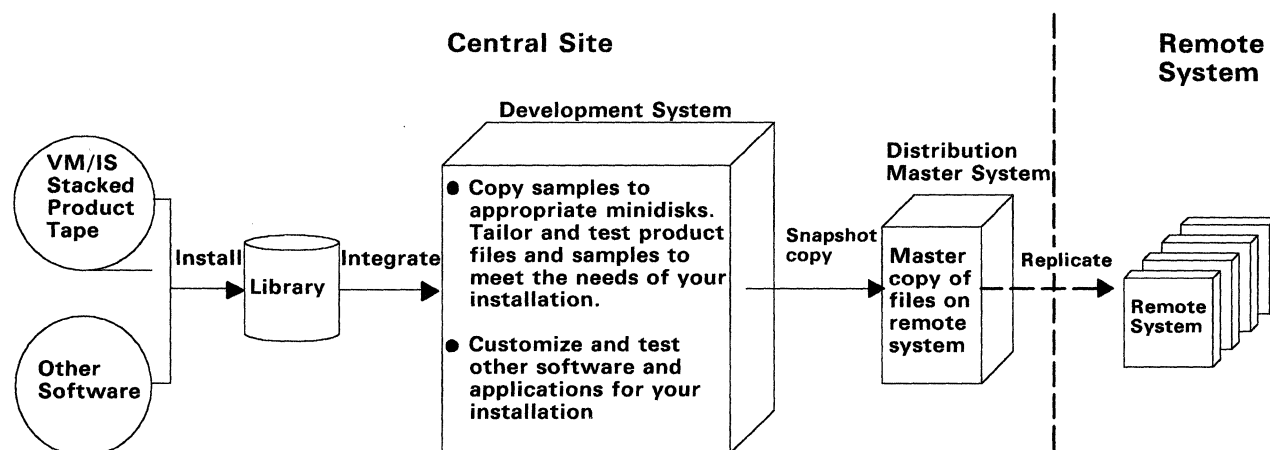


Figure 34. Sample Setup of a Central Site

System Requirements for the Central Site

Tasks done at the central site in a DDP network, together with any programs and system facilities needed to do these tasks, are as follows:

- Perform data processing on a local system
 - Any VM/SP supported processor might fill this need, but some items in this list require that the processor be an IBM 4300.
- Install processors and operating systems at the distributed systems from the central site system
 - The distributed processors installed in this way are IBM 4300 processors with Remote Operator Console Facility (ROCF). The central site operator installs this procedure through one of the following:
 - An emulated IBM 3275-2 Display Station on the central site processor. Emulation is provided by VM/Pass-Through.
 - An IBM 3275-2 Display Station
 - Each site must have the proper data transmission facilities (for example, modem and transmission lines.)
- Log on to and use systems at other network sites
 - VM/Pass-Through
 - ACF/VTAM for VM/SP
- Send certain commands to distributed systems or respond to certain messages that start at distributed systems, but must be handled by the central site
 - Programmable Operator Facility
 - NetView Release 2
 - RSCS Networking.
- Send messages and files to network sites
 - RSCS Networking.

- Manage operations, access to resources, and problem data collection in SNA networks
 - NetView Release 2.
- Perform problem diagnostics on distributed systems
 - IBM 4300 processor Remote Support Facility
 - NetView Release 2.

Resource Requirements for the Distributed Systems

In the DDP environment, network management tasks are handled at the central site. Therefore, the list of tasks done by users at distributed sites is understandably shorter than that for the central site.

In a DDP environment where the central site manages and maintains the distributed system, the following are the tasks of the system user at the remote system:

- Uses the system to complete daily tasks
- Mounts tapes
- Adds printer paper
- Communicates problem descriptions to the support personnel at the central site.

Also, you can reduce DASD requirements for your distributed systems by selectively *pruning out* or *trimming* portions of software that are either unnecessary to the distributed system or already provided by the distributed system's central site.

Mixed VM SNA and Non-VM SNA Networks

A mixed SNA network consisting of VM, MVS, and VSE nodes is also possible. For VM/SP, these nodes communicate using ACF/VTAM Version 3.

Reference

The *VM/SP Distributed Data Processing Guide*, SC24-5241, describes distributed data processing network requirements.

The *Systems Network Architecture Concepts and Products* book, GC30-3072, is an introduction to the benefits and concepts of a Systems Network Architecture (SNA) network. It also discusses IBM products that run in the SNA network.

The *Systems Network Architecture Technical Overview* book, GC30-3073, describes the major functions of the SNA network.

The *ACF/VTAM General Information* book, GC38-0254, describes ACF/VTAM, SNA concepts, installing and starting ACF/VTAM, commands, and ACF/VTAM application programs.

Reducing Message Traffic in a Distributed Network

OVERVIEW

Message traffic is reduced through filtering by using the Programmable Operator Facility and the NetView Automation Table.

The Programmable Operator Facility lets your virtual console receive messages or requests and handles them with preprogrammed actions.

Using the message automation table along with NetView CLISTs, you can define NetView to handle the many types of system and network operations. With the automation table, you can log all messages (even those not logged) and convert the messages into NetView alerts when necessary.

Using the Programmable Operator Facility for Message Filtering

The Programmable Operator Facility is a part of VM/SP. It runs in a CMS virtual machine and

- Increases efficiency in:
 - A single VM/SP system configuration
 - Both the central site and the distributed systems of a VM/SP network
 - The SNA environment with mixed VM, OS/VS, and VSE systems.
- Improves system operation efficiency by:
 - Handling routine system operator tasks automatically
 - Letting some operator tasks for distributed systems on a VM/SP network be done by an operator at a central site.
- Uses the facilities of RSCS Networking program product when doing tasks in a VM network environment
- Uses the facilities of the NetView Message Queuing Service when doing tasks in a mixed environment.

The programmable operator processes incoming messages according to entries in the *routing table*.

The programmable operator must have information available to help it do its tasks. This includes authorization for you to enter programmable operator commands. The source of this information is one or more CMS files called routing tables. One sample routing table is supplied with the Programmable Operator Facility, which can be edited to suit the needs of your system. Others can be produced for special purposes.

- The programmable operator at the central site:
 - Filters and logs messages that do not need the system operator's immediate attention. For example, it handles messages that show an operator originated action was successfully finished.
 - Routes certain messages to the person responsible for taking action on those messages. For example, messages that show some action must be taken on an I/O device.

- Automatically enters appropriate programmed responses to certain situations. For example, relocating information on direct access areas that are beginning to fill.
- The programmable operator at distributed systems:
 - Responds to messages and does tasks that do not need an on-site operator. For example, it stores user comments in the programmable operator feedback file.
 - Filters and logs messages that do not need the system operator's attention. For example, messages associated with users logging on to and off the system.
 - Routes messages that need on-site handling, such as mounting a magnetic tape, to the proper person to take action.
 - Routes messages that need handling by central support personnel to the central site. For example, a user at a distributed system requesting the time that the system will be shut down by the central site operator.

In a mixed environment, the NetView PROP command enters the programmable operator and VM commands to the programmable operator from a NetView operator's terminal. Messages are sent through the Programmable Operator/NCCF Message Exchange (PMX) part of the programmable operator. Commands to be entered are input as parameters to the NetView PROP command.

Using NetView for Message Filtering

The NetView message automation table is a standard facility of NetView that lets you specify message processing options such as:

- Selection criteria
- Actions
 - Message suppression from NetView terminals
 - Message distribution
 - Command or CLIST execution.

You can start the message automation activity by looking at a VM log and a NetView log and automating those operator action messages that require:

- Same simple reply every time
- Entry of a sequence of commands.

Reference

The *VM System Facilities for Programming* book, SC24-5288, describes the programmable operator, routing table, commands, and action routines. This book also explains how to install and use the programmable operator in single, distributed, and mixed environments.

Automated and Remote Operations in Distributed Networks

OVERVIEW

In a DDP environment, you want to automate as much as possible or automate remotely using NetView, the Programmable Operator Facility, or samples and examples.

NetView

NetView is an application program that lets you control, record, and automate various network and system operator tasks. It also provides many command lists (CLISTs) which simplify common tasks that the operator frequently executes. Online access that summarizes the status of all resource types within a domain (status monitor) and doing problem determination is also provided.

You can automate some of your network operations so that operators are not required at the remote systems and implement procedures for NetView to handle these unattended operations.

Programmable Operator Facility

This facility is designed to increase the efficiency of system operation and lets remote systems operate in a distributed environment.

Samples and Examples

Sample execs help you support distributed systems. Using these execs and VM/DSNX's example application samples, you can automate network and system operations, remotely administer distributed systems, and efficiently update and maintain distributed systems. These are available with the VM/Integrated System (VM/IS) system.

Reference

The *VM System Facilities for Programming* book, SC24-5288, describes the programmable operator, routing table, commands, and action routines. This book also explains how to install and use the programmable operator in single, distributed, and mixed environments.

The *NetView Installation and Administration Guide*, SC30-3476, helps system programmers install and prepare NetView for operation.

The *NetView Operations* book, SC30-3364, helps system programmers interpret and respond to the error messages issued by NetView.

The *Automated Operations Planning Guide*, SC30-3474, assists managers and technical staff who are responsible in planning for automated console operations.

The *Automated Operations Using NetView Command Lists* book, SC30-3477, is designed to help system programmers and network operators write NetView command lists (CLISTs).

The *VM/IS How to Support Your Distribution System* book, SC24-5355, describes and illustrates procedures to help you create, install, automate operations of, manage, or maintain distributed VM systems.

Distributing Changes to Remote Systems

OVERVIEW

Software maintenance is provided to your remote systems by your central site. Using the VM/DSNX example application, the central site system programmer can distribute a new or changed product to any or all of your remote systems.

NetView DM provides central site services for managing SNA communication networks.

VM/Distributed Systems Node Executive

Software maintenance is provided on your remote systems from the central site. If a product needs to be uplevelled or a software change is required, your personnel at the central site can develop and test the software change before distributing it to your remote systems.

NetView Distribution Manager

NetView DM assists you in centrally controlling data object distribution and in implementing software changes in SNA networks.

Some functions provided by NetView DM are:

- Storing and tracking data objects
- Tracking and defining software objects
- Identifying and authorizing central site users of this service
- Storing transmission plans.

Reference

The *VM/DSNX Guide and Reference*, SC24-5381, gives you more information on VM/DSNX.

The *NetView Distribution Manager General Information* book, GH19-6587, introduces NetView Distribution Manager and its benefits.

The *VM/IS How to Support Your Distribution System*, SC24-5355, describes and illustrates procedures to help you create, install, automate operations of, manage, or maintain distributed VM systems.

Chapter 9. Extending the Capabilities of VM

This chapter discusses the following licensed programs that extend the capabilities of VM.

- VM/Interactive Productivity Facility
- VM/Directory Maintenance
- VM/SP High Performance Option (VM/SP only)
- VM/Extended Architecture System Product.

VM/Interactive Productivity Facility

OVERVIEW

The VM/Interactive Productivity Facility (VM/IPF) provides a protected environment for the VM/SP user. Menu and Data Entry task panels are presented to assist you in performing VM/SP tasks. In addition, HELP panels are readily available for each Menu and Data Entry panel.

The VM/IPF licensed program runs in a CMS virtual machine. It lets you do VM/SP tasks without having to remember or look up related commands. VM/IPF is usually started by entering the DTRIPF command, either manually under CMS, or as part of an automated IPL procedure.

There are four different panel environments in VM/IPF. They are:

- General Use
- Operation
- Administration
- Maintenance.

Users are authorized to use particular environments when they are enrolled as VM/IPF users. Each environment determines the kinds of tasks you are allowed to perform.

Task selection in VM/IPF is easy. From a menu containing numbered selections, you type a number that corresponds to a task and press the ENTER key. Subsequent panels are presented to prompt you for additional information needed to complete the task. For many panels, VM/IPF will automatically position the cursor for keying in data. For other panels, the tab key quickly positions the cursor.

Figure 35 on page 199 shows a typical VM/IPF menu panel containing a list of VM/SP tasks. To use a magnetic tape, for example, you begin by typing the number 6 on the panel command line (==>). Then by supplying the requested information as subsequent panels are displayed, you can complete the task without knowing or issuing any VM/SP commands. To do this task without VM/IPF, you must know how to issue VM/SP commands, and which VM/SP commands to issue.

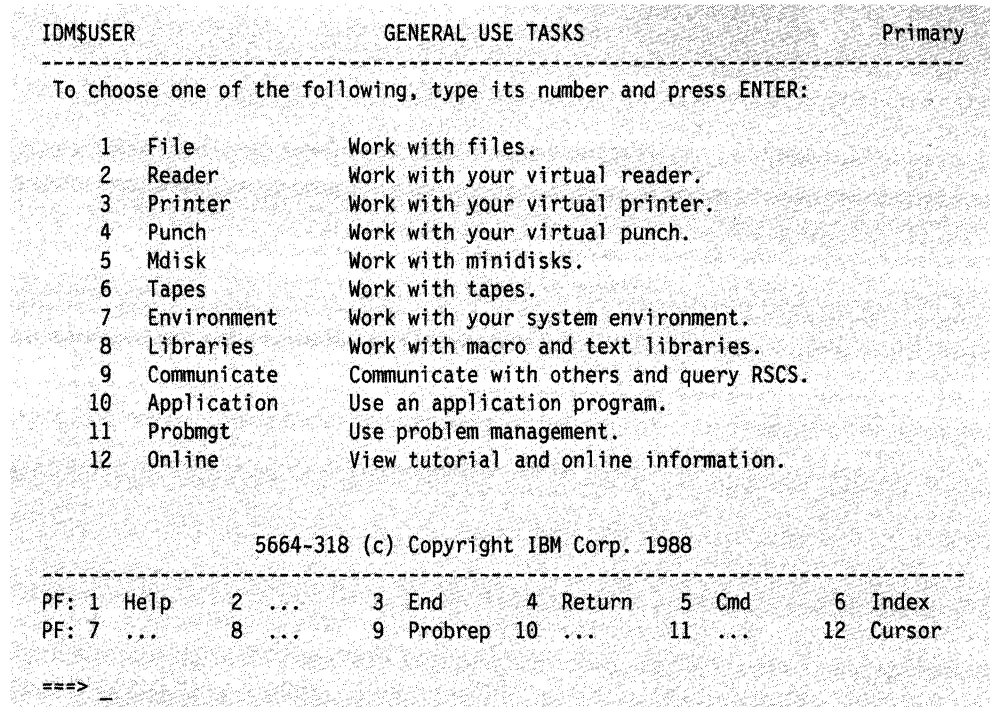


Figure 35. VM/IPF General Use Task menu panel

VM/IPF uses default values for many panel fields if they are not filled in. Under VM/IPF, program function keys are pre-set to do useful functions that make the facility easy to use. VM/IPF displays an abbreviated description of these keys at the bottom of each panel.

VM/IPF uses the Problem Control Facility (PCF) for handling many kinds of problems. PCF offers a panel-assisted way of recording and reporting problems, and sending them to the appropriate person. Problem reports can be generated and distributed within the same system, or, using facilities of the RSCS Networking licensed program, sent to other VM/SP systems that have VM/IPF installed.

For VM/IPF dialogs to work, the Interactive System Productivity Facility (ISPF) licensed program must be installed. ISPF provides dialog management services needed for the display and operation of VM/IPF panels. Refer to the *ISPF Dialog Management Services and Examples* book for additional information.

Reference

The *VM/IPF Operation* book, SC24-5319, describes how to use the VM/IPF panels to do system operator tasks.

The *VM/IPF Administration Guide*, SC24-5320, describes how to use the VM/IPF panels to do system administrator tasks.

The *VM/IPF General Use Guide*, SC24-5325, describes how to use the VM/IPF panels to do general user tasks.

The *ISPF Version 2 for VM/SP Dialog Management Services and Examples* book, SC34-4010, describes the services provided by ISPF.

VM/Directory Maintenance

OVERVIEW

The VM/Directory Maintenance program product offers:

- Directory update capability for the system administrator and the general user
- Optional enforcement of regular password changes
- Updating the directory using a controlled service virtual machine to reduce the chances of errors
- Improved directory availability if a program interrupt or a disabled wait state occurs
- Prevention of giving new minidisk space over existing minidisk extents
- Ability of the general users to make simple directory changes for themselves
- HELP panels.

Maintenance of the VM/SP system directory can be time-consuming and error prone. The VM/Directory Maintenance program product handles this task easily and efficiently. Using Directory Maintenance commands, changes to the VM/SP system directory are made interactively. The alternative, manually updating the directory, takes longer and is more likely to cause human error.

Directory Maintenance has two classes of commands: general user and system administrator.

By letting general users make simple updates to the directory, it reduces the necessity of doing full updates. Updating areas of the directory that need special access authorization is restricted to administrators, who have a privileged class of commands available to them for that purpose.

Directory Maintenance Facility System Requirements

Using Directory Maintenance calls for two virtual machines:

1. DIRMAINT - the primary directory manager.
2. DATAMOVE - copies and formats functions.

These virtual machines usually run in a disconnected state.

Directory Maintenance requests are accessed as the following file modes:

- File mode A.** Contains the current directory source file, a history file, and control files.
- File mode B.** Contains a directory source backup file, plus a journal of changes since the last backup was created.
- File mode C.** Contains Directory Maintenance programs.

Directory Maintenance from the User's Point of View

Requests are communicated to the directory maintenance virtual machines using options of the DIRMAINT command under CMS (see Figure 36). When the requested function is completed, your virtual machine receives a completion message.

A general user is authorized to use only a subset of the DIRMAINT command options. You request changes that affect virtual machine characteristics, such as password changes or automatic IPL. You cannot request changes that affect system resources, such as improving dispatch priority or changing virtual disk allocation. The system administrator is authorized to make directory changes that affect system resources, using DIRMAINT command options not available to the general user.

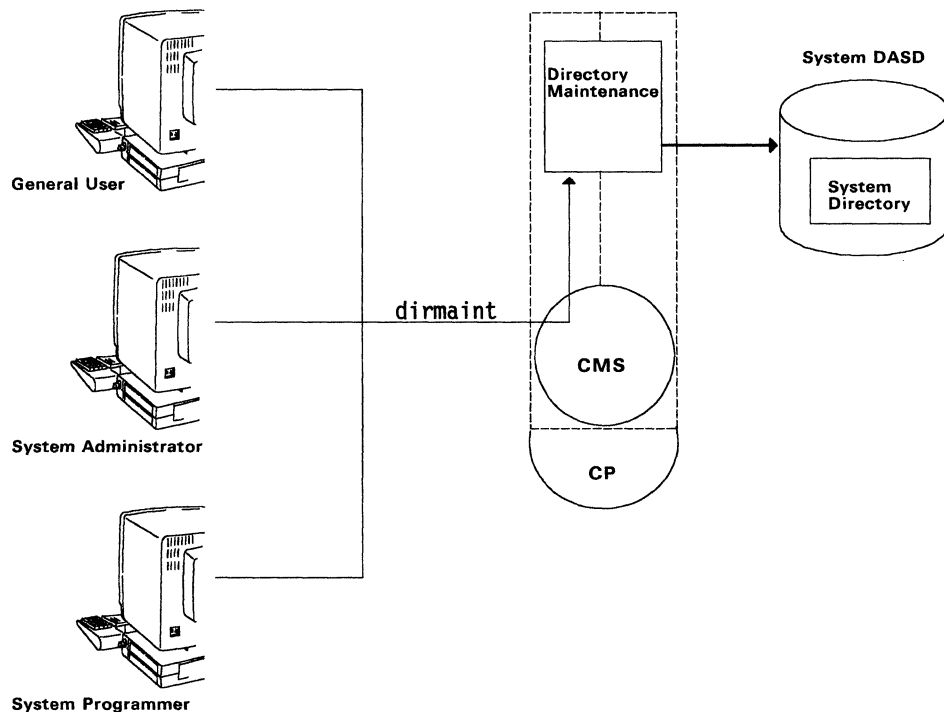


Figure 36. Updating the VM/SP System Directory

Reference

The *VM/Directory Maintenance General Information* book, GC20-1836, has an overview of the program.

The *VM/Directory Maintenance Guide for General Users*, SC20-1839, describes how to use the program.

The *VM/Directory Maintenance Installation and System Administrator's Guide*, SC20-1840, describes how to install, customize, and maintain the program.

The *VM/SP Planning Guide and Reference*, SC19-6201, describes VM/SP system directory entries for Directory Maintenance.

VM/SP High Performance Option

OVERVIEW

The VM/System Product High Performance Option (VM/SP HPO) licensed program extends the functional capabilities of the CP component of VM/SP. VM/SP is a required program for running VM/SP HPO.

What Does VM/SP HPO Add to VM/SP?

VM/SP HPO extends the capabilities of CP in the following areas:

- Improved system management
- Additional I/O support
- Additional processor support
- Improved performance and function for MVS/SP, VSE/AF, and VSE/SP guests
- Vector Facility support.

Improved System Management

Dispatching and scheduling modifications: VM/SP HPO uses a dispatch list, a dispatcher fast path, and scheduler design changes to improve its efficiency.

The dispatch list has only those virtual machines that are dispatchable, unlike the run list, which has all virtual machines, some of which are not executable. The control program scheduler still maintains the run list, but the dispatcher searches only the dispatch list when selecting a virtual machine to run.

The dispatcher fast path lets some virtual machines, such as CMS, be quickly redispached after events that interrupt the performance of the virtual machine, but do not change its status. This function is only available on attached processor (AP) and multi-processor (MP) systems.

The scheduler design changes maintain interactive response times under more varied system loads.

Free storage management improvements: VM/SP HPO manages free storage subpools more efficiently. Subpools are fixed-size storage groups. They are used to satisfy control program requests for free storage when CP needs to do system-related functions such as building control blocks, processing I/O operations, or building save areas.

Enhanced Paging Subsystem support: This support improves paging performance by using main storage as a high-speed buffer and by grouping together user pages that are likely to be used together. These groups of user pages are swapped in and out of main storage with one start I/O (SIO) operation. In addition, the system programmer uses the SYSPAG macro to use DASD more efficiently.

Expanded Storage support: Expanded Storage is a paging assist on the 3090 processor. The control program of VM/SP HPO uses it as a high-speed paging area.

N-select: N-select is a DASD page-slot selection algorithm that gives a selected number of consecutive requests to the same allocation area of DASD. This lets CP write multiple pages to the same device with a single SIO.

Active Wait: In attached processor (AP) or multi-processor (MP) systems, work becomes available without an accompanying interrupt. When one processor becomes idle, it goes into an active wait.

Active Wait lets an idle processor in an AP or MP system scan the dispatch request queues and dispatch lists looking for work.

SET RESERVE for multiple users: The system operator uses the SET RESERVE command to reserve real storage for multiple virtual machines. When a virtual machine has this option, during its execution, CP dynamically builds a set of reserved page frames until the maximum number *reserved* is reached.

The SET RESERVE command increases the efficiency of certain noninteractive virtual machines such as system control programs and special service machines.

Vector Facility: The Vector Facility is an instruction processor capable of high-speed manipulation of values, usually floating-point values. VM/SP HPO supports the Vector Facility in System/370 mode when it is available on a processor. A Vector Facility attaches to only one processor, but each processor in a processor complex can have a Vector Facility.

Capability for 9900 spool files per user: Subject to limited checkpoint space and SYSSPOOL's virtual storage size, an installation supports up to 9900 spool files per user. Recovery of spool files during checkpoint and force starts is redesigned by starting multiple tasks that overlap both processing and I/O activity.

Additional I/O Support

IBM 3880 Model 11 and 21 storage subsystems support: Excessive paging demands cause the sending of active pages from real storage to auxiliary storage. By transferring active pages between real storage and a high-speed buffer area called the *cache*, frequently used pages are kept more readily available to the system.

IBM 3880 Model 13 and 23 storage subsystems support: This support is for nonpaging applications such as MVS or CMS guest virtual machines running with 3380 DASDs. The performance improvement comes by maximizing the number of read accesses that are satisfied by accessing the cache (subsystem storage) copy.

Extended channel support: This lets an installation configure its resources over 48 channels for a 3090 processor.

Extended virtual device support: This extends the number of virtual devices that your system supports for a single user. The exact number of virtual devices that you define depends on the release of VM/SP HPO that you are using.

Additional Processor Support

Extended Storage support: This feature lets CP in VM/SP HPO manage processor storage that exceeds 16 megabytes. CP uses the storage above 16 megabytes for virtual machine pages. This results in improved productivity for systems that were previously constrained by limited availability of real storage.

Dyadic and Dual Processors support: The performance on dyadic and dual processors with store-in caches improves by implementing control block alignment, local dispatcher and free-storage queues for each processor, and dispatcher queue scanning as described below:

- Frequently used control blocks are aligned on cache-line boundaries
- Dispatcher and free-storage queues are provided for each processor. Thus, two processors modify and reference the same storage areas less frequently.
- When a processor finishes its dispatchable work, it scans the dispatcher queues of the other processor instead of going into a wait state. It is no longer necessary for the other processor to signal the idle processor when there is a piece of work for idle processor to do.

Improved Performance and Function for MVS/SP, VSE/AF, or VSE/SP Guests

Preferred Machine Assist support: VM/SP HPO supports the preferred machine assist feature. Preferred machine assist improves performance for the MVS/SP, VSE/AF, or VSE/SP V=R preferred guests by reducing control program processing.

As a preferred guest, MVS/SP, VSE/AF, or VSE/SP runs in supervisor state with direct control of its hardware resources and I/O operations. The preferred guest also uses storage greater than 16 megabytes. This has even more performance potential for storage-constrained MVS/SP, VSE/AF, or VSE/SP systems.

Preferred machine assist also removes restrictions for using single processor mode.

Control switch assist extensions to preferred machine assist: This support lets the MVS/SP preferred guest use IUCV, some DIAGNOSE instructions, and some Service Call instructions. It also reduces line timeout problems for such guests by letting CP reflect virtual I/O interruptions to the guest. You must install the control switch assist to get this support.

Enhanced availability in the MVS/SP V=R environment: VM/SP HPO attempts to save the status of the V=R (virtual addresses=real addresses) virtual machine which it is running, if an abnormal end of CP occurs. When CP is automatically restarted, running in the MVS/SP V=R virtual machine resumes. When the MVS/SP virtual machine is operating with preferred machine assist active, additional recovery capability is provided.

3033 Extension Feature (Virtual Machine Assist enhancement): This enhancement improves performance by reducing paging operations, improving real storage management, and increasing the performance of cross-memory services and page fault processing.

Single processor mode operational enhancements: Single processor mode lets you restrict the VM/SP HPO control program to a single processor in an AP or MP system, leaving the other processor for the exclusive use of the MVS/SP virtual machine. These enhancements make it easy to switch between single processor mode and either AP or MP mode.

Reference

The VM/SP High Performance Option licensed program is documented in its own library that is similar to the VM/SP library.

The *VM/SP High Performance Option Library Guide, Glossary, and Master Index*, GC23-0187, gives an overview of the library.

VM/Extended Architecture System Product

OVERVIEW

VM/Extended Architecture System Product (VM/XA SP) uses 370-XA architecture.

VM/XA SP is a VM operating system that is very similar to other IBM VM systems. Unlike System/370 mode VM systems (such as VM/SP and VM/SP HPO), VM/XA SP runs on 370-XA processors in 370-XA mode. This allows VM/XA SP to exploit 370-XA architectural features such as: large addressing, the 370-XA channel subsystem, and the Vector Facility. At the same time, VM/XA SP remains highly compatible with other IBM VM systems.

Like other IBM VM systems, VM/XA SP is an interactive system that provides file management, and an English-like command language, and a full-screen editor. A wide range of application programs can complement VM/XA SP's basic functions. As with other IBM VM systems, VM/XA SP lets you do:

- Problem solving
- Editing and text creation
- Program development
- Application testing
- System program testing.

Reference

The *VM/XA SP General Information* book, GC23-0362, describes the operating system at an introductory level.

Chapter 10. VM/Integrated System

VM/Integrated System (VM/IS) is a powerful and responsive computing system for business, engineering, and scientific professionals. It is a general purpose system that helps improve the productivity in the many daily activities of your organization.

VM/Integrated System

OVERVIEW

The VM/Integrated System (VM/IS) is based on VM/SP and provides a number of predefined system configurations that can be selected for installation. This concept frees you from the complex task of designing your configuration, reduces the skill level, and decreases the time you need to install your system.

VM/IS also has many software packages that contain a more specialized product set geared to a specific area of use. You can add many combinations of these packages to your base. These packages act as building blocks; you select only those products you need.

Installing VM/IS

VM/IS supplies several predefined system configurations for you. During the installation process, you choose one of these configurations.

The prepackaged software of VM/IS comes with its own installation procedure. This procedure is specifically for VM/IS and reduces the number of steps and decisions you must make to install the system.

The VM/IS documentation walks you through the installation process. It gives step-by-step instructions that tell you what to do and how the system will respond.

VM/IS makes its functions and products available to you by a menu interface, which describes the tasks you can do using VM/IS. This interface is a series of panels and menus that appear on your terminal display. These panels and menus help you select the correct product to complete a specific task.

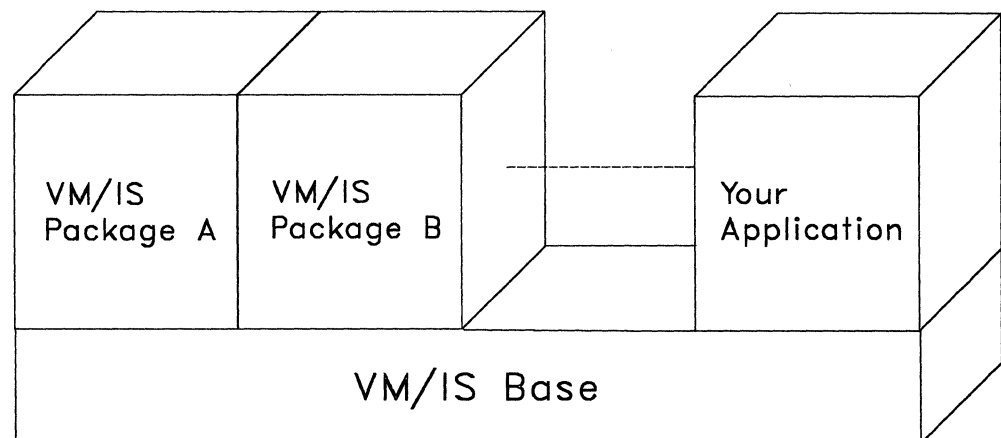


Figure 37. The VM/IS Base

Using the Base Products

The base products let you create documents, produce business graphics, manage your files, and send and monitor batch jobs. It acts as a foundation for your VM/IS system to use as a building block for the computer system you need. Without the base, you could not install and use the packaged products; however, the base is functional without the packaged products.

Creating Documents

VM/IS provides tools to help you format documents.

You create text files that have both the text of the document and *markup* that describes how you want the document to look. You then format the file using a text formatter which produces the finished product. The formatted document is sent to a printer or sent to other users for review.

Creating Graphics

VM/IS also has tools that help you produce diagrams and charts suitable for communicating business information.

You can produce a variety of different chart graphics that provide a visual representation of large quantities of complex data. The graphics can be displayed at a terminal or workstation, or they can be printed for distribution to others.

Using the VM/IS Packaged Products

The VM/IS packaged products include many software products that you can use to do different tasks. They let you achieve many combinations. The following are some of the tasks you can do:

- Administrative Support
 - Process schedules
 - Add automatic reminders
 - Handle mail
 - Process notes and messages
 - Prepare documents
 - Proofread documents
 - File and retrieve documents.
- Project Management
 - Collect and manage data
 - Retrieve and analyze information
 - Produce comprehensive business reports
 - Present information graphically
 - Use statistical analyses and forecasting packages
 - Develop business plans and evaluate alternatives
 - Manage projects with tools—like network drawings and resource allocation
 - Create and format letters and large documents.

- Network Development
 - Increase your communication with other sites
 - Transmit correspondence electronically.
- Communication Support
 - Send letters and documents electronically
 - Log on and use remote systems.
- Data Management
 - Produce a data base to share data and improve data integrity and control
 - Get data from the data base
 - Produce reports in many formats
 - Do calculations on the data to reflect totals, subtotals, averages, maximums, and minimums in your reports
 - Add, delete, or change data in the data base
 - Extract data to format for loading into other types of data bases or for storage into various types of files.
- Business Computing
 - Solve everyday business problems
 - Meet your data processing needs
 - Process requests or transactions by accessing shared data.
- Intelligent Workstation Support
 - Create a file on your 3270 PC and send it to your personal storage
 - Create a file on the VM/IS system and send it to your 3270 PC for revising or printing.
 - Copy files between your IBM PC and VM/IS
 - Use your personal storage disk on VM/IS as if it were an IBM PC disk
 - Access files in your personal storage on VM/IS from your IBM PC
 - Print files on the VM/IS printer
 - Store your IBM PC files in your personal storage on VM/IS.
- Engineering and Scientific Computing
 - Create and edit programs
 - Store and retrieve your programs effectively using libraries that you create
 - Compile and execute your programs through interactive or batch facilities
 - Perform general maintenance to your program libraries.
- Application Development
 - Define and validate data structures, user screens, and files for your applications
 - Test and debug your application programs
 - Maintain and document your application programs

- Generate and perform trial runs of your application programs
- Run the application programs that you create.
- Distributed System Support
 - Start and stop remote systems
 - Manage files and memory space
 - Change service machine passwords
 - Reduce message traffic.

Reference

The *VM/IS General Information* book, GH24-5119, describes what VM/IS is, what it can do, and how it can help your organization.

Chapter 11. Planning, Installing, and Servicing VM/SP

This chapter is an overview of three tasks that can apply to every VM/SP installation:

- Planning
- Installing
- Applying service updates.

For specific information on planning, installing, and servicing VM/SP, you should use the *VM/SP Planning Guide and Reference*, the *VM/SP Installation Guide*, and the *VM/SP Service Guide*. This chapter highlights the contents of those three books. You should not use this chapter for guide or reference information.

Planning VM/SP

OVERVIEW

Among the things considered when planning for a VM/SP installation are:

- System control files
- Storage requirements
- Device requirements
- Minidisk allocation
- Options that affect performance
- System libraries
- Programming language support
- Access method support
- Virtual machine operating systems other than CMS
- Special situations.

Planning is an important part of the VM/SP installation process. The better a VM/SP installation process is planned, the less time it takes to do the job. If properly done, the actual installation takes far less time than the planning process.

Before installing VM/SP there are many things that must be considered. Consequently, there are things that can be done in advance to make the installation go more smoothly. This topic briefly discusses a few of the more important aspects of system planning. Anyone who is taking part in the planning process for a VM/SP installation should also read the *VM/SP Planning Guide and Reference*.

Preparing the Files That Describe Your VM/SP System

A major task in getting ready to install VM/SP is the preparation of files that define your VM/SP system. These files are:

File	Function
System Directory	Defines virtual machines for all authorized users.
DMKRIO	Defines the real system I/O devices, control units, and channels.
DMKSYS	Defines many of the CP system control functions.
DMKSNT	Identifies saved systems for installations that plan to use them.

Depending on the installation method, these files can be ready to install at system generation time. For example, you can prepare control statements for setting up the system directory in advance. You can plan for tailoring the copies of system files supplied with VM/SP so that they require minimum time during installation.

Planning Storage Requirements

Planning for VM/SP system storage requirements falls into two general categories:

- Real storage planning
- Direct access storage planning.

Many factors can affect how much of each kind of storage a system might need.

Some of the factors that affect real storage requirements are:

- Nucleus size
- Number of system real devices, control units, and channels
- Types of system devices
- Number of virtual machines defined in the system directory.

Some of the factors that affect direct access storage requirements are:

- Nucleus size
- Maximum number and size of logged on virtual machines
 - Paging space
 - Spooling space
 - CMS minidisks.
- Operating systems running in virtual machines
- Other facilities
 - Error recording
 - System restart
 - System directory
 - System profile
 - Interactive Problem Control System
 - Saved systems
 - Number of national languages available on the system.

Reference

The *VM/SP Planning Guide and Reference*, SC19-6201, describes how to get ready to install VM/SP.

Installing VM/SP

OVERVIEW

Proper planning is the most important factor when generating a VM/SP system.

To install VM/SP, use one of the following procedures:

- Build a new system, using the Starter System shipped with VM/SP as the generating system.
- Upgrade to a new level, using an existing VM system as the generating system.

A VM/SP installation consists of the following basic processes (not necessarily in this order):

- Formatting DASD volumes and minidisks
- Loading files from tapes to minidisks and CMS Shared File System (SFS) directories
- Editing files
- Building system nuclei
- Building file pools.

These processes require you to enter commands and respond to system prompts to define the unique configuration of your VM/SP system.

Installation Tasks

You should be aware of what the major installation tasks are. For more information, see the *VM/SP Installation Guide*. If your responsibilities include VM/SP installation, you should become familiar with the contents of that book. Although the installation of VM/SP is not difficult, there is considerably more to it than outlined in this topic.

Major tasks when using the Starter System as the generating system are:

- Plan the complete installation
- Format the required DASD volumes and load the Starter System
- Load the installation and system generation tools, CP code, and sample system definition files from the product tape
- Tailor the sample CP directory and the other system definition files to define your unique system configuration
- Build a CP nucleus
- Load the CMS and REXX code from the product tape and build a CMS nucleus
- Load the IPCS code
- Build the system file pool
- Build the user file pool (if desired)
- Load the system HELP files (if desired)
- Load the GCS code and build a GCS nucleus (if desired)
- Load the TSAF and AVS code (if desired)

- Run the Installation Verification Procedure (IVP)
- Install saved segments (if desired).

Major tasks when using an existing VM system as the generating system are:

- Plan the complete installation
- Update the current CP directory and format any new minidisks
- Load the installation and system generation tools, sample system definition files, CP code, CMS code, and REXX code from the product tape
- Build a new CMS nucleus
- Tailor the system definition files
- Build a new CP nucleus
- Load the system HELP files (if desired)
- Load IPCS
- Build the system file pool
- Build the user file pool (if desired)
- Load GCS and build a GCS nucleus (if desired)
- Load TSAF and AVS (if desired)
- Run the Installation Verification Procedure (IVP)
- Install saved segments (if desired).

Reference

The *VM/SP Installation Guide*, SC24-5237, describes the installation process and contains step-by-step procedures for installing VM/SP.

Servicing VM/SP

OVERVIEW

You should be familiar with these concepts before you attempt to service your VM/SP system.

VM/SP service is the process of applying changes to a VM/SP system after installation. Service changes are not superficial, but involve rebuilding or replacing parts of the system. There are various reasons for servicing your VM/SP system, such as:

- Adding programming enhancements
- Changing the system configuration
- Fixing a problem.

The VM/SP service process consists of three major tasks:

- Receiving the service
- Applying the service
- Building the serviced parts.

Maintaining a VM/SP System

Each VM/SP component consists of numerous parts (types of files), including text decks, modules, control blocks, macros, execs, and so on. Some parts are maintained as *object* (executable) code. An object maintained part is serviced by replacing the part with a new one. Other parts are maintained as *source* (non-executable) code. A source maintained part is serviced by updating statements in a source file, then assembling the file (that is, combining the updates with the base part of the file) to produce a new executable object file. A source maintained part can also be serviced by replacing the generated object file with a new one in which the updates have already been applied.

Types of Service

There are several types of service that you can apply to your system:

- Program update service
 - Preventive service
 - Selective preventive service.
- Corrective service
- Local service
 - Circumventive service (patches and ZAPs)
 - User modifications.

You should apply service in a test environment before you install it on your production system. Also, make sure that you have a good backup copy of the production system, specifically the files being changed by the service.

Program Update Service

IBM Software Distribution (ISD) provides programming enhancements and fixes for known problems on program update tapes (PUTs). Application of the entire contents of a PUT is called *preventive service* because it is intended to enhance your system and prevent problems. However, depending on the configuration and level of your system, and whether you need or want all of the enhancements and fixes, you can also *selectively* apply preventive service.

A PUT is cumulative. When you receive service from the tape, you load down the service files to special minidisks and/or CMS Shared File System (SFS) directories for each component. For object maintained parts, the PUT contains serviced text decks (replacement object files). For source maintained parts, the PUT contains update files and serviced text decks. If you do not have any local updates that you want to combine with the IBM-supplied updates, you can simply replace the text decks.

ISD maintains a Customer Profile for you that lists the IBM program products for which you are licensed. You can choose to receive program update service on a regular basis or on request. Each PUT that IBM sends you is tailored to your Customer Profile and contains program update service for VM/SP and any other program products that you have.

Corrective Service

At your request, IBM can send you a tape containing changes to *correct* a specific problem that you are having, if the fix is not on an existing PUT. Corrective service for an object maintained part is supplied as one or more replacement text decks. Corrective service for a source maintained part contains the replacement text deck(s) plus the source updates.

Local Updates

If you have encountered a problem that is not known (there is no preventive or corrective fix), IBM might be able to supply information over the phone (or on tape) that you can use to *circumvent* the problem. The circumvention usually does not fix the problem, but prevents failure of the system by disabling the failing function until a corrective fix is available.

To apply circumventive service to a source maintained part, you update the source file, then assemble the file to produce a new text deck. For object maintained parts, circumventive service is supplied as either a patch (for the CP, CMS, or GCS nucleus) or a ZAP. Patches and ZAPs are direct temporary changes to the object code.

Local service also includes any changes that you originate (user modifications). Such a change could be:

- Adding a member to a macro library or text library
- Modifying the CP directory and rebuilding the CP nucleus
- Updating a printer module.

The VM/SP Product Parameter File

The VM/SP product parameter file contains installation and service parameters. It consists of a product area and an area for each component. A component area has three sections:

- The first section contains control options (such as the control file name, the system level and version, the national language, and so on).
- The second section contains minidisk and SFS directory assignments.
- The third section is a part type/function list that identifies specific execs for receiving service, applying service, and building component parts.

Programs for Servicing VM/SP

The VM/SP service process uses many programs to service the various parts of the system. The following table summarizes the functions of the service programs.

Program	Function
ASMGEND	Updates the VM/370 system assembler using GENMOD.
CMMSGEND	Rebuilds CMS command modules using GENMOD.
CSLGEN	Rebuilds a callable services library (CSL).
DCSSGEN	Rebuilds the CMS installation physical saved segment (CMSINST).
DOSGEN	Rebuilds the CMSDOS physical saved segment.
EXECUPDT	Applies updates to a \$ source file and creates an executable version of the program.
GENMOD	Creates a nonrelocatable module file on a CMS minidisk.
GENSERVE	Rebuilds CMS Shared File System (SFS) file pool server load modules.
GENTSAP	Rebuilds the RUNTSAP module.
PRELOAD	Collects multiple text files and reformats them into a single text file.
SAMGEN	Rebuilds the CMSBAM physical saved segment.
SEGEN	Rebuilds a physical saved segment that contains logical saved segments.
UTILITY	Rebuilds utility service programs (DDR, Format/Allocate, CP Directory).
VMFAPPLY	Creates and/or updates auxiliary control files for the program temporary fixes (PTFs) on the service tape.
VMFASM	Updates an ASSEMBLE source file according to entries in a control file, then assembles the updated file.
VMFBLD	Builds a new CP, CMS, or GCS nucleus and applies patches.
VMFHASM	Updates an ASSEMBLE source file according to entries in a control file, then assembles the updated file using the H-assembler.
VMFLKED	Invokes the CMS LKED command to link-edit modules into a load library (LOADLIB).

Program	Function
VMFLOAD	Punches a nucleus or stand-alone dump load deck based on a loadlist and a control file.
VMFMAC	Rebuilds macro libraries (MACLIBs) containing updated macro and copy files.
VMFNLS	Applies updates to national language files and compiles the updated versions.
VMFOVER	Overrides a portion of the product parameter file to define alternative service parameters.
VMFREC	Receives program update service or corrective service from tape.
VMFSETUP	Sets up a minidisk and/or SFS directory access order.
VMFTXT	Rebuilds text libraries (TXTLIB) containing serviced object files.
VRSIZE	Builds a DMKSLC TEXT file that is used to generate a virtual = real (V = R) area when rebuilding the CP nucleus.
VSAMGEN	Rebuilds the CMSVSAM and CMSAMS physical saved segments.
VSEVSAM	Builds a VSE/VSAM MACLIB containing the supported VSE/VSAM macros.
ZAP	Modifies or dumps MODULE, LOADLIB, or TXTLIB files.
ZAPTEXT	Modifies or dumps individual text files.

Specialized programs (VMFMERGE, VMFREMOV, and VMFZAP) are also available for servicing Systems Network Architecture (SNA) products.

These service programs are described in more detail in the *VM/SP Service Guide*, with the following exceptions:

- CSLGEN and SEGEN are described in the *VM/SP Application Development Guide for CMS* book.
- DCSSGEN, DOSGEN, SAMGEN, UTILITY, VRSIZE, and VSAMGEN are described in the *VM/SP Installation Guide*.
- EXECUPDT and GENMOD are described in the *VM/SP CMS Command Reference*.

Reference

The *VM/SP Service Guide*, SC24-5389, describes the service process and contains procedures for servicing the VM/SP components.

Glossary of Terms and Abbreviations

A

abend. (1) Abnormal end of task. (2) Synonym for *abnormal termination*.

abnormal end of task (abend). Termination of a task before its completion because of an error condition that cannot be resolved by recovery facilities while the task is executing.

abnormal termination. The ending of processing before planned termination. Synonymous with *abend*.

access mode. A method VM/SP uses to control user access to data files. Access modes let the user read and write data to a file, or only read data from a file. See *file mode*.

access security. Information that a target LU and target transaction program use to verify whether a source program is authorized to make a connection. This information consists of a user ID and, possibly, a password.

ACF/SSP. Advanced Communications Function for Systems Support Programs.

ACF/VTAM. Advanced Communications Function for Virtual Telecommunications Access Method.

Advanced Communications Function for Systems Support Programs (ACF/SSP). An IBM program product made up of a collection of utilities and small programs. SSP is required for operation of the NCP.

Advanced Communications Function for Virtual Telecommunications Access Method (ACF/VTAM). An IBM licensed program that controls communications and flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability.

Advanced Program-to-Program Communications (APPC). The inter-program communication service within SNA LU 6.2 on which the APPC/VM interface is based.

Advanced Program-to-Program Communications/VM (APPC/VM). An API for communicating between two virtual machines that is mappable to the SNA LU 6.2 APPC interface and based on IUCV functions. Along with the TSAF virtual machine, APPC/VM provides this communication within a single system and throughout a collection of systems.

alias. A pointer to a base file. An alias can be in the same directory as the base file or in a different

directory. There must always be a base file for the alias to point to. The alias references the same data as the base file. Data is not moved or duplicated.

alphanumeric. Pertaining to a character set that contains letters, digits, and usually other characters, such as punctuation marks. Synonymous with *alphameric*.

AP. Attached processor.

APAR. Authorized program analysis report.

APPC. Advanced Program-to-Program Communications.

APPC/VM. Advanced Program-to-Program Communications/VM.

APPC/VM VTAM Support (AVS). A component of VM/SP that lets application programs using APPC/VM communicate with programs anywhere in a network defined by IBM's SNA. AVS transforms APPC/VM into APPC/VTAM protocol.

API. Application program interface.

application program. A program written for or by a user that applies to the user's work, such as a program that does inventory control or payroll.

application program interface (API). The formally defined programming language interface between an IBM system component or program product and its user.

apply. When servicing a product or component, to generate an auxiliary control structure from a PTF.

assembler language. A source language that includes symbolic machine language statements in which there is a one-to-one correspondence with instruction formats and data formats of the computer.

attached processor (AP). A processor that has no I/O capability and is always linked to the processor initialized for I/O handling.

authority. In SFS, the permission to access a file or directory. You can have read authority or write authority (which includes read authority). You can also have file pool administration authority, which is the highest level of authority in a file pool.

authorized program. Synonym for *privileged program*.

authorized program analysis report (APAR). An official request to the responsible IBM Change Team to look

into a suspected problem with IBM code or documentation. APARs describe problems giving conditions of failure, error messages, abend codes, or other identifiers. They also contain a problem summary and resolution when applicable. See *program temporary fix (PTF)*.

authorized virtual machine. A GCS virtual machine identified by user ID.

auxiliary directory. In CMS, an extension of the CMS file directory for a minidisk, which contains the names and locations of certain CMS modules not included in the minidisk's CMS file directory.

auxiliary storage. Data storage other than main storage; in VM/SP, auxiliary storage is usually a direct access device.

AVS. APPC/VM VTAM Support.

AVS virtual machine. The virtual machine that manages a gateway that allows communication between VM systems and an SNA network.

B

base file. The first occurrence of a file. It remains the base for the life of the file, even if the file has been renamed. Aliases point to base files.

binary synchronous communication (BSC). Communication using binary synchronous line discipline in which transmission of binary-coded data between stations is synchronized by timing signals generated at the sending and receiving stations.

block. A unit of DASD space on FB-512 devices. For example, FB-512 devices can be the IBM 9335, 9332, 9313, 3370, and 3310 DASD using fixed-block architecture.

BSC. Binary synchronous communication.

buffer. An area of storage, temporarily reserved for performing input or output, into which data is read, or from which data is written.

byte. A unit of storage, consisting of eight adjacent binary digits that are operated on as a unit and constitute the smallest addressable unit in the system.

C

callable services library (CSL). A package of CMS assembler routines that can be stored as an entity and made available to application programs.

catalog storage group. The storage group in a file pool that contains information about the objects (such as files

and directories) and authorizations that exist in the file pool. See *file pool catalog*.

CAW. Channel address word.

CCS. Console communication service.

CCW. Channel command word.

central site/system. The main installation with skilled system support personnel such as system programmers and operations staff.

changes. In reference to installation and service, IBM and original equipment manufacturer (OEM) supplied service for their programs. In the IBM service process, there are many ways users can receive information they need to fix (change) a portion(s) of a product they are running on a VM system. These include PTFs, APARs, user modifications, and information received over the phone. All these types of information are called *changes*.

channel. A path in a system that connects a processor and main storage with an I/O device.

channel address word (CAW). An area in storage that specifies the location in main storage at which a channel program begins.

channel command word (CCW). A doubleword at the location in main storage specified by the channel address word. One or more CCWs make up the channel program that directs data channel operations.

channel status word (CSW). An area in storage that provides information about the termination of I/O.

channel-to-channel adapter (CTCA). A hardware device that connects two channels on the same computing system or on different systems.

channel-to-channel (CTC) device. A hardware device that connects two channels on the same computing system or on different systems. CTC devices include both CTCAs and 3088 MCUs.

checkpoint. An internal file pool server operation during which the changes recorded on the log minidisks are permanently made to the file pool.

checkpoint (CKPT) start. A VM/SP system restart that attempts to recover information about closed spool files previously stored on the checkpoint cylinders. The spool file chains are reconstructed, but the original sequence of spool files is lost. Unlike warm start, CP accounting and system message information is also lost. Contrast with *cold start*, *force start*, and *warm start*.

CICS/VM. Customer Information Control System for VM.

circumventive service. Information that IBM supplies over the phone or on a tape to circumvent a problem by disabling a failing function until a PTF is available to be shipped as a corrective service fix. See *patch* and *zap*.

class authority. Privilege assigned to a virtual machine user in the user's directory entry; each class specified allows access to a subset of all the CP commands. See *privilege class* and *user class restructure (UCR)*.

CMS. Conversational Monitor System.

CMS batch facility. A facility that lets the user run time-consuming or noninteractive CMS jobs in another CMS virtual machine dedicated to that purpose, thus freeing the user's own terminal and virtual machine for other work.

CMSDOS. The standard name of the CMS/DOS saved segment. See *saved segment*.

CMS/DOS. The functions of CMS that become available when the user enters the command: SET DOS ON. CMS/DOS is a part of the regular CMS system and is not a separate system. Users who do not use CMS/DOS are sometimes called OS users, because they use the OS simulation functions of CMS. Synonymous with *DOS simulation under CMS*. Contrast with *OS simulation under CMS*.

CMS EXEC language. A general-purpose, high-level programming language, particularly suitable for EXEC procedures and EDIT macros. The CMS EXEC processor executes procedures and macros (programs) written in this language. Contrast with *EXEC 2 language* and *Restructured Extended Executor (REXX) language*.

CMS file directory. A directory on each CMS disk that contains the name, format, size, and location of each of the CMS files on that disk. When a disk is accessed by the ACCESS command, its directory is read into virtual storage and identified with any letter from A through Z. Synonymous with *master file directory block* and *minidisk directory*.

CMS files. Refers exclusively to files in the fixed-block format used by CMS file system commands. VSAM and OS data sets and DOS files are not compatible with the CMS file format and cannot be manipulated using CMS file system commands.

CMS file system. A way to create files in the CMS system. CMS files are created by using an identifier consisting of three fields: file name, file type, and file mode. These files are unique to the CMS system and cannot be read or written using other operating systems.

CMS nucleus. The portion of CMS that is resident in the user's virtual storage whenever CMS is executing. Each CMS user receives a copy of the CMS nucleus

when the user IPLs CMS. See *saved system* and *shared segment*.

CMSSERV. A command that starts a CMS router in the Enhanced Connectivity Facilities environment of VM/SP.

cold start. A VM/SP system restart that ignores previous data areas and accounting information in main storage, and the contents of paging and spool files on CP-owned disks. Contrast with *checkpoint (CKPT) start*, *force start*, and *warm start*.

collection. See *TSAF collection*.

command. A request from a user at a terminal for the execution of a particular CP, CMS, IPCS, GCS, TSAF, or AVS function. A CMS command can also be the name of a CMS file with a file type of EXEC or MODULE. See *subcommand* and *user-written CMS command*.

command line. The line at the bottom of display panels that lets a user enter commands or panel selections. It is prefixed by an arrow (= = = >).

Common Programming Interface (CPI)

Communications. A set of program-to-program communication routines that let applications written in REXX and high-level languages access APPC/VM functions. These routines are part of IBM's Systems Application Architecture.

communications directory. A CMS facility that lets APPC/VM applications connect to a resource using symbolic destination names and special NAMES files.

compile. To translate a program written in a high-level programming language into a machine language program.

component. A collection of objects that together form a separate functional unit. A product may contain many components (for example, VM/SP has components of CP, CMS, GCS, TSAF, IPCS, AVS, and Procedures Language/VM). A component can be part of many products (CP spans both VM/SP and VM/HPO products).

concurrently. Concerning a mode of operation that includes doing work on two or more activities within a given (short) interval of time.

connect. Establishing a path to communicate with another virtual machine or with the user's own virtual machine.

connectivity program request block (CPRB). An interface control block that requesters and servers use to communicate information.

console. A device used for communications between the operator or maintenance engineer and the computer.

console communication service (CCS). A group of CP modules that interfaces with the VTAM service machine, providing full VM/SP console capabilities for SNA terminal users.

console spooling. Synonym for *virtual console spooling*.

console stack. Refers collectively to the program stack and the terminal input buffer.

contention. The situation where two LUs try to allocate a conversation over the same session at the same time.

control block. A storage area that a computer program uses to hold control information.

control file. (1) In service, a file with file type CNTRL that contains records that identify the updates to be applied and the macro libraries, if any, needed to assemble that source program. (2) A CMS file that is interpreted and directs the flow of a certain process through specific steps. For example, the control file could contain installation steps, default addresses, and PTF prerequisite lists as well as many other necessary items.

control minidisk. In a file pool, the minidisk that tracks the physical DASD blocks allocated to the file pool.

control program. A computer program that schedules and supervises the program execution in a computer system. See *Control Program (CP)*.

Control Program (CP). A component of VM/SP that manages the resources of a single computer so multiple computing systems appear to exist. Each virtual machine is the functional equivalent of an IBM System/370.

control statement. A statement that controls or affects program execution in a data processing system.

control unit. A device that controls I/O operations at one or more devices.

Conversational Monitor System (CMS). A virtual machine operating system and component of VM/SP that provides general interactive time sharing, problem solving, program development capabilities, and operates only under the control of the VM Control Program (CP).

corrective service. Service that IBM supplies on tape to correct a specific problem.

corrective service tape. A tape supplied by IBM at the user's request containing a fix for a specific problem.

CP. Control Program.

CP assist. A hardware function, available only on a processor with ECPS, that reduces CP overhead by doing the most frequently used tasks of CP routines.

CP command. A command available to all VM users. Class G CP commands let the general user reconfigure their virtual machine, control devices attached to their virtual machine, do input and output spooling functions, and simulate many other functions of a real computer console. Other CP commands let system operators, system programmers, system analysts, and service representatives manage the resources of the system.

CP directory. Synonym for *VM/SP directory*.

CPI Communications. Common Programming Interface Communications.

CPRB. Connectivity program request block.

CPTRAP. A CP debugging tool that creates a reader spool file of selected trace table entries, CP data, and virtual machine data in the order that they happen. The IPCS commands can help the user access and print this collected data.

CSL. Callable services library.

CSW. Channel status word.

CTC. Channel-to-channel.

CTCA. Channel-to-channel adapter.

Customer Information Control System for VM (CICS/VM). An IBM licensed program that provides a transaction processing capability for use in distributed departmental VM systems. It lets users in individual departments mix transaction processing requests with other office and commercial applications.

cylinder. In a disk pack, the set of all tracks with the same nominal distance from the axis about which the disk pack rotates.

D

DASD. Direct access storage device.

DASD Dump Restore (DDR) program. A service program that copies all or part of a minidisk onto tape, loads the contents of a tape onto a minidisk, or sends data from a DASD or from tape to the virtual printer.

DASD space. (1) Area allocated to DASD units on CKD devices. (2) Area allocated to DASD units on FB-512 devices. Note that *DASD space* is synonymous with *cylinder* when there is no need to differentiate between CKD devices and FB-512 devices. This term

applies to VM/370, VM/SP and VM/SP HPO program products.

data stream. A set of logical records sent one after the other.

DCF. Document Composition Facility.

DDP. Distributed data processing.

direct access storage device (DASD). A storage device in which the access time is effectively independent of the location of the data.

directory. See *auxiliary directory*, *CMS file directory*, *SFS directory*, or *VM/SP directory*.

directory name (dirname). A fully-qualified directory name that can incorporate a period (.) to indicate the user's own top directory (used in commands).

dirname. Directory name.

discontiguous saved segment. One or more 64K segments of storage that were previously loaded, saved, and assigned a unique name. The segment(s) can be shared among virtual machines if the segment(s) contains reentrant code. Discontiguous segments used with CMS must be loaded into storage at locations above the address space of a user's CMS virtual machine. They can be detached when no longer needed.

disk. A magnetic disk unit in the user's CMS virtual machine configuration. Also called a virtual disk.

dispatcher. The program in CP that places virtual machines or CP tasks into execution. The dispatcher selects the next virtual machine to run and prepares the virtual machine for problem state execution.

dispatching. The starting of virtual machine execution.

dispatch list. See *run list*.

display device. An I/O device that gives a visual representation of data.

display mode. A type of editing at a display terminal in which an entire screen of data is displayed at once and in which the user can access data through commands or by using a cursor. Contrast with *line mode*.

display terminal. A terminal with a component that can display information on a viewing surface such as a CRT or gas panel.

distributed data processing (DDP). Data processing in which processing, storage, and control functions, in addition to I/O operations, are distributed among remote locations and connected by transmission facilities.

Document Composition Facility (DCF). A text processing program; its main component is the text formatter, called *SCRIPT/VS*. See *SCRIPT/VS*.

DOS. Disk operating system.

DOS simulation under CMS. Synonym for *CMS/DOS*.

dump. To write the contents of part or all of main storage, or part or all of a minidisk, to auxiliary storage or a printer. See *abend dump*.

dynamic address translation (DAT). In System/370 virtual storage systems, the change of a virtual storage address to a real storage address during execution of an instruction.

E

ECPS:VM/370. Extended Control Program Support:VM/370.

edit. A function that makes changes, additions, or deletions to a file on a disk. These changes are interactively made. The edit function also generates information in a file that did not previously exist.

edit mode. The environment in which CMS EDIT subcommands and System Product Editor (XEDIT) subcommands can be entered by the user to insert, change, delete, or rearrange the contents of a CMS file. Contrast with *input mode*.

emulation. The use of programming techniques and special machine features to permit a computing system to execute programs written for another system.

Ethernet. See *IEEE 802.3*.

EXEC 2 language. A general-purpose, high-level programming language, particularly suitable for EXEC procedures and XEDIT macros. The EXEC 2 processor runs procedures and XEDIT macros (programs) written in this language. Contrast with *CMS EXEC language* and *Restructured Extended Executor (REXX) language*.

EXEC procedure. (1) A procedure defined by a frequently used sequence of CMS and CP commands to do a commonly required function. A user creates the procedure to save repetitious rekeying of the sequence, and invokes the entire procedure by entering a command (that is, the exec file's file name). The procedure could consist of a long sequence of CMS and CP commands, along with REXX, EXEC 2, or CMS EXEC control statements to control processing within the procedure. (2) A CMS file with a file type of EXEC.

expanded virtual machine assist. A hardware assist function, available only on a processor that has ECPS, that handles many privileged instructions not handled

by VMA, and extends the level of support of certain privileged instructions beyond that provided by VMA.

explicit lock. A lock on a file or directory that a user explicitly created by entering a CREATE LOCK command or executing a DMSCRLOC CSL routine.

extended control (EC) mode. A mode in which all features of a System/370 computing system, including dynamic address translation, are operational. Contrast with *basic control (BC) mode*.

Extended Control Program Support (ECPS:VM/370). A hardware assist feature that improves the performance of CP by reducing CP overhead. ECPS:VM/370 consists of CP assist, expanded virtual machine assist, and virtual interval timer assist.

F

FBA. Fixed-block architecture.

file access mode. A file mode number that designates whether the file can be used as a read-only or read/write file by a user. See *file mode*.

file definition. (1) Equating a CMS file identifier (file name, file type, file mode) with an OS data set name by the FILEDEF command; or equating a DOS file ID with a CMS file identifier by the DLBL command. (2) Identifying the input or output files used during execution of a program (by way of either the FILEDEF or DLBL commands).

file mode. A two-character CMS file identifier field comprised of the file mode letter (A through Z) followed by the file mode number (0 through 6). The file mode letter indicates the minidisk or SFS directory on which the file resides. The file mode number indicates the access mode of the file. See *file access mode*.

file name. A one-to-eight character alphanumeric field, comprised of A through Z, 0 through 9, and special characters \$ # @ + - (hyphen) : (colon) _ (underscore), that is part of the CMS file identifier and serves to identify the file for the user.

file pool. A collection of minidisks managed by SFS. It contains user files and directories and associated control information. Many users' files and directories can be contained in a single file pool.

file pool catalog. The part of a file pool that contains information about the objects stored in the file pool and the authorizations granted on those objects. See *catalog storage group*.

file pool server machine. A virtual machine that is properly configured to manage a file pool. (Its VM

system directory entries must, for example, contain the MDISK statements for a file pool.)

file space. A user's allocation of space within a file pool.

file type. A one-to-eight character alphanumeric field, comprised of A through Z, 0 through 9, and special characters \$ # @ + - (hyphen) : (colon) _ (underscore), that is used as a descriptor or as a qualifier of the file name field in the CMS file identifier. See *reserved file types*.

fixed-block architecture (FBA) device. A disk storage device that stores data in blocks of fixed size or records; these blocks are addressed by block number relative to the beginning of the particular file.

force start. A VM/SP system restart that attempts to recover information about closed spool files previously stored on the checkpoint cylinders. All unreadable or invalid spool file information is ignored. Contrast with *checkpoint (CKPT) start*, *cold start*, and *warm start*.

free storage. Storage not allocated. The blocks of memory available for temporary use by programs or by the system.

full-screen CMS. When a user enters the command SET FULLSCREEN ON, CMS is in a window and can take advantage of 3270-type architecture and windowing support, and various classes of output are routed to a set of default windows. Also, users can type commands anywhere on the physical screen and scroll through commands and responses previously displayed. See *windowing*.

full-screen editor. An editor used at a display terminal where an entire screen of data is displayed at once and where the user can access the data through commands or by using a cursor. See *full-screen CMS*.

G

gateway. The LU name of a TSAF collection that is a source for communications to an SNA-defined network or the target of communications from an SNA-defined network.

GCS. Group Control System.

GDDM. Graphical data display manager.

global gateway. (1) A gateway that programs outside a TSAF collection can use to access global resources inside the collection. (2) A gateway that global resource manager programs use to access resources outside a TSAF collection. Contrast with *private gateway*.

global resource. A resource accessible from anywhere within a TSAF collection and whose identity is known throughout the collection. A shared file system file pool is an example of a global resource. Contrast with *local resource* and *private resource*.

global resource manager. An application that runs in a server virtual machine and identifies itself to the TSAF collection as a global resource owner using *IDENT. Contrast with *local resource manager* and *private resource manager*.

graphical data display manager (GDDM). (1) A group of pictures that let pictures be defined and procedurally displayed through function routines that correspond to graphic primitives. Contrast with *presentation graphics routines (PGR)*. (2) An IBM licensed program that creates page segments.

group. Synonym for *virtual machine group*.

Group Control System (GCS). A component of VM/SP, consisting of a shared segment that the user can IPL and run in a virtual machine. It provides simulated MVS services and unique supervisor services to help support a native SNA network.

guest operating system (GOS). A second operating system that runs on the user's primary operating system. An example of a GOS is VSE running on VM/SP to support VM/VCNA.

H

host system. A data processing system that prepares programs and the operating environments for use by another computer or controller.

I

IEEE 802.3. A standard that describes the formats and protocols at the medium access level for a LAN. In this standard, hardware protocol requires carrier sense multiple access with collision detection. In addition, a transmitting data station that detects another signal while transmitting stops sending, sends a jam signal, and then waits for a variable time before trying again (Ethernet).

initial program load (IPL). The initialization procedure that causes an operating system to begin operation. A VM user must IPL the specific operating system into the virtual machine that will control the user's work. Each virtual machine can be loaded with a different operating system.

input mode. In the CMS Editor or System Product Editor (XEDIT), the environment that lets the user key in new lines of data. Contrast with *edit mode*.

input/output (I/O). (1) Pertaining to a device whose parts can do an input process and an output process at the same time. (2) Pertaining to a functional unit or channel involved in an input process, output process, or both, concurrently or not, and to the data involved in such a process.

input stream. The sequence of job control statements and data submitted (to an operating system) through an input unit especially started for this purpose by the operator.

installation verification procedure (IVP). A procedure distributed with VM/SP that exercises the newly generated VM/SP system. This procedure verifies that the basic facilities of VM/SP are correctly functioning.

interaction. A basic unit that records system activity, consisting of acceptance of a line of terminal input, processing of the line, and a response, if any.

interactive. The classification given to a virtual machine depending on this virtual machine's processing characteristics. When a virtual machine uses less than its allocation time slice because of terminal I/O, the virtual machine is classified as being interactive. Contrast with *noninteractive*.

Interactive Problem Control System (IPCS). A component of VM/SP that permits online problem management, interactive problem diagnosis, online debugging for disk related CP or virtual machine abend dumps or CPTRAP files, problem tracking, and problem reporting.

interface. A shared boundary between two or more entities. An interface might be a hardware or software component that links two devices or programs together.

interrupt. A suspension of a process, such as execution of a computer program, caused by an external event and done in such a way that the process can be resumed.

inter-user communication vehicle (IUCV). A VM/SP generalized CP interface that helps the transfer of messages either among virtual machines or between CP and a virtual machine.

I/O. Input/output.

IPCS. Interactive Problem Control System.

IPL. Initial program load.

IUCV. Inter-user communication vehicle.

IVP. Installation verification procedure.

K

K. kilobyte.

kilobyte (K). 1024 bytes.

L

LAN. Local area network.

line mode. The mode of operation of a display terminal that is equivalent to using a typewriter-like terminal. Contrast with *display mode*.

line number. A number located at either the beginning or the end of a record (line) that can be used during editing to refer to that line. See *prompting*.

link. (1) In RSCS, a connection, or ability to communicate, between two adjacent nodes in a network. (2) In TSAF, the physical connection between two systems.

load. In reference to installation and service, to move files from tape to disk, auxiliary storage to main storage, or minidisks to virtual storage within a virtual machine.

loader. A routine, commonly a computer program, that reads data into main storage.

load map. A map containing the storage addresses of control sections and entry points of a program loaded into storage.

local. Two entities (for example, a user and a server) are said to be local to each other if they belong to the same system within a collection or to the same node within an SNA system. Contrast with *remote*.

local area network (LAN). A data network located on the user's premises in which serial transmission is used for direct data communication among data stations. Contrast with *wide area network (WAN)*.

local resource. A resource accessible from only within a single VM system and whose identity is known only within a single VM system in the TSAF collection. Contrast with *global resource* and *private resource*.

local resource manager. An application that runs in a virtual machine and identifies itself to the local system in the TSAF collection as a local resource owner by *IDENT. Contrast with *global resource manager* and *private resource manager*.

local service. Changes manually applied to a product or component (that is, not using the program update service or corrective service procedures). See *circumventive service* and *user modification*.

lock. A tool for controlling concurrent usage of SFS objects. Implicit locks are acquired and automatically released when you run CMS commands and program functions in SFS. Explicit locks let you control the type and duration of the lock.

logical operator. The name given to the virtual machine from which OPERATOR functions requested by the Programmable Operator Facility virtual machine are done. This name can also describe the person who usually operates the Logical Operator virtual machine. In a mixed environment, an NCCF operator can be assigned as the logical operator to control a VM distributed system.

logical record. A formatted record that consists of a 2-byte logical record length and a data field of variable length.

logical saved segment. A portion of a physical saved segment that CMS can manipulate. Each logical segment can contain different types of program objects, such as modules, text files, execs, callable services libraries, language repositories, user-defined objects, or a single minidisk directory. A system segment identification file (SYSTEM SEGID) associates a logical saved segment to the physical saved segment in which it resides. See *physical saved segment* and *saved segment*.

logical unit (LU). An entity addressable within an SNA-defined network, similar to a node within a VM network. LUs are categorized by the types of communication they support. A TSAF collection in an SNA network is viewed as one or more LUs.

log minidisks. In a file pool, two duplicate minidisks that contain information about changes made to the file pool. File pool servers use the log minidisks to help protect the integrity of the file pool if a system failure occurs.

logoff. The procedure by which a user ends a terminal session.

logon. The procedure by which a user begins a terminal session.

LU. Logical unit.

LU type 6.2. A set of protocols and services defined by IBM's SNA for communication between application programs.

M

machine. A synonym for a virtual machine running under the control of VM/370 or VM/SP.

master file directory block. Synonym for *CMS file directory*.

Mb. Megabyte.

MDISK. (1) Another name for minidisk. (2) The user directory that describes a user's storage space.

megabyte (Mb). 1,048,576 bytes.

merge. When receiving files from a service tape using the VMFREC EXEC, the process of moving existing service files from each minidisk or SFS directory in the target string (as defined by the MERGE tag in the product parameter file) to the minidisk or SFS directory that contains the previous service level. The result is that the primary target minidisk or directory is left empty and ready to receive the latest service from the tape.

minidisk. A logical subdivision (or all) of a physical disk pack that has its own virtual device address, consecutive virtual cylinders (starting with virtual cylinder 0), and a VTOC or disk label identifier. Each user virtual disk is preallocated and defined by a VM/SP directory entry as belonging to a user.

minidisk directory. Synonym for *CMS file directory*.

module. (1) A unit of a software product that is discretely and separately identifiable with respect to modifying, compiling, and merging with other units, or with respect to loading and execution. For example, the input to, or output from, a compiler, the assembler, the linkage editor, or an exec routine. (2) A nonrelocatable file whose external references have been resolved.

MP. Multiprocessor.

Multiple Virtual Storage (MVS). An alternative name for OS/VS2.

multiprocessor (MP). A computer using two or more processing units under integrated control.

multitasking. Providing services for many tasks that are active at the same time.

MVS. Multiple Virtual Storage.

N

native mode. Refers to running an operating system stand-alone on the real machine instead of under VM/SP.

NCCF. Network Communication Control Facility.

NCP. Network control program.

network. Any set of two or more computers, workstations, or printers linked in such a way as to let data be transmitted between them.

Network Communication Control Facility (NCCF). An IBM licensed program consisting of a base for command processors that can monitor, control, and improve the operation of a network.

network control program (NCP). An IBM licensed program that provides communication controller support for single-domain, multiple-domain, and interconnected network capability.

node. (1) A single processor or a group of processors in a teleprocessing network. (2) A computer, workstation, or printer, when it is participating in a network.

node ID. Node identifier.

node identifier (node ID). The name by which a node is known to all other nodes in a network.

noninteractive. The classification given to a virtual machine depending on this virtual machine's processing characteristics. When a virtual machine usually uses all its allocated time slice, it is classified as being noninteractive or compute bound. Contrast with *interactive*.

nonprivileged program. In GCS, a program called by a GCS application that operates in problem state. Contrast with *privileged program*.

nucleus. The part of CP, CMS, and GCS resident in main storage.

O

object code. Compiler or assembler output that is executable machine code or is suitable for more processing to produce executable machine code. Contrast with *source code*.

object module. A module that is the output of an assembler or a compiler and is input to a linkage editor.

operand. Information entered with a command name to define the data on which a command processor operates and to control the execution of the command processor.

Operating System/Virtual Storage (OS/VS). A family of operating systems that control IBM System/360 and System/370 computing systems. OS/VS includes VS1, VS2, MVS/370, and MVS/XA.

OS/VS. Operating System/Virtual Storage.

overhead. The additional processor time charged to each virtual machine for the CP functions needed to simulate the virtual machine environment and for paging and scheduling time.

overlay. The technique of repeatedly using the same areas of internal storage during different stages of a program.

P

page. A fixed-length block that has a virtual address and can be transferred between real storage and auxiliary storage.

page frame. A block of 4096 bytes of real storage that holds a page of virtual storage.

page number. The part of a virtual storage address needed to refer to a page.

paging. Transferring pages between real storage and external page storage.

paging area. An area of direct access storage (and an associated area of real storage) that CP uses for the temporary storage of pages when paging occurs.

parameter. A variable that is given a constant value for a specified application and that may denote the application.

parameter list (PLIST). In CMS, a string of 8-byte arguments that call a CMS command or function. The first argument must be the name of the command or function to be called. General register 1 points to the beginning of the parameter list.

part. A CMS file provided on a product tape or service tape as input to the build process. See *build*. A part is the smallest serviceable unit of a component.

password. In computer security, a string of characters known to the computer system and a user, who must specify it to gain full or limited access to a system and to gain full or limited access to a system and to the data stored within it.

patch. A circumventive service change applied directly to object code in a text deck in a nucleus.

path. In APPC/VM or IUCV, a connection between two application programs that are on the same or different systems. Paths have names assigned to them.

PC. Personal computer.

performance option. One or more functions that can be assigned to a virtual machine to improve its performance, response time (if terminal-oriented) or throughput under VM/SP.

personal computer (PC). A desk-top, floor-standing, or portable microcomputer that usually consists of a system unit, a display monitor, a keyboard, one or more diskette drives, internal fixed-disk storage, and an optional printer.

PF key. Program function key.

physical saved segment. One or more pages of storage that have been named and retained on a CP-owned volume (DASD). Once created, it can be loaded within a virtual machine's address space or outside a virtual machine's address space. Multiple users can load the same copy. A physical saved segment can contain one or more logical saved segments. A system segment identification file (SYSTEM SEGID) associates a physical saved segment to its logical saved segments. See *logical saved segment* and *saved segment*.

physical screen. Synonym for *screen*.

preferred machine assist (PMA). The hardware feature of the IBM 308X processor complex or the IBM 3033 processor that improves MVS/SP (Release 1 enhancement, or later) V=R virtual machine performance. The MVS/SP guest virtual machine operates in supervisor state with direct control of its own I/O operations under VM/SP HPO. PMA is an extension of VMA, which eliminates CP simulation of certain instructions and interruptions.

preferred paging area. A special area of auxiliary storage where frequently used pages are paged out. It provides high-speed paging.

preventive service. The massive application of PTFs from the PUT. Contrast with *selective preventive service*.

private gateway. (1) A gateway that programs outside a TSAF collection can use to access private resources inside the collection. (2) A gateway that nonglobal resource manager programs can use to access resources outside a TSAF collection. Contrast with *global gateway*.

private resource. A resource accessible from anywhere within a TSAF collection or SNA network and whose identity is known only within a single virtual machine. Contrast with *global resource* and *local resource*.

private resource manager. An application that runs in a server virtual machine and provides a service for connecting programs, but that does not identify itself to the TSAF collection. Contrast with *global resource manager* and *local resource manager*.

privilege class. One or more classes assigned to a virtual machine user in a VM/SP directory entry; each privilege class specified lets a user access a logical subset of the CP commands. There are eight IBM-defined privilege classes that correspond to specific administrative functions. They are:

- Class A - primary system operator
- Class B - system resource operator
- Class C - system programmer
- Class D - spooling operator
- Class E - system analyst
- Class F - service representative
- Class G - general user
- Class H - reserved for IBM use
- Class Any - available to any user.

The privilege classes can be changed to meet the needs of an installation. See *class authority* and *user class restructure (UCR)*.

privileged instruction simulation. The CP-incurred overhead to handle privileged instructions for virtual machine operating systems that execute as if they were in supervisor state but which are executing in problem state under VM/SP. See *virtual machine assist (VMA)*.

privileged program. In GCS, a program called by a GCS application that operates in supervisor state and uses privileged functions. A privileged program is one that meets either of the following requirements:

- It runs in an authorized virtual machine.
- It is called through the AUTHCALL facility.

Synonymous with *authorized program*. Contrast with *nonprivileged program*.

problem state. A state during which the central processing unit cannot execute I/O and other privileged instructions. VM/SP runs all virtual machines in problem state. See *privileged instruction simulation*. Contrast with *supervisor state*.

Procedures Language/VM. A component of VM/SP. It contains the VM/SP System Product Interpreter, which processes the REXX language. The component contains the VM/SP implementation of the Systems Application Architecture Procedures Language in addition to the VM/SP System Product Interpreter function available in VM/SP Releases 3, 4, and 5. Procedures Language/VM provides a single source base for the VM/SP System Product Interpreter in both the CMS and GCS components.

process. A systematic sequence of operations to produce a specified result. A process is usually logical, not physical.

product parameter file. A file containing installation and service parameters for a product: control options, minidisk and SFS directory assignments, and component part type/function lists.

PROFILE EXEC. A special EXEC procedure with a file name of PROFILE that a user can create. The procedure is usually executed immediately after CMS is loaded into a virtual machine (also known as IPL CMS).

program function (PF) key. On a terminal, a key that can do various functions selected by the user or determined by an application program.

programmable operator facility. This facility enables automatic filtering and routing of messages from a specified virtual machine (for example the operator) to a logical operator virtual machine in a local distributed or mixed environment. It also permits installation defined actions to be automatically performed.

program stack. Temporary storage for lines (or files) being exchanged by programs that execute under CMS. See *console stack*.

program status word (PSW). An area in storage that indicates the order in which instructions are executed, and to hold and indicate the status of the computer system.

program temporary fix (PTF). Code changes needed to correct a problem reported in an APAR. The corrected code is included in later releases. A PTF contains one or more APAR fixes. For object maintained parts that are changed, the PTF includes replacement parts. For source maintained parts that are changed, the PTF includes update files and replacement parts. Each PTF is unique to a given release of a product. If the same problem occurs in multiple releases of a product, a separate PTF is defined for each release.

program update service. Receiving the contents of a PUT, applying all or some of the changes, and rebuilding the serviced parts. See *preventive service* and *selective preventive service*.

program update tape (PUT). A tape containing a customized collection of service tapes (preventive service) to match the products listed in a customer's ISD (IBM Software Distribution) profile. Each PUT contains cumulative service for the customer's products back to earlier release levels of the product still supported. The tape is distributed to authorized customers of the products at scheduled intervals or on request.

prompt. A displayed message that describes required input or gives operational information.

prompting. An interactive technique that lets the program guide the user in supplying information to a program. The program types or displays a request, question, message, or number, and the user enters the desired response. The process is repeated until all the necessary information is supplied.

protocol. A set of rules for communication that are mutually understood and followed by two communicating stations or processes. The protocol specifies actions that can be taken by a station when it receives a transmission or detects an error condition.

PSW. Program status word.

PTF. Program temporary fix.

PUBLIC. In reference to a file pool, all valid users of the system.

PUT. Program update tape.

PVM. VM/Pass-Through Facility.

Q

queue-drop. The action by the system scheduler, DMKSCH, of removing a virtual machine from the list of virtual machines that can be given control of a processor.

queue-drop elimination. A VM/SP performance option that eliminates the dropping of a virtual machine from the active queue if the virtual machine is determined to be idle.

R

read authority. The authority to read the contents of a file without being able to change them. For a directory, read authority lets the user view the names of the objects in the directory.

read-only access. An access mode associated with a virtual disk or SFS directory that lets a user read, but not write or update, any file on the disk or SFS directory.

read/write access. An access mode associated with a virtual disk or SFS directory that lets a user read and write any file on the disk or SFS directory.

receive. (1) Bringing into the specified buffer data sent to the user's virtual machine from another virtual machine or from the user's own virtual machine. (2) To load service files from a service tape.

remote operator console facility (ROCF). A 4300 Series Support Processor microcode function that permits communication from a remote console for functions like IML or IPL using a switched line. The VM/Pass-Through Facility program provides a communication vehicle that lets any of its supported display stations serve as this remote console.

Remote Spooling Communications Subsystem Networking (RSCS). An IBM licensed program and special-purpose subsystem that supports the reception and transmission of messages, files, commands, and jobs over a computer network.

reply. (1) A response to an inquiry. (2) In SNA, a request unit sent only in reaction to a received request unit.

requester. (1) The name given to a virtual machine containing a user program that requests a resource. (2) The program that relays a request to another computer through the SRPI. Contrast with *server*.

reserved file types. (1) File types recognized by the CMS editors (EDIT and XEDIT) as having specific default attributes that include: record size, tab settings, truncation column, and uppercase or lowercase characters associated with that particular file type. The CMS Editor creates a file according to these attributes. (2) File types recognized by CMS commands; that is, commands that only search for and use particular file types or create one or more files with a particular file type.

resource. A program, a data file, a specific set of files, a device, or any other entity or a set of entities that the user can uniquely identify for application program processing in a VM system.

resource ID. A one-to-eight character name that identifies a resource.

resource manager. An application running in a server virtual machine that directly controls one or more VM resources. There are three categories of VM resource managers: global, local, and private.

Restructured Extended Executor (REXX) language. A general-purpose programming language, particularly suitable for EXEC procedures, XEDIT macros, or programs for personal computing. Procedures, XEDIT macros, and programs written in this language can be interpreted by the System Product Interpreter. Also, a component of VM/SP. Contrast with *CMS EXEC language* and *EXEC 2 language*.

REXX language. Restructured Extended Executor language.

ROCF. Remote operator console facility.

route. A connection to another system by a logical link and one or more intermediate systems. In TSAF, a number of links and possible intermediate systems that allow the connection of one system to another.

routing table. A CMS file that contains the information that controls the operation of the Programmable Operator Facility. It lets the Programmable Operator Facility recognize a message as a command, determine the action to take when a message comes in, and recognize the authorized users of programmable operator functions.

RSCS. Remote Spooling Communications Subsystem Networking.

run list. A queue of virtual machines that are executable and currently competing for processor resources. Virtual machines take turns being dispatched for short periods of time (time slices) until they either complete a queue slice or go into a long WAIT state. Virtual machines in the run list can be briefly nonrunnable—for instance, waiting for a page swap—without being dropped from the run list. The virtual machines in the run list are sorted by deadline priority. See *eligible list*.

S

saved segment. A segment of storage that has been saved and assigned a name. The saved segment(s) can be physical saved segment(s) that CP recognizes or logical saved segments that CMS recognizes. The segments can be loaded and shared among virtual machines, which helps use real storage more efficiently, or a private, nonshared copy can be loaded into a virtual machine. See *logical saved segment* and *physical saved segment*.

saved system. A special nonrelocatable copy of a virtual machine's virtual storage and associated registers kept on a CP-owned disk and loaded by name instead of by I/O device address. Loading a saved system by name substantially reduces the time it takes to IPL the system in a virtual machine. In addition, a saved system such as CMS can also share one or more 64K segments of reenterable code in real storage between virtual machines. This reduces the cumulative real main storage requirements and paging demands of such virtual machines.

scale. A line on the System Product Editor's (XEDIT) full-screen display, used for column reference.

screen. An illuminated display surface; for example, the display surface of a CRT. Synonymous with *physical screen*.

SCRIPT/VS. A component of the IBM Document Composition Facility program product available from IBM for a license fee.

secondary user. When a user is disconnected — that is, has no virtual console on line — a secondary user can be designated to receive the disconnected user's console messages and to enter commands to the disconnected user's console.

segment table. In System/370 virtual storage systems, a table used in DAT to control user access to virtual storage segments. Each entry indicates the length, location, and availability of a corresponding page table.

selective preventive service. The selective application of PTFs from the PUT. Contrast with *preventive service*.

server. (1) The general name for a virtual machine that provides a service for a requesting virtual machine. (2) The program that responds to a request from another computer through SRPI. Contrast with *requester*.

server-requester programming interface (SRPI). (1) A protocol between requesters and servers in an enhanced connectivity network. Includes the protocol to define a cooperative processing subsystem. (2) The interface that enables enhanced connectivity between requesters and servers in a network.

service. Changing a product after installation. See *corrective service*, *local service*, and *program update service*.

service machine. A virtual machine running a program that provides system-wide services.

service routines. CP or CMS routines used for addressing and updating directories; formatting or initializing disks; or doing disk, tape, or terminal I/O functions.

service tape. A tape containing service changes for one or more products. See *corrective service tape* and *program update tape (PUT)*.

session. The SNA term for a connection between two LUs. The LUs involved allocate conversations across sessions.

sever. Ending communication with another virtual machine or with the user's own virtual machine.

SFS. Shared file system.

SFS directory. A group of files. SFS directories can be arranged to form a hierarchy in which one directory can contain one or more subdirectories as well as files.

shared file system (SFS). A part of CMS that lets users organize their files into groups known as *directories* and to selectively share those files and directories with other users.

shared read-only system residence disk. A system residence disk tailored so that most of the system residence information is read-only and accessible to all relevant virtual machines, leaving a relatively smaller private read/write system disk that must be dedicated to each virtual machine. This technique can substantially reduce the disk requirements of an installation by avoiding needless duplication of disk packs by virtual machines that use the same operating system. See *saved system*. Synonymous with *read-only system residence disk*.

shared segment. A feature of a saved system or physical saved segment that lets one or more segments of reentrant code or data in real storage be shared among many virtual machines. For example, if a saved CMS system was generated, the CMS nucleus is shared in real storage among all CMS virtual machines loaded by name; that is, every CMS machine's segment of virtual storage maps to the same 64K of real storage. See *discontiguous saved segment* and *saved system*.

shared system. See *saved system* and *shared read-only system residence disk*.

single console image facility (SCIF). (1) Lets a user, who is disconnected from a primary virtual console, continue to have console communications by way of the console of the secondary user. See *secondary user*. (2) Enables a virtual machine operator to control multiple virtual machines from one physical terminal.

SIO. Start I/O.

SNA. Systems Network Architecture.

source code. The input to a compiler or assembler, written in a source language. Contrast with *object code*.

source file. A file that contains source statements for such items as high-level language programs and data description specifications.

spooling. The processing of files created by or intended for virtual readers, punches, and printers. The spool files can be sent from one virtual device to another, from one virtual machine to another, and to real devices. See *virtual console spooling*.

SRPI. Server-requester programming interface.

SSP. System service program.

stack. See *console stack* and *program stack*.

stand-alone. Pertaining to an operation independent of another device, program, or system.

stand-alone dump. A dump acquired without regular system functions. For example, to obtain a CP dump when the regular system is unable to dump the machine,

the stand-alone dump facility gets a CP stand-alone dump.

starter system. A very basic VM/SP system that the user can use to build a production VM/SP system.

starter system tape. The tape that contains the starter system. Starter system tapes are DASD-type specific; for instance, a 3350 starter system cannot be used on any DASD type other than a 3350.

subcommand. The commands of processors such as EDIT or System Product Editor (XEDIT) that run under CMS.

subdirectory. Any directory below a user's top directory. The CREATE DIRECTORY command creates subdirectories. There can be up to eight levels of subdirectories with no limit on the number of them at each level, other than overall DASD space limits. Each level of a subdirectory is an additional identifier of up to 16 characters that is appended to next higher level subdirectory.

supervisor call instruction (SVC). An instruction that interrupts a program being executed and passes control to the supervisor so that it can do a specific service indicated by the instruction.

supervisor state. A state during which the processor can execute I/O and other privileged instructions. Only CP can execute in the supervisor state; all virtual machine operating systems run in problem state. Contrast with *problem state*.

SVC. Supervisor call instruction.

symbolic destination name. A name an APPC/VM or CPI-Communications connection uses. Symbolic destination names index SNA routing and security parameters, which are stored in communications directory NAMES files, to complete a connection.

syntax. The rules for the construction of a command or program.

system administrator. The person responsible for maintaining a computer system.

system load. The combination of active devices, programs, and users that use the system resources of the processor and storage.

System Product Editor. The CMS facility, comprising the XEDIT command and XEDIT subcommands and macros, that lets a user create, change, and manipulate CMS files.

System Product Interpreter. The language processor of the VM/SP operating system that processes procedures, XEDIT macros, and programs written in the REXX language.

system profile. An EXEC (SYSPROF) that resides in a saved system or on a system disk and called by CMS initialization. It contains some initialization functions, and provides a means for installations to override the default CMS environment by tailoring the exec to suit the installation.

system restart. The restart that allows reuse of previously initialized areas. System restart usually requires less time than IPL. See *warm start*.

Systems Application Architecture. A defined set of interfaces, conventions, and protocols that can be used across various IBM systems.

system service program (SSP). In ACF/TCAM, an IBM-supplied or user-supplied program that does system-oriented auxiliary functions in support of the message control program. System service programs run under control of the initiator as attached subtasks.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through and controlling the configuration and operation of networks.

T

target. One of many ways to identify a line to be searched for by the System Product Editor. A target can be specified as an absolute line number, a relative displacement from the current line, a line name, or a string expression.

terminal. A device, usually equipped with a keyboard and a display, capable of sending and receiving information.

terminal input buffer. Holds lines entered at the user's terminal until CMS processes them.

terminal user. Anyone who uses a terminal to log on to VM/SP.

text library. A CMS file that contains relocatable object modules and a directory that indicates the location of each of these modules within the library.

token. An eight-character symbol created by the CMS EXEC processor when it scans an EXEC procedure or EDIT macro statements. Symbols longer than eight characters are truncated to eight characters.

tokenized PLIST (parameter list). A string of doubleword aligned parameters occupying successive doublewords.

top directory. The directory created for a user when the user is enrolled in a file pool. The name of the top directory is the same as the person's user ID.

Transparent Services Access Facility (TSAF). A component of VM/SP that handles communication between systems by letting APPC/VM paths span multiple VM systems. TSAF lets a source program connect to a target program by specifying a name that the target has made known, instead of specifying a user ID and node ID.

TSAF. Transparent Services Access Facility.

TSAF collection. A group of VM processors, each with a TSAF virtual machine, connected by CTC, binary synchronous lines, or LANs.

TSAF virtual machine. The virtual machine that lets user programs connect to and communicate with virtual machines on different VM systems.

U

UCR. User class restructure.

update service. Servicing a part by applying a change to a source file statement, then assembling or compiling the source file to produce a new object file.

user. Anyone who requests the services of a computing system.

user class. A privilege category assigned to a virtual machine user in the user's directory entry; each class specified allows access to a logical subset of all the CP commands. See *privilege class*.

user class restructure (UCR). The extension of the class structure of CP instructions from 8 to 32 classes for each user, command, and diagnose code within the system. This extension allows the installation greater flexibility in authorizing CP instructions.

user data. In reference to a file pool, any data that resides in storage groups 2 through 32767.

user ID. User identification.

user modification. Any change that a user originates for a product or component.

user program. A transaction program that requests a service from a resource manager program. User programs reside in requester virtual machines.

user-written CMS command. Any CMS file created by a user that has a file type of MODULE or EXEC. Such a file can be executed as if it were a CMS command by issuing its file name, followed by any operands or options expected by the program or EXEC procedure.

V

virtual address. The address of a location in virtual storage. A virtual address must be translated into a real address in order to process the data in processor storage.

virtual console. A console simulated by CP on a terminal such as a 3270. The virtual device type and I/O address are defined in the VM/SP directory entry for that virtual machine.

virtual console spooling. The writing of console I/O on disk as a printer spool file instead of, or in addition to, having it typed or displayed at the virtual machine console. The console data includes messages, responses, commands, and data from or to CP and the virtual machine operating system. The user can invoke or terminate console spooling at anytime. When the console spool file is closed, it becomes a printer spool file. Synonymous with *console spooling*.

virtual disk. A logical subdivision (or all) of a physical disk storage device that has its own address, consecutive storage space for data, and an index or description of the stored data so that the data can be accessed. A virtual disk is also called a minidisk. See *disk*.

virtual file. (1) A file or data set residing at a remote computer being used as though it were residing at the local computer (for example, an IBM host computer file being used from a PC as though it were a PC file).
(2) An agent that transforms a request for a DOS file to a request for a VM or MVS System/370 file and lets the user access the file as if it were a PC file.

virtual machine (VM). A functional equivalent of a real machine.

virtual machine communication facility (VMCF). A CP function that provides a method of communication and data transfer between virtual machines operating under the same VM/SP system.

virtual machine group. The concept in GCS of two or more virtual machines associated with each other through the same named system (for example, IPL GCS1). Virtual machines in a group share common read/write storage and can communicate with one another through facilities provided by GCS. Synonymous with *group*.

Virtual Machine/System Product (VM/SP). An IBM licensed program that manages the resources of a single computer so that multiple computing systems appear to exist. Each virtual machine is the functional equivalent of a *real* machine.

Virtual Machine/VTAM Communications Network Application (VM/VCNA). A program that runs in the

VTAM service machine. VM/VCNA controls the physical appearance of the screen when displaying output on a VM/SP terminal attached to an SNA network.

virtual printer (or punch). A printer (or card punch) simulated on disk by CP for a virtual machine. The virtual device type and I/O address are usually defined in the VM/SP directory entry for that virtual machine.

virtual = real option. A VM/SP performance option that lets a virtual machine run in VM/SP's virtual = real area. This option eliminates CP paging and, optionally, CCW translation for this virtual machine. Synonymous with $V = R$.

virtual screen. A functional simulation of a physical screen. A virtual screen is a *presentation space* where data is maintained. The user can view pieces of the virtual screen through a window on the physical screen.

virtual storage. Storage space that can be regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses. The size of virtual storage is limited by the addressing scheme of the computing system and by the amount of auxiliary storage available, and not by the actual number of main storage locations.

virtual storage access method (VSAM). An access method for direct or sequential processing of fixed and variable-length records on direct access devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry-sequence), or by relative-record number.

virtual storage extended (VSE). The generalized term that indicates the combination of the DOS/VSE system control program and the VSE/Advanced Functions program product. Note that in certain cases, the term DOS is still used as a generic term; for example, disk packs initialized for use with VSE or any predecessor DOS or DOS/VS system are sometimes called DOS disks. Also note that the DOS-like simulation environment provided under the VM/SP CMS component and CMS/DOS exists on VM/SP and VM/SP HPO program products and continues to be called CMS/DOS.

Virtual Telecommunications Access Method (VTAM). An IBM licensed program that controls communication and the flow of data in a computer network. It provides single-domain, multiple-domain, and multiple-network capability. VTAM runs under MVS, OS/VS1, VM/SP, and VSE.

VM. Virtual machine.

VMCF. Virtual machine communication facility.

VMLIB. The name of the CSL supplied with VM/SP and that contains routines to do various VM functions.

VM/Pass-Through Facility. A facility that lets VM users interactively access remote system and processor nodes. These can be remote IBM 4300 processors, other VM systems, with or without this facility installed, or System/370-compatible non-VM systems.

VM/SP. Virtual Machine/System Product.

VM/SP directory. A CP disk file that defines each virtual machine's typical configuration; the user ID, password, regular and maximum allowable virtual storage, CP command privilege class or classes allowed, dispatching priority, logical editing symbols to be used, account number, and CP options desired. Synonymous with *CP directory*.

VM/VCNA. Virtual Machine/VTAM Communications Network Application.

V = R. Synonym for *virtual = real option*.

VSAM. Virtual storage access method.

VSCS. VTAM SNA Console Support.

VSE. Virtual storage extended.

VTAM. Virtual Telecommunications Access Method.

W

WAN. Wide area network.

warm start. (1) The result of an IPL that does not erase previous system data. (2) The automatic reinitialization of the VM/SP control program that occurs if the control program cannot continue processing. Closed spool files and the VM/SP accounting information are not lost. Contrast with *checkpoint (CKPT) start*, *cold start*, and *force start*.

wide area network (WAN). A network that provides communication services to a geographic area larger than that served by an LAN. Contrast with *local area network (LAN)*.

window. An area on the physical screen where virtual screen data can be displayed. Windowing lets the user do such functions as defining, positioning, and overlaying windows; scrolling backward and forward through data; and writing data into virtual screens.

windowing. A set of functions that lets the user view and manipulate data in user-defined areas of the

physical screen called *windows*. Windowing support lets the user define, position, and overlay windows; scroll backward and forward through data; and write data into virtual screens.

write authority. The authority to read or change the contents of a file or directory. Write authority implies read authority.

X

XEDIT. See *System Product Editor*.

Z

zap. To modify or dump an individual text file, using the ZAP command or the ZAPTEXT EXEC.

Numerics

3033. Refers to the IBM 3033 Processor.

3088. Refers to the IBM 3088 Multisystem Communications Unit, Models 1 and 2.

3270. Refers to a series of IBM display devices; for example, the IBM 3275, 3276 Controller Display Station, 3277, 3278, and 3279 Display Stations, the 3290 Information Panel, and the 3287 and 3286 printers. A specific device type is used only when a distinction is required between device types. Information about display terminal usage also refers to the IBM 3138, 3148, and 3158 Display Consoles when used in display mode, unless otherwise noted.

3350. Refers to the IBM 3350 Direct Access Storage Device when used in native mode.

3380. Refers to the IBM 3380 Direct Access Storage Device.

3725. Refers to the IBM 3725 Communication Controllers.

3800. Refers to the IBM 3800 Printing Subsystems. A specific device type is used only when a distinction is required between device types.

3880. Refers to the IBM 3880 Storage Control Units.

9370. Refers to a series of processors, namely the IBM 9373 Model 20 and 30, the IBM 9375 Models 40, 50, and 60, and the IBM 9377 Model 80 and 90.

Bibliography

The following lists the abstracts of books in the VM/SP library. The charts show how the books relate to general tasks. To find specific information in the VM/SP library, use the *VM/SP Library Guide and Master Index*, GC19-6207.

For descriptions of programs that run on VM/SP, see the references cited in related topics in this book. For a complete list of IBM documentation, see the *IBM System/370, 30xx, 4300, and 9370 Processors Bibliography*, GC20-0001.

These abstracts are to help you select the book(s) best suited as reference information for your needs. The titles are listed alphabetically.

Administration, SC24-5285

This is a reference book for the system administrator and others who carry out and extend the functions of IBM's Virtual Machine/System Product (VM/SP). It assumes some experience with programming concepts and techniques. This reference consists of three parts and an appendix:

Part 1: Tailoring Your VM/SP (Chapters 1-9) describes some functions of VM/SP and provides guidance to the system administrator in tailoring those functions.

Part 2: Performance (Chapters 10-11) describes options available in VM/SP to analyze and improve the performance of virtual machines and operating systems.

Part 3: Printer Information (Chapters 12-13) provides information about various types of printers.

Appendix, "VM/SP Monitor Tape Format and Content" describes the format and contents of data records for classes and codes of MONITOR CALL.

Application Development Guide for CMS, SC24-5286

This book provides application programmers and system programmers with information about CMS. This information includes details on the CMS programming interface and CMS architecture, using CMS native services (to handle interrupts, obtain free storage, perform I/O, and process abends), and managing CMS programs (using CMS services to build, load, execute, and update programs and program packages).

It also includes information on developing OS/MVS and VSE programs under CMS, and using Access Method Services and VSAM under CMS and CMS/DOS. Much of the information in this book was formerly found in the *VM/SP CMS for System Programming* book.

Application Development Guide for FORTRAN and COBOL, SC24-5247

This book tells FORTRAN and COBOL application programmers how to compile, link, load, run, test, and debug programs using CMS. It also has information on using the Interactive System Productivity Facility (ISPF) for dialog management and the Structured Query Language/Data System (SQL/DS) for data base management.

Application Development Reference for CMS, SC24-5284

This book gives CMS users detailed reference information about CMS assembler language macro instructions and CMS functions.

CMS Command Reference, SC19-6209

This book gives users of the Conversational Monitor System (CMS) component of VM/SP detailed reference information about command syntax and usage for: CMS commands; XEDIT subcommands; HELP format words; DEBUG commands; exec control statements, special variables, and integrated functions.

CMS Primer, SC24-5236

This book teaches you, as a new user, how to do your work using VM/SP and a video display terminal. The primer presents only a subset of all the functions and commands available with VM/SP. The material is presented in an interesting manner with suggested exercises included in the text.

CMS Primer for Line-Oriented Terminals, SC24-5242

This book is an interactive tutorial for VM/SP users of line-oriented (line mode) video display terminals. The book, which is similar in scope and content to the *VM/SP CMS Primer*, quickly gives the reader a working knowledge of VM/SP. Topics include logging on; editing, managing, and printing files; using the Document Composition Facility (SCRIPT/VS) to format files; and writing execs.

CMS Shared File System Administration, SC24-5367

This book is for those who will be administering the Shared File System (SFS) on VM/SP. It assumes that you are familiar with DASD management on VM/SP and with VM/SP system directory control statements. You should also be familiar with the use of the SFS as described in the *CMS User's Guide*. SFS is a part of the CMS component of VM/SP.

SFS administration includes generating file pools and managing their operation and use. This book is organized in two parts:

- **Part 1: Guidance.** This part explains the concepts and procedures needed to effectively administer one or more file pools. The first chapter provides a brief overview of SFS and what one must do to administer a file pool. The remaining chapters contain concepts and procedures related to a particular area of file pool administration. There are chapters on operation, recovery, security, and managing users. You do not need to read those chapters in any particular order. Each is self contained and can be read as needed.
- **Part 2: Reference.** This part of the book contains all administration commands and file pool server commands in reference form. It is **not** recommended that you use the reference until you have gained some familiarity with the procedures in Part 1.

CMS User's Guide, SC19-6210

This book has general information on the Conversational Monitor System (CMS) component of VM/SP. It has information and examples regarding the CMS file system, the CMS batch facility, the HELP facility, full-screen CMS, and window functions. Also included are examples on organizing and sharing your files using the shared file system, using the System Product Editor to create and edit CMS files, and using the System Product Interpreter to create and use execs.

Connectivity Planning, Administration, and Operation, SC24-5378

This book is intended for someone who administers or operates the components of VM that allow programs to communicate within a single TSAF collection and an SNA network. This book provides an overview of the products and the VM/SP components related to connectivity and an overview of the programming interface, Advanced Program-to-Program Communication/Virtual Machine (APPC/VM) and Common Programming Interface (CPI) Communications. This book also contains all of the reference material needed to use the Transparent Services Access Facility (TSAF) and APPC/VM VTAM Support (AVS). It includes information about how to run the TSAF virtual machine, how to run the AVS virtual machine, and what system services are provided with them.

Connectivity Programming Guide and Reference, SC24-5377

This book is intended for someone writing an application program to communicate with another application program using APPC protocol. The communication can be within a single TSAF collection, across multiple TSAF collections, or between a TSAF collection and an SNA network.

The book explains two ways to do this programming:

1. Using the Common Programming Interface (CPI) Communications from *high-level* programming languages such as VS FORTRAN.

(Only communication routines unique to VM are described in this book; common routines are contained in the *SAA Common Programming Interface Communications Reference*, SC26-4399.)

2. Using the APPCVM and IUCV macro interface from assembler language programs.

CP General User Command Reference, SC19-6211

This publication is a reference manual for class G and class Any users who are running systems such as OS, OS/VS, DOS, DOS/VS, and VSE systems, CMS, and networking systems in a virtual machine under VM/SP. Control Program (CP) commands available to privilege classes G and Any are listed alphabetically. Each command description contains general usage information, the command line format, descriptions of all allowable operands, and default values for operands.

CP System Command Reference, SC24-5402

This publication is a reference manual for class A, B, C, D, E, and F users who are running systems such as OS, OS/VS, DOS, DOS/VS, and VSE systems, CMS, and networking systems in a virtual machine under VM/SP. Control Program (CP) commands available to privilege classes A, B, C, D, E, and F are listed alphabetically. Each command description contains general usage information, the command line format, descriptions of all allowable operands, and default values for operands.

Distributed Data Processing Guide, SC24-5241

This book describes the concepts and facilities of distributed data processing when using VM/SP and related programs.

EXEC 2 Reference, SC24-5219

This book defines the EXEC 2 language. It has all the formats, syntax rules, and descriptions of the arguments for EXEC 2 statements. An EXEC 2 primer for new users is included. It summarizes the language and its capabilities. A detailed discussion of the different types of EXEC 2 statements is followed by examples. It lists the error messages and return codes issued by the EXEC 2 interpreter.

General Information, GC20-1838

This book introduces and describes the features and facilities of VM/SP, and provides customer management and technical staffs with information needed to evaluate the applicability of VM/SP to their installations. It has information needed for a basic understanding of using, programming, and installing VM/SP. This book also lists some program products that work with VM/SP.

Group Control System Command and Macro Reference, SC24-5250

This book has detailed reference material that describes the functions and use of all macros supported in the Group Control System (GCS). Each macro description has information on general usage, format, all available parameters, messages, and return codes. The book is for personnel who are developing application programs to run on GCS.

Installation Guide, SC24-5237

This book is for system programmers and anyone responsible for installing VM/SP. It contains step-by-step procedures for using the Starter System or an existing VM/SP system to install VM/SP. In addition, there are procedures for verifying an installed system, installing saved segments, and installing a new system national language.

Interactive Problem Control System Guide and Reference, SC24-5260

This book describes the major functions of IPCS and gives insight into its facilities. It describes the operation and generation procedures of IPCS.

Operator's Guide, SC19-6202

This book is for users responsible for the operation and administration of VM/SP. It includes descriptions of all the commands that affect the I/O resources and operating characteristics of VM/SP, the associated virtual machines, and the real hardware configuration. Also included is information on spooling, resource allocation, system startup and shutdown procedures, and VM/SP service programs.

Planning Guide and Reference, SC19-6201

This book is for people responsible for the planning, installing, and updating of a VM/SP system. This publication includes information about:

- Planning for system generation
- Defining your VM/SP system
- Generating a 3704/3705/3725 control program that runs with VM/SP
- Updating VM/SP.

The reader is expected to have a general understanding of data processing and teleprocessing techniques.

Programmer's Guide to the Server-Requester Programming Interface for VM/SP, SC24-5291

This book gives an application programmer information on how to write and install IBM System/370 to IBM Personal Computer Enhanced Connectivity Facilities servers in a VM/SP system. The workstation user (for example, IBM Personal Computer) receives information on how to start IBM System/370 to IBM Personal Computer Enhanced Connectivity Facilities communications on VM/SP.

Release 6 Guide, SC24-5368

This book describes the functional changes to VM/SP for Release 6. It includes details for migrating from VM/SP Release 5 to VM/SP Release 6.

RSCS Networking General Information, GH24-5055

This book has an overview of the Remote Spooling Communication Subsystem (RSCS) Networking program product. It includes a functional description, discussion of requirements for using the product, and a program summary.

RSCS Planning and Installation, SH24-5057

This book describes how to prepare to install RSCS, what to consider, and what options you have to choose from. Also, it describes the procedures for installing, customizing, and automating RSCS.

Running Guest Operating Systems, GC19-6212

This book is for people interested in running their operating system (VSE, MVS, VM/SP, and VM/SP HPO) host systems. The book presents the basic processes so that the working system is set up quickly. Experienced people improve the efficiency of their installation, running a guest system under VM/SP and VM/SP HPO, with the included recommendations. The book assumes the audience is knowledgeable about their own system, but new to VM/SP and VM/SP HPO.

Service Guide, SC24-5389

This book is for system programmers and anyone responsible for servicing VM/SP. It contains step-by-step procedures for program update (preventive) service, corrective service, and local service. It also contains descriptions of the various service tools and service files, including the product parameter file.

System Facilities for Programming, SC24-5288

This book supplies system programmers with detailed information on facilities available in VM/SP and VM/SP HPO. These facilities include the DIAGNOSE instruction, the Inter-User Communications Vehicle (IUCV) for CP and CMS, CP System Services, the Virtual Machine Communication Facility (VMCF), the programmable operator facility, and information on getting national languages on your system.

System Messages and Codes, SC19-6204

This book has messages and codes, and restrictions, that can be met when using the Conversational Monitor System, Control Program, Interactive Problem Control System, Group Control System, and Transparent Services Access Facility. Conditions that generate these messages and codes are explained, the resulting action is described, and appropriate responses are suggested.

System Messages Cross-Reference, SC24-5264

This book has cross-reference information for the messages in the book, *VM/SP System Messages and Codes*, SC19-6204. The cross-reference information relates messages to the commands that caused them to be entered, lists messages alphanumerically by message identifier, relates messages to the module that issued them, and lists the messages in alphanumeric order by message text. The book has five sections, one for each of the VM/SP components: Conversational Monitor System, Control Program, Interactive Problem Control System, Group Control System, and Transparent Services Access Facility.

System Product Editor Command and Macro Reference, SC24-5221

This is a reference book that has all the command formats, syntax rules, and operand and option descriptions for the XEDIT subcommands and macros. It tells how to enter XEDIT commands, subcommands and macros. It has the format description, and operand and option list for the XEDIT command, which is used to run the editor. It lists EDIT subcommands and their XEDIT counterparts. It tells how to define windows and virtual screens used with the CMS Session Services. You should be familiar with the information in the *VM/SP System Product Editor User's Guide* before attempting to use this book.

System Product Editor User's Guide, SC24-5220

This book is for the individual who has limited data processing experience. It gives the user a working knowledge of the System Product Editor (executed by the XEDIT command). XEDIT provides many functions for text processing and programming development. Both a full screen and a line mode editor are used on display and typewriter terminals.

System Product Interpreter Reference, SC24-5239

This book has statement syntax, format, and usage notes for using Restructured Extended Executor (REXX), the command language for writing procedures to run under System Product Interpreter, the command language processor of CMS.

System Product Interpreter User's Guide, SC24-5238

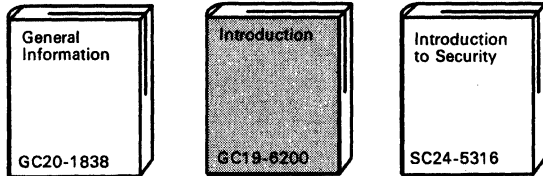
This tutorial book describes and has examples of how to use Restructured Extended Executor (REXX), the command language for writing procedures to run under System Product Interpreter, the command language processor of CMS.

Terminal Reference, GC19-6206

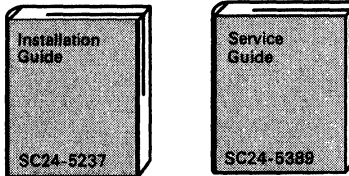
This book describes some of the many devices that are used as VM/SP I/O terminals. It includes descriptions of physical characteristics and examples of how to use the terminals.

VM/SP RELEASE 6 LIBRARY

Evaluation



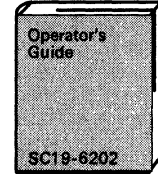
Installation and Service



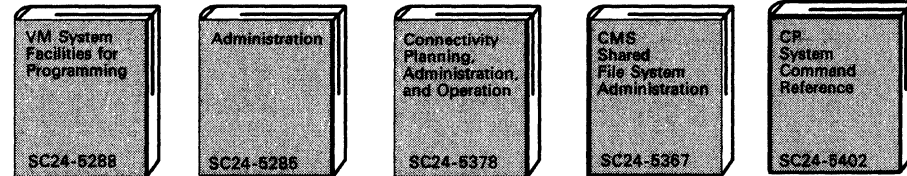
Planning



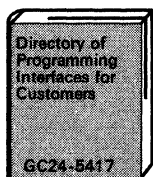
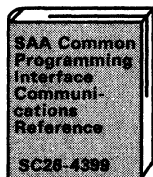
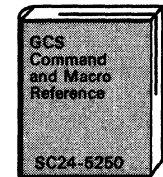
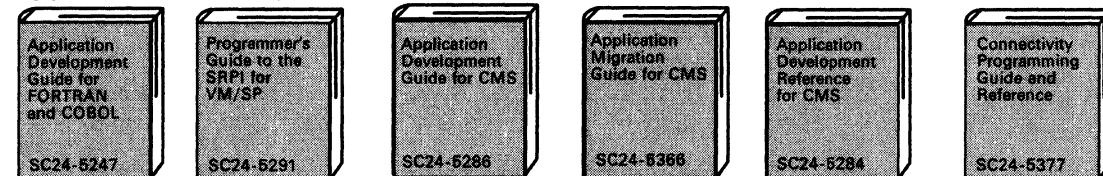
Operation



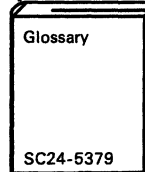
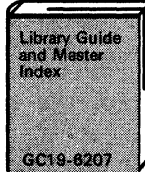
Administration



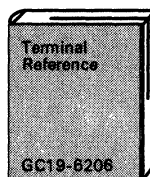
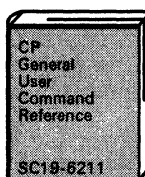
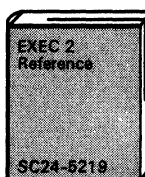
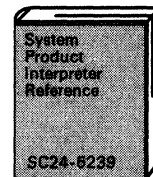
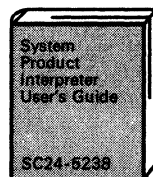
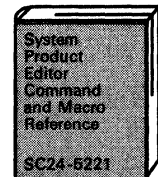
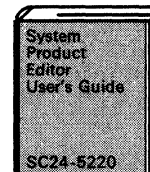
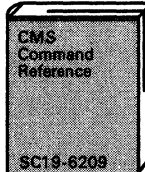
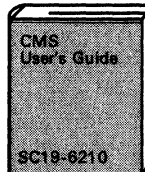
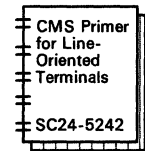
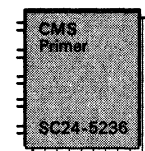
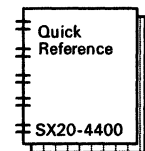
Application Development




Index/Glossary



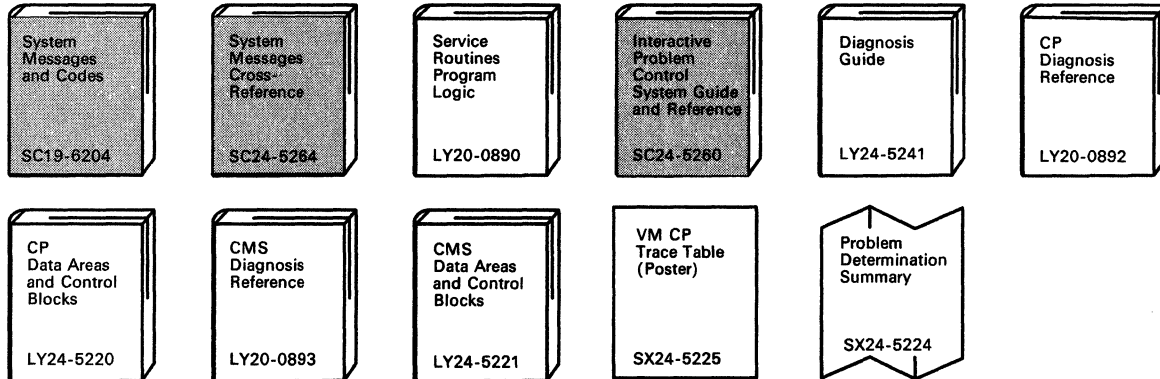
End Use



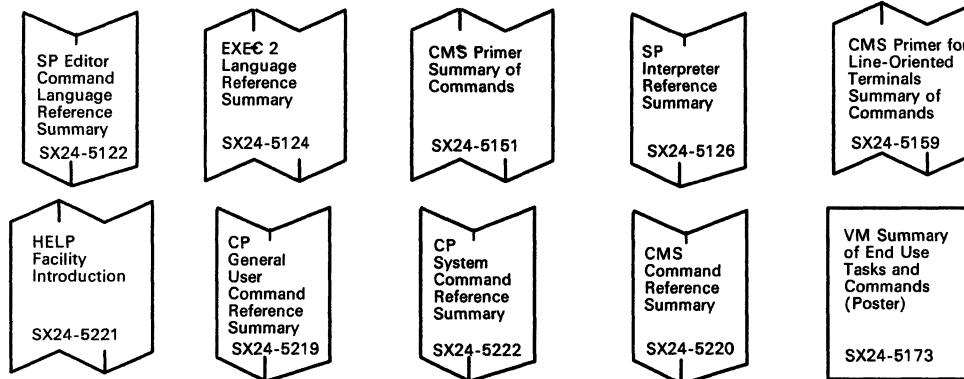
 one copy of each shaded manual received with product tape

VM/SP RELEASE 6 LIBRARY

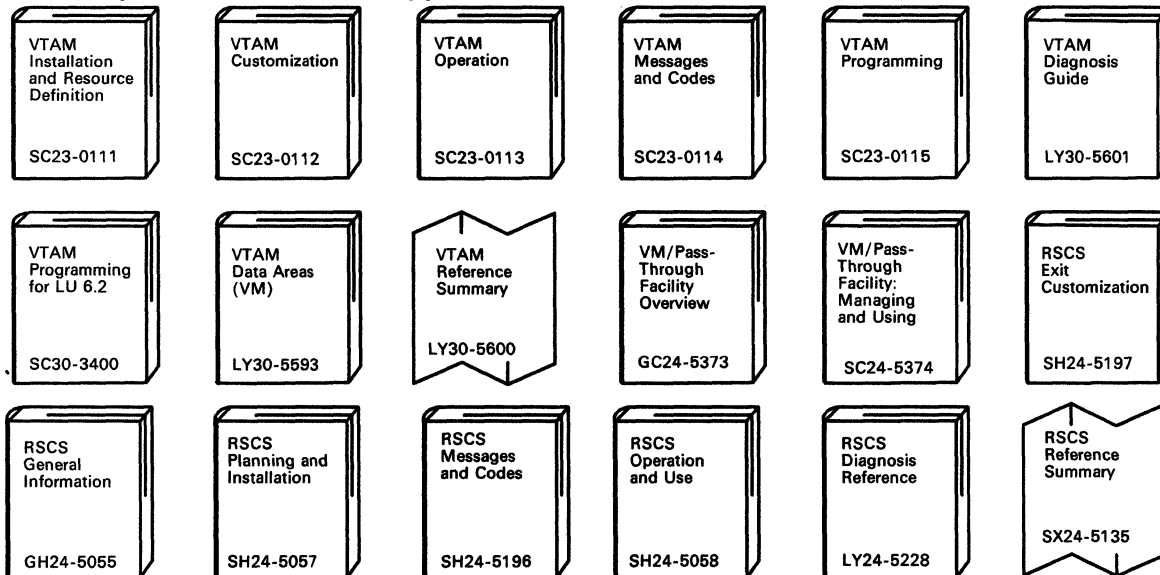
Diagnosis



Reference Summaries



Auxiliary Communication Support



Index

A

- abstracts of VM/SP manuals
 - Administration 241
 - Application Development Guide for CMS 241
 - Application Development Guide for FORTRAN and COBOL 241
 - Application Development Reference for CMS 241
 - CMS Command Reference 241
 - CMS Primer 242
 - CMS Primer for Line-Oriented Terminals 242
 - CMS Shared File System Administration 242
 - CMS User's Guide 242
 - Connectivity Planning, Administration, and Operation 242
 - Connectivity Programming Guide and Reference 242
 - CP General User Command Reference 243
 - CP System Command Reference 243
 - Distributed Data Processing Guide 243
 - EXEC 2 Reference 243
 - General Information 243
 - Group Control System Command and Macro Reference 243
 - Installation Guide 243
 - IPCS Guide and Reference 244
 - Operator's Guide 244
 - Planning Guide and Reference 244
 - Programmer's Guide to the Server-Requester Programming Interface for VM/SP 244
 - Release 6 Guide 244
 - RSCS Networking Version 2 General Information 244
 - RSCS Planning and Installation 244
 - Running Guest Operating Systems 244
 - Service Guide 245
 - System Facilities for Programming 245
 - System Messages and Codes 245
 - System Messages Cross-Reference 245
 - System Product Editor Command and Macro Reference 245
 - System Product Editor User's Guide 245
 - System Product Interpreter Reference 245
 - System Product Interpreter User's Guide 245
 - Terminal Reference 246
- ACCESS command
 - accessing directories 97
 - accessing minidisks 94
 - batch job statement 52
 - example of use 19
- Access Method (ACF/VTAM) for VM/SP 57
- ACF/SSP (Advanced Communications Function/System Support Programs) 123
 - ACF/VTAM (Advanced Communications Function for VTAM) 116, 122
 - address space 6
 - address translation 16
 - ADSTOP command, debugging aid 44
 - advanced application, COBOL 166
 - Advanced Communications Function for VTAM (ACF/VTAM) 116, 122
 - Advanced Communications Function/System Support Programs (ACF/SSP) 123
 - Advanced Communications Function/Virtual Telecommunications 57
 - Advanced Program-to-Program Communication 110
 - AFFINITY option for system generation 75
 - aid
 - PL/I debugging 169
 - program development, COBOL 166, 168
 - allocation of processor time 14
 - APL2 180
 - APPC/VM 110
 - connectivity 115
 - description 119
 - APPC/VM VTAM Support (AVS)
 - See also* AVS (APPC/VM VTAM Support)
 - support 113, 116
 - application
 - advanced, COBOL 166
 - checking new programs 43
 - debugging applications 43
 - developing under CMS 37
 - facility for debugging 43
 - IBM Enhanced Connectivity Facilities 127
 - program products 133, 165
 - running under CMS 48
 - steps in developing 37
 - Application System (AS) 134
 - apply local updates to VM/SP 219
 - ASMGEND EXEC 220
 - ASSEMBLE command
 - definition 49
 - example of use 38
 - ASSEMBLER H 175
 - Assembler language 175
 - ASSEMBLER XF 38
 - assemble, defined 37
 - assist, virtual machine
 - See* virtual machine assist
 - ATTACH command 11
 - attribute, LRECL 92
 - authorization to use system 71
 - AUTHORIZE command 33
 - automated CMS initialization 80
 - automated procedures 82

AUTOSAVE option, example of use 89
AVS (APPC/VM VTAM Support)
 See also APPC/VM VTAM Support (AVS)
 support 113, 116

B

BACKSPAC command 24
base file 97
BASIC, IBM 182
batch job statement
 ACCESS 52
 LINK 52
batch processing under CMS 52
batch processing, VM Batch Subsystem 53
buffer, terminal input 106
build, definition 150
business professional product
 APL2 180
 SQL/DS with QMF 155

C

cache, description 203
Callable Services Library (CSL)
 See also CSL (Callable Services Library)
 description 50
 OS/VS COBOL 168
 PL/I 171
 VS COBOL II 168
 VS FORTRAN 178
 VS Pascal 173
central site, DDP defined 185
 description 189
CHANGE
 batch machine output 53
 command 22
 example of use 22
CICS/VM 40, 110
class
 device 69
 privilege 60
CLASS control statement 60, 63
CLASS OVERRIDE control statement 60
CLOSE command, batch machine output 53
CMS Shared File System (SFS)
 See also SFS (CMS Shared File System)
 accessing directories 97
 creating aliases 97
 description 95
 locking files and directories 99
 PUBLIC key word 100
 sharing files or directories 94, 98
CMS virtual machine 81
CMS (Conversational Monitor System)
 See also Conversational Monitor System (CMS)
 assembling programs 38
 batch processing 52

CMS (Conversational Monitor System) (*continued*)
 communicating with 80
 compiling programs 39
 definition 3
 developing applications 37
 dump 45
 exec procedure 86
 facility for debugging programs 43
 file system 91
 general description 79–108
 HELP facility 104
 initializing 80
 introductory description 3
 message 28
 minidisk 18
 running programs 48
 subcommands 82
 user task 3, 84
 user's view 84
CMSBATCH command, example of use 52
MSGEND EXEC 220
COBOL 166
color graphics 148
 GDDM 148
command language processor 86
commands
 ACCESS 19
 ADSTOP 44
 ASSEMBLE 38
 ATTACH 11
 AUTHORIZE 33
 BACKSPAC 24
 CHANGE 22
 CLOSE 53
 CMSBATCH 52
 COPYFILE 104
 CP 32
 DEBUG 43
 DEFINE 20, 64
 DETACH 20, 64
 DIAL 126
 DIRMAINT 201
 DISCARD 23
 DISCONN 64
 DISPLAY 44
 DRAIN 24
 DTRIPF 198
 EXECIO 107
 FILEDEF 43, 48, 49
 FLUSH 25
 FORCE 32
 FREE 25
 general 32
 GENMOD 49
 GLOBAL 43
 HASM 175
 HOLD 25
 INCLUDE 49

commands (continued)

INDICATE 12
LINK 20, 64
LOAD 49
LOADBUF 25
MIGRATE 17
MONITOR 74
NAMES 35
NetView PROP 193
ORDER 24, 29
PASSTHRU 126
PEEK 23
PER 44
PLIOPT 49
privileged 32
PURGE 24, 35
QUERY 12
RDRLIST 23, 35
RECEIVE 23, 35, 81
REPEAT 25
RUN 50
SCRIPT 141
SENDFILE 23, 35
SET 28, 33, 64, 86
SET MSG 33
SHUTDOWN 29, 32
SPACE 25
SPMODE 76
SPOOL 22, 23, 25
SPTAPE 25
START 49
SVCTRACE 44
TERMINAL 64
TRACE 44
TRANSFER 24
VARY 32
VMDUMP 45
XEDIT 81

communicating
 between virtual machines 67
 between VM/SP and workstations 127
 system operator with VM/SP 28
 user with VM/SP 27
 with CMS 80
 with CP 32, 60
 with other virtual machines 33
 with system operator 34

compilers
 COBOL 166
 documentation references 41
 OS/VS COBOL 166
 PL/I 169
 supported by VM/SP 39
 VS COBOL II 167
 VS Pascal 172

compile, defined 37
component, network 189

concepts of VM/SP 30
configurations for VM/IS 208
configuration, defining device 69
connecting
 workstations to VM/SP 127

Connectivity Facilities on VM/SP
 APPC 119
 APPC/VM VTAM Support (AVS) 113
 applications 127
 CPI Communications 119
 needs of 127
 TSAF 113

console 9
 disconnected 61
 single console image facility 61
 spooling 25, 53
 stack 106
 virtual console management 61

Console Communication Service (CCS) 67
CONSOLE directory control statement 61, 63
console spooling 25
console stack 106
console usage by system operator 9

Control Program (CP)
 See also CP (Control Program)
 command 32
 communicating with 32
 definition 2
 facilities for debugging programs 44
 general description 59-77
 introductory description 2
 message 28
 processor time allocation 14
 real resource management 13, 60
 real storage allocation 14
 running operating systems 55
 pool management 15, 21
 system service 67
 Console Communication 67
 DASD Block I/O 67
 Message 67
 Signal System Service 67
 virtual machine creation 13
 virtual storage management 15

control statements
 CLASS 60, 63
 CLASS OVERRIDE 60
 CONSOLE 63
 DEDICATE 61
 MDISK 63
 USER 63

control word, SCRIPT 139

Conversational Monitor System (CMS)
 See also CMS (Conversational Monitor System)
 assembling programs 38
 batch processing 52
 communicating with 80
 compiling programs 39

Conversational Monitor System (CMS) (*continued*)

- definition 3
 - developing applications 37
 - dump 45
 - exec procedure 86
 - facility for debugging programs 43
 - file system 91
 - general description 79–108
 - HELP facility 104
 - initializing 80
 - introductory description 3
 - message 28
 - minidisk 18
 - running programs 48
 - subcommands 82
 - user task 3, 84
 - user's view 84
- COPYFILE command, example of use 104
- corrective fixes, description 219
- CP (Control Program)
- See also* Control Program (CP)
 - command 32
 - communicating with 32
 - definition 2
 - facilities for debugging programs 44
 - general description 59–77
 - introductory description 2
 - message 28
 - processor time allocation 14
 - real resource management 13, 60
 - real storage allocation 14
 - running operating systems 55
 - spool management 15, 21
 - system service 67
 - Console Communication 67
 - DASD Block I/O 67
 - Message 67
 - Signal System Service 67
 - virtual machine creation 13
 - virtual storage management 15
- CSL (Callable Services Library)
- See also* Callable Services Library (CSL)
 - description 50
 - OS/VS COBOL 168
 - PL/I 171
 - VS COBOL II 168
 - VS FORTRAN 178
 - VS Pascal 173
- CSLGEN EXEC 220

D

- DASD Block I/O System Service 67
- DASD (direct access storage device)
 - See also* direct access storage device (DASD)
 - factors affecting requirements 215
 - minidisk 8, 11, 18
 - preferred DASD area 17

DASD (direct access storage device) (*continued*)

- shared DASD 11
 - specified in system directory 18
 - virtual storage 6
- data file, sharing 12
- data integrity 71
- database (SQL) 152
- DCF (Document Composition Facility) 139
- DCSSGEN EXEC 220
- DDP (distributed data processing)
- See also* distributed data processing (DDP)
 - advantages 187
 - definition 185
 - distributed system requirements 191
 - network components 189
 - programmable operator facility 192
 - related terms defined 185
 - VM/SP facilities 187
- DEBUG command described 43
- debugging
- ADSTOP command 44
 - aid, PL/I 169
 - application programs 43
 - VM/SP 77
- DEDICATE control statement 61
- dedicated tape device 11
- DEFINE command, example of use 20, 64
- defining system device configuration 69
- demand paging 16
- definition 17
- DETACH command 20
- example of use 64
- developing applications 37
- devices
- class 69
 - configuration, defining 69
 - VM/SP I/O 69
- device, unit record 8
- DIAGNOSE '94' 45
- diagnosing system programming problems 77
- DIAL command 126
- direct access storage device (DASD)
- See also* DASD (direct access storage device)
 - factors affecting requirements 215
 - minidisk 8, 11, 18
 - preferred DASD area 17
 - shared DASD 11
 - specified in system directory 18
 - virtual storage 6
- Directory Maintenance 200
- directory, system
- CMS Shared File System 95
 - data contained 13
 - definition 8
 - for RSCS systems 124
 - maintenance 200
 - virtual machine definition 13, 61, 63

DIRMAINT command, uses 201
 dirname 96
 DISCARD command 23
 DISCONN command, example of use 64
 disconnected console 61
 DISPLAY command, as debugging aid 44
 Display Management System for CMS
 (DMS/CMS) 161
 DisplayWrite/370 142
 distributed data processing (DDP)
See also DDP (distributed data processing)
 advantages 187
 definition 185
 distributed system requirements 191
 network components 189
 programmable operator facility 192
 related terms defined 185
 VM/SP facilities 187
 distributed system 185
 DDP defined 185
 description 189
 distributed system requirements 191
 DMKRIO file, defined 214
 DMKSNT file, defined 214
 DMKSYS file, defined 214
 DMS/CMS (Display Management System for
 CMS) 161
 document
 preparation 135
 using PROFS 136
 Document Composition Facility (DCF) 139
 document, proofreading 136
 DOSGEN EXEC 220
 DRAIN command 24
 DTRIPF command, example of use 198
 dump data used by IPCS 45
 dynamic address translation 16

E
 ECPS:VM/370 option, SET command 76
 editing, full-screen 88
 editor, image symbol 147
 editor, VM/SP 88
 Enhanced Connectivity Facilities on VM/SP
 applications 127
 for VM/SP 127
 needs of 127
 exec procedure described 86
 EXECIO command options 107
 EXECUPDT EXEC 220
 EXECUTE facility of SQL/DS 160
 executing Assembler H 175
 explicit locks 99
 extended control program support:VM/370 option, SET
 command 76

F
 FAVORED option of SET command 75
 file identifier
 file mode, definition and use 93
 file name, definition and use 93
 file type, definition and use 93
 file mode, definition and use 93
 file name, definition and use 93
 file pool, description
 CMS Shared File System 95
 file sharing 94
 file space 19, 95
 File Storage Facility program offering 94
 FILE subcommand of XEDIT, example of use 89
 file system of CMS 91
 file transfer 127
 file type, definition and use 93
 FILEDEF command
 definition 49
 example of use 43, 48
 fixed-length record, description 91
 fixing system problems 77
 FLUSH command 25
 FORCE command 32
 forecasting 134
 formatting and printing documents 137
 formatting minidisks 20
 FORTRAN, VS 177
 frame 16
 FREE command, operator use 25
 full-screen editing 88
 full-screen mode, HELP facility 104

G
 gateways 117
 GCS (Group Control System)
See also Group Control System (GCS)
 dump 45
 installing 216
 introductory description 3, 116
 profile 72
 virtual machine group 116
 GDDM (Graphical Data Display Manager) 147
 general command 32
 general user
 communicate with VM/SP 27
 console usage 9
 view of spooling 23
 Generalized Markup Language (GML) 140
 GENMOD command, definition 49
 GENMOD EXEC 220
 GENSERVE EXEC 220
 GENTSAF EXEC 220
 getting documents, PROFS 137
 GLOBAL command
 definition 49

GLOBAL command (*continued*)
 example of use 43, 48
GML (Generalized Markup Language) 140
Graphical Data Display Manager (GDDM) 147
 graphics 135
 graphics, color 148
 GDDM 148
Group Control System (GCS)
 See also GCS (Group Control System)
 dump 45
 installing 216
 introductory description 3, 116
 profile 72
 virtual machine group 116
group, virtual machine 116
guest operating system 11

H

HASM command to execute Assembler H 175
HELP facility
 description 104
 example 81
High Performance Option, VM/SP
 See also VM/SP HPO (High Performance Option)
 free storage management improvements 202
 page migration enhancement 202
 single processor mode 76
HOLD command, operator use 25
host data base access 127

I

IBM BASIC 182
identifier, file 93
image symbol editor for GDDM 147
implicit locks 99
INCLUDE command, defined 49
INDICATE command, defined 12, 74
initializing CMS 80
initiating Pass-Through sessions 126
input buffer, terminal 106
input device 69
installing VM/IS 208
installing VM/SP
 basic processes 216
 starter system 216
instruction, macro 175
Integrated System (VM/IS) 207
integrity
 data 71
 data, during editor session 89
 power failure 72
 SQL database 153
Inter-User Communications Vehicle (IUCV)
 See also IUCV (Inter-User Communications Vehicle)
 communication between virtual machines 67
 communication with CP system services 67

Inter-User Communications Vehicle (IUCV) (*continued*)
 description 66
interactive menus 161
Interactive Problem Control System (IPCS)
 See also IPCS (Interactive Problem Control System)
 description 45
Interactive Productivity Facility (IPF) 198
Interactive SQL (ISQL) 153
Interactive System Productivity Facility (ISPF) 199
interactive, defined 2
IPCS (Interactive Problem Control System)
 See also Interactive Problem Control System (IPCS)
 description 45
IPF (Interactive Productivity Facility) 198
IPL control statement 63
ISPF (Interactive System Productivity Facility) 199
ISQL (Interactive SQL) 153
IUCV (Inter-User Communications Vehicle)
 See also Inter-User Communications Vehicle (IUCV)
 communication between virtual machines 67
 communication with CP system services 67
 description 66

K

keys for program function (PF) 89

L

language processors 86
libraries
 COBOL subroutine 167
 PL/I subroutine 170
 VS Pascal 172
licensed program
 APL2 180
 COBOL 166
 defined 39
 Directory Maintenance 200
 Display Management System (DMS/CMS) 161
 Document Composition Facility (DCF) 139
 examples 57
 Graphical Data Display Manager (GDDM) 147
 IBM BASIC 182
 Interactive Problem Control System (IPCS) 45
 Interactive Productivity Facility (IPF) 198
 OS/VS COBOL 166
 Pass-Through Facility 126
 PC/VM Bond 129
 PL/I 169
 Professional Office System (PROFS) 136
 Remote Spooling Communications Subsystem
 (RSCS) Networking 124
 service provided 5
 SQL/Data System 158
 SQL/Data System with QMF 155
 Systems Application Architecture 163
 VM/SP High Performance Option (HPO) 202

licensed program (*continued*)
 VS COBOL II 167
 VS FORTRAN 177
 VS FORTRAN Interactive Debug 178
 VS Pascal 172
LINK command, example of use 64
 LINK batch job statement 52
 LINK control statement 63
LISTING file 38, 39, 43
listing file, defined 37
LOAD command
 defined 49
 example of use 43
LOADBUF command 25
local and global resources 117
Logging on to VM/SP 13
LRECL attribute, example of use 92

M

macro instruction, definition and example 175
magnetic tape device 11
main storage 7
 See also real storage
manual, VM/SP
 See abstracts of VM/SP manuals
mapping virtual to real storage 7
maximum data items on minidisk 92
maximum storage size 6, 15
MDISK control statement 63
memos 135
menus for interactive environment 161
Message System Service 67
messages
 sending to system operator 34
 system 28
message, definition of 66
MIGRATE command, example of use 17
migration process 55
minidisks
 access to 18
 concept 8
 description 8, 11, 18
 how defined 18
 initializing 20
 maximum data items 92
 sharing 9
MODULE file 48
MONITOR command, defined 74
multiple access, defined 2
multiple shadow table option, SET command 75

N

NAMES command, example of use 35
National Language Support (NLS)
 See also NLS (National Language Support)
 CMS 84

National Language Support (NLS) (*continued*)
 general 13, 215
 language support 145
 service 221
NetView 122
NetView Distribution Manager (NetView DM) 188,
 189
 description 195
 SNA 195
NetView Message Queueing Service 192
NetView PROP command 193
network component 189
network, DDP defined 185
NLS (National Language Support)
 See also National Language Support (NLS)
 CMS 84
 general 13, 215
 language support 145
 service 221

O

object program, defined 37
office task program product
 DCF 139
 GDDM 147
 PROFS 137
operating system
 executing 55
 guest under VM/SP 11
 shared DASD 11
optimizing compiler, PL/I 169
option, performance 74
ORDER command
 batch machine output 53
 example of use 24, 29
OS/VS COBOL 166
 CSL 168
output device 69

P

page
 frame 16
 migration 17
 of storage 16
page fault, definition 17
paging
 description 17
 overview 16
 preferred paging area 17
panel
 HELP 104
 interactive 161
Pass-Through Facility (PVM) 113, 126
PASSTHRU command 126
password, minidisk access 71

PCF (Problem Control Facility) 199
 PC/VM Bond 129
 PEEK command, example of use 23
 PER command, description 44
 performance option 74
 personal minidisk 19
 PF (program function) key 89
 planning for VM/SP
 device configuration 69
 preparing system files 214
 storage requirement 215
 planning model 134
 PLIOPT command, defined 49
 PL/I
 CSL 171
 debugging aid 169
 description 169
 optimizing compiler example 39
 procedures or subroutines 170
 power failure, data protection 72
 POWERINP description 88
 preferred DASD area 17
 preferred paging 17
 PRELOAD EXEC 220
 PREPARE facility of SQL/DS 160
 preparing system files 214
 preprocessor for SQL/DS 158
 preventive service 219
 preventive service tapes, description 219
 primary user 61
 printers
 spooled virtual 22
 virtual 34
 priority
 operator's virtual machine 28
 spooled file 22
 virtual machine 14, 25
 private resources 117
 privilege classes 60
 privileged command 32
 Problem Control Facility (PCF) 199
 problem diagnosis 77
 problem reporting 45
 procedures, description 170
 procedure, automated 82
 processing batch jobs
 under CMS 52
 VM Batch Subsystem 53
 processing unit 9
 processor
 command language 86
 sharing usage 9
 time allocation 14
 Professional Office System (PROFS) 136
 PROFILE EXEC 80
 PROFILE GCS 72
 PROFS (Professional Office System) 136
 program function (PF) key 89
 programmable operator facility 192
 how it works 192
 NetView PROP command 193
 routing table 192
 tasks performed 192
 programming problem diagnosis 77
 programs
 development aid, COBOL 166, 168
 object 37
 product 133, 165
 source 37
 stack 106
 update tape (PUT) 219
 program, licensed
 See licensed program
 project management 134
 proofreading documents using PROFS 136
 publications for VM/SP
 See abstracts of VM/SP manuals
 PURGE command
 batch machine output 53
 example of use 24, 35
 PUT (program update tape) 219
 PVM dump 45
 PVM (VM/Pass-Through Facility) 113, 126

Q
 QBE (query by example) 156
 QMF (Query Management Facility) 155
 query by example (QBE) 156
 QUERY command
 defined 12
 example of use 29, 34, 63, 81
 Query Management Facility (QMF) 155
 queue drop elimination option of SET command 76

R
 RDRLIST command, example of use 23, 35
 reader, virtual
 defined 34
 handling input files 34
 readshare password 72
 read/write authority 71
 real resource
 managed by CP 2
 real console 9
 real storage 7
 sharing 2
 real storage
 allocation 14
 mapping to virtual 7
 security 72
 RECEIVE command, example of use 23, 35, 81
 RECFM attribute, example of use 92

record, data 91
 record, variable-length 91
 Remote Spooling Communications Subsystem (RSCS)
 Networking 57, 124
 remote system, DDP defined 185
 REPEAT command 25
 reporting, problem 45
 request, status information 12
 reserved page option, SET command 75
 resource management, real 13, 60
 Restructured Extended Executor (REXX) language 87
 review, definition 151
 REXX (Restructured Extended Executor) language 87
 routing table for programmable operator 192
 RSCS dump 45
 RSCS (Remote Spooling Communications Subsystem)
 Networking 57, 124
 RUN command 50
 Running programs under CMS 48

S

SAMGEN EXEC 220
 SAVE subcommand of XEDIT, example of use 89
 screen, DMS/CMS 161
 SCRIPT command 139
 control word 139
 SCRIPT/VS (DCF) 139
 secondary user 61
 security
 data 71
 real storage 72
 SEGGEN EXEC 220
 segment 16
 SENDFILE command, example of use 23, 35
 sending and receiving documents 136
 sending messages to system operator 34
 server
 description 95
 servicing VM/SP
 corrective fixes 219
 local service 219
 program update tapes (PUT) 219
 types of services 218
 session manager, APL2 180
 SET command
 example of use 28, 33, 64, 86
 extended control-program support
 option:VM/370 76
 favored execution option 75
 multiple shadow table option 75
 queue drop elimination option 76
 reserved page option 75
 user priority option 75
 virtual machine assist option 76
 SET SMSG command 33
 SET subcommand of XEDIT, definition 89

SFS (CMS Shared File System)
 See also CMS Shared File System (SFS)
 accessing directories 97
 creating aliases 97
 description 95
 locking files and directories 99
 PUBLIC key word 100
 sharing files or directories 94, 98
 shared DASD 11
 shared minidisk 20
 shared segment 72
 sharing data files 12, 94
 sharing system resources 6
 data integrity 71
 direct access storage 11
 minidisk 9, 18
 processor usage 9, 14
 real storage segment 72
 through spooling 21
 user CMS file 71
 SHUTDOWN command, example of use 29, 32
 Signal System Service 67
 single console image facility 61
 single processor mode
 option, SPMODE command 76
 slicing, time 9
 SNA (Systems Network Architecture)
 See also Systems Network Architecture (SNA)
 benefits 110
 defined 110
 introductory description 3
 network components 189, 190
 node 124
 SolutionPac 138
 SORT subcommand of XEDIT, example of use 89
 source module, description 176
 source program, defined 37
 SPACE command 25
 SPMODE command, example of use 76
 SPOOL
 batch job statement 52
 batch machine output 53
 command 22, 34
 command, example of use 22, 23, 25
 control statement 63
 SPOOL CONSOLE command 23
 spool operator task 24
 spooled virtual printer 22
 spooling
 CP management 15, 24
 description 21
 example 21
 overview 21
 priority 22
 resulting from I/O requests 23
 summary of CP activity 25
 system operator task 24
 user control 24

- spooling (*continued*)
 - virtual console data 25
 - virtual punch 22
 - virtual reader 22
- SPTAPE command 25
- SQL/DS (Structured Query Language/Data System)
 - See also* Structured Query Language/Data System (SQL/DS)
 - database 153
 - EXECUTE facility 160
 - general 152
 - PREPARE facility 160
 - preprocessor 158
 - with QMF 155
- STACK subcommand of XEDIT, example of use 89
- stack, console 106
- stack, program 106
- stand-alone dump 45
- START command
 - defined 49
 - example of use 43
 - operator use 24
- starter system 216
- statistics 134
- storage
 - address translation 16
 - allocation 14
 - how CP manages 14
 - how organized 16
 - maximum size 6
 - planning for requirements 215
 - virtual address 16
 - virtual storage 6
 - virtual versus real 7
- STORE command 44
- Structured Query Language/Data System (SQL/DS)
 - See also* SQL/DS (Structured Query Language/Data System)
 - database 153
 - EXECUTE facility 160
 - general 152
 - PREPARE facility 160
 - preprocessor 158
 - with QMF 155
- subcommand, CMS 82
- subroutines, description 170
- summary of VM/SP concepts 30
- SVCTRACE command described 44
- symbol editor for GDDM 147
- system directory
 - See* directory, system
- system directory, defined 214
- system file, preparing 214
- system generation option
 - AFFINITY 75
 - virtual=real 75
- system message 28

- system operator
 - batch virtual machine 52
 - communication with VM/SP 28
 - console usage 9
 - controlling performance options 74
 - dedicating devices 11
 - privileged command 32
 - sending messages to 34
 - use of CMS commands 85
 - view of spooling 24
 - VM/IPF panel 199
- System Product Editor
 - description 88
 - using for program development 37
- System Product Interpreter 87
- Systems Application Architecture 163
- Systems Application Architecture Common Programming Interface (CPI) Communications 110
- Systems Network Architecture (SNA)
 - See also* SNA (Systems Network Architecture)
 - benefits 110
 - defined 110
 - introductory description 3
 - network components 189, 190
 - node 124

T

- tag, DCF 140
- tape device, magnetic 11
- task
 - CMS user 3, 84
 - spool operator 24
- TCP/IP 110, 113
- TERMINAL command, example of use 64
- terminal input buffer 106
- terminals 9
- TEXT files 38, 39, 48
- text file, defined 37
- text preparation program product
 - DCF 139
 - PC/VM Bond 129
 - PROFS 136
- text processing 135
- time allocation, processor 14
- time slicing 9, 14
- top directory 19, 95
- TRACE command described 44
- TRANSFER command, example of use 24
- Transparent Services Access Facility (TSAF) 4
 - description 115
 - Shared File System 96
 - support 113
- TSAF (Transparent Services Access Facility) 4
 - description 115
 - Shared File System 96
 - support 113

TXTLIB (TEXT file library) 48

U

unit record device

- defined 8
- in a virtual machine 11
- virtual device 8

update

program update tapes (PUTs) 219

update procedures

- applying local updates 219
- ASMGEND 220
- CMSGEND 220
- CSLGEN 220
- DCSSGEN 220
- deciding which procedure to use 219
- DOSGEN 220
- EXECUPDT 220
- GENSERVE 220
- GENTSAF 220
- PRELOAD 220
- SAMGEN 220
- SEGEN 220
- UTILITY 220
- VMFAPPLY 220
- VMFASM 220
- VMFBLD 220
- VMFHASM 220
- VMFLKED 220
- VMFMAC 221
- VMFNLS 221
- VMFOVER 221
- VMFTXT 221
- VRSIZE 221
- VSAMGEN 221
- VSEVSAM 221
- ZAP 221
- ZAPTEXT 221

update, definition 151

user class 60

USER control statement 63

user priority option for SET command 75

user task, CMS 3, 84

user view, CMS 84

User-written programs 57

user, general

See general user

UTILITY EXEC 220

V

variable-length records 91

VARY command, example of use 32

vector symbol editor (GDDM) 147

virtual address 16

virtual concept 6-9

- device 8
- console 9

virtual concept (*continued*)

device (*continued*)

- direct access storage 8
- magnetic tape 8
- unit record 8

storage 6

compared to real 7

need for 6

size limitation 6

summary 30

virtual console data, spooling 25

virtual console management 61

virtual disks 127

virtual file 127

virtual machine assist

option of SET command 76

virtual machines

batch machine 52

changing the definition 13

console 9, 61

defining in directory 13

description 10

display status 32

group 116

I/O 34

make changes 32

operating system 11

priority 14

requirement 10

resource available to 10

share of processor time 14

status information request 12

use of real devices 15

virtual I/O device 11

working in virtual environment 32

Virtual Machine/Interactive Productivity Facility (VM/IPF) 198

Virtual Machine/Personal Computer 131

Virtual Machine/System Product (VM/SP)

defined 2

recommended updating procedures 219

virtual print 127

virtual printer 22, 34

virtual punch

description 34

spooling 22

usage 34

virtual reader

handling input files 34

spooling 22

usage 34

virtual storage

mapping to real 7

virtual storage management 15

Virtual Telecommunications Access Method (VTAM) 110, 113, 116, 122

virtual = real system generation option 75

VM Batch Subsystem 53
 VMBACKUP Management Subsystem
 (VMBACKUP-MS) 189
 VMDUMP 45
 VMFAPPLY EXEC 220
 VMFASM EXEC 220
 VMFBLD EXEC 220
 VMFHASM EXEC 220
 VMFLKED EXEC 220
 VMFLOAD EXEC 221
 VMFMAC EXEC 221
 VMFMERGE 221
 VMFNLS EXEC 221
 VMFOVER EXEC 221
 VMFREC EXEC 221
 VMFREMOV 221
 VMFSETUP EXEC 221
 VMFTXT EXEC 221
 VMFZAP 221
 VMLIB 50
 OS/VS COBOL 168
 PL/I 171
 VS COBOL II 168
 VS FORTRAN 178
 VS Pascal 173
 VM/Distributed Systems Node Executive (DSNX) 188,
 189
 description 195
 example application 195
 samples 194
 VM/Extended Architecture System Product (VM/XA
 SP)
 description 206
 programming interface 38
 VM/Integrated System 207
 VM/Interactive Productivity Facility (VM/IPF) 198
 VM/IPF (VM/Interactive Productivity Facility) 198
 VM/IS (Integrated System) 207
 VM/Pass-Through 57
 VM/PC 131
 VM/SP concepts 30
 VM/SP High Performance Option (HPO)
 See VM/SP HPO (High Performance Option)
 VM/SP HPO (High Performance Option)
 See also High Performance Option, VM/SP
 free storage management improvements 202
 page migration enhancement 202
 single processor mode 76
 VM/SP manuals
 See abstracts of VM/SP manuals
 VM/SP (Virtual Machine/System Product)
 defined 2
 recommended updating procedures 219
 VRSIZE EXEC 221
 VS COBOL II 167
 CSL 168
 VS FORTRAN 177
 CSL 178
 VS FORTRAN Interactive Debug 178
 VS Pascal 172
 CSL 173
 VSAMGEN EXEC 221
 VSEVSAM EXEC 221
 VTAM (Virtual Telecommunications Access
 Method) 110, 113, 116, 122

W

writeshare password 72

X

XEDIT command, example of use 81, 88
 FILE subcommand 89
 POWERINP subcommand 88
 SAVE subcommand 89
 SET subcommand 89
 SORT subcommand 89
 STACK subcommand 89

Z

ZAP EXEC 221
 ZAPTEXT EXEC 221

VM/SP
Introduction
Order No. GC19-6200-05

**READER'S
COMMENT
FORM**

Is there anything you especially like or dislike about this book? Feel free to comment on specific errors or omissions, accuracy, organization, or completeness of this book.

If you use this form to comment on the online HELP facility, please copy the top line of the HELP screen.

_____ Help Information line ____ of ____

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you, and all such information will be considered nonconfidential.

Note: Do not use this form to report system problems or to request copies of publications. Instead, contact your IBM representative or the IBM branch office serving you.

Would you like a reply? YES NO

Please print your name, company name, and address:

IBM Branch Office serving you:

Thank you for your cooperation. You can either mail this form directly to us or give this form to an IBM representative who will forward it to us.

Reader's Comment Form

CUT
OR
FOLD
ALONG
LINE

Fold and tape

Please Do Not Staple

Fold and tape



BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



INTERNATIONAL BUSINESS MACHINES CORPORATION
DEPARTMENT G60
PO BOX 6
ENDICOTT NY 13760-9987

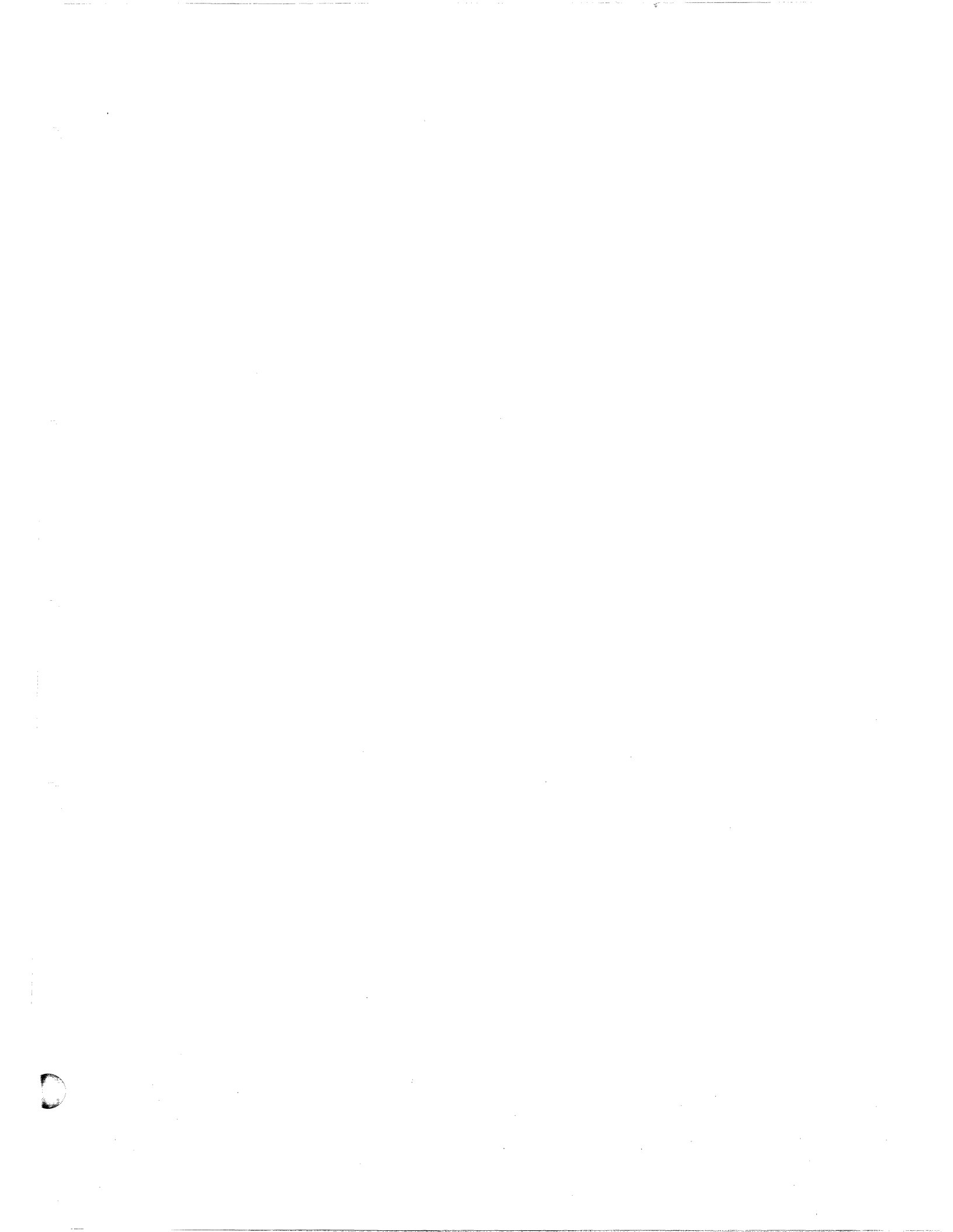


Fold and tape

Please Do Not Staple

Fold and tape







Program Number
5664-167

File Number
S370/4300-20



Program Number
5664-167

File Number
S370/4300-20

