Advanced Operating System/Virtual Storage (AOS/VS) Programmer's Manual

Volume 1

System Concepts

093-000335-00

For the latest enhancements, cautions, documentation changes, and other information on this product, please see the Release Notice (085-series) supplied with the software.

Ordering No. 093-000335

Revision 00, March 1983

NOTICE

Advanced Operating
System/Virtual Storage
(AOS/VS)
Programmer's Manual
Volume 1
System Concepts
093-000335

Revision History:	Effective with:
Original Release - March 1983	AOS/VS Rev. 2.00

A vertical bar in the margin of a page indicates substantive change from the previous revision. Two volumes supersede the 'AOS/VS Programmer's Manual,' 093-000241-00:

- o Volume 1, 093-000335-00 (System Concepts); and
- o Volume 2, 093-000241-01 (System Calls).

PREFACE

This manual supersedes the 'Advanced Operating System/Virtual Storage (AOS/VS) Programmer's Manual' (093-000241-00). It is intended for use by experienced assembly language programmers.

In this revision, we describe new features and enhancements to existing features of the Advanced Operating System/Virtual Storage (AOS/VS) software for Release 2.00.

This manual is divided into two volumes:

o Volume 1 contains explanations of basic AOS/VS concepts and how families of system calls work together.

o Volume 2 contains the AOS/VS system calls. The system calls are arranged alphabetically for your convenience.

If you are not experienced with assembly language, we suggest that you read the following manuals before you read this book:

o 'Fundamentals of Small Computer Programming' (093-000090), a general introduction to Data General computers.

o 'Advanced Operating System/Virtual Storage (AOS/VS) Macroassembler (MASM) Reference Manual' (093-000242), which gives detailed information about the syntax of AOS/VS assembly language and about the Macroassembler utility.

The information in Volume 1, System Concepts, is divided into the following chapters:

Chapter 1 --	Introduces AOS/VS.
Chapter 2 --	Describes virtual memory concepts and memory management.
Chapter 3 --	Describes processes and how to manage them.
Chapter 4 --	Describes files and how to create them.
Chapter 5 --	Describes input/output (I/O) concepts and file I/O.
Chapter 6 --	Describes the interprocess communications (IPC) facility and how to use it.
Chapter 7 --	Describes tasks and how to manage them in a multitasking environment.

Chapter 8 --	Describes the connection-management facility and how to use it.
Chapter 9 --	Describes binary synchronous communications (BSC).
Chapter 10 --	Describes how to define user devices under AOS/VS.
Chapter 11 --	Describes the functions of various miscellaneous system calls.
Chapter 12 --	Describes the system calls that are exclusively for 16-bit users.

Certain features of AOS/VS may change from revision to revision. Therefore, please refer to the current AOS/VS Release Notice for the most up-to-date information about functional changes and enhancements. The Release Notice is in the utilities directory (:UTIL) on your system tape.

Reader Please Note:

We use the following conventions in this manual:

Symbol	Meaning
< >	Angle brackets indicate the paraphrase of an argument or statement. (You supply the actual argument or statement.)
*	One asterisk indicates multiplications. For example, 2*3 means 2 multiplied by 3.
**	Two asterisks indicate exponentiation. For example, 2**3 means 2 raised to the third power.

Unless the text supplies a specific radix (as it often does), all memory addresses are octal value and all other numbers are decimal values. To specify a radix, we use the notation "octal value" to indicate that the radix is octal.

When we refer to magnetic tape, we means 9-track magnetic tape.

Contacting Data General

If you have comments on this manual, please use the prepaid Remarks Form that appears after the Index. If you require additional manuals, please use the enclosed TIPS order form (USA only) or contact your local Data General sales representative. If you experience software problems, please notify Data General Systems Engineering.

End of Preface

TABLE OF CONTENTS

Page

CHAPTER 1 INTRODUCTION TO AOS/VS....1-1

Virtual Memory....1-2
Ring Structure....1-3
AOS Compatibility....1-4
Inner-Ring Management Terms....1-5
System Calls....1-6

CHAPTER 2 MEMORY....2-1

Ring Structure....2-2
Demand Paging....2-4
Shared and Unshared Memory Pages....2-4
Protected Shared Files....2-7
Coordinated Shared-File Update....2-8
Dedicated and Undedicated Memory Pages....2-9
User Context....2-9

CHAPTER 3 PROCESSES....3-1

Memory Scheme....3-3
Process Types....3-5
Priority Numbers....3-6
Process Identification....3-6
Process Creation....3-8

Process Privileges....3-9
Process Creation Parameters....3-11
Superuser Mode/Superprocess Mode....3-12

Process States....3-13
Process Scheduling....3-14

TABLE OF CONTENTS (Cont.)

Page

CHAPTER 3 PROCESSES (Cont.)

Process Blocking....................................3-14
Changing Process Priorities.........................3-15
Process Information.................................3-15
Execute-Protection Status...........................3-16
Process Traps.......................................3-17
Break Files and Memory Dumps........................3-17
Linking Programs Together with the ?CHAIN System Call.................3-19
Inner Rings...3-20
Process and Memory Sample Programs..................3-23

SON Subroutine..................................3-23
RUNTIME Program.................................3-27
RINGLOAD Program................................3-35
INRING Program..................................3-39
GATE.ARRAY Program..............................3-43

CHAPTER 4 FILE CREATION AND MANAGEMENT........................4-1

Disk File Structures................................4-2
Directory Creation..................................4-4
Directory Entries...................................4-4
File Types..4-5
Directory Access....................................4-7
Filenames...4-8
Pathnames...4-9
Link Entries..4-12

Use of ?CREATE and ?DELETE System Calls on Link Entries..................4-13
File Access...4-13
Access Control Lists................................4-15

ACL Templates...................................4-16
The Permanent Attribute.........................4-17

Logical Disks.......................................4-17
Disk Space Control..................................4-19
File Creation and Management Sample Programs........4-21

FILCREA Program.................................4-21

TABLE OF CONTENTS (Cont.)

Page

CHAPTER 5 FILE INPUT/OUTPUT (I/O)....5-1

File I/O Concepts....5-3

Blocks and Records....5-3
Channels....5-3
File I/O Operation Sequence....5-4
File Pointer....5-5

Block I/O....5-6
Physical Block I/O....5-7
Record I/O....5-8
Device Names....5-9
Generic Filenames....5-9
Multiprocessor Communications Adapters....5-12
Character Devices....5-13

Full-Duplex Modems....5-15
Card Readers....5-17
Character Device Assignment....5-17

Line-Printer Format Control....5-19
Console Format Control....5-19
The IPC Facility as a Communications Device....5-21
Labeled Magnetic Tapes....5-22

Formats....5-23
Label Types....5-24

File I/O on Labeled Magnetic Tapes....5-35
File I/O on Unlabeled Magnetic Tapes....5-37
File I/O Sample Programs....5-39

RITE Program....5-39
DLIST Program....5-45

CHAPTER 6 TASKS....6-1

Task Concepts....6-3
Task-Protection Schemes....6-4
Task Identifiers and Priority Numbers....6-5
Task Initiation....6-6

TABLE OF CONTENTS (Cont.)

Page

CHAPTER 6 TASKS (Cont.)

Stack Space Allocation and Stack Definition.........6-7
Inner-Ring Stacks..................................6-8
Task Scheduling....................................6-10
Task Suspension....................................6-11
Task Readying......................................6-12
Task Redirection...................................6-13
Inner-Ring Task Redirection Protection.............6-13
Task Termination...................................6-15
Task Creation and Termination Detection............6-16
Console-to-Task Communication......................6-17
Task-to-Task Communication.........................6-17
Critical Region Locking/Unlocking..................6-18
MV/8000 Floating-Point Registers...................6-19
Multitasking Sample Programs.......................6-21

NEWTSK Program.................................6-21
BOOMER Program.................................6-25

CHAPTER 7 THE INTERPROCESS COMMUNICATIONS (IPC) FACILITY......7-1

Sending Messages Between IPC Ports.................7-2
Typical IPC System Call Sequence...................7-4
Send and Receive Headers...........................7-5

System and User Flags..........................7-7
User Flag Word.................................7-9

Process Termination Messages in a Customer/Server Relationship............7-9

Termination Messages for 16-Bit Processes.......7-10
Termination Messages for 32-Bit Processes.......7-14

?ISEND and ?IREC System Call Logic.................7-14
IPC Sample Programs................................7-19

HEAR Program...................................7-19
SPEAK Program..................................7-25

TABLE OF CONTENTS (Cont.)

Page

CHAPTER 8 CONNECTION MANAGEMENT....................8-1

Connection Creation....................8-2

Server Process....................8-4

Connection Termination....................8-5
Obituary Messages....................8-5
Inner-Ring Connection Management....................8-6
Fast Interprocess Synchronization....................8-8

CHAPTER 9 BINARY SYNCHRONOUS COMMUNICATIONS (BSC)....................9-1

BSC Concepts....................9-2
Line Configurations....................9-3

Multipoint Line Selection and Polling....................9-4
Relative Consoles....................9-6

BSC Protocol....................9-6
BSC Error-Recovery Procedures....................9-11
BSC Implementation....................9-13

CHAPTER 10 USER DEVICE SUPPORT....................10-1

?IDEF System Call Options....................10-3
User Interrupt Service....................10-7
User Stacks....................10-8
Communicating from an Interrupt Service Routine....................10-9
Enabling and Disabling Access to All Devices....................10-9
LEF Mode....................10-10
Power-Failure/Auto-restart Routine....................10-11

CHAPTER 11 MISCELLANEOUS SYSTEM CALL FUNCTIONS....................11-1

Console Interrupts....................11-2
Clock/Calendar System Calls....................11-3

TABLE OF CONTENTS (Cont.)

Page

CHAPTER 11 MISCELLANEOUS SYSTEM CALLS FUNCTIONS (Cont.)

Error Message File....................................11-4
Program Information/Control System Calls...........11-5
System Information...................................11-5
Utility Interfaces...................................11-6
Bias Factors...11-7
CLI Messages...11-7
Symbols..11-7
Host Information.....................................11-7
Address/Access Validation............................11-8
TIMEOUT Sample Program...............................11-9

CHAPTER 12 16-BIT PROCESSES..................................12-1

Memory Modification with Disk Images................12-2
Overlay Concepts.....................................12-2
Resource System Calls................................12-4
Procedures Entries...................................12-6
Alternate Return from Resources.....................12-7
System Management of Resource System Calls.........12-7
Runtime Relocatability Requirements.................12-8
Primitive Overlay System Calls......................12-10
Extended State Save Area............................12-11

CHAPTER 13 SYSTEM CALLS......................................13-1

?ALLOCATE..13-7
?ASSIGN..13-9
?BLKPR...13-11
?BNAME...13-13
?BRKFL...13-15

?CDAY..13-19
?CGNAM...13-21
?CHAIN...13-23
?CKVOL...13-27
?CLOSE...13-29

?CON...13-33
?CPMAX...13-37

TABLE OF CONTENTS (Cont.)

Page

CHAPTER 13 SYSTEM CALLS (Cont.)

?CREATE..13-39
?CRUDA...13-51
?CTERM...13-53
?CTOD..13-55
?CTYPE...13-57

?DACL..13-59
?DADID...13-61
?DCON..13-63
?DDIS..13-65
?DEASSIGN......................................13-67

?DEBL..13-71
?DELAY...13-73
?DELETE..13-75
?DFRSCH..13-77
?DIR...13-79

?DQTSK...13-81
?DRCON...13-85
?DRSCH...13-87
?ENBRK...13-89
?ENQUE...13-93

?ERMSG...13-97
?ERSCH...13-101
?ESFF..13-103
?EXEC..13-105
?EXPO..13-123

?FDAY..13-127
?FEDFUNC.......................................13-129
?FEOV..13-133
?FLUSH...13-135
?FSTAT...13-137

?FTOD..13-147
?GACL..13-149
?GBIAS...13-151
?GCHR..13-153
?GCLOSE..13-159

Page

CHAPTER 13 SYSTEM CALLS (Cont.)

?GCPN....13-161
?GCRB....13-163
?GDAY....13-165
?GECHR....13-167
?GHRZ....13-171

?GLINK....13-173
?GLIST....13-175
?GMEM....13-177
?GNAME....13-179
?GNFN....13-181

?GOPEN....13-185
?GPORT....13-191
?GPOS....13-193
?GPRNM....13-197
?GRNAME....13-199

?GSHPT....13-201
?GSID....13-203
?GTACP....13-205
?GTMES....13-207
?GTNAM....13-215

?GTOD....13-217
?GTRUNCATE....13-219
?GTSVL....13-223
?GUNM....13-225
?GVPID....13-227

?HNAME....13-229
?IDEF....13-233
?IDGOTO....13-241
?IDKIL....13-243
?IDPRI....13-245

?IDRDY....13-247
?IDSTAT....13-249
?IDSUS....13-251
?IESS....13-253
?IFPU....13-255

TABLE OF CONTENTS (Cont.)

Page

CHAPTER 13 SYSTEM CALLS (Cont.)

?IHIST....................................13-257
?ILKUP....................................13-261
?IMERGE...................................13-263
?IMSG.....................................13-265
?INIT.....................................13-267

?INTWT....................................13-271
?IQTSK....................................13-273
?IREC.....................................13-275
?IRMV.....................................13-281
?ISEND....................................13-283

?ISPLIT...................................13-287
?IS.R.....................................13-289
?ITIME....................................13-295
?IXIT.....................................13-297
?IXMT.....................................13-299

?KCALL....................................13-301
?KHIST....................................13-303
?KILAD....................................13-305
?KILL.....................................13-307
?KINTR....................................13-309

?KIOFF....................................13-311
?KION.....................................13-313
?KWAIT....................................13-315
?LABEL....................................13-317
?LEFD.....................................13-321

?LEFE.....................................13-323
?LEFS.....................................13-325
?LMAP.....................................13-327
?LOGCALLS.................................13-329
?LOGEV....................................13-333

?MBFC.....................................13-337
?MBTC.....................................13-341
?MDUMP....................................13-345
?MEM......................................13-347
?MEMI.....................................13-349

TABLE OF CONTENTS (Cont.)

Page

CHAPTER 13 SYSTEM CALLS (Cont.)

?MYTID....................13-351
?ODIS....................13-353
?OEBL....................13-355
?OPEN....................13-357
?OVEX....................13-377

?OVKIL....................13-379
?OVLOD....................13-381
?OVREL....................13-385
?PCNX....................13-387
?PMTPF....................13-389

?PNAME....................13-393
?PRCNX....................13-395
?PRDB/?PWRB....................13-397
?PRI....................13-403
?PRIPR....................13-405

?PRKIL....................13-409
?PROC....................13-411
?PRRDY....................13-425
?PRSUS....................13-427
?PSTAT....................13-429

?RCALL....................13-441
?RCHAIN....................13-443
?RDB/?WRB....................13-445
?RDUDA/?WRUDA....................13-453
?READ/?WRITE....................13-455

?REC....................13-473
?RECNW....................13-475
?RECREATE....................13-477
?RELEASE....................13-479
?RENAME....................13-481

?RESCHED....................13-483
?RESIGN....................13-485
?RETURN....................13-487
?RINGLD....................13-491
?RNAME....................13-493

TABLE OF CONTENTS (Cont.)

Page

CHAPTER 13 SYSTEM CALLS (Cont.)

?RNGPR.....13-495
?RNGST.....13-499
?RPAGE.....13-501
?RUNTM.....13-503
?SACL.....13-507

?SATR.....13-509
?SBIAS.....13-511
?SCHR.....13-513
?SCLOSE.....13-519
?SDAY.....13-521

?SDBL.....13-523
?SDLM.....13-525
?SDPOL.....13-529
?SDRT/?SERT.....13-533
?SEBL.....13-535

?SECHR.....13-545
?SEND.....13-549
?SERMSG.....13-553
?SERVE.....13-555
?SGES.....13-557

?SIGNL.....13-561
?SIGWT.....13-563
?SINFO.....13-565
?SLIST.....13-569
?SOPEN.....13-571

?SOPPF.....13-573
?SPAGE.....13-577
?SPOS.....13-581
?SRCV.....13-585
?SSHPT.....13-597

?SSID.....13-599
?SSND.....13-601
?STMAP.....13-611
?STOM.....13-617
?SUPROC.....13-619

TABLE OF CONTENTS (Cont.)

Page

CHAPTER 13 SYSTEM CALLS (Cont.)
?SUS..13-621
?SUSER......................................13-623
?SYLOG......................................13-625
?TASK.......................................13-627
?TERM.......................................13-637

?TIDSTAT....................................13-641
?TLOCK......................................13-643
?TPID.......................................13-647
?TPORT......................................13-649
?TRCON......................................13-651

?TRUNCATE...................................13-655
?TUNLOCK....................................13-657
?UBLPR......................................13-659
?UIDSTAT....................................13-661
?UNWIND.....................................13-663

?UNWIRE.....................................13-665
?UPDATE.....................................13-667
?VALAD......................................13-669
?VCUST......................................13-671
?VRCUST.....................................13-673

?WALKBACK...................................13-675
?WDELAY.....................................13-677
?WHIST......................................13-679
?WIRE.......................................13-683
?WTSIG......................................13-691

?XMT..13-693
?XMTW.......................................13-695

APPENDIX A ERROR CODES.......................A-1

LIST OF ILLUSTRATIONS

Figure		Page
2-1	Segments and Their Protection Rings	2-3
2-2	Working Sets in Memory	2-5
2-3	Memory Context	2-10
3-1	Working Sets in Memory	3-4
3-2	Process Names	3-7
3-3	Process Hierarchy	3-9
3-4	Sample Process Tree	3-12
3-5	Ring Structure	3-21
4-1	File Growth Stages	4-3
4-2	Sample Directory Tree	4-5
4-3	Directory Structure	4-11
4-4	Initializing a Logical Disk	4-18
4-5	Control Point Directories (CPDs)	4-20
5-1	Labels and Data on a Labeled Magnetic Tape	5-26
6-1	Task States	6-11
7-1	Structure of IPC Send and Receive Headers	7-6
7-2	Structure of Offset ?UIFL	7-10
7-3	?ISEND Logic Flowchart	7-17
7-4	?IREC Logic Flowchart	7-18

LIST OF ILLUSTRATIONS (Cont.)

Figure		Page
8-1	Model Customer/Server Configuration	8-2
8-2	Multilevel Customer/Server Configuration	8-3
8-3	Double Connection	8-3
9-1	Point-to-Point/Multipoint Line Configurations	9-4
9-2	?SSND System Call, Initial, Point-to-Point	9-14
9-3	?SSND System Call, Continue, Point-to-Point	9-15
9-4	?SRCV System Call, Initial and Continue, Point-to-Point	9-16
9-5	?SSND System Call, Multipoint Control Station	9-17
9-6	?SRCV System Call, Multipoint Control Station	9-18
9-7	?SRCV System Call, Multipoint Tributary Station	9-19
9-8	?SEBL System Call, Point-to-Point	9-20
10-1	Device Control Table (DCT)	10-2
10-2	Structure of Map Definition Table	10-5
12-1	Basic Overlay Area Equals Size of Largest Overlay	12-3
12-2	Multiple Overlay Area (total area = basic size * 2)	12-4
12-3	Passing a Procedure Entry Descriptor via the Stack	12-6
12-4	Resource System Call Stack after ?RSAVE System Call	12-8
12-5	Invalid Return Address from ?RCALL System Call	12-9
13-1	Parametric Coding Example	13-4
13-2	Structure of ?CLOSE Packet	13-30
13-3	Structure of ?CREATE IPC Packet	13-41
13-4	Structure of ?CREATE Time Block	13-43
13-5	Structure of ?CREATE Directory Packet	13-44
13-6	Structure of ?CREATE Packet for Other File Types	13-46

LIST OF ILLUSTRATIONS (Cont.)

Figure		Page
13-7	Extended Task Definition Packet	13-82
13-8	Structure of ?ENBRK Packet	13-91
13-9	Error Code Structure in ERMES File	13-98
13-10	Structure of ?EXEC Packet for Unlabeled Mount Function ?XFMUN	13-107
13-11	Structure of ?EXEC Extended Packet for Unlabeled Mount Function ?XFXUN	13-107
13-12	Structure of ?EXEC Packet for Labeled Mount Function ?XFMLT	13-108
13-13	Structure of ?EXEC Extended Packet for Labeled Mount Function ?XFXML	13-108
13-14	Structure of ?EXEC Packet for Dismounting a Tape, ?XFDUN	13-109
13-15	Structure of ?EXEC Packet for Queue Requests	13-111
13-16	Structure of ?EXEC Packet for Holding, Unholding, or Canceling Queue Requests	13-118
13-17	Structure of ?EXEC Packet for Status Information	13-119
13-18	Structure of ?FEDFUNC Packet to Change Radix	13-129
13-19	Structure of ?FEDFUNC Packet to Open Symbol Table File	13-130
13-20	Structure of ?FEDFUNC Packet to Evaluate FED String	13-130
13-21	Structure of ?FSTAT Unit File Packet	13-139
13-22	Structure of ?FSTAT IPC File Packet	13-140
13-23	Structure of ?FSTAT Directory File Packet	13-141
13-24	Structure of ?FSTAT Other File Types Packet	13-142
13-25	?SSTS Structure	13-143
13-26	Structure of ?GNFN Packet	13-182
13-27	Structure of ?GOPEN Packet for IPC Files	13-186
13-28	Structure of Standard ?GOPEN Packet	13-187
13-29	Standard I/O Packet for ?READ, ?WRITE, ?OPEN, and ?CLOSE	13-194
13-30	Structure of ?GTMES Packet	13-208
13-31	Structure of ?GTRUNCATE Packet	13-220
13-32	Structure of Device Control Table (DCT) for 32-Bit Processes	13-234
13-33	Structure of Device Control Table (DCT) for 16-Bit Processes	13-235
13-34	Structure of Map Definition Table	13-236

LIST OF ILLUSTRATIONS (Cont.)

Figure		Page
13-35	Structure of ?IHIST Packet	13-258
13-36	Structure of ?IREC Header	13-276
13-37	Structure of Offset ?IUFL (User Flag Word)	13-278
13-38	Structure of ?ISEND Header	13-284
13-39	Structure of ?IS.R Header	13-290
13-40	Structure of ?LABEL Packet	13-318
13-41	?LOGEV Event Logging Format	13-334
13-42	Structure of ?MBFC Packet	13-338
13-43	Structure of ?MBTC Packet	13-342
13-44	Structure of ?OPEN Packet	13-359
13-45	Sample Delimiter Table	13-367
13-46	Structure of Labeled Magnetic Tape Packet Extension	13-370
13-47	Structure of ?PMTPF Packet	13-390
13-48	Structure of ?PRDB/?PWRB Packet	13-398
13-49	Structure of ?PROC Packet	13-413
13-50	Structure of ?PSTAT Packet	13-430
13-51	Structure of ?PSTAT Memory Descriptor	13-434
13-52	Structure of ?RDB/?WRB Packet	13-446
13-53	Structure of ?READ/?WRITE Packet	13-456
13-54	Structure of Screen Management Packet Extension	13-463
13-55	Structure of Selected Field Translation Packet Extension	13-467
13-56	Selected Field Translation Packet Sample Listing	13-469
13-57	Structure of ?RNGPR Packet	13-496
13-58	Structure of ?RUNTM Packet	13-504
13-59	Polling List Defined by a Control Station	13-531
13-60	Poll and Select Addresses Defined by a Tributary	13-531
13-61	Structure of ?SEBL Packet	13-536
13-62	Structure of ?SERMSG Packet	13-554
13-63	Structure of ?SGES Packet	13-558
13-64	Structure of ?SINFO Packet	13-566
13-65	Structure of ?SOPPF Packet	13-574
13-66	Structure of ?SPAGE Packet	13-578
13-67	Structure of ?SRCV Packet	13-586
13-68	ITB Receive buffer Format	13-594
13-69	Structure of ?SSND Packet	13-602
13-70	Structure of Standard Task Definition Packet	13-628

LIST OF ILLUSTRATIONS (Cont.)

Figure		Page
13-71	Stack Parameters for Initiating One or More Tasks	13-631
13-72	Extended Task Definition Packet	13-632
13-73	Structure of ?UIDSTAT Packet	13-662
13-74	Structure of ?WHIST Packet	13-680

LIST OF TABLES

Table		Page
2-1	Memory-Management System Calls	2-10
3-1	Process Privileges	3-10
4-1	File Types	4-6
4-2	Filename Conventions	4-9
4-3	Valid Pathname Prefixes	4-10
4-4	File Access Privileges	4-14
4-5	Valid ACL Templates	4-16
5-1	File I/O Operation Sequence	5-4
5-2	File Types You Can Create with the ?OPEN System Call	5-5
5-3	AOS/VS Devices and Device Names	5-10
5-4	Generic Filenames	5-11
5-5	Modem Flags	5-15
5-6	Control Characters and Their Functions	5-20
5-7	Control Sequences and Their Functions	5-21
5-8	Label Formats and Levels; Files per Volume Set, Record Types	5-23
5-9	Types of Labels	5-27
5-10	Contents of VOL1 Volume Labels	5-28
5-11	Contents of User volume Labels (UVLs)	5-29
5-12	Contents of HDR1 File Header Labels	5-30
5-13	Contents of HDR2 File Header Labels	5-32
5-14	Contents of UHL and UTL User Labels	5-34

LIST OF TABLES (Cont.)

Table		Page
7-1	Structure of IPC Send and Receive Headers	7-7
7-2	Contents of System Flag Word (Offset ?ISFL)	7-8
7-3	Process Termination Codes in Offset ?IUFL for ?IREC and ?ISEND Headers	7-11
7-4	Termination Codes for 16-Bit Processes	7-12
7-5	?TRAP Termination Messages for 16-Bit Processes	7-13
7-6	?TEXT Code Termination Messages Sent on 32-Bit Process User Trap	7-15
9-1	BSC Protocol Data-Link Control Characters (DLCC)	9-7
9-2	BSC Error-Recovery Procedures	9-11
10-1	Contents of Map Definition Table Entry	10-6
10-2	LEF Mode and Device Access System Call Functions Summary	10-10
13-1	Valid ?CREATE File Types	13-40
13-2	Contents of ?CREATE IPC Packet	13-42
13-3	Contents of ?CREATE Directory Packet	13-45
13-4	Contents of ?CREATE Packet for Other File Types	13-47
13-5	Contents of ?ENBRK Packet	13-91
13-6	Switches for ?ENQUE Specifications String	13-94
13-7	Flags for EXEC Functions ?XFXUN and ?XFXML	13-109
13-8	Contents of ?EXEC Packet for Queue Requests	13-112
13-9	Contents of ?EXEC Packet for Holding, Unholding, or Canceling Queue Requests	13-118
13-10	Contents of ?EXEC Packet for Status Information	13-120
13-11	Contents of ?FEDFUNC Packet to Evaluate a FED String	13-131
13-12	Flags Returned in Offset ?SSTS	13-143
13-13	Character Device characteristics Words	13-154

LIST OF TABLES (Cont.)

Table		Page
13-14	Commonly Used Device Characteristics	13-156
13-15	Contents of ?GNFN Packet	13-182
13-16	Contents of ?GOPEN Packet for IPC Files	13-186
13-17	Contents of Standard ?GOPEN Packet	13-187
13-18	Positioning the file Pointer	13-195
13-19	Contents of ?GTMES Packet	13-209
13-20	Input Parameters for Offset ?GREQ (Request Types)	13-210
13-21	Output from ?GTMES Requests	13-212
13-22	Contents of ?GTRUNCATE Packet	13-220
13-23	Contents of Map Definition Table	13-237
13-24	Structure of ?IHIST Array	13-259
13-25	Contents of ?INIT Packet	13-268
13-26	Contents of ?IREC Header	13-277
13-27	Contents of ?ISEND Header	13-284
13-28	Contents of ?IS.R Header	13-291
13-29	Contents of ?LABEL Packet	13-319
13-30	Contents of ?OPEN Packet	13-360
13-31	File Creation Options for Offset ?ISTI	13-364
13-32	File Types You Can Create with ?OPEN	13-365
13-33	Contents of Labeled Magnetic Tape Packet Extension	13-371
13-34	Contents of ?PMTPF Packet	13-391
13-35	Contents of ?PRDB/?PWRB Packet	13-399
13-36	?PRDB/?PWRB Packet: Controller Status Words	13-400
13-37	Error Reports Returned in ?PRDB/?PWRB Offsets	13-401
13-38	Contents of ?PROC Packet	13-414
13-39	Privilege bits in Offset ?PPRV	13-419
13-40	Contents of ?PSTAT Packet	13-431
13-41	Contents of ?RDB/?WRB Packet	13-447
13-42	Contents of ?READ/?WRITE Packet	13-458
13-43	Contents of Screen-Management Packet Extension	13-464
13-44	Contents of Selected Field Translation Packet Extension	13-468
13-45	Contents of ?RNGPR Packet	13-496
13-46	Character Device Characteristics Words	13-515
13-47	Contents of ?SEBL Packet	13-537
13-48	BSC Protocol Data-Link Control Characters (DLCC)	13-539

LIST OF TABLES (Cont.)

Table		Page
13-49	Contents of ?SERMSG Packet	13-554
13-50	Contents of ?SGES Packet	13-558
13-51	Contents of ?SOPPF Packet	13-574
13-52	Contents of ?SPAGE Packet	13-578
13-53	File-Pointer Position	13-582
13-54	Contents of ?SRCV Packet	13-587
13-55	Masks Returned on ?SRCV System Calls	13-593
13-56	Contents of ?SSND Packet	13-603
13-57	?SSND Call Types	13-607
13-58	Contents of Standard Task Definition Packet	13-629
13-59	Contents of Extended Task Definition Packet	13-633
13-60	Histogram Array Structure	13-681

GLOSSARY

Access control list (ACL)

A system-maintained list for each file and directory (for example, logical disk) that contains the names of users who can access that file or directory and the types of access to which they are entitled.

Access privilege

The basis of AOS/VS file access protection. You can be assigned up to four types of access privileges for files: Execute, Read, Write, and Owner. for directories, you can be assigned Execute, Read, Write, Owner, and Append access privileges.

ACL (See "Access control list (ACL)".)

Address space (See "Logical address space".)

Batch job

One or more programs submitted as a unit to batch.

Block

A logical grouping of contiguous words of code in main memory or on a peripheral storage device. Except for disk blocks, the size can vary.

Blocked process

One of three process states, in which a process is waiting for a specific external event to occur so that it can gain control of the central processor. A process can block itself or become blocked involuntarily.

Block I/O

One of two input/output modes in which you can access a file. Information is transferred in 512-byte disk blocks, magnetic tape blocks, MCA blocks.

AOS/VS always performs I/O in block units, whether you employ block or record I/O.

Block length

The number of bytes per block. (See also, "Block".)

Break file

A status file in which AOS/VS, under certain conditions, saves the state of a terminated process.

Character device

A device that performs I/O in byte units. CRT consoles and hard-copy terminals are typical character devices.

CLI (See "Command line interpreter (CLI)".)

Command line interpreter (CLI)

A utility that is the main interface between you and the system. The CLI accepts your command lines and (among other functions) translates this input into commands for other utilities, or into commands that directly perform functions such as file maintenance.

Connection table

A table in which AOS/VS writes an entry to manage exchanges between customers and servers.

Control character

A keyboard character that you type while you press the CTRL key.

Control point directory (CPD)

A directory in an LD that contains two variables: CS, the amount of space currently allocated; and MS, the maximum amount of space available in the directory. CPDs allow you to control the system's disk space allocation.

Control sequence

A CTRL-C followed by any control character. (See also, "Control character".)

CPD (See "Control point directory (CPD)".)

Critical region

A procedure or databased shared by all tasks, but available to only one task at a time.

CS (current space)

The amount of space currently allocated in a CPD. (See also, "Control point directory (CPD)".)

Data-link control character

A synchronization character mutually recognized by sending and receiving BSC stations.

Data-sensitive record type

A record type whose records consist of character strings terminated by one of the default delimiters, NEW LINE, carriage return, null, or form feed, or terminated by a user-defined delimiter.

Dedicated line

A communications line that continuously connects two or more stations, regardless of the amount of time the line is actually in use.

Dedicated pages

Memory pages that AOS/VS reserves for specific purposes, including physical pages occupied by the resident portion of the operating system and pages wired to a resident process.

Demand paging

Moving logical pages from the disk to memory as a proces refers to (demands) those pages.

Device

A hardware peripheral component; each type of device has unique operating characteristics. Devices are either character-oriented (send or receive single bytes of data) or block-oriented (send or receive data in multibyte blocks).

Device independence

The ability of a process to communicate with a device without regard to the unique nature of the device.

Directory

A file that catalogs files and allows qualified users to access them. Directories are connected in a structure that resembles an inverted tree. On this tree, the lower directories are inferior to the higher directories. Each directory contains an entry for any directory that is immediately inferior to itself.

Directory entry

A unit of information contained in a directory; a directory can contain multiple entries. A common type of entry is that which lists certain information about a file in the directory. Examples of other types of entries are IPC entries and links. (See also, "File status".)

Disk

A magnetic recording medium (for example, disk pack, disk cartridge, diskette, fixed-head disk).

Disk address

The location of a block on a disk. (See also, "Disk block".)

Disk block

The smallest allocatable unit of disk memory, standardized as 512 bytes.

Disk controller

A mechanism that directs the operator of one or more disk units. A program can direct the operation of a disk controller.

Disk controller name

The name of a disk controller, consisting of three letters and possibly one decimal digit; for example, DPE and DPE1.

Disk drive (See "Disk unit".)

Disk unit

A mechanism that physically reads from and writes to disk.

Disk unit name

The name of a disk unit, consisting of the name of a disk controller followed by a decimal digit; for example, DPE0 and DPE10.

Dormant state

One of four task states, in which a task exists that has not yet been initiated (made known to the operating system) or that has terminated execution.

Double connection

A connection in which each process can act as either the customer or the server of the other, depending on the action to be performed.

Dynamic record type

A record type in which you specify the record length when you read or write.

Eligible process

A process that has been allocated main memory, which allows it to compete for control of the CPU with other such processes, based on its proces type and its priority. (This is one of three process states.)

Error code

A 32-bit unsigned value that AOS/VS returns in AC0 to indicate an exceptional condition. (This exceptional condition may or may not indicate an actual error.) Each error code has a text string associated with it. (See the description of the ?ERMSG system call for information on getting the text string associated with a particular error code.)

Exceptional condition code (See "Error code".)

Executable file

A binary memory-image file that you can read into main memory from a peripheral storage device for exection; a program that can run.

Executable task

A task that has control of the CPU. Only one task at a time can be executing. (This is one of four task states.)

File

A collection of related data treated as a unit. A file can contain up to 2**32 bytes of data. Disk and magnetic tape can contain one or more files.

File element

The basic unit of storage in the AOS/VS disk file organization. Each file element consists of one or more contiguous blocks. You specify file element size when a file is first created. If a file grows, it grows in units of the file element size.

File status

A collection of information about each file. This information includes the file size, time of creation, and other details.

File system (See "Hierarchical file system".)

Filename

An alphanumeric file identifier. All filenames in a single directory must be unique, and each can contain no more than 31 characters.

Fixed-length record type

A record type in which you specify a predefined, common record length.

Form name

The name of a file in the :UTIL:FORMS directory, which was created with the CLI Forms Control Utility (FCU). The form name must contain from 1 through 31 legal filename characters.

Gate

An entry point to code in an inner ring.

Global port number

A number made up of a port's PID, ring number, and local port number, which uniquely identifies that port system-wide.

Global server

A separate process that performs functions on behalf of a customer process. (The servers that are described in Chapter 8 are global servers.)

Hierarchical file system

The inverted tree structure in which AOS/VS organizes files and directories. The highest directory in the hierarchy is the system root, which points to inferior directories; these, in turn, point to inferior directories. Any process with proper privileges can access any file within any directory.

High-order bits

The 16 most significant bits in a 32-bit value; that is, Bits 0 through 15.

Histogram

A data array that provides a global view of CPU activity.

Index

A single block that lists the address of each file element.

Ineligible process

An process that has not been allocated main memory, but in all other ways is ready to run. (This is one of thre process states.)

Initial task

The first task that executes in a process. AOS/VS assigns the initial task TID 1, priority 0.

Interprocess communication facility (IPC)

A generalized AOS/VS facility that sends free-format messages of any length between any two processes. IPC messages are sent between ports (See also, "Port".)

IPC (See "Interprocess communication facility (IPC)".)

Jobname

A name that identifies a batch job. A jobname must contain from 1 through 31 legal filename characters.

K

An abbreviation for the decimal number 1024. Thus, 32K bytes of memory are 32,768 bytes.

Keyword switch

A two-part switch of the following form: /keyword=value. For example, /L=filename is a keyword switch.

Kill-processing routine

A user-defined routine that guarantees an orderly release of a task's user-related resources.

LD (See "Logical disk (LD)".)

LEF mode

The CPU state that protects the system's I/O devices from unauthorized access. I/O instructions and LEF instructions use the same bit patterns. AOS/VS determines how to interpret these instructions by checking the state of LEF mode and the state of I/O mode. (LEF mode and I/O mode are mutually exclusive states.)

Link entry

A file that contains a pathname to another file.

Link-to-link reference

A link entry that is another link entry.

Load effective address mode (See "LEF mode".)

Local root

A single directory that acts as the foundation for a directory structure on a logical disk.

Local server

A server that shares the same logical address space as its customer. (Local servers can be loaded into the inner rings of your process.)

Locality of reference

Clustering instructions and data by writing code in modular pieces.

Logical address space

The entire range of locations that a process can address. A process's user-visible logical address space can be up to 512 megabytes for each ring.

Logical context

The total pages available to you (the user), including shared, unshared, and unused pages.

Logical disk (LD)

One or more physical disk units that you want to consider as a single logical unit.

Logical disk address

The location of a logical block on a logical disk. The address must include a disk pointer and a disk address to access the block.

Logical disk name

The filename of a logical disk's root directory.

Low-order bits

The 16 least significant bits in a 32-bit value; that is, Bits 16 through 31.

LRU chain

A list of released shared pages arranged in least recently used (LRU) order.

Main memory (physical)

Core or semiconductor storage, which contains computer instructions or data.

Master LD

A logical disk (LD) whose root becomes the system root (identified by a colon (:)). You must select the master LD.

MCA (See "Multiprocessor communications adaptor (MCA)".)

Modem

A communications device that translates analog signals to digital signals, and vice versa, over telephone lines.

MS (maximum space)

The maximum amount of space avilable in a CPD. (See also, "Control point directory (CPD)".)

Multidrop line

See Multipoint line.

Multilevel connection

A process that acts as both a server and a customer in a customer/server relationship.

Multipoint line

One of the two types of BSC line configurations (the other type is point-to-point). There is no contention between stations on a multipoint line.

Multiprocessor communications adaptor (MCA)

A device that permits communcation between two Data General central processors using the processors' data channels.

Multiprogramming

The ability to run an arbitrary number of independent processes. The system allocates its resources among these processes based on their priorities, types, or certain software events.

Multitasking process

A process in which more than one task is currently active.

Obituary message

A zero-length IPC message that is sent when a customer or a server disconnects. (Obituary messages use the IPC system calls.) (See also, "Obituary notice".)

Obituary notice

A signal that is sent when a customer or a server disconnects. (Obituary notices use the ?SIGNL, ?WTSIG, and SIGWT system calls.) (See also, "Obituary message".)

Object code

Code, consisting of 32-bit instruction words and data words, which has been assembled or compiled from a source code file but not yet bound with other modules by the Link utility to make an executable program.

Object code file

A file containing object code, usually created by the Macroassembler or one of several high-level language compilers and having a filename ending in ".OB".

Overlay (Not used by AOS/VS)

A portion of a larger program that can be brought into main memory when it is needed.

Overlay area (Not used by AOS/VS)

A fixed-length storage area in a program in which different overlays can be read at different times while a program is executing.

Packet

A group of words in your address space that AOS/VS uses to get your input specifications and/or return output values. Many system calls require a packet.

Page

Memory storage area of 2K (2048) bytes, starting on a 2K-byte boundary.

Page fault

A reference to a page that is not currently in the working set.

Parameter packet (See "Packet".)

Parametric code

Code in which system call packet offsets are cited by their mnemonic names, regardless of how the offsets are ordered in the packet figures.

Pathname

A name that identifies the location of a file within the system's files. A pathname may be a filename, or an optional list of directories followed by the file's name.

Physical disk

Same as disk.

PID (See "Process identifier (PID)".)

PID 2

The initial operator process (See also, "Process identifier (PID)".)

PID/ring tandem

Process identifier (PID)/ring-within-PID ordered pair. The connection management facility uses PID/ring tandems to identify all connections.

Point-to-point line

One of the two types of BSC line configurations (the other type is multipoint). Each station must bid for a point-to-point line.

Polling list

A series of contiguous words that contains each BSC tributary's poll address and device address.

Port

A data path to or from a process. The IPC facility sends messages between ports, which are full-duplex and can therefore send and receive data simultaneously. Each port is assigned a unique number (see also, "Interprocess communication facility (IPC)".)

Port numbers

The identification mechanism that allows two processes to send and receive messages via the IPC facility. The system maintains a directory of process numbers and associated port numbers.

Pre-emptible process

A process that the scheduler treats as a high-priority swappable process. (See also, "Swappable process".)

Priority numbers

Values in the range from 0 (the highest priority) to 255 (the lowest priority) that determine the order in which tasks or processes execute.

Process

An executing set of segment images, plus all of the system resources that the process image needs to execute. (A process is a dynamic entity.) (See the definition of "Process image" below.)

Process identifier (PID)

A number from 1 through 32 that you assign to identify each process.

Process image

A union of user segment images and of system segment images. (A process image is a static entity.) (See also, "Segment image".)

Process name

A character string consisting of a username and a simple process name, with a colon (:) separating the two elements. AOS/VS uses process names and PIDs to identify each process.

Process priority

One of the factors that governs how the system allocates CPU time to a process. More than one process can have the same priority. (See also, "Priority number".)

Process state

One of the factors that AOS/VS considers to determine the order in which it executes processes.

Process type

Process type governs when and for how long a process acquires main memory. The three process types are: resident, pre-emptible, and swappable.

Program

The current executable contents of a process's address space. A program contains the code paths executed by tasks. A process contains only one program at any given time; but during the execution of a process the current program may change many times.

Program file

A segment image linked for any one ring. (See also, "Segment image".)

Resident process

A process that always remains in memory somewhere. (See also, "Swappable process" and "Pre-emptible process".)

Ring maximization

A protection scheme in which AOS/VS considers a task that is executing in a user ring to be less privileged than another task that is executing in a lower user ring. AOS/VS uses ring maximization to validate user-supplied channels, word pointers, or byte pointers for system calls. (See also, "Ring specification.")

Ring specification

A protection scheme in which AOS/VS protects tasks executing in one user ring from interference by tasks executing in any other user rings. The connection management and IPC facilities use ring specification as their protection scheme.

Root process

The most superior process in the system hierarchy. All system processes and the initial process are sons of the root process.

Scalar notation

A time or date notation in which the current time equals the number of biseconds that have elapsed since midnight, and in which the date equal the number of days that have elapsed since 31 December 1967.

Search list

A list of directories that AOS/VS searches if it fails to find a specified file in your working directory. Each process has its own search list.

Segment

One of eight independent 512-megabyte units connected by strict protocols that make up your logical address space (See also, "Logical address space".)

Segment image

A .PR file that AOS/VS has made part of a process's logical address space. (A segment image is a static entity.)

Shared library (See "Shared routine facility".)

Shared page

A page in your logical address space that more than one process can access. Shared pages are usually write-protected to prevent overwriting.

Shared routine facility

The facility whereby AOS/VS implicitly calls one or more library routines on disk into main memory areas in page increments; processes share these.

Source code

Code, consisting of byte-packed words of ASCII characters, which can be converted by an assembler or compiler into object code. Usually, you compose source code.

Source code file

A file that contains source code. Usually you use the CLI or a text editor under AOS/VS to create source code.

Stack

A block of consecutive memory locations set aside for task-specific information. (Every task that uses system calls must have a unique stack.) (See also, "Wide stack".)

Stack base

The starting address of a stack.

Stack fault handler

A routine that gains control when there is a stack error.

Station

The origin (sender) or destination (receiver) of data over a BSC line.

Swappable process

A process that is swapped into memory and written out to disk at the discretion of the scheduler. Swappable processes have the lowest priority of the three process types; they acquire memory only after the scheduler has satisfied all resident and pre-emptible processes.

Swapping

A procedure whereby AOS/VS writes a process out to disk and then reassigns the main memory occupied by that process to another process that is waiting to run. This procedure is invisible to the process.

Switched line

A communications line on which you use a dialing procedure to establish a connection between local and remote stations.

System call

A request to the operating system to act on your behalf.

System generation

The process of tailoring AOS/VS to the particular hardware configuration and application environment at your installation.

Task

A path through a process. A task is an asynchronously controllable entity to which the CPU is allocated for a specific time. A task can only execute code within the bounds of the address space allocated to its process.

Task call (See "System call".)

Task control block (TCB)

A block of data maintained by AOS/VS that contains a memory image of the CPU registers and other context data for each task.

Task identifier (TID)

A user-specified number in the range from 1 through 32 that identifies a task within a particular process. (See also, "Unique task identifier (TID)".)

Task priority

Governs which is the executing task within a process. The executing task is always the highest priority task ready to run in the process with control of the central processor.

Task states

A task in a process exists in one of four states: dormant, ready, suspended, or executing.

TCB (See "Task control block (TCB)".)

Template

Certain charaters to be matched, plus one or more expansion operator characters that allow specified parts of the template to accept any character as legal.

Tick

A real-time clock pulse.

TID (See "Task identifier (TID).")

Time-out value

The length of time AOS/VS will wait for a response from the target device before it takes an error return or begins error-recovery procedures. The shortest possible time-out value is 2 seconds.

Timesharing

A multiprogramming scheme in which processes share the CPU on a timed basis; that is, a process takes control of the CPU for a unit of time called a time slice. When this time slice expires, control goes to the next process that is waiting. In this way, no process monopolizes the CPU.

Trapping

Encountering a hardware fault.

Undedicated pages

Pages that AOS/VS can assign to a process as it requires them.

Unique task identifier (TID)

A system-assigned number that uniquely identifies each task, system-wide. (See also, "Task identifier (TID)".)

Unshared page

A page in your logical address space that only one process can access. Unshared pages cannot be write-protected.

Unused page

A page in your logical addres space that is neither shared nor unshared. (See Chapter 3 for information on the relationships among shared, unshared, and unused pages in a typical logical context.)

Variable-length record type

A record type whose records have a 4-byte ASCII header that specifies their byte length. Files that contains records of varying lengths have the variable-length record type.

Wide stack

A 32-bit stack. (See also, "Stack".)

Wired pages

Pages that are permanently bound to the working set.

Word

A 32-bit (2-byte) location of memory.

Working directory

A process's reference point in the overall directory structure and its starting point for file access. Any directory can be a working directory, as long as you have proper access to it.

Working set

The subset of each process's logical address space that is memory resident. The working set of a process changes in size and content as the process references pages and then stops referencing them.

CHAPTER 1
INTRODUCTION TO AOS/VS

AOS/VS (Advanced Operating System/Virtual Storage) is a 32-bit, demand-paged, virtual-memory operating system that runs on Data General's ECLIPSE® MV/Family of machines.

AOS/VS combines the flexibility and convenience of minicomputer architecture with the processing power of a large mainframe computer. Because of its 32-bit addressing capability, its multiuser and multiprogramming support, and its compatibility with existing Data General software, AOS/VS is uniquely suited to both commercial and scientific applications. Specifically, AOS/VS provides you with the following:

- A logical address space of up to 2048 megabytes per process
- Virtual memory management
- Sophisticated process-protection schemes
- Support for concurrent 16- and 32-bit programs
- Compatibility with the Advanced Operating System (AOS)
- A wide range of system and applications utilities
- High-level language support
- Full functional support for inner rings

Full functional support for the inner rings allows you to write multitasked programs that will execute in more than one user ring (the user rings are Rings 4 through 7). Specifically, full functional support for the inner rings provides you with the following advantages:

- Improved software performance

  You can take better advantage of the large logical address space of the MV-series hardware by using the inner user rings to create local servers. (Local servers are servers that share the same logical address space as their customers. You can load a local server into the inner rings of a process.)

Local servers are faster than global servers because they do not need to use the interprocess communications (IPC) facility system calls or the ?MFBC and ?MTBC system calls to move data between customer and server. Instead, because a local server resides in the same logical address space as its customer, local servers can use MV-series hardware instructions to perform identical synchronization and data movement.

o Improved accounting

  When you use the inner rings to implement local servers, the server becomes part of the logical address space of the process that uses it. Therefore, the server is no longer a separate process. A local server's use of resources is accounted for by AOS/VS as part of the resources used by the customer's process.

o Larger logical address space

  By using the inner rings, you can expand your logical address space from 512 megabytes (the capacity of one user ring) to 2048 megabytes (the capacity of the four user rings).

Virtual Memory

Virtual memory allows you to run programs that are larger than the physical memory configuration of your system. With virtual memory, AOS/VS can move the active portions of a program from disk to memory while the program is executing. Then, when the system needs more memory, AOS/VS returns the inactive portions of the program to disk. This process of moving portions of the program in and out of memory is called demand paging.

The portion of an executing program (called a process) that is in physical memory at any given time is its working set. The size of each process's working set changes as the demands of the process change. AOS/VS determines the working set size by examining the number of pages the process currently needs as well as its history of page faults.

Page faults are references to memory locations that are not currently in physical memory. When a page fault occurs, the AOS/VS demand-paging mechanism moves the page that is needed from disk into physical memory.

AOS/VS allocates a large working set to a process that has a history of many page faults. Therefore, to run your system as efficiently as possible, you must reduce the number of page faults. To do this, write your code in modules that cluster the instructions and data together as closely as possible. (See Chapter 13 for information on writing modular code.) The fewer page faults your process causes, the smaller and more stable is its working set. However, some page faults are unavoidable.

Ring Structure

The entire range of memory locations that a process can address is called its logical address space. The logical address space is divided into eight 512-megabyte units called segments. Although these segments are connected by strict protocols, they are independent of one another. Therefore, AOS/VS can use each segment for a different function. This makes your virtual memory system very efficient and reliable.

Each segment is protected by a ring that is permanently bound to that segment. Thus, Ring 0 (the innermost ring) protects Segment 0, Ring 1 protects Segment 1, and so forth through Ring 7 (the outermost ring) and Segment 7.

These rings prevent segments from interfering with one another, even though each segment may be performing a different function. If a program that is executing in one segment needs to change or access the contents of another segment, it must follow strict protocols established by the rings. (The system follows these protocols without your knowledge.)

The eight segments (and their rings) are arranged hierarchically. Segment 0 has the greatest ability to change or access the contents of other segments, and Segment 7 has the least. Similarly, Ring 0 gives Segment 0 the greatest protection from interference by other segments, and Ring 7 gives Segment 7 the least protection.

Segments 0 through 3 contain the AOS/VS operating system. Segments 4 through 7 contain user programs. Because the user programs and the AOS/VS operating system share the single large logical address space, context switching is unnecessary. In fact, system calls and calls to routines that are in another segment become subroutine calls. This means that when you issue a system call, there is no need for AOS/VS to change contexts. AOS/VS does take part, however, in the execution of most system calls.

Ordinarily, a segment can only change or access the contents of segments whose segment and ring numbers are higher than or equal to its own segment and ring number. For example, the rings will not allow a program that is executing in Segment 4 to access the contents of Segments 0 through 3, but they would allow that same process to access Segments 4 through 7.

With a subroutine call, however, a segment whose segment number is higher than or equal to the target segment can access the segment in which the subroutine actually resides. In this case, the ring that protects the target segment allows the subroutine call to pass through a gate. This gate points to the starting location of the subroutine.

Although you cannot make a cross-ring subroutine call directly to the starting location of the subroutine, you can return directly from the subroutine. Subroutine returns do not have to pass through gates.

In fact, the only restriction on subroutine returns is that they must originate from a segment whose number is lower than or equal to the target segment.

Refer to the 'Principles of Operation ECLIPSE 32-Bit Systems' manual for information on the hardware instructions that allow you to define gates and reference code in the outer rings.

AOS Compatibility

You can execute both 32-bit and 16-bit programs concurrently under AOS/VS. In addition, AOS/VS is compatible with AOS. In most cases, you need only relink AOS-written programs to execute them under AOS/VS; however, you may also need to reassemble or recompile, depending on your program.

Your AOS/VS system's compatibility with AOS extends to the file structure, magnetic-tape formats, and peripheral devices. You can transport disk files and tapes developed under AOS to AOS/VS without rewriting them. In addition, 16-bit device drivers written under AOS/VS can coexist with their 32-bit counterparts.

Overlays are not necessary for 32-bit programs, and therefore, are not supported for them. However, AOS/VS does support overlays for 16-bit programs.

Certain system databases, such as task control blocks (TCBs) and the user status table (UST) are in the system's address space. Thus, AOS programs that manipulate these databases without using task system calls need modification. AOS/VS does provide each program with a copy of the program's UST, but for reading purposes only.

The AOS/VS system calls use 32-bit packets. If your AOS assembly language program uses the appropriate mnemonics for the packet offsets, you need only reassemble them with the new 16-bit parameter file (PARU.16) and relink them to run under AOS/VS.

Inner-Ring Management Terms

Previously, this manual defined a process as: "A program file executing under the operating system, plus all of that program's system resources." For the purposes of inner-ring management, however, the term "process" is not precise enough, nor are some of the other common AOS/VS terms. Therefore, in this manual, we use the following terms:

o Segment image

  A .PR file that AOS/VS has made part of a process's logical address space. (A segment image is a static entity.)

o Process image

  A union of user segment images and of system segment images. (A process image is a static entity.)

o Program file

  A segment image linked for any one ring.

o Process

  An executing set of segment images, plus all of the system resources that the process image needs to execute. (A process is a dynamic entity.)

o Task

  An asynchronous flow of control within a process. (A task is a dynamic entity.)

o Global server

  A separate process that performs functions on behalf of a customer process.

o Local server

  A server that shares the same logical address space as its customer. (Local servers can be loaded into the inner rings of your process.)

For a complete list of AOS/VS definitions, see the "Glossary" in this manual.

System Calls

AOS/VS supports a wide variety of system calls. System calls are command macros that call on predefined system routines. There are various categories of system calls, which allow you to do the following:

- Create and manage processes.
- Manage the logical address space.
- Establish interprocess communications.
- Create and maintain disk files and directories.
- Perform file input and output.
- Create and manage a multitasking environment.
- Define and access user devices.
- Establish binary synchronous communications.
- Establish customer/server connections between processes.
- Perform input and output in blocks, rather than in records or lines.

This manual groups the system calls into functional categories, with a chapter that describes each category. The individual system call descriptions are arranged alphabetically in Chapter 13.

End of Chapter

CHAPTER 2
MEMORY

The memory-management system calls are:

?ESFF	Flushes shared file memory pages to disk.
?GMEM	Returns the current number of undedicated pages.
?GSHPT	Lists the current size of the shared partition.
?LMAP	Maps a lower ring.
?MEM	Lists the current unshared memory parameters.
?MEMI	Changes the number of unshared memory pages.
?PMTPF	Permits access to an open, protected shared file.
?RPAGE	Releases a shared page and decrements its use count.
?SCLOSE	Closes a shared file.
?SOPEN	Opens a shared file.
?SOPPF	Opens a protected shared file.
?SPAGE	Reads a shared page and increments its use count.
?SSHPT	Establishes a new shared partition size.

This chapter describes how memory is organized under AOS/VS and how each process can manipulate its own logical context.

To understand this chapter, you must be familiar with the following terms and what they mean to AOS/VS:

o Logical context

  Logical context refers to the total pages available to you (the user), including shared, unshared, and unused pages.

o Logical address space

  Logical address space is the entire range of locations that a process can address. A process's user-visible logical address space can be up to 512 megabytes for each user ring.

- Shared page

  A shared page is a memory-resident page in your logical address space that more than one process can access. Shared pages are usually write-protected to prevent overwriting. (See "Shared Pages" in this chapter for more information on shared pages.)

- Unshared page

  An unshared page is a page in your logical address space that only one process can access. Unshared pages cannot be write-protected. (See "User Context" in this chapter for more information on the relationships among shared, unshared, and unused pages in a typical logical context.)

- Unused page

  An unused page is a page in your logical address space that is neither shared nor unshared. (See "User Context" in this chapter for more information on the relationships among shared, unshared, and unused pages in a typical logical context.)

- Working set

  Working set is the subset of each process's logical address space that is memory resident. The working set of a process changes in size and content as the process references pages and then stops referencing them.

Ring Structure

The system's logical address space is divided into eight units of 512 megabytes each, which are called segments. Although these segments are independent, they are connected by clearly defined protocols that allow AOS/VS to use each segment for a different function.

The software modularity that the segments provide means that there must be protection mechanisms. To this end, AOS/VS provides hardware protection rings, which maintain the necessary independence or interdependence of the different software modules. If a program that is executing in one segment needs to alter or access the contents of another segment, the program must follow protocols established by the rings.

There are eight rings, Ring 0 (the innermost ring) through Ring 7 (the outermost ring), which surround and protect each segment. In turn, each segment is permanently bound to a particular ring. Thus, Ring 0 is bound to and protects Segment 0. Similarly, Ring 1 is bound to and protects Segment 1, and so forth through Ring 7 and Segment 7. (See Figure 2-1.)

The eight segments (with their associated rings) are hierarchically arranged: Segment 0 has the greatest ability to alter or access the contents of other segments and is afforded the greatest protection by Ring 0. Segment 7 has the least ability to alter or access other segments and is afforded the least protection by Ring 7.

Therefore, Segment 0 contains the kernel of the AOS/VS operating system, while Segment 7 is reserved for user programs. AOS/VS uses the other segments for various system or user functions. Rings 0 through 3 are the system rings, while Rings 4 through 7 are the user rings. (In the future, however, Rings 4 and 5 may contain Data General-supplied software.)

```
  ////////////////Segment 3////////////////
||// |==============Ring 2=============| //||
||// ||///////////Segment 2///////////|| //||
||// ||// |=========Ring 1========| //|| //||
||// ||// ||//////Segment 1//////|| //|| //||
||// ||// ||// |====Ring 0===| //|| //|| //||
||// ||// ||// ||///////////|| //|| //|| //||
||// ||// ||// ||/Segment 0/|| //|| //|| //||
||// ||// ||// ||///////////|| //|| //|| //||
||// ||// ||// |=============| //|| //|| //||
etc. ||// ||/////////////////////|| //|| etc.
     ||// |=======================| //||
```

Figure 2-1. Segments and Their Protection Rings

Ring 7, which has a maximum logical size of 512 megabytes, is the default user ring. However, you can load a program file into one of the other user rings by issuing the ?RINGLD system call. (See Chapter 3 for more information on rings and the ?RINGLD system call.)

If you are a privileged user, you can use the ?LMAP system call to monitor the kernel in Ring 0, provided the caller is a resident process and Superprocess mode is turned on. (See Chapter 3 for information on process types and Superprocess mode.)

Demand Paging

AOS/VS is a demand-paged, virtual-memory operating system. Demand paging is the AOS/VS method of moving logical pages from the disk to memory as the process "demands" (refers to) those pages. As AOS/VS moves pages into memory, it can free memory pages to accommodate the new entries.

At any given time, only a subset of each process's logical address space is in memory. This subset, which is called the working set, changes in size and content as the process references pages.

Every process starts with a working set large enough to accommodate Page 0 (the first 2K bytes of the logical address space) and the program counter (PC) page. The PC points to the current control point in a program.

The rest of the logical address space -- the pages outside the working set -- is virtual address space. Figure 2-2 shows the working sets and virtual address space of several processes.

Shared and Unshared Memory Pages

There are two kinds of memory pages:

- Shared pages

  Shared pages are memory-resident pages that may or may not be initialized. When the use count of an initialized memory-resident shared page is 0, the shared page stays in memory. However, when the use count of a non-initialized memory-resident shared page is 0, AOS/VS releases the shared page from memory to the free memory chain.

- Unshared Pages (See "User Context" in this chapter.)

  Unshared pages are pages in your logical address space that only one process can access. You cannot write-protect unshared pages.

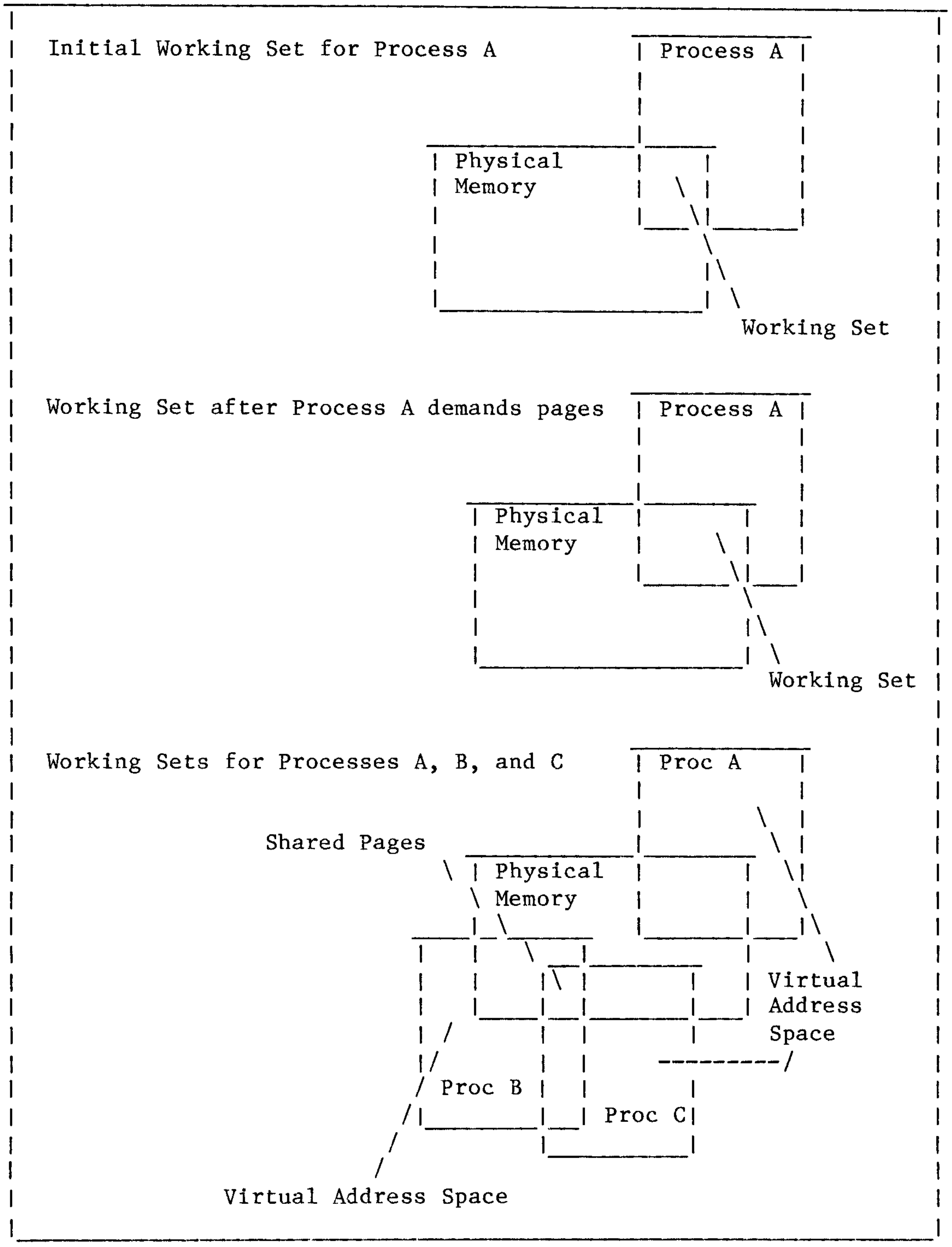

Figure 2-2. Working Sets in Memory

You can conserve memory by using shared pages, because they allow more than one process to use the same re-entrant code or data. Also, shared pages reduce disk I/O, because AOS/VS does not immediately swap them to disk when a process releases them. Instead, it retains shared pages in a cache-like collection in memory for other processes to use.

If a shared page is not currently in use, AOS/VS places it on an LRU chain. An LRU chain is a list of released shared pages, which is arranged in least recently used (LRU) order. The shared pages on the LRU chain are candidates for re-use by a process of any type (that is, resident, pre-emptible, and swappable).

The ?SPAGE system call reads one or more contiguous pages of a disk file into the shared area of the caller's logical address space. Its complement, the ?RPAGE system call, releases one or more shared pages from the caller's logical address space, but may retain them in memory.

When you issue the ?RPAGE system call, AOS/VS does not immediately release the shared page from memory. If you modified the page and, therefore, want to release and update it immediately, you must issue either a ?FLUSH system call or a modified version of the ?RPAGE system call, which implies a ?FLUSH system call. The ?FLUSH system call writes the updated contents of a shared page or pages to disk.

Before you can use the ?SPAGE, ?RPAGE, or ?FLUSH system calls, you must use the ?SOPEN system call to open the target file for shared access. A file opened this way is called a shared file. The ?SOPEN system call gives you the option of opening your shared file for Read-only access. To close a shared file, you must issue the ?SCLOSE system call.

There are three ways to use shared memory pages:

o Explicitly, by using the shared-page system calls, such as ?SSHPT, ?SOPEN, ?SPAGE, and so forth

o Implicitly, by defining a shared area with assembly language pseudo-ops

o By opening a file for shared access with a special form of the ?OPEN system call

The .NREL and .PART pseudo-ops allow you to define shared areas in an assembly language program. The .NREL pseudo-op directs the macroassembler (MASM) to place the code or data that comes after it into one of the predefined NREL (normal relocatable) memory partitions. To specify which partition you want, use the appropriate nonzero argument with the pseudo-op.

For example, the statement .NREL 5 tells MASM to place all subsequent source statements in the predefined shared-data partition. The statements .NREL 1 and .NREL 7 tell MASM to place all subsequent source statements in the predefined shared-code partition.

To define your own partitions in NREL memory, use the .PART pseudo-op. This pseudo-op allows you to define a variety of attributes (characteristics) for the partition, including whether it is part of shared or unshared memory.

When you link your source code, the Link utility uses your .NREL and .PART specifications to create shared (and unshared) partitions in the final program file. The shared areas become part of the logical address space of any process that uses the program file.

For information on using the ?OPEN system call for page sharing, see Chapter 5.

Protected Shared Files

A set of common local servers can use shared memory files to coordinate access to a common resource. Each local server that wants to share the memory must first open and then read from or write to the same shared file.

Inner-ring servers may need to limit access to their shared files. They may not want any segments other than themselves to have access to their shared memory. However, the access control list (ACL) protection mechanism cannot protect a local server, because all segments within a process share the same username. The ?SOPPF and the ?PMTPF system calls permit a more private form of protecting shared files.

You can use the ?SOPPF system call to open a shared file in a protected manner. Once a shared file has been opened in a protected manner, the opener can issue the usual shared-page system calls, just as if the channel were opened by a ?SOPEN system call. To close a shared file, whether or not it was opened in a protected manner, you can use the ?SCLOSE system call.

The first ?SOPPF system call behaves differently than subsequent ?SOPPF system call opens of the same shared file that you want to open in a shared manner. (See the individual system call description of the ?SOPPF system call for more information on the difference between first and subsequent opens.)

After a segment image uses the ?SOPPF system call to open a protected shared file for the first time, that segment image is called the "first opener" of the file. The first opener of a protected shared file can use the ?PMTPF system call to permit other segment images to access the file. The ?PMTPF caller also informs AOS/VS of the type of file access privileges that the caller wants to pass to another segment image.

Only the first opener of a protected shared file can issue a ?PMTPF system call against that file. Also, there must be a valid connection between the PID/ring tandem from which the ?PMTPF system call is issued (the server) and the PID/ring tandem of the target (the customer).

A first opener that issues the ?PMTPF system call cannot pass access privileges it does not have itself. In addition, access privileges are not cumulative.

An access grant remains active until one of the following events occurs:

- o The connection between the first opener of the protected shared file and the target segment image is broken.
- o The first opener closes the file.
- o The first opener revokes the access grant.

This means that a segment image possesses only the access privileges specified by the most recent ?PMTPF system call that addressed that segment image. Thus, a ?PMTPF system call that specifies no privileges can revoke a segment image's access privileges.

Coordinated Shared-File Update

Periodically, inner-ring servers may need to checkpoint the updated status of a set of shared memory pages. Such checkpointing may be critical for recovery from system failure.

The ?ESFF system call helps checkpoint shared memory by flushing to disk all modified pages associated with a specified shared file, no matter where they are in system memory. AOS/VS tries to flush all modified shared pages, even if it encounters an I/O error while it is flushing the pages.

The ?ESFF system call makes only one pass through the pages in a shared file. Therefore, if another process -- or other tasks within the same process -- concurrently updates the shared file, the checkpoint state will be uncertain.

Dedicated and Undedicated Memory Pages

Just as AOS/VS distinguishes between shared and unshared pages, it also distinguishes between dedicated and undedicated memory pages:

o Dedicated pages are memory pages that AOS/VS reserves for specific purposes. They include physical pages occupied by the resident portion of AOS/VS and pages wired to a resident process by the ?WIRE system call.

o Undedicated pages are pages that AOS/VS can assign to any process as the process needs them. Undedicated pages are not necessarily "unused" pages; they are simply available for reassignment. The ?GMEM system call returns the current number of undedicated pages available to the calling process.

User Context

The user's unshared area starts at the first word of the logical address space in the current ring, and expands toward numerically higher addresses. The shared page area occupies the numerically highest portion of the address space and expands upward and downward.

Between the shared and unshared portions of the logical context, there can be an "unused" area. You can allocate this area with the system calls ?MEMI and ?SSHPT. The ?MEMI system call modifies the unshared area's upper boundary, while the ?SSHPT system call modifies the number of shared pages in the logical address space and the position of the shared area in your user address space.

Figure 2-3 shows the relationship among the unshared, unused, and shared areas in a typical user context. Table 2-1 lists the system calls that are available for managing a process's logical context.

```
 ______________________________________________________________________
|                                                                      |
|                    0 ------------------   |                          |
|                      |    Unshared     |  |                          |
|                      |_________________|  V _                        |
|                      |\\\\\\\\\\\\\\\\\|     |                       |
|                      |\\\\\\\\\\\\\\\\\|     |                       |
|                      |\\\\\\\\\\\\\\\\\|     > Unused Area           |
|                      |\\\\\\\\\\\\\\\\\|     |                       |
|                      |-----------------|  ^ _|                       |
|                      |                 |  |                          |
|               (1 MB) |-----Shared------|  |                          |
|                      |                 |  |                          |
|         77777777777  |_________________|  V                          |
|                                                                      |
|______________________________________________________________________|
```

Figure 2-3. Memory Context

Table 2-1. Context-Management System Calls

System Call	Function
?ESFF	Flushes shared file memory pages to disk (shared pages only).
?GSHPT	Lists shared-partition information for this context (shared pages only).
?MEM	Lists the maximum number of unshared pages available, the number of unshared pages used, and the highest currently used unshared address in Ring 7 (unshared pages only).
?MEMI	Increases or decreases the number of unshared pages in Ring 7 (unshared pages only).
?PMTPF	Permits access to an open, protected shared file (shared pages only).
?RPAGE	Releases a shared page (shared pages only).
?SCLOSE	Closes a shared file (shared pages only).
?SOPEN	Opens a file for shared access (shared pages only).
?SOPPF	Opens a protected shared file (shared pages only).

Table 2-1. Context-Management System Calls (Cont.)

System Call	Function
?SPAGE	Reads a shared page (shared pages only).
?SSHPT	Establishes a new shared-partition size (shared pages only).

End of Chapter

CHAPTER 3
PROCESSES

The process-management system calls are:

?BLKPR	Blocks a process.
?BRKFL	Terminates a process and creates a break file.
?CHAIN	Passes control to a new program.
?CTYPE	Changes a process's type.
?DADID	Gets the PID of a process's father.
?ENBRK	Enables a break file.
?EXPO	Sets, clears, or examines execute-protection status.
?GUNM	Gets the username of a process.
?KHIST	Kills a histogram.
?MDUMP	Dumps the memory image from a specified ring to a file.
?PNAME	Gets a process name.
?PRIPR	Changes the priority of a process.
?PROC	Creates a process.
?PSTAT	Returns status information on a process.
?RESCHED	Schedules another process for execution.
?RETURN	Terminates the calling process.
?RINGLD	Loads a program file into a specified ring.
?RNGPR	Returns the .PR filename for a ring.
?RNGST	Stops lower rings from being ringloaded.
?RUNTM	Gets runtime statistics on a process.
?SUPROC	Enters, leaves, or examines Superprocess Mode.
?SUSER	Enters, leaves, or examines Superuser Mode.
?TERM	Terminates a process.
?UBLPR	Unblocks a process.
?UNWIRE	Unwires pages previously wired.
?WHIST	Starts a histogram.
?WIRE	Wires pages to the working set.

This chapter defines processes and how AOS/VS uses them. Also, this chapter describes the system calls that allow you to manage processes.

To understand this chapter, you must be familiar with the following terms and what they mean to AOS/VS:

o Program

  A program is user code that all tasks within a particular process execute. A program file contains user code and control information that is supplied by the Link utility.

o Process

  | A process is made up of a program file for each segment, with a set of constraints on the use of resources, such as memory and I/O channels, and a set of resources that are currently in use. (In addition to user code, a program file contains status information supplied by the Link utility.)

o Task

  A task is a path through a program file. It is an asynchronously controllable entity to which the CPU is allocated for a specific time. A task can only execute code within the bounds of the address space allocated to its process. A task is the basic element of a process.

Each process is made up of one or more tasks, which execute asynchronously. You can design your code so that several tasks execute a single re-entrant sequence of instructions, or you can create a different instruction path for each task. Control always goes to the highest priority ready process, and within that process, to the highest priority ready task. (For more information on tasks, see Chapter 6.)

When you create a process, it exists until one of the following events occurs:

o The process traps. (See "Process Trapping" in this chapter.)

o The process terminates voluntarily. (See the description of the ?TERM system call in Chapter 13.)

- o Another process terminates the process. (See the description of the ?TERM system call in Chapter 13.)
- o The process's father terminates.

Memory Scheme

Each process competes independently for system resources, such as memory and CPU time. When AOS/VS has allocated main memory to a process, that process is eligible for CPU time. AOS/VS allocates memory and CPU to each process based on its process type and priority.

The entire range of locations addressed by a process is its logical address space. Under AOS/VS, a process's user-visible address space can consist of up to 512 megabytes for each of the four user rings (Rings 4 through 7). (See Chapter 2 for information on memory, how it is organized, and what protections are available to you.)

At any given time, only a subset of each process's logical address space is in memory. This subset, which is called the working set, changes in size and content as the process references pages and then returns them to disk.

Every process starts with a working set large enough to accommodate Page 0 (the first 2K bytes of the logical address space) and the program counter (PC) page. The program counter points to the current control point in a program.

The rest of the logical address space -- the pages outside the working set -- is virtual address space. Figure 3-1 depicts the working sets and virtual address space of several processes.

The size of a process's working set directly relates to the number of memory pages the process currently needs or is likely to need. When a process refers to a page or set of pages outside its working set, the hardware signals a page-fault condition. AOS/VS responds by adjusting the size of the working set. The ?WIRE and ?UNWIRE system calls give a process with sufficient privileges control over its working set. The ?WIRE system call wires (that is, permanently binds) pages to the working set. The ?UNWIRE system call releases previously wired pages.

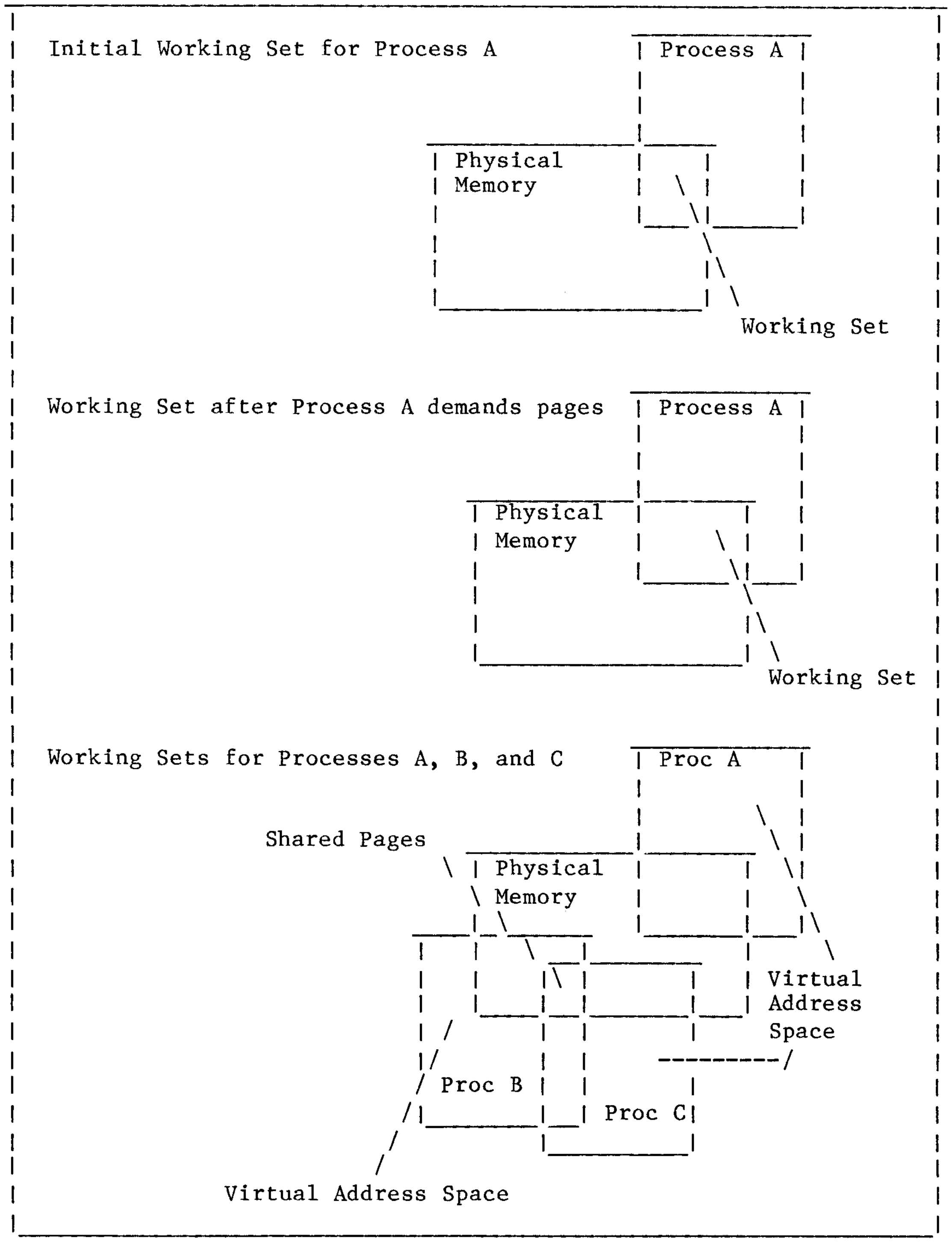

Figure 3-1. Working Sets in Memory

Process Types

To manage the multiprocess environment, AOS/VS allocates main memory to processes based on their types and priorities. There are three process types:

- Resident

  A resident process is always in memory somewhere. In general, only the most critical processes in your system environment should be resident.

- Swappable

  A swappable process is swapped into memory and written out to disk at the discretion of the scheduler. Swappable processes have the lowest priority of the three process types; they acquire memory only after the scheduler has satisfied all resident and pre-emptible processes.

- Pre-emptible

  A pre-emptible process is a hybrid of the other two types of processes in that the scheduler treats it as a high-priority swappable process.

When you create a process with the ?PROC system call, you can define it as one of these three process types. (By default, a newly created process has the same process type as the ?PROC caller.)

All three types of processes can issue the ?WIRE system call to wire pages. However, although AOS/VS can write wired pages out to disk for a swappable process, it cannot do so for a resident process. Therefore, a resident process, whose wired and unwired pages remain in memory, is capable of wiring enough pages to degrade the system. This means that you can run out of memory if you create one or more resident processes with a large number of wired pages. Therefore, you should avoid creating resident processes whenever possible.

As a general rule, AOS/VS keeps interactive swappable processes in memory longer than non-interactive swappable processes. You can change this, however, by setting the bias factors. (Refer to Chapter 8 and to the 'Managing AOS/VS' manual for information on bias factors.)

AOS/VS treats a pre-emptible process as a high-priority swappable process. However, when a resident process or a higher priority pre-emptible process requires memory, AOS/VS swaps the pre-emptible process out to disk. Also, when another process explicitly blocks a pre-emptible process (with the ?BLKPR system call), AOS/VS can swap the pre-emptible process out to disk if it needs more memory.

Priority Numbers

Eligible pre-emptible and eligible resident processes compete with each other for CPU time, based on their individual priority numbers. (See "Process States" in this chapter for a definition of eligible processes.)

Priority numbers are values AOS/VS uses to determine each process's priority, relative to other processes of the same type. By using priority numbers and other factors, such as each process's past behavior, AOS/VS determines which processes of a specific type should run before others of that same type. When you create a process, you can assign it a priority number as well as a process type.

The priority numbers for resident and pre-emptible processes range from 1 (the highest priority) through 255 (the lowest priority). The priority numbers for swappable processes are 1 (high priority), 2 (normal priority), and 3 (low priority).

Process Identification

A process identifier and a process name identify each process. When you create a process, AOS/VS assigns it a unique process identifier (PID) in the range from 1 through 255. At the same time, you must assign a process name to that process.

A full process name is a character string that consists of a username and a simple process name, with a colon (:) between the two elements. Each element can contain up to 15 valid filename characters. The valid filename characters are:

- o Letters A through Z. (AOS/VS treats uppercase and lowercase letters the same.)
- o Numbers 0 through 9.
- o Period (.), dollar sign ($), question mark (?), and underscore (_).

A username functions like a family surname. AOS/VS uses this part of the process name to determine the process's geneology and its access rights to files. By default, each son process bears its father's username. A father process can assign its sons a different username only if the father was created (by issuing the ?PROC system call) with the privilege to do so.

You can use either a full process name or a simple process name as input to the system calls. When you supply a simple process name, AOS/VS expands it. (See Figure 3-2.)

```
                 / ______________ \
                /                  \
          FAY PROC1             SAM PROC1
              /\                    /\
             /  \                  /  \
   FAY PROC2  FAY PROC3     SAM PROC2  SAM PROC3

         FAY PROC1              SAM PROC1
         FAY PROC2              SAM PROC2
         FAY PROC3              SAM PROC3
```

Figure 3-2. Process Names

Figure 3-2 shows a process with the full process name SAM:PROC2, where SAM is the username and PROC2 is the simple process name. If you issue a system call from SAM:PROC1 with the simple process name PROC2 as an input parameter, AOS/VS recognizes the target process as SAM:PROC2.

You cannot assign the same simple process name to processes that have the same username. If you do, AOS/VS returns error code ERPNU (process name already in use).

Depending on your input specifications, the following system calls return the process name and/or PID of a target process:

- ?PNAME returns the full process name or PID of either the calling process or another target process

- ?GUNM returns the username associated with a specific simple process name or PID
- ?DADID returns the PID of a father process (the father of either the calling process or of another process)

Process Creation

To create a process, define its privileges, and define its characteristics, issue the ?PROC system call.

The antecedent of every other process is a process called the system root. AOS/VS creates the system root when you initialize the system. From the system root, AOS/VS creates certain system processes, such as the peripheral manager (PMGR), which manages character I/O. Also, AOS/VS creates at least one user process, called the initial (operator) process. The initial process can create subordinate processes, or sons, and assign them a process type and priority number.

AOS/VS manages processes by organizing them into a hierarchical tree structure, where processes on the lower branches are subordinate to their relatives on the higher branches. (See Figure 3-3.)

The system root is the highest process in the system hierarchy; every other process is a son of the system root. User processes are sons of the initial process.

Once a process has been created, it continues to exist until one of the following events occurs:

- The process traps. (See "Process Trapping" in this chapter.)
- The process terminates voluntarily. (See the descriptions of the ?TERM and the ?RETURN system calls in Chapter 13.)
- Another process terminates the process. (See the description of the ?TERM system call in Chapter 13.)
- The process′s father terminates.

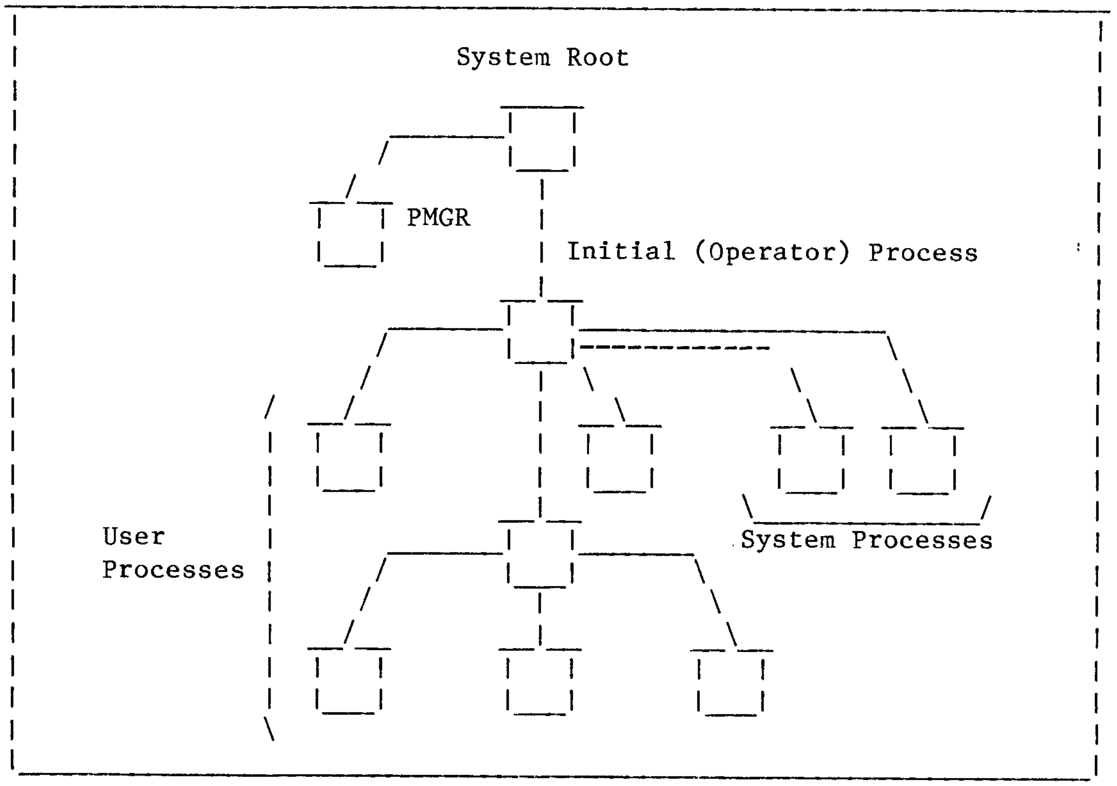

Figure 3-3. Process Hierarchy

Process Privileges

Within the ?PROC packet, you can define a number of privileges for a newly created process; for example, the right to create sons and to assign those sons minimum and maximum working-set parameters, and the right to override the usual file access controls. However, you cannot assign the new process privileges that the ?PROC caller does not have.

Table 3-1 lists the bit masks in offset ?PPRV of the ?PROC packet that define process privileges.

Table 3-1. Process Privileges

Privilege	Meaning
?PVPC	The new process can create an unlimited number of sons.
?PVWS	The new process can create sons of a different program file type (that is, 16-bit or 32-bit program files).
?PVEX	The new process can remain unblocked while one of its sons executes.
?PVWM	The new process can define working-set parameters for its sons.
?PVPR	The new process can use the ?PRIPR system call to change its own priority or to assign its sons higher priorities than its own.
?PVTY	The new process can use the ?CTYPE system call to change its process type or to create sons of any process type.
?PVIP	The new process can issue the ?ISEND and ?IS.R primitive IPC system calls. (See Chapter 7 for information on IPC system calls.)
?PVUI	The new process can create sons that have usernames different from its own.
?PVDV	The new process can define and access user devices. (See Chapter 10 for information on devices.)
?PVSP	The new process can issue the ?SUPROC system call to turn on Superprocess mode. (See "Superuser Mode/ Superprocess Mode" in this chapter.)
?PVSU	The new process can issue the ?SUSER system call to turn on Superuser mode. (See "Superuser Mode/ Superprocess Mode" in this chapter.)

Process Creation Parameters

AOS/VS determines the number of offspring a process can create by checking its ?PROC packet for:

- The ?PVPC privilege, which specifies that the new process can ?PROC an unlimited number of sons.

  This privilege overrides every other creation parameter in the ?PROC packet. When a process that does not have the ?PVPC privilege tries to create a son, AOS/VS performs the following steps to check the other creation parameters:

  1. Does the number of sons and their combined ?PPCR count exceed the caller's ?PPCR value? If yes, signal an error. If no, perform Step 2.

  2. Is bit ?PVEX set? If yes, allow the caller to create the son. If no, perform Step 3.

  3. Does the caller have the ?PVEX privilege? If yes, allow the caller to create the son. If no, do not allow the caller to create the son.

- The ?PVEX privilege, which specifies that the new process can remain unblocked while one of its sons executes.

- The presence of offset ?PPCR, which specifies the maximum number of offspring.

  Offset ?PPCR is a cumulative value. That is, if a process with a ?PPCR value of 10 creates 2 sons, each with a ?PPCR value of 4, the original process cannot create any other sons, because 2 sons plus 2*4 (8 potential grandsons) equals 10.

- The presence of the ?PFEX mask within offset ?PLFG, which determines whether the new process blocks while its sons execute.

You can use ?PROC system calls in your program if you want to create son processes, which, in turn, can create other sons. Figure 3-4 shows a process tree of this kind: process A created processes B, C, and D; process B created process F; and process D created processes G and E.

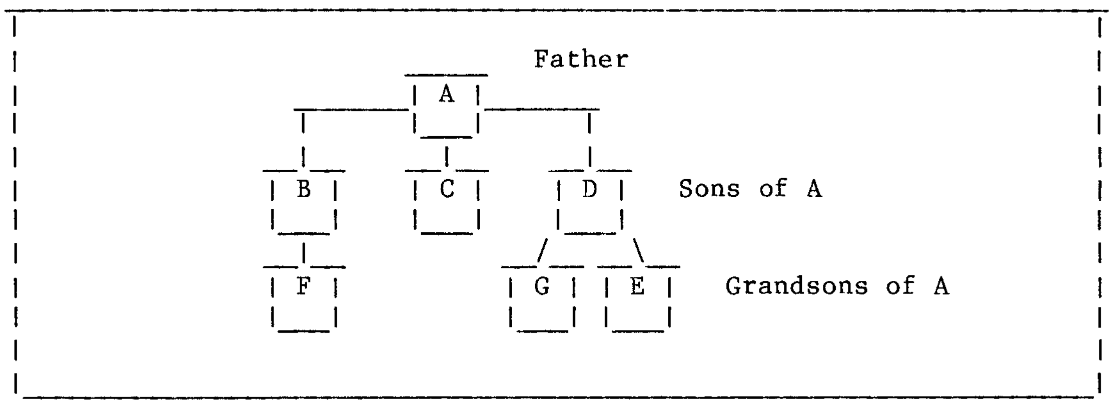

Figure 3-4. Sample Process Tree

Superuser Mode/Superprocess Mode

By default, a process can issue certain system calls only against its subordinates, and can use only those files for which it has the proper access privileges. You can override these restrictions, however, by assigning a process Superuser and Superprocess privileges when you create it.

A process that is in Superuser mode can access any file, regardless of the file's access control list, and can also determine any other process's access to any file. To assign the Superuser privilege to a process, set ?PVSU in offset ?PPRV of the ?PROC packet. The ?SUSER system call turns on Superuser mode.

AOS/VS allows processes that have the Superuser privilege to pass it on to their sons. Moreover, sons created with the Superuser privilege are in Superuser mode at their inception. Each process in Superuser mode remains in that state until it issues a complementary ?SUSER system call to turn off Superuser mode.

A process in Superprocess mode can change the state of any process, not just its subordinate processes, by issuing one of the following system calls:

o ?BLKPR, which blocks a process.

o ?UBLPR, which unblocks a process.

- ?BRKFL, which terminates a process and creates a break file. (See "Break Files" in this chapter for more information on break files.)

- ?CTYPE, which changes a process's type.

- ?GTACP, which gets access control privileges. (See Chapter 4.)

- ?PRIPR, which changes a process's priority.

- ?TERM, which terminates a process.

(See the individual system call descriptions in Chapter 13 for more information on these system calls.)

To assign the Superprocess privilege to a process, set mask ?PVSP in offset ?PPRV of the ?PROC packet. The ?SUPROC system call turns on Superprocess mode for the calling process.

A process with the Superprocess privilege can also pass that privilege to its sons, although sons created with this privilege are not in Superprocess mode initially. A process remains in Superprocess mode until it issues a complementary ?SUPROC system call to explicitly turn off Superprocess mode.

You should restrict the right to enter Superuser and Superprocess modes, because a process in Superuser mode can delete any file, and a process in Superprocess mode can terminate any process.

Process States

When a process has gained memory, it competes for CPU time. At this point, AOS/VS looks at both the priority and state of a process to determine its order of execution. A process is always in one of the following three states:

- Eligible

  A process is eligible for CPU time when it has acquired memory and is ready to run.

- Ineligible

  A process is ineligible when it has not acquired memory, even if it is otherwise ready to run. Every process is ineligible at its inception.

o Blocked

A process is blocked if its execution is suspended to wait for a specific event that may or may not occur. A process can block voluntarily, another process can block it (generally via the ?BLKPR system call), or AOS/VS can block it. (See "Blocking Rules" in this chapter for information on blocking rules.)

Process Scheduling

AOS/VS schedules eligible processes based on their process types and individual priority numbers. To schedule processes, AOS/VS performs the following steps in round-robin fashion:

1. AOS/VS executes the highest priority eligible resident process or eligible pre-emptible process. If there are no processes in these categories, AOS/VS performs Step 2.

2. AOS/VS executes other eligible resident and eligible pre-emptible processes, according to their individual priorities. If there are no processes in these categories, AOS/VS performs Step 3.

3. AOS/VS executes eligible swappable processes according to their individual priorities.

If an executing process cannot proceed, you can issue the ?RESCHED system call, which allows the calling process to give up the remainder of its time slice and force AOS/VS to immediately schedule another process for execution.

Process Blocking

AOS/VS blocks a process under the following conditions:

o When another process explicitly blocks it, using the ?BLKPR system call.

o When the process creates a subordinate process, called a son, and voluntarily blocks itself until the son terminates. (See "Process Creation" in this chapter for information on the process hierarchy.)

o When the process issues a system call that suspends its only active task.

The last condition implies that the process has only one task or that all of its other tasks are suspended. ?IREC and ?WDELAY are two examples of system calls that can cause a process to block. (See Chapter 13 for more information on the ?IREC and ?WDELAY system calls and see Chapter 6 for more information on tasks.)

AOS/VS unblocks a process under the following conditions:

- When the process previously blocked with ?BLKPR is explicitly unblocked with ?UBLPR. (?BLKPR and ?UBLPR work as a pair; ?UBLPR unblocks only those processes that were previously blocked with ?BLKPR.)

- When a son created by the process terminates (provided the father voluntarily blocked to wait for the son to terminate)

- When a task within the process becomes ready to run (AOS/VS blocked the process because it had no ready task)

When memory contention occurs, AOS/VS is more likely to swap blocked processes or to remove pages from them. The processes that have been blocked the longest are the prime candidates for these actions.

Keep in mind that resident processes cannot be explicitly blocked.

Changing Process Priorities

To change a process's own priority, you can issue the ?PRIPR system call. However, if you want to change the priority of another process, the calling process must be in Superprocess mode. (See "Superuser Mode/Superprocess Mode" in this chapter.)

Process Information

You can get information about a process's use of system resources by issuing the ?PNAME, ?RUNTM, ?WHIST, or ?PSTAT system calls.

If you want to know the PID or process name of a process, issue the ?PNAME system call. Often, other system calls require this information as input.

The ?RUNTM system call returns the following information:

- The real time that has elapsed since process creation (in seconds, within the range 0 through (2**32)-1).
- The CPU time that the process used (in milliseconds).
- The number of blocks read or written.
- The page usage over a period of time (in page-seconds). AOS/VS calculates page-seconds by multiplying CPU usage by main memory usage.

The ?WHIST system call generates a histogram. A histogram is a data array that provides a global view of CPU activity. To issue the ?WHIST system call, a process must be resident. Also, you can activate one histogram at a time. To terminate a histogram, the process msut issue the ?KHIST system call.

Each histogram shows how often CPU control passes to the target process and, optionally, at what points. It also shows how often control passed to other processes, including system processes. In addition, a histogram records the amount of time the system remained idle, waiting for a process to become eligible for execution.

AOS/VS updates the histogram statistics after each "tick," or real-time clock pulse.

Note that the ?WHIST system call does not zero out existing histograms in a data array. This allows you to stop a histogram and restart it without losing data. Thus, unless you want to aggregate data, you should explicitly reset the array to zero before you use it for another histogram.

The ?PSTAT system call returns internal statistics about a process and performance information about all programs that are currently executing.

Execute-Protection Status

To make it easier to find errors in your code, you may want to prevent your program from executing certain logical pages, such as pages that contain data. Therefore, AOS/VS provides execute protection. The ?EXPO system call allows you to set, clear, or examine a process's execute-protection status.

Process Traps

A process trap is a hardware error. Each process exists until it terminates voluntarily, becomes terminated by another process, or encounters a process trap (that is, "traps"). Any one of the following conditions can cause a process to trap:

- The process tries to reference an address that is outside its logical address space or refers to an invalid address within Ring 7.
- The process tries to use more than 16 levels of indirection in a memory reference instruction.
- The process tries to read, write, or execute code that is protected against any of these actions (for example, it attempts to write to the write-protected shared area of its logical address space).
- The process uses I/O instructions while LEF is disabled and I/O protection is enabled.
- A process tries to execute a privileged instruction in a user ring.

When a process traps or terminates voluntarily, AOS/VS uses the IPC facility to send that process's father a termination message. If the process terminated on a trap, the IPC message describes the cause. (See Chapter 7 for more information on termination messages.)

Break Files and Memory Dumps

When a process terminates, you can save the state of certain memory parameters and tables (for example, the process's UST and TCBs) in two ways:

- You can create a break file.

  A break file is a status file in the terminated process's working directory that contains this information. You must be logged on to examine a break file.

- You can dump the contents of a particular ring to a dump file.

  A dump file contains all of the information that a break file contains, plus a copy of the memory image. Also, you do not have to be logged on to examine a dump file.

  To perform a dump, issue the ?MDUMP system call, which creates a dump file wherever you specify.

There are two ways to terminate a process and explicitly create a break file:

- Issue the ?BRKFL system call.

- Type a CTRL-C CTRL-E sequence from the process console. (See Chapter 5 for a full description of console control characters and control sequences.)

To create a break file every time a process traps, set bit ?PBRK in offset ?PFLG of the process's ?PROC packet. (See the description of the ?PROC system call in Chapter 13 for more information on the ?PROC packet.)

AOS/VS copies the following words to the break file:

Status Word	Contents
?BRAC0	Value of AC0
?BRAC1	Value of AC1
?BRAC2	Value of AC2
?BRAC3	Value of AC3
?BRPC	Value of the program counter (PC)
?BRTID	TID
?BRFP	Value of the stack frame pointer
?BRSP	Value of the stack pointer
?BRSL	Value of the stack limit
?BRSB	Value of the stack base

(Refer to the current AOS/VS Release Notice for more information on the contents of a break file.)

Unless you specify another pathname, AOS/VS assigns the break file the default pathname:

?pid.time.BRK

where:

- pid is the 3-digit PID of the terminated process
- time is the time of the termination, in the form hours_minutes_seconds

AOS/VS only creates a break file if the terminated process has Write or Append access to its working directory and if the working directory has enough disk space for the break file.

The ?ENBRK system call, unlike the ?BRKFL system call, which terminates a process and creates a break file, does not terminate the process. Instead, if the process traps, issues a CTRL-C CTRL-E, or is the target of a TERM/BREAK, the ?ENBRK system call allows AOS/VS to create a break file of whatever user ring you specified as its target ring. The ?ENBRK system call allows AOS/VS to create a break file, it does not explicitly direct it to do so.

Linking Programs Together with the ?CHAIN System Call

The ?CHAIN system call allows you to link together several steps of a long, complex program set, where each program is a separate program file. The ?CHAIN system call actually releases the system resources that one process is using, and then executes a new program. In addition, the ?CHAIN system call transfers the following attributes to the new program:

- The username, process name, PID, console, search list, default ACL, and working directory of the calling process.
- The generic file associations of the calling process (for example, the filenames associated with the generic files @INPUT, @OUTPUT, @LIST, and @DATA).
- The privileges, process type, and priority of the calling process.

When a process chains to a new program, AOS/VS performs the following steps:

1. Unloads all of the process's inner user rings.

2. Terminates all son processes that were previously created by ?PROC system calls issued from the inner user rings.

3. Breaks the connection, which, in turn, causes AOS/VS to revoke access privileges to protected shared files.

Inner Rings

To load program files into a specific rings, you can issue the ?RINGLD system call. Then, to find out what program was loaded into the ring, you can issue the ?RNGPR system call. If you want to prevent the ?RINGLD system call from loading a runtime routine into a particular ring, you can issue the ?RINGST sytem call. (See Chapter 2 for more information on the ring structure.)

To cross from an outer ring to an inner ring, a program must have access to the proper gates; that is, entry points to the code in the inner ring. When you write a program to execute in Rings 4, 5, or 6, you must define an array of the legal entry points (gates).

In the module in which you define your gate array, you must declare the gate entry points as .EXTG (external gate). Also, in your source module, you must declare your gate entry points as .ENT (entry point). (See the GATE.ARRAY sample program in Chapter 5 for an example of using the .EXTG pseudo-op.) The 'Principles of Operation ECLIPSE 32-Bit Systems' manual explains how to reference gates and how to set up gate arrays.

Figure 3-5 shows how a process can span rings. For the purpose of the figure, assume that the main program has used the ?RINGLD system call to load a program file into Ring 6.

```
 ---------------------------------------------------------------------
|                                                                    |
|                                                                    |
|          __________________________________                        |
|           |                  Main Program      Access from         |
|           |  -------"--------------------      Ring 6 through      |
|           |  | Program loaded with ?RINGLD     gate in Ring 6      |
|           |  |  --------"-------------------                       |
|           |  |  |  -----------------------                         |
|           |  |  |  |  ------"--------------                        |
|           |  =  |  |  |  -----------------                         |
|           |  |  |  =  |  |  ---"-------------     Full access from |
|           |  |  |  |  |  =  |  -----------        inner rings to   |
|           | 7| 6| 5| 4| 3| 2| 1=    0             outer rings      |
|           |  |  |  |  |  |  =  |_______                            |
|    Gate   |  |  |  |  =  |  |_______________                       |
|    ------------> =  |  |  |_______________                         |
|           |  |  |  |  |_______________                             |
|           |  |  |  |____________________                           |
|           |  |  |___________________________                       |
|           |  |____________________________                         |
|           |______________________________                          |
|                                                                    |
|____________________________________________________________________|
```

Figure 3-5. Ring Structure

Process and Memory Sample Programs

The following subroutine, SON, creates a swappable son process. The son process runs program SPEAK.PR which is an IPC sample program. (See Chapter 7.)

To use the SON subroutine, you must have the Create Without Block privilege in your user profile.

```
                        .TITLE   SON
                        .ENT     SON
                        .NREL                    ;Default partition 4.

;Get program name to ?PROC:

SON:    WSSVS   0                                ;Save return from XJSR.
        XLEFB   0,PRGNM*2                        ;Byte pointer to the program
                                                 ;name.
        XWSTA   0,PKT+?PSNM                      ;Put in ?PROC packet.
        ?PROC   PKT                              ;Create process.
        WBR     ERROR                            ;Report error and quit.
        WRTN                                     ;Return to caller.

PRGNM:  .TXT    "SPEAK.PR"

ERROR:  WLDAI   ?RFEC!?RFCF!?RFER,2              ;Error flags: Error code is in
                                                 ;AC0 (?RFEC), message is in
                                                 ;CLI format (?RFCF), and
                                                 ;father should handle this as
                                                 ;an error (?RFER).
        ?RETURN                                  ;Return to CLI.
        WBR     ERROR                            ;Report error and quit.

;?PROC packet:

PKT:    .BLK    ?PLTH                            ;Allocate enough space for
                                                 ;packet.
        .LOC    PKT+?PFLG
        .WORD   0                                ;Default process creation
                                                 ;specifications.  (See the
                                                 ;description of ?PROC in
                                                 ;Chapter 13.)

        .LOC    PKT+?PPRI
        .WORD   -1                               ;Default priority of son
                                                 ;process to same as father.
```

SON Subroutine (Cont.)

```
          .LOC     PKT+?PSNM
          .DWORD   PRGNM*2               ;Byte pointer to pathname of
                                         ;program file for son to
                                         ;execute.

          .LOC     PKT+?PIPC
          .DWORD   -1                    ;No IPC message header to
                                         ;send to son (default is -1).

          .LOC     PKT+?PNM
          .DWORD   -1                    ;Default son's simple process
                                         ;name to ASCII representation
                                         ;of its PID.

          .LOC     PKT+?PMEM
          .DWORD   -1                    ;Default maximum number of
                                         ;son's logical pages to same
                                         ;as father.

          .LOC     PKT+?PDIR
          .DWORD   -1                    ;Default name of son's working
                                         ;directory to same as father

          .LOC     PKT+?PCON
          .DWORD   0                     ;Default name of son's
                                         ;@CONSOLE device to same as
                                         ;father

          .LOC     PKT+?PCAL
          .WORD    -1                    ;Default number of system
                                         ;calls son can issue
                                         ;concurrently is two.

          .LOC     PKT+?PWSS
          .WORD    -1                    ;Default son's maximum working
                                         ;set size to no limit.

          .LOC     PKT+?PUNM             ;Byte pointer to son's
                                         ;username.
          .DWORD   -1                    ;Default son's username to
                                         ;same as father.

          .LOC     PKT+?PPRV             ;Son's privileges.
          .WORD    ?PVIP                 ;Son can issue ?ISEND and
                                         ;?IS.R.

          .LOC     PKT+?PPCR
          .WORD    0                     ;Son can create no sons.
```

SON Subroutine (Cont.)

```
        .LOC    PKT+?PWMI
        .WORD   -1                      ;Default son's minimum working
                                        ;set size to no minimum.

        .LOC    PKT+?PIPF
        .DWORD  0                       ;Son has no @INPUT file.

        .LOC    PKT+?POFP
        .DWORD  0                       ;Son has no @OUTPUT file.

        .LOC    PKT+?PLFP
        .DWORD  0                       ;Son has no @LIST file.

        .LOC    PKT+?PDFP
        .DWORD  0                       ;Son has no @DATA file.

        .LOC    PKT+?SMCH
        .DWORD  -1                      ;Default maximum CPU time
                                        ;allotted for son to remainder
                                        ;of father's time limit.

        .LOC    PKT+?PLTH               ;End of packet.

        .END    SON                     ;End of SON program.
```

The following program, RUNTIME, gets its own runtime statistics and displays these statistics on the console.

First, RUNTIME opens the console, then it issues the ?RUNTM system call, and finally, it converts the runtime statistics to ASCII decimal values and displays them on the console. Although RUNTIME gets its own runtime statistics, you can use it to get any process's runtime statistics by passing the process's filename.PR or the process's PID. To use RUNTIME as a subroutine, start with a proper save and end with a proper return.

```
                        .TITLE  RUNTIME
                        .ENT    RUNTIME, CONVERT
                        .NREL

;Open console for I/O.

RUNTIME:?OPEN   CON                     ;Open console (CON) for I/O.
        WBR     ERROR                   ;Report error and quit.

        ?WRITE  CON                     ;Display message on console.
        WBR     ERROR                   ;Report error and quit.

;Call ?RUNTM to get statistics.

LOOP:   WLDAI   -1,0                    ;Check self.
        ?RUNTM  RPKT                    ;Get statistics in RPKT.
        WBR     ERROR                   ;Report error and quit.
        XWLDA   1,MSECS                 ;Get time in milliseconds from
                                        ;RPKT.

        XLEFB   2,MSECMSG*2             ;Byte address of message that
                                        ;describes milliseconds
                                        ;elapsed.
        XJSR    CONVERT                 ;Convert milliseconds elapsed
                                        ;to ASCII decimal and put
                                        ;converted value in
                                        ;milliseconds elapsed message.

        XLEFB   0,MSECMSG*2             ;Get byte pointer to
                                        ;milliseconds elapsed message.
        XWSTA   0,CON+?IBAD             ;Put milliseconds elapsed
                                        ;message in I/O packet.

        ?WRITE  CON                     ;Display milliseconds elapsed
                                        ;message on console.
        WBR     ERROR                   ;Report error and quit.
```

RUNTIME Program (Cont.)

```
          XWLDA    1,PSECS                  ;Get page-seconds from RPKT.
          XLEFB    2,PSECMSG*2              ;Byte address of message that
                                            ;describes page-seconds
                                            ;elapsed.

          XJSR     CONVERT                  ;Convert page-seconds elapsed
                                            ;to ASCII decimal and put
                                            ;converted value in
                                            ;page-seconds elapsed message.
          XLEFB    0,PSECMSG*2              ;Get byte pointer to
                                            ;page-seconds elapsed message.

          XWSTA    0,CON+?IBAD              ;Put page-seconds elapsed
                                            ;message in I/O packet.

          ?WRITE   CON                      ;Display page-seconds elapsed
                                            ;message on console.
          WBR      ERROR                    ;Report error and quit.

   ;See if user wants to stop.

          XLEFB    0,BUF*2                  ;Get byte pointer to I/O
                                            ;buffer.
          XWSTA    0,CON+?IBAD              ;Put in I/O packet.

          ?READ    CON                      ;Look for terminator.
          WBR      ERROR                    ;Report error and quit.

          NLDAI    'ST',0                   ;Put ST in AC0.
          XNLDA    1,BUF                    ;Put first word of buffer in
                                            ;AC1.

          WSNE     0,1                      ;Skip next if first word is
                                            ;not ST.
          WBR      BYE                      ;If first word is ST, go to
                                            ;BYE.
          WBR      LOOP                     ;If first word is not ST, do
                                            ;LOOP again.

   ;Error handler and return.

   ERROR: NLDAI    ?RFEC!?RFCF!?RFER,2      ;Error flags: Error code is
                                            ;in AC0 (?RFEC), message is in
                                            ;CLI format (?RFCF), and
                                            ;father should handle this
                                            ;as an error (?RFER).
```

RUNTIME Program (Cont.)

```
BYE:    WSUB    2,2                  ;Good return flags.
        ?RETURN                      ;Return to father.
        WBR     ERROR                ;?RETURN error return.

;Open and I/O packet for console.

CON:    .BLK    ?IBLT                ;Allocate enough space for
                                     ;packet.

        .LOC    CON+?ISTI            ;File specifications.
        .WORD   ?ICRF!?RTDS!?OFIO    ;Change format to data-
                                     ;sensitive records and open
                                     ;for input and output.

        .LOC    CON+?IMRS
        .WORD   -1                   ;Default physical block size
                                     ;to 2K bytes.

        .LOC    CON+?IBAD
        .DWORD  ITEXT*2              ;Byte pointer to record I/O
                                     ;buffer.

        .LOC    CON+?IRCL
        .WORD   120.                 ;Record length is 120
                                     ;characters.

        .LOC    CON+?IFNP
        .DWORD  CONS*2               ;Byte pointer to pathname.

        .LOC    CON+?IDEL            ;Delimiter table address.
        .DWORD  -1                   ;Use default delimiters: null,
                                     ;NEW LINE, form feed, and
                                     ;carriage return (default is
                                     ;-1).

        .LOC    CON+?IBLT            ;End of packet.

;Filename, start message, and buffer.  A .NOLOC 1 follows.

CONS:   .TXT    "@CONSOLE"           ;Use generic name.

ITEXT:  .TXT    "I give runtime statistics on a process.
                Type ST[NL] to return to father.<212><12>"

BUF:    .BLK    (BUF-CONS)*2)        ;Use number of bytes in
                                     ;message.
        .NOLOC  0                    ;Resume listing all.
```

RUNTIME Program (Cont.)

```
   ;Messages to include converted statistics. .NOLOC here.

   MSECMSG: .TXT    "       milliseconds elapsed.<12>"
   PSECMSG: .TXT    "       page-seconds elapsed.  Type
            ST[NL] to stop, type another character to loop.<212><12>"

            .NOLOC  0

   ;?RUNTM packet.

   RPKT:    .BLK    ?GRLTH                        ;Allocate enough space for
                                                  ;packet.

   SECS:    .LOC    RPKT+?GRRH
            .DWORD  0                             ;AOS/VS returns elapsed time
                                                  ;in seconds.

   MSECS:   .LOC    RPKT+?GRCH
            .DWORD  0                             ;AOS/VS returns elapsed CPU
                                                  ;time in milliseconds.
    ;See if user wants to stop.

            XLEFB   0,BUF*2                       ;Get byte pointer to I/O
                                                  ;buffer.
            XWSTA   0,CON+?IBAD                   ;Put in I/O packet.

            ?READ   CON                           ;Look for terminator.
            WBR     ERROR                         ;Report error and quit.

            NLDAI   'ST',0                        ;Put ST in AC0.
            XNLDA   1,BUF                         ;Put first word of buffer in
                                                  ;AC1.

            WSNE    0,1                           ;Skip next if first word is
                                                  ;not ST.

            WBR     BYE                           ;If first word is ST, go to
                                                  ;BYE.
            WBR     LOOP                          ;If first word is not ST, do
                                                  ;LOOP again.

   ;Error handler and return.

   ERROR:   NLDAI   ?RFEC!?RFCF!?RFER,2           ;Error flags: Error code is
                                                  ;in AC0 (?RFEC), message is in
                                                  ;CLI format (?RFCF), and
                                                  ;father should handle this
                                                  ;as an error (?RFER).
```

RUNTIME Program (Cont.)

```
BYE:    WSUB    2,2                     ;Good return flags.
        ?RETURN                         ;Return to father.
        WBR     ERROR                   ;?RETURN error return.

;Open and I/O packet for console.

CON:    .BLK    ?IBLT                   ;Allocate enough space for
                                        ;packet.

        .LOC    CON+?ISTI               ;File specifications.
        .WORD   ?ICRF!?RTDS!?OFIO       ;Change format to data-
                                        ;sensitive records and open
                                        ;for input and output.

        .LOC    CON+?IMRS
        .WORD   -1                      ;Default physical block size
                                        ;to 2K bytes.

        .LOC    CON+?IBAD
        .DWORD  ITEXT*2                 ;Byte pointer to record I/O
                                        ;buffer.

        .LOC    CON+?IRCL
        .WORD   120.                    ;Record length is 120
                                        ;characters.

        .LOC    CON+?IFNP
        .DWORD  CONS*2                  ;Byte pointer to pathname.

        .LOC    CON+?IDEL               ;Delimiter table address.
        .DWORD  -1                      ;Use default delimiters: null,
                                        ;NEW LINE, form feed, and
                                        ;carriage return (default is
                                        ;-1).

        .LOC    CON+?IBLT               ;End of packet.

;Filename, start message, and buffer.  A .NOLOC 1 follows.

CONS:   .TXT    "@CONSOLE"              ;Use generic name.

ITEXT:  .TXT    "I give runtime statistics on a process.
                Type ST[NL] to return to father.<212><12>"

BUF:    .BLK    (BUF-CONS)*2)           ;Use number of bytes in
                                        ;message.
        .NOLOC  0                       ;Resume listing all.
```

RUNTIME Program (Cont.)

```
;Messages to include converted statistics. .NOLOC here.

MSECMSG: .TXT   "       milliseconds elapsed.<12>"
PSECMSG: .TXT   "       page-seconds elapsed.  Type
         ST[NL] to stop, type another character to loop.<212><12>"

        .NOLOC  0

;?RUNTM packet.

RPKT:   .BLK    ?GRLTH                          ;Allocate enough space for
                                                ;packet.

SECS:   .LOC    RPKT+?GRRH
        .DWORD  0                               ;AOS/VS returns elapsed time
                                                ;in seconds.

MSECS:  .LOC    RPKT+?GRCH
        .DWORD  0                               ;AOS/VS returns elapsed CPU
                                                ;time in milliseconds.

IO:     .LOC    RPKT+?GRIH
        .DWORD  0                               ;AOS/VS returns number of
                                                ;blocks read or written.
PSECS:  .LOC    RPKT+?GRPH                      ;Page usage over elapsed CPU
                                                ;time.
        .DWORD  0                               ;AOS/VS returns page usage
                                                ;over elapsed CPU time in
                                                ;page-seconds.

        .LOC    CON+?GRLTH                      ;End of packet.

;CONVERT routine converts binary value into its decimal equivalent and
;puts it in a text string.  AC1 contains the value and AC2 contains
;the byte address of the text message.

CONVERT: WSSVS  0                               ;Save return.
        WMOV    2,3                             ;Use AC3 to shift byte pointer.

     WADI    3,3                          ;Add integer 3 to byte address.
     NLDAI   10.,2                        ;Put 10 in AC2
```

RUNTIME Program (Cont.)

```
DLOOP:  WSUB    0,0                     ;Zero AC0 (high-order portion
                                        ;of dividend).  AC1 still
                                        ;contains low-order portion of
                                        ;dividend.
        WDIVS                           ;Divide by 10, put quotient in
                                        ;AC1, and put remainder in AC0.

        IORI    60,0                    ;OR in 60 for ASCII number.
        WSTB    3,0                     ;Store AC0 byte in byte
                                        ;address of AC3.

        WSBI    1,3                     ;Decrement byte address.
        MOV     1,1,SNR                 ;Is quotient 0?
        WRTN                            ;If quotient is 0, return to
                                        ;caller.

        WBR     DLOOP                   ;If quotient is not 0, do
                                        ;another digit.

        .END    RUNTIME                 ;End of RUNTIME program.
```

The following program, RINGLOAD, loads program INRING into an inner ring. Then, RINGLOAD uses an LCALL instruction to call INRING. RINGLOAD assumes that file INRING.PR, which was linked from INRING and GATE.ARRAY exists. (GATE.ARRAY is at the end of this section.)

```
                        .TITLE  RINGLOAD
                        .ENT    RINGLOAD
                        .NREL

;Open console for I/O and issue ?RINGLD to load program into inner
;ring.

RINGLOAD: ?OPEN CON                       ;Open CON (console) for I/O.
          WBR     ERROR                   ;?OPEN error return.

          ?WRITE  CON                     ;Display message on console.
          WBR     ERROR                   ;?WRITE error return.

          XLEFB   0,PNAME*2               ;Get byte pointer to INRING
                                          ;name.

          ?RINGLD                         ;Load INRING.
          WBR     ERROR                   ;?RINGLD error return.

          LCALL   INRING,0,0              ;Call INRING and set the index
                                          ;and argument count to 0.
          WBR     INERROR                 ;Report INRING error.

;Back from INRING.  Depart with message for CLI.

          XLEFB   0,MES2*2                ;Get byte pointer to farewell
                                          ;message.

          XWSTA   0,CON+?IBAD             ;Put in I/O packet.

          ?WRITE  CON                     ;Display message on console.
          WBR     ERROR                   ;?WRITE error return.

          WSUB    2,2                     ;Set return flags for normal
                                          ;return.
          WBR     BYE                     ;Done.  Give message and
                                          ;depart.

;Inner-ring program and current program error handlers.

INERROR:  LLDFB   0,INMES                 ;Get inner-ring program error
                                          ;message.
          LWSTA   0,CON+?IBAD             ;Put in I/O packet.

          ?WRITE  CON                     ;Display message on console.
          WBR     ERROR                   ;?WRITE error message.
```

RINGLOAD Program (Cont.)

```
ERROR:  WLDAI   ?RFEC!?RFCF!?RFER,2         ;Error flags: Error code is in
                                            ;ACO (?RFEC), message is in
                                            ;CLI format (?RFCF), and
                                            ;father should handle this as
                                            ;an error (?RFER).

BYE:    ?RETURN                             ;Return to CLI.
        WBR     ERROR                       ;?RETURN error return.

;Definition of inner-ring program ring bracket and gate.  AOS/VS uses
;this, instead of the program name (INRING) to access gate and
;inner-ring program messages.  A .NOLOC 1 follows.

        INRING = 5S3+0                      ;Ring 5 + first gate.

PNAME:  .TXT    "INRING.PR"                 ;Program name is INRING.

MES1:   .TXT    "I'm RINGLOAD.  I am about to ?RINGLOAD program
                INRING.<12>"

MES2:   .TXT    "<212>I'm RINGLOAD.  I'm back from INRING, and I'm
                terminating.<12>"

INMES:  .TXT    "<212>ERROR IN INNER-RING PROGRAM.<12>"
        .NOLOC  0                           ;Resume listing all.

;Open and I/O packet.  (You need this packet for I/O.)

CON:    .BLK    ?IBLT                       ;Allocate enough space for
                                            ;packet.

        .LOC    CON+?ISTI                   ;File specifications.
        .WORD   ?ICRF!?RTDS!?OFIO           ;Change format to data-
                                            ;sensitive records and open
                                            ;for input and output.

        .LOC    CON+?IMRS
        .WORD   -1                          ;Default physical block size
                                            ;to 2K bytes.

        .LOC    CON+?IBAD
        .DWORD  MES1*2                      ;Byte pointer to record I/O
                                            ;buffer.

        .LOC    CON+?IRCL
        .WORD   120.                        ;Record length is 120.
                                            ;characters.
```

RINGLOAD Program (Cont.)

```
        .LOC    CONS+?IFNP
        .DWORD  CONS*2                      ;Byte pointer to pathname.

        .LOC    CON+?IDEL                   ;Delimiter table address.
        .DWORD  -1                          ;Use default delimiters: null,
                                            ;NEW LINE, form feed, and
                                            ;carriage return (default is
                                            ;-1).

        .LOC    CON+?IBLT                   ;End of packet.

CONS:   .TXT    "@CONSOLE"                  ;Use generic name.

        .END    RINGLOAD                    ;End of RINGLOAD program.
```

Program RINGLOAD loads the following program, INRING, into Ring 5. Then, RINGLOAD uses an LCALL instruction to call INRING.

INRING saves the return address, opens the console, writes messages, and invokes the Debugger (so you can explore the inner ring). To return to RINGLOAD in Ring 7, type ESC R.

Except for the I/O packet, all code in INRING is shared.

You must link INRING with GATE.ARRAY (the last program in this section). Depending on the LCALL name definition in GATE.ARRAY, the link that the gate defines in GATE.ARRAY, and the Link switch that you use, you can execute INRING in Rings 4, 5, or 6. In this case, RINGLOAD defined the LCALL name as Gate 5 (5S3), GATE.ARRAY defined Gate 5, and the Link command line was X LINK/RING=5 INRING GATE.ARRAY.

```
                         .TITLE  INRING
                         .ENT    INRING
                         .NREL   1
                            .EXTL   GATE.ARRAY

;Define a 2-word pointer to the gate array.

        .LOC     34                        ;Locations 34 and 35.
        .DWORD   GATE.ARRAY                ;Pointer.

;Save the return, open the console, write the message, and enter the
;debugger.

INRING: WSAVR    0                         ;Save frame (ACs, PC in AC3).
        ?OPEN    CON                       ;Open console (CON) for I/O.
        WBR      ERTN                      ;?OPEN error return.

        ?WRITE   CON                       ;Display message from Ring 5
                                           ;on console.
        WBR      ERTN                      ;Report error and quit.

        ?DEBUG                             ;Enter debugger.
        WBR      ERTN                      ;Report error and quit.

        LLEFB    0,MES2*2                  ;Get byte pointer to return
                                           ;message.
        LWSTA    0,CON+?IBAD               ;Put in I/O packet.

        ?WRITE   CON                       ;Display another message on
                                           ;console.
        WBR      ERTN                      ;?WRITE error return.
```

INRING Program (Cont.)

```
   ;Done.  Ready for good return to caller.

           LDAFP    3                      ;Get frame pointer in AC3.
           XWISZ    0,3                    ;Increment return address for
                                           ;normal return to LCALLer.
           WRTN                            ;Return to caller.

   ;INRING error handler.  Returns error to outer-ring caller, not CLI.

   ERTN:   LDAFP    3                      ;Get frame pointer in AC3.
           LWSTA    0,?OAC0,3              ;Put error code (AC0) in saved
                                           ;frame's AC0.
           WRTN                            ;Return to LCALLer's error return.

   ;Text messages.  .NOLOC 1 follows.

   MES1:   .TXT     "I'm INRING. I'm in the inner ring. I'm about to debug.
                    Type ESC R to proceed.<212><12>"

   MES2:   .TXT     "<212><212>From INRING.  I'm about to WRTN.<12>"

           .NOLOC   0                      ;Resume listing all.

           .NREL                           ;Use unshared code for packet,
                                           ;because program and AOS/VS
                                           ;write into it.

   ;Open I/O packet for @CONSOLE.

   CON:    .BLK     ?IBLT                  ;Allocate enough space for
                                           ;packet.

           .LOC     CON+?ISTI              ;File specifications.
           .WORD    ?ICRF!?RTDS!?IFIO      ;Change format to data-
                                           ;sensitive records and open
                                           ;for input and output.

           .LOC     CON+?IMRS
           .WORD    -1                     ;Default physical block size
                                           ;to 2K bytes.

           .LOC     CON+?IBAD
           .DWORD   MES1*2                 ;Byte pointer to record I/O
                                           ;buffer.

           .LOC     CON+?IRCL
           .WORD    120.                   ;Record length is 120
                                           ;characters.
```

INRING Program (Cont.)

```
        .LOC    CON+?IFNP
        .DWORD  CONS*2                  ;Byte pointer to pathname.

        .LOC    CON+?IDEL               ;Delimiter table address.
        .DWORD  -1                      ;Use default delimiters: null,
                                        ;NEW LINE, form feed, and
                                        ;carriage return (default is
                                        ;-1).

        .LOC    CON+?IBLT               ;End of packet.

;Filename.  A .NOLOC 1 follows:

CON:    .TXT    "@CONSOLE"              ;Use generic name.
        .NOLOC  0                       ;Resume listing all.

        .END    INRING                  ;End of INRING program.
```

The following program, GATE.ARRAY, defines the gate array. Generally, you must define the gate array in a separate module. A gate array module must contain .EXTG PROG-ENTRY-NAME, where PROG-ENTRY-NAME is the start entry name in the program that will be accessed through the gate. Also, you must link the gate array module with the inner-ring program. In this case, the Link command line is:

```
X LINK/RING=5 INRING GATE.ARRAY
```

```
                        .TITLE  GATE.ARRAY
                        .EXTG   INRING
                        .ENT    GATE.ARRAY

        .NREL    1                          ;Shared code for general use.
        .ENABLE ABS

GATE.ARRAY:     .DWORD  1                   ;Gate array, one gate.
        .DWORD  (RING7-RING5)!INRING        ;LINK will determine the
                                            ;address.  A program in any
                                            ;ring can access the gate.

RING7 = 7S3                                 ;Bits 1 3 specify Gate 7.
RING5 = 5S3                                 ;Bits 1 3 specify Gate 5.

        .END    GATE.ARRAY                  ;End of GATE.ARRAY program.
```

End of Chapter

CHAPTER 4
FILE CREATION AND MANAGEMENT

The file creation and management system calls are:

?CGNAM	Gets a complete pathname from a channel number.
?CPMAX	Sets maximum size for a control point directory (CPD).
?CREATE	Create a file or directory.
?DACL	Sets, clears, or examines a default access control list.
?DELETE	Deletes a file entry.
?DIR	Changes the working directory.
?FSTAT	Gets file status information.
?GACL	Gets a file entry's access control list.
?GLINK	Gets the contents of a line entry.
?GLIST	Gets the contents of a search list.
?GNAME	Gets a complete pathname.
?GRNAME	Returns complete pathname of generic file.
?GNFN	Lists a particular directory's entries.
?GTACP	Gets access control privileges.
?INIT	Initializes a logical disk.
?RECREATE	Recreates a file.
?RELEASE	Releases an initialized logical disk (LD).
?RENAME	Renames a file.
?SACL	Sets a new access control list.
?SATR	Sets or removes attributes for a file or directory.
?SLIST	Sets the search list.

The previous chapter describes your program's image as it executes under AOS/VS. This chapter describes the AOS/VS file structure and the system calls you use to create and maintain files and directories.

A file is a collection of related data that is treated as a unit. "File" also refers to the disk blocks used to store files. Each file has a filename by which you and AOS/VS address that file. You can create files and assign them filenames by using the ?CREATE system call, the CLI, or one of the text editors AOS/VS supports. Or, you can create files as you assemble, compile, and link your source code. In the latter case, the utilities assign the filenames.

There are two general types of devices that allow you to store and retrieve file information. You can use multifile devices, such as disks and magnetic tape, to perform file I/O and to store and retrieve files. Other devices, such as consoles, you can use strictly for file I/O.

Disk File Structures

Each file consists of one or more file elements. A file element is a set of contiguous 512-byte disk blocks. (Contiguous disk blocks are blocks with sequential addresses). The default file-element size is four (four disk blocks per element), or whatever file-element size you selected during the system-generation procedure. (Refer to the 'Managing AOS/VS' manual for more information on the system-generation procedure.) You can also specify a file-element size when you create a file.

AOS/VS always rounds file-element sizes to the next higher multiple of the default file-element size. For example, if you create a file with a file-element size of five and the default file-element size is four, AOS/VS rounds the file-element size to eight.

AOS/VS allocates disk space to a file based on its file-element size. For example, a file with a file-element size of four "grows" in units of four contiguous blocks.

The blocks that make up a file element are always contiguous, although the file elements may not be. For example, a file with a file-element size of four may consist of a number of "scattered" 4-block elements.

To keep track of each file's file elements, AOS/VS maintains one or more index levels for each disk file. An index is a single block that lists the address of each file element. As a file exhausts one index, AOS/VS provides a superior index, to a maximum of three index levels. A pointer in each index level links that level with its immediate subordinate. Figure 4-1 shows typical growth stages for a file with a file-element size of four.

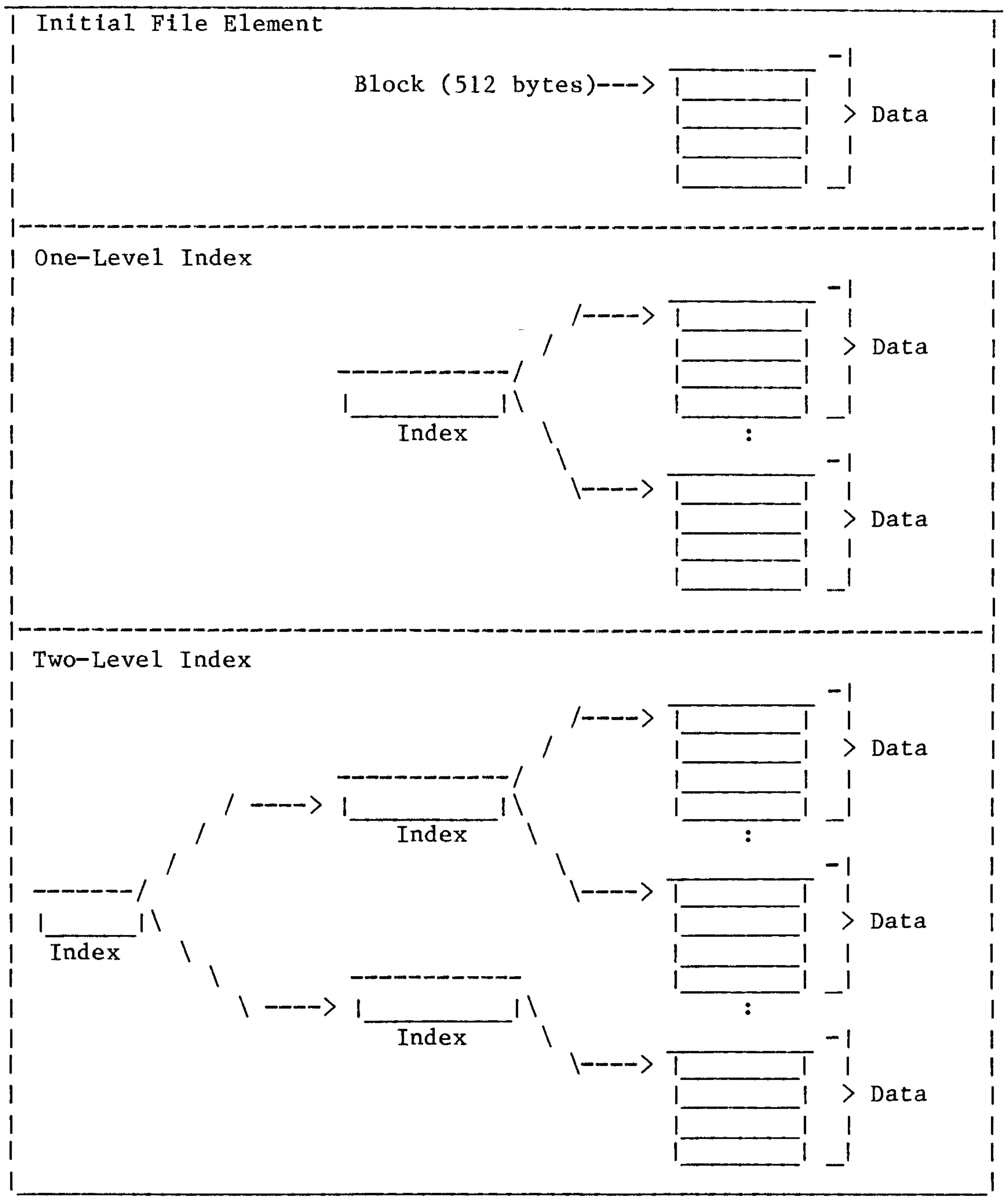

Figure 4-1. File Growth Stages

Files with larger file-element sizes have fewer separate elements and, therefore, require fewer index levels. Files with smaller file-element sizes are easier to store, however, because each block in a file element must be contiguous. (It is easier for AOS/VS to find eight contiguous blocks, for example, than to find 500.)

The maximum size for a disk file is 2**23 blocks. You cannot use all the blocks in the total disk storage, however, because AOS/VS must reserve some for index blocks, to store disk bootstraps, and for other purposes.

Directory Creation

Generally, you group related disk files into directories for convenience. A directory is a file that contains information about a particular set of files. For example, you might create a directory called PL_1 to group all PL/1 source files, or a directory called UPD to contain all user profiles. The AOS/VS filename conventions also apply to directory names.

AOS/VS organizes directories into a hierarchical tree structure similar to the process tree structure. (See Figure 4-2.) The initial directory, called the root, is superior to all others in the hierarchy. A colon (:) represents the root.

Directory Entries

Each directory contains a directory entry for every one of its subordinate files. A typical directory entry contains the name of the file, its file type, a list of the access privileges for various users, and other information unique to the file type. For example, a directory entry for an IPC file contains such additional information as the PID of the process that created the file and the file's local port number. AOS/VS recognizes 256 different types of directory entries, numbered from 0 through 255.

Data General reserves types 0 through 127; the user parameter files PARU.32 and PARU.16 define these types. Users can define directory entry types 128 through 255.

```
                                           : Root
                                          /|\
                                        /  |  \
                                      /    |    \
                                 UTIL    UDD   UPD
                                 /\        |     \
                                /  \       |      \
Directory----------------> LANG  SPEED  COMMON    BIFF
                            /\    SED      |       IAN
                          /    \           |      FUMBLE
                        /        \     DFSHEET       .
                      /            \    PROGS        .
Subdirectory----> PL_1             DG_L    .         .
                    |                |     .
                    |                |     .
File Entries----> PL1.PR          DGL.PR
                 PL1.TEMP         DGL.ST
                    .                .
                    .                .
                    .                .
```

Figure 4-2. Sample Directory Tree

File Types

A file's characteristics and function determine its file type. Table 4-1 lists the AOS/VS file types.

User data files (file type ?FUDF) are not executable files. Typically, you use ?FUDF files to store the object files or text files you create with one of the text editors.

As Table 4-1 indicates, there are two types of program files:

o ?FPRV files, which are developed under AOS/VS.

o ?FPRG files, which are developed under AOS.

Table 4-1. File Types

Mnemonic	Type	Comments
?FUDF	User Data File	Usually applies to source or object files.
?FTXT	Text File	Should contain ASCII text.
?FPRG	AOS Program File	Program file for use under AOS (16-bit code).
?FPRV	AOS/VS Program File	Program file for use under AOS/VS (32-bit code).
?FDIR	Disk Directory	None.
?FCPD	Control Point Directory	(See "Disk Space Control" in this chapter.)
?FLNK	Link File	None.
?FSTF	Symbol Table File	Produced by the Link utility and used primarily by AOS/VS.
?FUPF	User Profile File	Used by PREDITOR (user profile editor) and EXEC.
?FSDF	System Data File	None.
?FIPC	IPC Port Entry	(See Chapter 7.)
?FMTF	Magnetic Tape File	None.
?FGFN	Generic Filename	Refers to the generic filenames; that is, @OUTPUT, @LIST, @DATA, etc.
?FGLT	Generic Labeled Tape	None.
?FDKU	Disk Unit	None.
?FSPR	Spoolable Peripheral Directory	None.
?FQUE	Queue Entry	None.
?FLDU	Logical Disk	Cannot create with the ?CREATE system call. (See "Logical Disks" in this chapter.)

Table 4-1. File Types (Cont.)

Mnemonic	Type	Comments
?FMCU	Multiprocessor Communications Unit	Cannot create with the ?CREATE system call.
?FMTU	Magnetic Tape Unit	Device you use to access magnetic tape files; cannot create with the ?CREATE system call.
?FLPU	Data Channel Line Printer	Cannot create with the ?CREATE system call.
?FNCC ?FPCC ?FFCC ?FOCC	FORTRAN Carriage Control	None.
?FCRA	Card Reader	Cannot create with the ?CREATE system call.
?FPLA	Plotter	Cannot create with the ?CREATE system call.
?FCON	Console (hard-copy or video display)	Cannot create with the ?CREATE system call.
?FSYN	Synchronous Communications Line	Cannot create with the ?CREATE system call.

You cannot execute an AOS-written program under AOS/VS unless you relink it with the AOS/VS Link utility. (In some cases, you must re-assemble or re-compile an AOS program file to execute it under AOS/VS.) If you try to execute an ?FPRG program file under AOS/VS, it returns error code ERIFT (illegal file type).

Directory Access

Each process that runs under AOS/VS has a working directory. A working directory is a process's reference point in the overall directory structure and its starting point for file access. (In other words, your working directory is the directory you are working in.) You can use any directory as a working directory, provided you have proper access to it.

In most cases, you will probably access files from your current working directory. When you refer to a file that is not in your working directory, you must refer to it by a pathname, unless you've included the file's parent directory in the search list for your process.

If you want to change your working directory so that you can access files that are not curretly in it, issue the ?DIR system call. Also, the ?DIR system call allows you to return to your intiial working directory after you are finished working elsewhere.

A search list is a list of directories that AOS/VS searches if it fails to find the file that you want in your working directory. You can use the ?SLIST system call to create a search list or to change the contents of an existing search list. To examine your current search list, issue the ?GLIST system call.

Filenames

A filename is a byte string that consists of at least one, and as many as 31, ASCII characters. The legal filename characters are:

- Uppercase and lowercase letters
- Numerals 0 through 9
- Period (.)
- Dollar sign ($)
- Question mark (?)
- Underscore (_)

AOS/VS treats uppercase and lowercase letters alike.

To rename a file, issue the ?RENAME system call.

In general, you can use any conventions you like to name files and families of files. Table 4-2 lists the filename conventions used by AOS/VS and its utilities.

Table 4-2. Filename Conventions

File	Filenames End In
Assembly language source files	.SR
CLI macro files	.CLI
Object files	.OB
Program files	.PR
Temporary files	.TMP and begin with ?
Library files	.LB

You create source files for a program's source code, and then assemble or compile them to produce object files. One or more linked object modules and/or library files make up an executable program file. In general, you use temporary files for data that requires only short-term disk storage.

Pathnames

A pathname specifies the exact location of a directory or file in the file structure. For example, you could use the following pathname to locate directory EAGLE, an entry in the superior directory PAT:

```
:UDD:PAT:EAGLE
```

Directory PAT is inferior to directory UDD, which, in turn, is inferior to the system root, which the colon (:) represents.

A pathname can consist of:

o A prefix alone (such as a colon to indicate the system root).

o An optional prefix followed by the name of a directory or file.

o Pairs of prefixes and directory names or filenames.

The prefix directs AOS/VS to a particular point in the file structure. Table 4-3 lists the valid pathname prefixes.

Table 4-3. Valid Pathname Prefixes

Prefix	Meaning
:	Start at the system root directory.
=	Start at the current working directory.
^	(Uparrow) Move up to the immediately superior directory. (You can use more than one uparrow in a pathname.)
@	Start at the peripheral directory (:PER).

The peripheral directory (:PER), which is inferior to the root, contains the names of generic filenames, which refer to classes of I/O devices, and the names of system devices. (See Chapter 5 for more information on generic filenames and the peripheral directory.)

The = prefix directs AOS/VS to search only the working directory. Generally, when a pathname has no prefix and the file that you want is not in the working directory, AOS/VS checks the search list. The = prefix prevents AOS/VS from doing this.

To construct a pathname to a directory other than your working directory, use either a single prefix, or one or more pairs of prefixes and directory names. For example, the prefixes ^^ cause AOS/VS to move to the directory two levels above your current working directory. The pathname :UDD:PAT explicitly directs AOS/VS to directory PAT, which is subordinate to both UDD and the root.

A full pathname traces the path of a particular file all the way from the root to the file's parent directory. The last entry in a full pathname is :filename, where filename is the name of the file you want to access. The following is a complete pathname to the file GLOSSARY, which is an entry in directory EAGLE:

```
:UDD:PAT:EAGLE:GLOSSARY
```

Figure 4-3 illustrates the use of pathname strings for a sample directory structure.

The ?GNAME and ?CGNAM system calls both return a file's complete pathname, starting with the root. However, they are not the same in that the ?GNAME system call requires a filename or portion of a pathname as input, while the ?CGNAM system call, requires the file's channel number as input. (See Chapter 5 for information on channels.)

```
                                           : Root
                                           /\
                                          /  \
                                         /    \
                                     UTIL      A
                                      /\       /\
                                     /  \     /  \
   Directory----------------> LANG    SPEED  /    \
                               /            B      C
                              /            / \     |
                             /            /   \    |
                            /            /     \  file3
   Subdirectory----> PL_1              D        E
                      |                |        |
                      |                |        |
   File Entries----> PL1.PR           file1    file2
                     PL1.TEMP
```

Working Directory	Pathname	System Action
D	^E:file2	From working directory D, move up to directory B, and down to directory E. Locate file2 in E.

Figure 4-3. Directory Structure

Assuming that the directory structure is the one shown in Figure 4-3, and that D is the working directory, issuing the ?GNAME system call would yield the following results:

Your Input	?GNAME Output
file1	:A:B:D:file1
^E	:A:B:E
^	:A:B
^E:file2	:A:B:E:file2

The ?GRNAME system call is similar to the ?GNAME system call, except that it returns the complete pathname of a generic file. You cannot use the ?GNAME system call to get the "true" pathname of a generic file. For example, given the input pathname @DATA, the ?GNAME system call would return :PER:DATA as the complete pathname, even though the complete pathname of the file is actually :UDD:USER:DATA. In this case, the ?GRNAME system call would return :UDD:USER:DATA. (See Chapter 5 for more information on generic files.)

Link Entries

A link entry (file type ?FLNK) is a file that contains a pathname to another file.

Link entries act as a pathname shorthand. When you specify a link entry in a pathname, AOS/VS substitutes the contents of the link for its name. In Figure 4-3, for example, you can create a link called G that contains the pathname :A:B:D. Thereafter, whenever you refer to link G, AOS/VS resolves that link to :A:B:D. Link entries work differently as input to the system calls ?CREATE and ?DELETE. The next section discusses these two exceptions.

A prefix is optional in a link-entry pathname. If there is a prefix, AOS/VS starts resolving the pathname at the directory that the prefix specifies. If there is no prefix, AOS/VS starts resolving the pathname at the link entry's parent directory.

In addition to acting as pathname abbreviations, link entries serve another purpose. A process can access a file without copying the actual file into its working directory. To do this, the process must include the appropriate link entry in its working directory.

Another way to avoid copying the file is to include the directory that contains the file in a search list. This works only if no other directory in the search list contains a file with the same name. The ?SLIST system call sets a search list for the calling process. Note that a search list cannot contain more than eight pathnames.

One of the entries of a link can be another link. This is called a link-to-link reference. Too many link-to-link references can cause the system call that is referencing the link to overflow its stack. If a stack overflow does occur, AOS/VS returns the stack overflow error message, ERSTO.

Because the number of link-to-link references that you can use depends on both your program and AOS/VS, it is impossible to predict how many link-to-link references will cause a stack overflow. Therefore, if a stack overflow occurs while you are using a pathname, examine the pathname. Then, if the pathname contains link-to-link references, remove them.

To find out what a particular link entry represents, issue the ?GLINK system call. The ?GLINK system call is particularly useful if you cannot decide whether to delete an existing link entry and/or create a new one.

Use of ?CREATE and ?DELETE System Calls on Link Entries

You can use the ?CREATE and ?DELETE system calls to create and delete link entries just as you would other files. When you apply these calls to link entries, however, AOS/VS creates or deletes the link itself, not its contents.

For example, suppose in directory :A you create link entry B, which contains the pathname D:D. If you issue ?DELETE against pathname :A:B, AOS/VS deletes link B without resolving its contents. Directories D and E remain intact, however, as does directory A. (Directory A is simply the "path" to link entry B.)

AOS/VS resolves a link if it is simply part of the pathname of a file you wish to create or delete. Consider the preceding example. If you issue ?DELETE against file C in the pathname :A:B:C, AOS/VS resolves link B to :D:E, and then deletes file C in directory :D:E. Again, directories A, D, and E remain intact.

File Access

To read, write, or execute a file, you must have the proper access to it. Under AOS/VS there are five kinds of access for every file:

- Owner access
- Write access
- Append access
- Read access
- Execute access

Table 4-4 lists the access privileges and their meaning for directories and all other file types.

Table 4-4. File Access Privileges

Privilege	For Nondirectory Files	For Directories
Owner	Allows you to change the file's ACL.	Allows you to change the directory's ACL, to delete and/or rename its files, and to initialize a logical disk (described in the next section).
Write	Allows you to modify the data in the file.	Allows you to create and/or delete the directory's files and to modify each file's ACL.
Append	(No meaning.)	Allows you to add files to the directory.
Read	Allows you to examine data in the file.	Allows you to list the name and file status of each file in the directory.
Execute	Allows you to execute the file.	Allows you to name the directory in a pathname. (This is essential if you want to use the directory or refer to it.)

Execute access is the most essential kind of access to directories, because it allows you to use the directory name in a pathname. Without this privilege, all other access privileges to a directory are meaningless.

Owner access to a directory allows you to initialize logical disks in that directory with the ?INIT system call. (See "Logical Disks" in this chapter.)

Access Control Lists

AOS/VS maintains a unique access control list (ACL) for every file that is not a link entry. An ACL is an ordered list of the users who can access the file and the type of access granted to each user. When you try to read, write, or execute a file, AOS/VS checks your username against each entry in the parent directory's ACL and against each entry in the file's ACL.

For example, if the ACL for file GLOSSARY.CLI allows username TJ Read and Execute access, users that log on under username TJ can execute the file and read its data. However, these same users cannot modify the contents of GLOSSARY.CLI or change its ACL, unless they also have Write access to GLOSSARY.CLI's parent directory.

There are several ways to set an ACL for a file or a directory. One way is to use the CLI command ACL. Another way is to define a file's ACL from your source code via the ?CREATE, ?SACL, or ?DACL system calls. The ?CREATE system call allows you to define the ACL along with the other specifications for the new file or directory. The ?SACL system call allows you to set an ACL for a file or directory.

To determine a particular file or directory's ACL, issue the ?GACL system call. The ?GTACP system call is more specific in that it returns the ACL for a specific file and username. If you are in Superuser mode, the ?GTACP system call allows you to find out if a given user has access to a particular file.

Depending on your input parameters, the ?DACL system call sets, clears, or examines the default ACL mode for one or more processes that have specific usernames. Default ACL mode is process specific, rather than file specific. For example, a process can issue the ?DACL system call to turn on default ACL mode and define a specific ACL for all files it will later create. A default ACL defined with the ?DACL system call exists until the ?DACL caller terminates or until it redefines that default by issuing another ?DACL system call.

The ?CREATE, ?DACL, and ?SACL system calls take the following bit masks as ACL specifications:

Mask	Meaning
?FACA	Append access
?FACE	Execute access
?FACR	Read access
?FACW	Write access
?FACO	Owner access

See the descriptions of the ?CREATE, ?DACL, and ?SACL system calls in Chapter 13 for information on how to combine these masks.

ACL Templates

When you create an ACL, you can define access privileges for specific usernames, or you can use ACL templates to represent certain username/character combinations. Table 4-5 lists the valid ACL templates and the character combinations they represent.

Table 4-5. Valid ACL Templates

Template	Meaning
+	Matches any character string. For example, the ACL username specification PA+ matches any character string that begins with PA, such as PAT, PAM, PAUL, PA_B, and PA.M.
-	Matches any character string except those that contain one or more periods. For example, PA- matches PAT, PAM, PAUL, and PA_B, but not PA.M.
*	Matches any single character except the period. For example, PA* matches PAT and PAM, but not PAUL, PA_B, or PA.M.

AOS/VS scans ACL entries from left to right. Thus, you should not place the plus sign (+) template first, because it will override more specific templates or usernames. For example, the following ACL specification begins with +<?FACR> (the zeros are delimiters), which gives all users Read access only (?FACR), even though the second element assigns Owner access to a specific username (PAT):

|

```
+<0><?FACR>PAT<0><?FACO><0>
```

The Permanent Attribute

Any user with Owner access can easily delete a directory or file. Therefore, AOS/VS provides the permanent attribute for additional protection.

The permanent attribute prevents users from deleting a directory or file, regardless of its ACL. The ?SATR system call sets the permanent attribute, or removes it, if the target directory or file already has permanent status. The ?FSTAT system call returns various information about a directory or file, including whether or not it has the permanent attribute.

If you set the permanent attribute for a file, you should also set it for the file's parent directory. Otherwise, a process can delete the file by deleting the parent directory.

Logical Disks

A logical disk (LD) is one or more physical disk units that you treat as a single logical unit. Each file is completely contained within a single LD.

Each LD is a complete collection of disk space that contains a directory tree structure. In fact, each LD has a single directory called the local root. It is the local root that acts as the foundation for constructing a directory structure. You specify an ACL for the local root when you construct the LD.

When you bootstrap AOS/VS, you select one LD as the Master LD. The root of this LD becomes the system root, which is identified by the colon (:).

Before you can use any LD except the Master LD, you must initialize it with the ?INIT system call or the CLI INITIALIZE command. To use the ?INIT system call, you must have Owner access to the LD's local root directory. The ?INIT system call grafts the LD's local root to a specified directory. (See Figure 4-4.)

No disk structure can have more than eight directory levels, excluding the local root (directory level zero).

```
Master LD (before ?INIT)
------------------------
                                 :<----system root
                                 |
                                 |
                                UTIL
                                / \
                               /   \
                              /     \
                           DGL       FORT4
```

Assume that the LD to be initialized is LD ALPHA. ALPHA's local root, directory UDD, contains two inferior directories: USERA and USERB. If you issue the ?INIT system call for ALPHA, and you specify 0 in AC1, AOS/VS grafts ALPHA to the system root, and the directory tree becomes:

```
Master LD (after ?INIT)
-----------------------
                                 :<----system root
                                 |
                                 |
                                / \
                               /   \
                              /     \
                           UTIL      UDD
                            /\        /\
                           /  \      /  \
                        DGL  FORT4 USERA USERB
```

Figure 4-4. Initializing a Logical Disk

An LD remains initialized until you release it by issuing the ?RELEASE system call. You may want to release an LD to remove its component volumes from the disk drives and mount other volumes onto those disk drives.

Disk Space Control

You can control how AOS/VS allocates disk space by designating certain directories in an LD as control point directories (CPDs). CPDs function exactly like other directories, but they contain two additional variables:

- Current space (CS), which is the amount of space currently allocated.
- Maximum space (MS), which is the maximum amount of space available in the directory.

Current space (CS) is the current number of disk blocks occupied by the CPD and all its inferior files, except for files in an inferior LD. When you create a CPD, AOS/VS initializes CS to zero. Maximum space (MS) is the maximum number of disk blocks available to the CPD and all its inferior files, except for files in an inferior LD. To specify MS, issue the ?CPMAX system call.

Each LD's local root is a CPD. Thus, a local root's CS is the total space currently used in the LD, and its MS is the maximum number of disk blocks the LD can contain.

CPDs restrict a file's disk space to a predefined limit. When a file requires more disk space, AOS/VS first checks the MS and CS of its CPD. AOS/VS allocates more disk space to that file only if it can do so without causing the CPD's CS to exceed its MS. If a file's pathname contains more than one CPD, AOS/VS compares the CS to the MS at every point, starting with the CPD closest to the file.

Figure 4-5 shows a simple directory structure with two CPDs.

Assume that the LD root and directory CP1 in Figure 4-5 are CPDs. If file1 needs an additional n blocks, AOS/VS first adds n to the CS of CP1, which is the control point closest to file1. If CS+n is greater than the MS for CP1, any attempt to allocate additional space for file1 will fail.

If CS+n is less than or equal to the MS for CP1, AOS/VS checks the next control point, in this case the LD root. AOS/VS adds n to the CS for the root. If CS+n is less than or equal to the MS at this level, then AOS/VS allocates the additional disk blocks; otherwise, the allocation attempt fails.

```
 ____________________________________________________________
|                                                            |
|                         LD Root                            |
|                            |                               |
|                            |                               |
|                     control point CP1                      |
|                         /       \                          |
|                        /         \                         |
|                       /           \                        |
|                 directory A     directory B                |
|                      |                |                    |
|                      |                |                    |
|                 :CP1:A:file1     :CP1:B:file2              |
|____________________________________________________________|
```

Figure 4-5. Control Point Directories (CPDs)

When you create a CPD, AOS/VS does not initially check its MS against those of the other CPDs in the file tree. In fact, AOS/VS permits oversubscription, as long as the tree's total CS does not exceed the MS in any superior control point, up to and including the local root. Note that you cannot set a CPD's MS to less than its CS.

File Creation and Management Sample Programs

The following program, FILCREA, opens the console and asks you for the name of the file you want to create. Then, if the file already exists, FILCREA deletes the file and recreates it for you.

```
                        .TITLE  FILCREA
                        .ENT    FILCREA
                        .NREL

FILCREA:?OPEN   CON                         ;Open CON (console) for I/O.
        WBR     ERROR                       ;Error out.

        ?WRITE  CON                         ;Write message.
        WBR     ERROR                       ;Quit.

        XLEFB   0,BUF*2                     ;Get byte pointer to buffer.
        XWSTA   0,CON+?IBAD                 ;Put in I/O packet.

        ?READ   CON                         ;Read filename.
        WBR     ERROR                       ;Quit.

CREATE: ?CREATE CPKT                        ;Create file (AC0 still
                                            ;contains byte pointer to
                                            ;filename.)
        WBR     TEST                        ;Try to handle the error.

        ?WRITE  CON                         ;Echo the filename.
        WBR     ERROR                       ;Quit.

        XLEFB   0,TMES*2                    ;Get byte pointer to
                                            ;confirmation message.
        XWSTA   0,CON+?IBAD                 ;Put in I/O packet.
        ?WRITE  CON                         ;Display confirmation message
                                            ;on console.
        WBR     ERROR                       ;Quit.

        WSUB    2,2                         ;Good return flags.
        ?RETURN                             ;Return to the CLI.
        WBR     ERROR                       ;?RETURN error return.


;Here we deal with errors from ?CREATE.

TEST:   WLDAI   ERNAE,2                     ;Is the error code
        WSEQ    2,0                         ;"file name already exists"?
        WBR     ERROR                       ;No. Report error and quit.
```

FILCREA Program (Cont.)

```
   ;File already exists. Delete it and start again.

           XLEFB   0,BUF*2                 ;Get byte pointer to buffer.
           ?DELETE                         ;Delete file.
           WBR     ERROR                   ;NOW what?
           WBR     CREATE                  ;Resume processing.


   ;All errors except those from ?CREATE come here. We just return with
   ;an error code.

   ERROR:  WLDAI   ?RFEC!?RFCF!?RFER       ;Error flags:  Error code is
                                           ;in ACO (?RFEC), message is
                                           ;in CLI format (?RFCF), and
                                           ;caller should handle this as
                                           ;an error (?RFER).
           ?RETURN                         ;Return to the CLI.
           WBR     ERROR                   ;?RETURN error return.

   ;?CREATE packet.

   CPKT:   .BLK    ?CLTH                   ;Allocate enough space for
                                           ;packet.

           .LOC    CPKT+?CFTYP             ;Record type in left byte and
           .WORD   ?ORDS*400!?FUDF         ;data type in right byte.

           .LOC    CPKT+?CCPS              ;File control parameters.
           .WORD   0                       ;Ignore.

           .LOC    CPKT+?CTIM              ;Address of time block.
           .DWORD  -1                      ;Set all values to current
                                           ;time (default is -1).

           .LOC    CPKT+?CACP              ;Set up byte pointer to ACL.
           .DWORD  ACL*2

           .LOC    CPKT+?CDEH              ;Reserved
           .WORD   0                       ;Set to 0.

           .LOC    CPKT+?CDEL              ;File element size.
           .WORD   -1                      ;Set to default.

           .LOC    CPKT+?CMIL              ;Maximum number of index
           .WORD   -1                      ;levels.  Default.

           .LOC    CPKT+?CLTH              ;End of packet.

   ACL:    .TXT "Username<0><?FACO!?FACW!?FACR><0>"        ;Set ACL to OWR.
```

```
;Open an I/O packet for the console.

CON:    .BLK    ?IBLT                   ;Allocate enough space for
                                        ;packet.

        .LOC    CON+?ISTI               ;File specifications.
        .WORD   ?ICRF!?RTDS!?OFIO       ;Change format to data-
                                        ;sensitive records and open
                                        ;for input and output.

        .LOC    CON+?IMRS               ;Physical block size (in
                                        ;bytes).
        .WORD   -1                      ;Default to 2K bytes.

        .LOC    CON+?IBAD               ;Byte pointer to record I/O
        .DWORD  ITEXT*2                 ;buffer.

        .LOC    CON+?IRCL
        .WORD   120.                    ;Record length is 120
                                        ;characters.

        .LOC    CON+?IFNP               ;Byte pointer to pathname.
        .DWORD  CONS*2

        .LOC    CON+?IDEL               ;Delimiter table address.
        .DWORD  -1                      ;Use default delimiters: null,
                                        ;NEW LINE, form feed, and
                                        ;carriage return.

        .LOC    CON+?IBLT               ;End of packet.

;Filename, message, and buffer.

CONS:   .TXT "@CONSOLE"                 ;Use generic name.

ITEXT:  .TXT "Type filename of file you want to create.  "

BUF:    .BLK    50.                     ;Allocate enough space for
                                        ;buffer.

TMES:   .TXT "  created with ACL of WSR.<12>"

        .NOLOC  0

        .END    FILCREA                 ;End of FILCREA program.
```

End of Chapter

CHAPTER 5
FILE INPUT/OUTPUT (I/O)

The file I/O system calls are:

?ALLOCATE	Allocates disk blocks.
?ASSIGN	Assigns a device to a process for record I/O.
?CLOSE	Closes a file previously opened for record I/O.
?CRUDA	Creates a user data area (UDA).
?DEASSIGN	Deassigns a character device.
?GCHR	Gets the characteristics of a character device.
?GCLOSE	Closes a file previously opened for block I/O.
?GECHR	Gets extended characteristics of a character device.
?GOPEN	Opens a file for block I/O.
?GPOS	Gets the file pointer position.
?GTRUNCATE	Truncates a disk file (block I/O).
?LABEL	Creates a label for a magnetic tape.
?OPEN	Opens a device for record I/O.
?PRDB/?PWRB	Performs physical block I/O.
?RDB/?WRB	Performs block I/O.
?RDUDA	Reads a user data area (UDA).
?READ	Reads a record for record I/O.
?RELEASE	Releases an initialized logical disk (LD).
?SCHR	Sets the characteristics of a character device.
?SDLM	Sets delimiter table.
?SECHR	Sets extended characteristics of a character device.
?SEND	Sends a message to an operator.
?SPOS	Sets the position of the file pointer.
?STOM	Sets the time-out value for a device.
?TRUNCATE	Truncates a disk file or magnetic tape file (record I/O).
?UPDATE	Flushes file descriptor information.
?WRITE	Writes a record for record I/O.
?WRUDA	Writes a user data area (UDA).

Writing to or reading data from a device is called file input/output (I/O). Before you can use file I/O system calls, you must understand file I/O. Therefore, this chapter is divided into the following major sections:

o File I/O Concepts

  This section defines blocks, records, and channels, describes how AOS/VS stores and accesses files, and describes the steps that you normally perform to use file I/O. (See "I/O Concepts.")

o Block I/O

  This section describes block I/O and how it allows you to access a file directly controlling the device on which the file exists. (See "Block I/O.")

o Physical Block I/O

  This section describes physical block I/O and how it is a low-level form of block I/O that only allows you to access disk files. (See "Physical Block I/O.")

o Record I/O

  This section describes record I/O and how it allows you to access a file without knowing on which device that file exists. (See "Record I/O.")

o I/O device sections

  The I/O device sections describe the I/O devices and how to use them to perform file I/O. (See "Device Names," "Generic Filenames," "Labeled Magnetic Tapes," "File I/O on Labeled Magnetic Tapes," "File I/O on Unlabeled Magnetic Tapes," "Multiprocessor Communications Adapters (MCAs), "Character Devices," "Line-Printer Format Control," and "The IPC Facility as a Communications Device.")

o Sample Programs

  This section contains assembled listings of two sample programs that illustrate the use of various file I/O system calls. (See "Sample Programs.")

File I/O Concepts

This section defines blocks, records, and channels, describes how AOS/VS performs file I/O, and describes the file I/O operation sequence.

Blocks and Records

AOS/VS stores files (data) in physical units called blocks. In general, there are two methods of accessing these files:

- Block I/O
- Record I/O

Block I/O system calls allow you to directly access the blocks in which your files are stored. Blocks vary in size from device to device. Therefore, when you access a file using a block I/O system call, you must specify the block size, the starting block number, and exactly how many blocks you want to transfer.

Record I/O system calls allow you to indirectly access the blocks in which your files are stored. When you issue a record I/O system call, AOS/VS sees the file as a collection of logical units called records. Then, AOS/VS selects the correct file and records based on the record type that you specified when you created the file. The record type defines the format of a file's records. AOS/VS uses this information along with other parameters, such as the file's pathname, to associate physical blocks on a device with a certain file and its records.

Channels

File I/O, which includes both block I/O and record I/O, takes place across paths called channels. When you issue a system call to open a file, AOS/VS assigns the file a channel and a unique channel number to identify that channel. The mnemonic ?LOCHN represents the lowest possible channel number and the mnemonic ?HICHN represents the the highest possible channel number.

To disassociate a channel number from a file, close the channel. When you close a channel, it becomes unavailable for further file I/O. AOS/VS assigns a new channel number every time you reopen the file.

File I/O Operation Sequence

File I/O usually involves performing the following steps:

1. Open the file with the ?OPEN system call (for record I/O) or the ?GOPEN system call (for block I/O).
2. Read or write the file data with the ?READ/?WRITE system call (for record I/O) or the ?RDB/?WRB system call (for block I/O).
3. Close the file with the ?CLOSE system call (for record I/O) or ?GCLOSE system call (for block I/O).

The sequence for block I/O is similar to the sequence for record I/O. (See Table 5-1.)

Table 5-1. File I/O Operation Sequence

Operation	Record I/O Call	Block I/O Call
Open the file.	?OPEN	?GOPEN
Read or write.	?READ/?WRITE	?RDB/?WRB
Close the file.	?CLOSE	?GCLOSE

Many file I/O system calls require a packet of file specifications. In general, the ?OPEN, ?READ, ?WRITE, and ?CLOSE system calls use similar specification packets, as do the ?GOPEN, ?RDB, ?WRB, and ?GCLOSE system calls. However, some packet offsets and masks apply to certain system calls only. For example, the Exclusive Open option applies to the ?OPEN system call, but not to the ?READ, ?WRITE, or ?CLOSE system calls. At various points in the file I/O cycle, you can change certain information in the file specification packet.

You can open a file repeatedly without issuing a ?CLOSE system call after each ?OPEN system call. AOS/VS maintains an open count for each ?OPEN system call and closes the file only when the open count equals zero.

The Creation option in the ?OPEN packet allows you to simultaneously create and open certain file types. Table 5-2 lists the file types you can create with this option. When you select the creation option and default the file type parameter in the ?OPEN packet, AOS/VS creates the new file as a user data file (type ?FUDF). You generally use user data files for storing text, data, and variables. User data files are not executable program files.

Table 5-2. File Types You Can Create with the ?OPEN System Call

File Type	Meaning	Comments
?FUDF	User Data File	This is the default file type. (To take this default, set the right byte of offset ?ISTO to 0.)
?FTXT	Text File	This type of file should contain ASCII code.
?FPRV	32-bit Program File	This type of file is an executable 32-bit program file; it should contain linked, executable code.
?FPRG	16-bit Program File	This type of file is an executable 16-bit program file; it should contain linked, executable code.
?FDIR	Disk Directory	If you use the ?OPEN system call to create this type of file, you can default only the following parameters: hash frame size, maximum number of index levels, and access control list.
?FIPC	IPC File	This type of file directs AOS/VS to create an IPC file or open an existing IPC file to allow full-duplex communications between two processes.
?FCPD	Control Point Directory	Although you can use the ?OPEN system call to create a control point directory, we recommend that you use the ?CREATE system call instead.

Unless you have Exclusively Opened a file (an option available in the ?OPEN packet), more than one process with Write access can update any record in the file simultaneously.

| By issuing the ?UPDATE system call, you can guarantee the integrity
| of all previous ?WRITE system calls issued against a file if the
| system crashes while that file is still open. The ?UPDATE system
| call flushes memory-resident file descriptor information to disk.
| Note, however, that the ?UPDATE system call does not write a file's
| data to disk, just its file descriptor information.

File Pointer

To manage repeated I/O sequences, AOS/VS maintains a separate file pointer for each open channel. The file pointer keeps track of the character position for the next read or write sequence on a file.

When you open a file, AOS/VS positions the file pointer, by default, to the first character (byte) in the file. AOS/VS then moves the file pointer forward as it reads or writes each record or byte string. Three ways to override the default position of the file pointer are:

- Select the Append option in the ?OPEN packet (?APND in offset ?ISTI).

  This option moves the file pointer to the last byte in the file, which allows you to append data with the ?WRITE system call.

- Manipulate the file pointer in the ?READ or ?WRITE packet during an I/O sequence.

- Issue the ?SPOS system call to reposition the file pointer without performing I/O.

The ?GPOS system call returns the current position of the file pointer. The ?TRUNCATE system call deletes all data that follows the file pointer in a disk file, and writes two end-of-file marks after the file pointer in a magnetic tape file.

Block I/O

Block I/O means reading or writing files that exist on a device in physical units called blocks. The sizes of these blocks vary from device to device. (See "I/O Devices and Generic Filenames" for information on devices.)

To perform block I/O on a file, you must know the number of blocks you want to transfer (block count), the starting block number, and the block length (number of bytes per block). You specify this information in a block I/O packet. (See the description of the ?RDB and the ?WRB system calls for the packet structure.) The ?GTRUNCATE system call allows you to reduce the size of a disk file that is currently open for block I/O.

The ?ALLOCATE system call allocates blocks for specified data elements and zeroes those data elements that do not actually exist. You can use the ?ALLOCATE system call to make sure that subsequent I/O will not cause a calling process to exceed its control point directory's maximums. (See Chapter 4 for information on control point directories.)

Physical block lengths vary from device to device. To find the block length for a particular device, refer to the 'Programmer's Reference Peripherals' manual. The standard block length for disks is 512 bytes. Magnetic tape block length is whatever length you specify when you issue the ?GOPEN system call. You must specify an MCA unit's block length with each read or write operation.

Physical Block I/O

AOS/VS supports physical block I/O for disks. Physical block I/O is more primitive than block I/O. To perform physical block I/O, you must issue the system calls ?PRDB (read physical blocks) and ?PWRB (write physical blocks.) These system calls require a packet that is similar to the packet for the ?RDB and ?WRB system calls, but that also includes a packet extension.

Physical block I/O allows you to bypass AOS/VS's usual retries for disk errors. You can also use the ?PRDB or the ?PWRB system call to check for bad blocks on a disk, or for problems with an I/O device. When AOS/VS encounters a bad block (transfer error) while it is executing a ?PRDB/?PWRB system call, it takes the normal return, but flags the bad block and reports the reason for the error in the ?PRDB/?PWRB packet. When a device error occurs during a ?PRDB/?PWRB system call, AOS/VS performs the block transfer, but returns the device error code to the ?PRDB/?PWRB packet.

In summary, physical block I/O differs from block I/O in that physical block I/O has:

o No remapping.

  If a physical block transfer fails because of a bad block, AOS/VS continues to read or write the additional blocks, and then takes the normal return from the ?PRDB or ?PWRB system call.

Offset ?PRBB of the system call packet indicates the relative block number of the last good block (successful transfer), and offset ?PCS1 indicates the reason for the error. (Under standard block I/O, AOS/VS avoids bad blocks by referring to the bad block table that the DFMTR utility built when it formatted the disk. (Refer to the 'Managing AOS/VS' manual for more information about the disk formatter and formatting procedures.)

- No retries.

  If a physical block transfer fails, AOS/VS does not try to read or write the block(s) again. (This is different from block I/O in which AOS/VS retries the block read or block write.)

- No ECC corrections.

  If data errors occur during a physical block transfer, AOS/VS completes as much of the transfer as possible, and takes the normal return from the system call. Packet offset ?PRBB contains the relative block number of the last good block, and offsets ?PCS3 and ?PCS3 contain the error-correction code (ECC) words for that particular device.

?PRDB and ?PRWB work in conjunction with the assembly language block status instructions DIA, DIB, and DIC. (For details on the syntax and function of these instructions and the error-correction codes for devices, refer to the 'Programmer's Reference Peripherals' manual.)

Record I/O

Record I/O means reading or writing files that exist on a device in logical groupings called records. There are four types of records:

- Dynamic-length

  When you read to or write from a file that contains dynamic-length records, you must specify the length of each dynamic record in that file.

- Fixed-length

  When you read to or write from a file that contains fixed-length records, you must specify a record length that is common to every record in that file.

- Data-sensitive

  When you read to or write from a file that contains data-sensitive records, you must specify the maximum record length in offset ?IRCL of your I/O packet. Then, AOS/VS transfers data until it either encounters a delimiter or reaches the maximum record length that you specified. In the latter case, your I/O system call fails and returns error code ERLTL (line too long) in AC0.

  The default delimiters are: NEW LINE, CR (carriage return), NULL, or FORM FEED. You can override the default delimiters by specifying a 16-word delimiter table when you open the file or by issuing the ?SDLM system call after you open the file.

- Variable-length

  When you read to or write from a file that contains variable-length records, you must specify the length of each record in a 4-byte ASCII header. This means that each record in a single file can be a different length.

Device Names

During system initialization, AOS/VS records the names of all available I/O devices in its peripheral directory, :PER. Because the standard device names are not reserved words, you must precede each one with the prefix @. As a pathname template, @ represents the :PER directory. Thus, when you use @ as a filename prefix, AOS/VS recognizes the filename as either a device name or a generic filename. (See "Generic Filenames." Also, see Table 5-3 for a complete list of the AOS/VS devices and their device names.

Generic Filenames

The peripheral directory (:PER) also contains generic filenames. Generic filenames are names that refer to devices or files of a particular type, such as input files, output files, and list files.

Generic filenames represent common classes of devices and files. By coding with generic filenames, you can change the filenames associated with the generic names without recoding the program. For example, you might code a program with the generic filename @LIST to represent the list file. Then, before you execute the program, you can set the list file to a specific filename.

Table 5-3. AOS/VS Devices and Device Names

Name	Device
ALM	Asynchronous Line Multiplexor.
@CON0	System Control Processor (SCP).
@CON2 through @CONn	DASHER® display consoles or asynchronous communications lines 1 through n on Lines 0 through n-2 (for example, CON2 is on Line 0, CON3 is on Line 1, etc.).
@CRA and @CRA1	First and second card readers.
@DKB0 through @DKB6	6063 or 6064 fixed-head disk unit 0 through 6.
@DPN0 through @DPN17	Moving-head disk units 0 through 7 on the first controller, and 10 (octal) through 17 (octal) on the second controller where n is a single alphabetic character that indicates the disk unit type. (Refer to the 'Managing AOS/VS' manual for descriptions of these types.)
@LPB, @LPB1 through @LPB7	Data channel line printers 0 through 17.
@LMT	Labeled magnetic tape.
@MCA, @MCA1	Multiprocessor communications adapter controllers (unit names).
@MTB0 through @MTB17	Magnetic tape controller units 0 through 7 on the first controller, and 10 (octal) through 17 (octal) on the second controller.
@PLA and @PLA1	First and second digital plotters.

Table 5-4 lists the six generic filenames and the files they represent.

Table 5-4. Generic Filenames

Filename	Refers To
@CONSOLE	Any interactive device associated with a process (usually a CRT console).
@LIST	A mass output file.
@INPUT	A command input file.
@OUTPUT	Any output file.
@DATA	Any mass input file.
@NULL	A file that remains empty.

Like device names, generic filenames requires the @ prefix.

For an interactive process, your console usually serves as both the @INPUT and the @OUTPUT file. @NULL is not a strict generic filename, in that you cannot associate it with an actual pathname. When you write data to the @NULL file, AOS/VS does not output the data to any other file or device. When you try to read the @NULL file, AOS/VS returns an end-of-file condition.

When you create a process with the ?PROC system call, you can set any generic filename except @NULL to a specific pathname. For example, you can set a process's @LIST file to the following pathname:

```
:UDD:USERNAME:MYDIR:LPT
```

where:

- MYDIR is the current working directory
- LPT is the list file

The ?PROC packet provides the following parameters for generic filename associations:

Offset	Generic Filename
?PCON	@CONSOLE
?PIFP	@INPUT
?POFP	@OUTPUT
?PLFP	@LIST
?PDFP	@DATA

The ?PROC packet also allows the ?PROC caller to pass its own generic filename associations to a newly created son. (See the description of the ?PROC system call in Chapter 13 for more information on the ?PROC packet.)

Usually, AOS/VS copies the data it reads from the @INPUT file to the @OUTPUT file. However, if @INPUT and @OUTPUT are both consoles, then the @INPUT function echoes data to the @OUTPUT console. The generic filenames @INPUT, @OUTPUT, and @LIST acquire all the characteristics of the devices associated with them. For example, if you associate the generic @LIST file with the line printer, a separate listing prints each time you open and close @LIST or any other file.

The @DATA file is similar to the @INPUT file, except that it does not copy data to the @OUTPUT file.

Multiprocessor Communications Adapters

AOS/VS supports type 42006 Multiprocessor Communications Adapters (MCAs). The I/O protocol that AOS/VS uses for these devices is the same MCA protocol that Data General's AOS, RDOS, and RTOS operating systems use.

Each MCA enables two or more central processing units (CPUs) to communicate across a data channel. The MCA units are connected by hardware links. A single MCA can connect a CPU to as many as 14 other CPUs. By adding a second MCA (MCA1), you can connect another 15 CPUs.

Each MCA link consists of two devices: an MCAT, which transmits data from one processor to another, and an MCAR, which receives the data. The MCA pathname takes the following forms:

```
@MCAT:n
@MCAT1:n
@MCAR:n
@MCAR1:n
```

where n is the number of the MCA link, in the range from 0 through 15

The link number indicates which remote CPU you are communicating with when your local CPU is linked to more than one remote CPU.

Character Devices

Character devices are devices that perform I/O in bytes. CRT and hard-copy consoles are typical character devices.

Character devices can operate in one of two modes: binary mode or text mode. Text mode is the default, but you can specify binary mode when you issue a ?READ or a ?WRITE system call against the device. When a character device is in binary mode, AOS/VS recognizes only delimiters. Therefore, AOS/VS passes each byte of any other character without interpretation.

When a character device is in text mode, AOS/VS interprets each byte according to the device's characteristics, or distinguishing features. The device characteristics include:

- The line length of the output.
- Whether the device is ANSI standard or non-ANSI standard.
- Whether the device echoes characters.
- Whether the device uses hardware tab stops or form feeds.

To qualify text mode further, you can set the character device to the Page Mode characteristic. When a character device is in page mode, AOS/VS automatically stops its output at the line length (lines per page) you specify, or when it encounters a FORM FEED character.

To display the next page while the device is in page mode, type the CTRL-Q console control character. (See "Console Format Control" in this chapter for a description of the console control characters.)

The ?GCHR and ?GECHR system calls return the current characteristics of a character device and the extended characterstics of a character device, respectively. The ?SCHR and ?SECHR system calls set or remove device characteristics or extended device characteristics, respectively, depending on your input specifications.

To define characteristics for a character device, you must set certain characteristic flags in a 4-word buffer in AC2 when you issue the ?GCHR system call or the ?GECHR system call. Usually, you will probably set characteristic flags in the first three words of this buffer. If you set characteristic flags in the fourth word, Word 3, then you are setting an "extended" characteristic.

The extended characteristics control XON/XOFF data flow over console lines. By setting the following extended characteristics, you can prevent character loss when the host computer or line device input buffer is overfilling:

Extended Characteristic (in Word 3)	Meaning
?XIFC	Console line is enabled to recognize CTRL-S interrupts from the host computer. The console line stops sending data until the host computer issues CTRL-Q.
?XOFC	Console line is enabled to send CTRL-S interrupts to the host computer; the host computer stops sending data until the console line issues CTRL-Q. (This is useful on D400, D450, or G300 graphics console lines.)

The initial operator process (PID 2) can override characteristics that were set during the system-generation procedure. However, if you are not PID 2, you can only set the modem control and monitor ring indicator characteristics during the system-generation procedure. (For more information on the system-generation procedure, refer to the 'Managing AOS/VS' manual.)

The ?SEND system call allows you to pass a message from a process to a console without opening and closing the console. This means that you can pass messages from real-time processes without consoles to a system process, such as OP CLI.

Full-Duplex Modems

A full-duplex modem is a communications device that translates analog signals to digital signals, and vice versa, over telephone lines. AOS/VS supports I/O over full-duplex modems, which AOS/VS treats as character devices.

You must define modems and set the modem control characteristic (?CMOD) during the AOS/VS system-generation procedure. You cannot set or remove this characteristic with the ?SCHR system call.

AOS/VS supports both auto-answer modems and non-auto-answer modems. The following sections describe the operating procedures for each modem type. Table 5-5 lists the flags used in modem operation.

Table 5-5. Modem Flags

Flag	Meaning
CD	Carrier detect; if set, the communications line is conditioned for data transmissions.
DSR	Dataset ready; if set, AOS/VS is connected to a communications line.
DTR	Data console ready; if set, AOS/VS is ready to connect with a remote user.
RTS	Request to send; if set, AOS/VS has made a request to send data.

Auto-Answer Modems

The following steps summarize the operating sequence for auto-answer modems:

1. During modem initialization, both DTR and RTS are off, which indicates that the modem is off.

2. Upon execution of the first ?OPEN system call, AOS/VS sets DTR and RTS, and changes the modem status to on.

3. No I/O will take place until both DSR and CD are on, which indicates that the modem is connected.

4. The I/O call terminates with an error return if DSR lapses during the I/O sequence, or if CD lapses for more than 5 seconds.

Non-Auto-Answer Modems

If you are receiving data over a non-auto-answer modem, and you are not PID 2, which can override characteristics set during the system-generation procedure, you can select the Monitor Ring Indicator characteristic during the system-generation procedure. This characteristic appears as parameter ?CMRI in the second device characteristics word. (See the descriptions of the ?GCHR and ?SCHR system calls in Chapter 13.) Like the ?CMRI characteristic, you can only set the ?CMOD characteristic during the system-generation procedure, unless you are PID 2.

AOS/VS uses the Monitor Ring Indicator to detect incoming calls (rings) to a non-auto-answer modem. If you select the ring-indicator option, AOS/VS begins monitoring the ring indicator as soon as you open the local modem-controlled device. When a remote user places a call to your device, the hardware signals a modem interrupt and sets the ring indicator. AOS/VS then raises the DTR flag and sets a timer. If AOS/VS does not detect a DSR signal and a valid carrier signal within 5 seconds of the modem interrupt, it posts a disconnect against the line. When this occurs, you must close the modem-controlled device and re-open it.

The following steps summarize the operating sequence for non-auto-answer modems with the Monitor Ring Indicator option:

1. During modem initialization, both DTR and RTS are off, which indicates that the modem is off.

2. Upon execution of the first ?OPEN system call to the modem-controlled device, AOS/VS begins monitoring the ring indicator, provided you selected this characteristic (?CMRI) during the system-generation procedure.

3. When a remote user places a call, the MV/8000 hardware signals an interrupt for the local modem and sets the ring indicator; AOS/VS then sets the DTR flag and starts the ring indicator timer.

4. AOS/VS begins checking for a DSR signal and a CD signal; if these occur within 5 seconds of the modem interrupt, the modem is connected; otherwise, the system posts a disconnect against the line.

5. No I/O takes place until the modem is connected.

6. I/O terminates with an error return if the modem becomes disconnected during the I/O sequence; this state occurs when either the DSR flag changes from on to off, or the carrier signal lapses for longer than 5 seconds.

NOTE: If you have selected the ring-indicator option, you cannot use the communications line for manual dial-outs. To use the line for manual transmissions, you must generate it again, without the ring-indicator option.

Card Readers

AOS/VS also recognizes card readers as character devices. The following steps summarize how AOS/VS handles these devices:

1. When you open a card reader, AOS/VS starts it for input. The card reader then reads ahead as many cards as wil fit in its ring buffer. AOS/VS does not restart the card reader until there is room in the ring buffer for an entire card.

2. AOS/VS performs the Hollerith-to-ASCII conversion if the card reader is in text mode when you issue a ?READ system call. If the card reader is not in text mode, AOS/VS does not convert Hollerith code to ASCII code.

3. If AOS/VS encounters a non-Hollerith card when you issue the ?READ system call, it returns the file read error code ERFIL.

4. AOS/VS returns an end-of-file condition when it reads a card that has all rows punched in column 1.

5. AOS/VS assumes that all cards are at most 81 columns long. Because it does not check column length on input, mark-sense card readers are compatible with AOS/VS.

6. If the card reader is in binary mode, you can set the packed characteristic for its input. This allows you to pack four 12-bit columns into three 16-bit words. Without the packing option, AOS/VS right-justifies the 12 bits in the buffer and uses the 4 upper bits for the following octal status codes:

100000	end of file
040000	hopper empty or stack full
020000	pick fail
010000	read error

7. If you set the Trailing Blanks (?CTSP) characteristic, AOS/VS retains all trailing blanks on the cards. If you omit this characteristic, AOS/VS discards all trailing blanks and writes a NEW LINE character after the last character on each card. You can fit more cards into the ring buffer if you omit this characteristic.

8. The No NEW LINE (?CNNL) characteristic directs AOS/VS to ignore all NEW LINE characters in each card.

Character Device Assignment

AOS/VS allows you to open a device for the exclusive use of one and only one process by "assigning" the device to that process. You can do this explicitly by issuing the ?ASSIGN system call, or you can do this implicitly by opening the file. You cannot issue the ?ASSIGN system call against a file that is already open.

If you assign a file with the ?ASSIGN system call, you must issue the ?DEASSIGN system call to break the assignment. If you assign a device with the ?OPEN system call, you can break the assignment by closing the device or by terminating the process. A process can open a device more than once without breaking an ?OPEN system call assignment; AOS/VS does not break the assignment until the last ?CLOSE system call (when the ?OPEN system call count drops to zero).

Device assignment works somewhat differently for consoles. All son processes can share their father's console, even if the consle was specifically assigned to the father. However, only the most recently created son can actually control the console by issuing ?OPEN, ?CLOSE, ?ASSIGN, ?RELEASE, ?GCHR, ?GECHR, ?SCHR, and ?SECHR system calls against it. The father process and all other sons can issue only ?READ and/or ?WRITE system calls against an assigned console.

Line-Printer Format Control

When you write a file to a data channel line printer controlled by EXEC, you can tailor the format of the output by creating a user data area (UDA) for the file. The ?CRUDA system call creates a UDA. The ?RDUDA and ?WRUDA system calls read and write UDA information, respectively. Typically, you use UDAs to specify file formats, although you can use them for other purposes.

In addition to the ?CRUDA system call, you can also use the AOS/VS Forms Control Utility (FCU) to create UDAs for format specifications. To do this, you must perform the following steps:

1. Create a file with the filename of the UDA that you want to create.

   This file can contain format specifications or, if you wish, it can be empty.

2. Execute FCU. (Refer to the 'Command Line Interpreter (CLI) User's Manual (AOS and AOS/VS)' for more information on FCU.)

3. Move the newly created UDAs to the :UTIL:FORMS directory so that EXEC can access them.

If you want the contents of a particular UDA to override EXEC's default format specification, use the CLI switch /FORMS when you print the file on the line printer. If you omit the /FORMS switch or if the file has no format specifications, AOS/VS uses the current default EXEC format settings. (Refer to the 'Command Line Interpreter (CLI) User's Manual (AOS and AOS/VS)' for more information on the CLI switches.)

Console Format Control

Several control characters and control sequences allow you to control the output that prints on your console.

A control character is any character that you type while you press the CTRL key at the same time. By default, AOS/VS does not pass control characters to your program. However, if you want to override this default, set binary mode or type CTRL-P immediately before you type a control character. Either method will cause AOS/VS to pass the control character to your program. Table 5-6 lists the control characters and what they do.

Table 5-6. Control Characters and Their Functions

Control Character	Function
CTRL-C	Begins a control sequence.
CTRL-D	An end-of-file character; terminates the current read and directs AOS/VS to return an end-of-file condition.
CTRL-O	Suppresses output to your console until you type CTRL-O again. (If AOS/VS detects a BREAK condition, then its output resumes immediately.)
CTRL-P	Signals AOS/VS to accept the next character as a literal, not as a control character.
CTRL-S	Freezes all output to your console, but does not discard it. (To disable CTRL-S, type CTRL-Q.)
CTRL-Q	Disables CTRL-S; if the device is in page mode, CTRL-Q displays the next page.
CTRL-U	Erases the current input line on your console.
CTRL-T and CTRL-V	Reserved for future use by Data General. (Currently, these control sequences do nothing. However, if you precede either one with CTRL-P, AOS/VS passes them to your program.)

A control sequence is a CTRL-C immediately followed by any control character from CTRL-A through CTRL-Z. What happens when you type the second control character depends on the internal state of the process with which the console is associated. If the process has not explicitly redirected the control character, then AOS/VS ignores the control sequence and treats the second control character as it normally would. However, AOS/VS ignores control sequences that do not have a default action.

Table 5-7 lists the control sequences and what they do.

Table 5-7. Control Sequences and Their Functions

Control Sequence	Function
CTRL-C CTRL-A	Generates a console interrupt (provided you used the ?INTWT system call to define a console interrupt task). (See Chapter 6 for more information on the ?INTWT system call.)
CTRL-C CTRL-B	Generates a console interrupt and aborts the current process.
CTRL-C CTRL-C	Echoes the characters ^C ^C on the console, and empties your type-ahead buffer. (This is useful when you want to revoke a command you have typed ahead.)
CTRL-C CTRL-D through CTRL-C CTRL-Z	Reserved for use by Data General.

The IPC Facility as a Communications Device

In addition to the interprocess communications (IPC) procedures described in Chapter 7, you can use IPC files as a communications device, and perform I/O against them. When you perform I/O against an IPC file, AOS/VS buffers the IPC messages in first-in/first-out order. To use the IPC facility as a communications device, AOS/VS performs the following steps:

1. The calling process creates an IPC file entry with the ?OPEN creation option (bit ?OFCR in offset ?ISTI) and sets the file type to ?FIPC (the file type for IPC files).

2. AOS/VS issues a global ?IREC system call for the IPC entry, which indicates that the entry is open. (Note that global ?IREC system calls issued from a particular ring can receive only IPCs destined for that particular ring.) (See Chapter 7 for a description of the global ?IREC option.)

3. The other process issues a complementary ?OPEN system call on the IPC entry.

4. AOS/VS responds with an ?ISEND system call to synchronize the two processes.

After AOS/VS performs these steps, either process can issue ?READ or ?WRITE system calls through the established IPC file. When one of the processes closes the IPC entry or terminates, the system sends the other process an end-of-file condition (error code EREOF) when it tries another ?READ system call against that file.

When you perform I/O on an IPC entry, AOS/VS synchronizes all ?READ and ?WRITE system calls. Thus, for a process to receive another process's termination message, it must read it in the proper sequence. Otherwise, the process could repeatedly attempt to write to the closed IPC entry with no results, because in that case, there is no error return.

Note that the process that creates the IPC file (by issuing the first ?OPEN system call) owns the file.

Labeled Magnetic Tapes

A labeled magnetic tape contains both user data and information about that data--the latter in the form of system and user labels. Labeled magnetic tapes provide the following advantages over unlabeled magnetic tapes:

o ANSI-standard and IBM formats, which enable you to use a labeled magnetic tape on another operating system.

o A naming facility, so you can reference your tape file by name rather than by tape number.

o Volume identifiers, so that a logical file can span several physical tape reels.

o Detailed information about when and how much I/O is actually performed for a particular device.

You can use either the CLI LABEL utility or the ?LABEL system call to create labels for a magnetic tape. After you complete the labeling procedures, you can create files on the tape.

Formats

AOS/VS supports two primary labeling formats: ANSI format (Levels 1, 2, or 3), which uses the ASCII character set, and IBM format (Level 1 or 2), which uses the EBCDIC character set. This allows you to select a format and labeling level suitable for use on another operating system. The formats and levels differ in the number of files allowed in a volume set, the allowable record types, the types of labels, and the contents of the labels. Table 5-8 defines the number of files and record types allowed for each label format and level.

Table 5-8. Label Formats and Levels: Files per Volume Set, Record Types

	Specification	Format	Level
No. of Files	Single file, single volume	ANSI	1, 2, 3
		IBM	1, 2
	Single file, multiple volume	ANSI	1, 2, 3
		IBM	1, 2
	Multiple file, single volume	ANSI	2, 3
		IBM	1, 2
	Multiple file, multiple volume	ANSI	2, 3
		IBM	1, 2
Record Types	Fixed-length	ANSI	1, 2, 3
		IBM	1, 2
	Variable-length	ANSI	3
		IBM	1, 2
	Variable-length spanning blocks	IBM	2
	Undefined-length	IBM	1, 2
	Data-sensitive	n/a	n/a
	Dynamic	n/a	n/a

If you do not set any flags in offset ?IRES of the ?OPEN packet, AOS/VS assumes that you want to use labeled tapes in AOS format. However, if you want to use labeled tapes in ANSI or IBM format, you must set one of the following flags in offset ?IRES:

- Set ?OANS to use labeled tapes in ANSI format.

- Set ?OIBM to use labeled tapes in IBM format.

  AOS/VS does not write the data in EBCDIC. To do this, you must select the field translation packet when you issue the ?READ or the ?WRITE system call.

You should select the labeling level based on the label support of the operating system on which you will use the tape. ANSI Level 3 and IBM Level 2 are the default levels, but you can select a lower level within the ?LABEL system call packet or the ?OPEN system call packet extension for labeled tapes.

If you select a lower level before writing to the tape, AOS/VS records less information about your data in the labels. If you select a lower level before reading, AOS/VS ignores some of the information in the labels. Because AOS/VS can read a tape to a lower level than you specify (for example, an ANSI Level 1 tape even if you define it as an ANSI Level 3 tape), you should default to the highest level.

Label Types

There are four types of labels:

- Volume labels

  These labels identify the volume (reel) of magnetic tape; they occur only at the start of each volume.

- File header labels

  These labels identify the file and its characteristics; they occur before every file on a labeled tape. If the file spans volumes, each file section starts with file header labels.

- End-of-file labels

  These labels identify the file and its characteristics; they occur after every file on a labeled tape.

- End-of-volume labels

  These labels identify the file and its characteristics; these occur at the end of a volume of tape to indicate that the file spans volumes.

Figure 5-1 shows how labels and data are written on a labeled tape.

Each type of label contains one or more individual labels. Some labels are necessary and must be present, or AOS/VS returns an error. Other labels are used if present, but are not required, and some are permitted but are not used. (The "permitted" labels do not cause errors; AOS/VS ignores the information in them.) Table 5-9 lists the different types of labels for the various formats and levels.

Volume Labels

As Table 5-10 indicates, each labeled tape volume must begin with a volume 1 label (VOL1) of 80 bytes (characters). Table 5-10 lists the required contents of the VOL1 labels. The system supplies the characters in quotation marks (for example, "VOL1").

The volid, or tape volume identifier, must consist of up to six characters from the following character set:

- Alphabetic characters A through Z, uppercase only

- Numerals 0 through 9

- Special characters ! " % ( ) * + , - . / ; < > = ?

The volid is part of the pathname you use to refer to a labeled tape file.

The Access field, which is used for ANSI tapes, defines the users allowed to access the tape. You must use a blank space character (ASCII 40) in this field. Otherwise, AOS/VS does not allow access to the volume. The space character allows anyone access to the volume.

```
 _________________________________________________________________________
|  Single File,             Single File, Multiple Volumes                 |
|  Single Volume                          ^                               |
|     Reel 1        _____________________________________________         |
|                  /   Reel 1        Reel 2         Reel 3       \        |
|                                                                         |
|  ____________     _____________   ____________   ____________           |
|  |    VOL   |     |    VOL    |   |    VOL   |   |    VOL   |           |
|  |==========|     |===========|   |==========|   |==========|           |
|  |    HDR   |     |    HDR    |   |    HDR   |   |    HDR   |           |
|  |\\\\\\\\\\|     |\\\\\\\\\\\|   |\\\\\\\\\\|   |\\\\\\\\\\|           |
|  |   File   |     |   File    |   |   File   |   |   File   |           |
|  |          |     |  (first   |   | (second  |   |  (last   |           |
|  |          |     | section)  |   | section) |   | section) |           |
|  ~          ~     ~           ~   ~          ~   ~          ~           |
|                                                                         |
|  |\\\\\\\\\\|     |\\\\\\\\\\\|   |\\\\\\\\\\|   |\\\\\\\\\\|           |
|  |    EOF   |     |    EOV    |   |    EOV   |   |    EOF   |           |
|  |\\\\\\\\\\|     |\\\\\\\\\\\|   |\\\\\\\\\\|   |\\\\\\\\\\|           |
|  |\\\\\\\\\\|     |\\\\\\\\\\\|   |\\\\\\\\\\|   |\\\\\\\\\\|           |
|                                                                         |
|  Multiple Files,          Multiple Files, Multiple Volumes              |
|  Single Volume                          ^                               |
|     Reel 1        _____________________________________________         |
|                  /   Reel 1        Reel 2         Reel 3       \        |
|                                                                         |
|  ____________     _____________   ____________   ____________           |
|  |    VOL   |     |    VOL    |   |    VOL   |   |    VOL   |           |
|  |==========|     |===========|   |==========|   |==========|           |
|  |    HDR   |     |    HDR    |   |    HDR   |   |    HDR   |           |
|  |\\\\\\\\\\|     |\\\\\\\\\\\|   |\\\\\\\\\\|   |\\\\\\\\\\|           |
|  |  File A  |     |  File A   |   |  File B  |   |  File C  |           |
|  |          |     |           |   |  (last   |   |  (last   |           |
|  |          |     |           |   | section) |   | section) |           |
|  ~          ~     ~           ~   ~          ~   ~          ~           |
|                                                                         |
|  |\\\\\\\\\\|     |\\\\\\\\\\\|   |\\\\\\\\\\|   |\\\\\\\\\\|           |
|  |    EOF   |     |    EOF    |   |    EOF   |   |    EOF   |           |
|  |\\\\\\\\\\|     |\\\\\\\\\\\|   |\\\\\\\\\\|   |\\\\\\\\\\|           |
|  |    HDR   |     |    HDR    |   |    HDR   |   |    HDR   |           |
|  |----------|     |-----------|   |----------|   |----------|           |
|  |  File B  |     |  File B   |   |  File C  |   |  File D  |           |
|  |          |     |  (first   |   |  (first  |   |          |           |
|  |          |     | section)  |   | section) |   |          |           |
|  ~          ~     ~           ~   ~          ~   ~          ~           |
|                                                                         |
|  |\\\\\\\\\\|     |\\\\\\\\\\\|   |\\\\\\\\\\|   |\\\\\\\\\\|           |
|  |    EOF   |     |    EOV    |   |    EOV   |   |    EOF   |           |
|  |\\\\\\\\\\|     |\\\\\\\\\\\|   |\\\\\\\\\\|   |\\\\\\\\\\|           |
|  |\\\\\\\\\\|     |\\\\\\\\\\\|   |\\\\\\\\\\|   |\\\\\\\\\\|           |
|                                                                         |
|   KEY:     VOL     Volume labels                                       |
|            HDR     Header labels                                       |
|            EOF     End-of-file labels                                  |
|            EOV     End-of-volume labels                                |
|            \\\     Tape mark; separates data  (Two consecutive         |
|                    tape marks represent an end-of-tape mark.)          |
|_________________________________________________________________________|
```

Figure 5-1. Labels and Data on a Labeled Magnetic Tape

Table 5-9. Types of Labels

Labels	ANSI(1)	ANSI(2)	ANSI(3)	IBM(1)	IBM(2)
Volume Labels:					
VOL1 (Volume 1)	N	N	N	N	N
UVL1-9 (User Volumes 1 through 9)	P	P	P	P	P
File Header Labels:					
HDR1 (Header 1)	N	N	N	N	N
HDR2 (Header 2)	P	P	U	P	U
HDR3-9 (Headers 3 through 9)	P	P	P	P	P
UHL1-9 (User Headers 1 through 9)	P	U	U	U	U
End-of-File Labels:					
EOF1 (End of File 1)	N	N	N	N	N
EOF2 (End of File 2)	P	P	U	P	U
EOF3-9 (End of Files 3 through 9)	P	P	P	P	P
UTL1-9 (User Trailers 1 through 9)	P	U	U	U	U
End-of-Volume Labels:					
EOV1* (End of Volume 1)	N	N	N	N	N
EOV2* (End of Volume 2)	P	P	U	P	U
EOV3-9* (End of Volumes 3 through 9)	P	P	P	P	P
UTL1-9 (User Trailers	P	U	U	U	U

KEY:		
	N	Necessary
	U	Used if present, but not required
	P	Permitted, but not used
	*	End-of-volume labels are necessary only if the file spans reels

Table 5-10. Contents of VOL1 Volume Labels

Byte Position	ANSI(1)	ANSI(2)	ANSI(3)	IBM(1)	IBM(2)
01-04	"VOL1"	"VOL1"	"VOL1"	"VOL1"	"VOL1"
05-10	Volid	Volid	Volid	Volid	Volid
11	Access	Access	Access	"0"	"0"
12-37	Blank	Blank	Blank	Blank	Blank
38-41	Owner name	Owner name	Owner name	Blank	Blank
42-51	Owner name	Owner name	Owner name	Owner name	Owner name
52-79	Blank	Blank	Blank	Blank	Blank
80	Version number	Version number	Version number	Blank	Blank

The optional Owner Name field identifies the owner of the volume. AOS/VS ignores this field when you reference a file on the volume. The default value for this field is a blank space.

The Version Number field specifies the ANSI label format (version) you want for labeled tape processing. This field must contain 1, 2, or 3 if you intend to read the tape. AOS/VS uses version number 3 when you write to the tape.

If you use the ANSI label format, you can follow the VOL1 label with as many as nine optional user volume labels (UVLs) to record additional data about all files on the volume. Note that you cannot use UVLs for tapes that are in IBM format.

Each UVL can contain up to 76 bytes of data. Bytes 1 through 3 contain the character string "UVL", which AOS/VS supplies. AOS/VS numbers UVLs consecutively from 1 through 9. Byte 4 contains the label number.

Table 5-11 lists the contents of a UVL.

Table 5-11. Contents of User Volume Labels (UVLs)

Byte Position	ANSI(1)	ANSI(2)	ANSI(3)
01-03	"UVL"	"UVL"	"UVL"
04	Label number	Label number	Label number
05-80	User data	User data	User data

Header 1 Labels

A Header 1 (HDR1) label of 80 bytes must follow the VOL1 label, regardless of the tape's format or labeling level. Table 5-12 describes the contents of HDR1 labels. AOS/VS supplies the characters in quotation marks (for example, "HDR1").

AOS/VS assigns a sequence number to each file on a labeled tape volume set. If the file spans volumes, AOS/VS divides it into sections and assigns each a section number. AOS/VS uses the File

Sequence Number and File Section Number fields, and a third field, Block Count, for error detection, as follows:

o File Section Number

  The File Section Number indicates which section of the file AOS/VS is currently processing; AOS/VS checks this field to see that the file is processed in order, and that the volume contains the proper file section.

o File Sequence Number

  The File Sequence Number indicates whether or not the file was written correctly. (An incorrect sequence number means the file was written incorrectly.)

o Block Count

  The Block Count indicates the number of blocks written to the file; if the block number on the end-of-file (EOF) or end-of-volume (EOV) label is not the number actually read, a block may have been skipped.

Table 5-12. Contents of HDR1 File Header Labels

Byte Position	ANSI(1)	ANSI(2)	ANSI(3)	IBM(1)	IBM(2)
01-04	"HDR1"	"HDR1"	"HDR1"	"HDR1"	"HDR1"
05-21	Filename	Filename	Filename	Filename	Filename
22-27	File ID set	File ID set	File ID set	File ID set	File ID set
28-31	File section number	File section number	File section number	File section number	File section number
32-35	File sequence number	File sequence number	File sequence number	File sequence number	File sequence number
36-39	"0001"	"0001"	"0001"	Blank*	Blank*
40-41	"00"	"00"	"00"	Blank*	Blank*
42-47	"00000"	"00000"	Creation date	Creation date	Creation date
48-53	Expiration date	Expiration date	Expiration date	Expiration date	Expiration date
54	" " (blank space character)	Access	Access	Access	Access
55-60	Block count	Block count	Block count	Block count	Block count
61-73	System ID	System ID	System ID	System ID	System ID
74-80	Blank	Blank	Blank	Blank	Blank

* For IBM levels 1 and 2, Bytes 36 through 41 contain information that AOS/VS does not use during processing.

The Expiration Date field prevents AOS/VS from overwriting the data on a labeled tape before the specified date. The default expiration date is 90 days after the tape's creation date.

The Access field (like the Access field in the VOL1 label) defines the users allowed to access the tape. For ANSI format, the default for this field is a blank space character (ASCII 40). For IBM format, the default is 0. This gives all users access to the data. Be sure to use the proper default value. If you use another character in this field, AOS/VS assumes that additional access privileges are required, and will not allow access to the tape.

AOS/VS checks the following fields to see that the file on the tape matches the file you requested for I/O:

o File Set Identifier

  The File Set Identifier field identifies the file set. (A file set is a group of files that occupies one or more volumes.) AOS/VS checks the File Set Identifier to see that the newly mounted volume belongs to the file set. By default, the File Set Identifier is the volid (volume identifier) or the first volume in the file set.

o Filename

  The Filename field identifies the file you want to process. There is no default value for this field.

o Generation Number

  The Generation Number field indicates the file's generation. (The default generation number is 0001.) A file can appear on a tape more than once, if each occurrence has a different generation number. This is useful for recording changes to a file.

o Version Number

  The Version Number field indicates which version of a certain file generation you are referencing. (The default version number is 00.) Only one version of a file's generation can appear on a tape.

Header 2 Labels

AOS/VS allows additional header labels (HDR2 and HDR3-9), but these are not required. In fact, AOS/VS uses only Header 2 labels (HDR2), if present, or the ANSI Level 3 and IBM Level 2 formats; it ignores Header 3 through 9 (HDR3-9) for all formats and levels.

If you do use HDR2 labels, they must contain the information shown in Table 5-13. AOS/VS enters the characters in quotation marks (for example, "HDR2").

Table 5-13. Contents of HDR2 File Header Labels

Byte Position	ANSI(3)	IBM(2)
01-04	"HSR2"	"HDR2"
05	Record type	Record type
06-10	Block length	Block length
11-15	Record length	Record length
16-38	Blank	Blank*
39	Blank	Block attribute
40-50	Blank	Blank*
51-52	Buffer offset	Blank*
53-80	Blank	Blank*

* For IBM Level 2, Bytes 40 through 80 contain information that AOS/VS does not use during processing.

The HDR2 labels describe the record type, record length, and block length of the data. The ?OPEN packet conveys this information to AOS/VS. The following list describes the HDR2 fields:

- Record Format

  The Record Format field is dynamic, fixed-length, data-sensitive, or variable length. The record format field must match the specification in offset ?ISTI of the ?OPEN packet. (See the description of the ?OPEN system call in Chapter 13.) You cannot default this value if you intend to write to the tape. If you intend to read the tape and there is no HDR2 label, the record type defaults to fixed-length.

- Block Attribute

  The Block Attribute field states whether the records are blocked (several records per physical block), unblocked (only one record per block), or spanned (a record occupies two or more consecutive blocks). AOS/VS writes all records in blocked format. (You can specify spanned for variable-length records with the special variable-block record type, ?RTVB.)

- Block Length

  The Block Length field states the maximum length of each physical block on the tape; offset ?IMRS in the ?OPEN packet governs this value. If you choose the ?IMRS default (-1), AOS/VS uses 2048 bytes as the block length when it is writing, and the value of the HDR2 field when it is reading.

- Record Length

  The Record Length field states the maximum length of each record; offset ?IRCL in the ?OPEN packet conveys this value. If you choose the ?IRCL default (-1), AOS/VS uses 210 as the record length when it is writing, and this value in the HDR2 field when it is reading.

- Buffer Offset

  The Buffer Offset field states the number of non-data bytes at the start of each physical block. AOS/VS ignores this field.

User Header and User Trailer Labels

In addition to file header labels and file trailer labels (end-of-file, end-of-volume), you can define user header and user trailer labels to supply further information about a labeled tape file. AOS/VS reads or writes these labels via the ?OPEN packet extension for labeled tapes. AOS/VS does not record these user-defined labels in the system labels.

Table 5-14 defines the contents of user header and user trailer labels. Notice that these labels have the same format as UVLs, except that bytes 1 through 3 contain the required strings "UHL" or "UTL", as appropriate.

Table 5-14. Contents of UHL and UTL User Labels

Byte Position	ANSI(2)	ANSI(3)	IBM(1)	IBM(2)
01-03	"UHL" or "UTL"	"UHL" or "UTL"	"UHL" or "UTL"	"UHL" or "UTL"
04	Label number	Label number	Label number	Label number
05-80	User data	User data	User data	User data

End-of-Volume 1, End-of-File 1 Labels

End-of-volume 1 (EOV1) and end-of-file 1 (EOF1) labels have the same format as HDR1 labels, except that Bytes 1 through 4 contain either "EOV1" or "EOF1", as appropriate. (See Table 5-9 for the format.)

End-of-Volume 2, End-of-File 2 Labels

End-of-volume 2 (EOV2) and end-of-file 2 (EOF2) labels have the same format as HDR2 labels, except that Bytes 1 through 4 contain either "EOV2" or "EOF2", as appropriate. (See Table 5-10 for the format.)

File I/O on Labeled Magnetic Tapes

To use labeled tapes for file I/O, you must be logged on under the EXEC utility, either in batch or at a console. You cannot issue I/O system calls against a labeled tape from the operator's console, because the operator process is not a son of EXEC. The OP username must mount all labeled tapes, and the CLI command CONTROL @EXEC OPERATOR ON must be in effect. This command signals EXEC that the operator is available to mount the tapes.

There are two ways to mount a labeled tape: explicitly, by issuing the CLI MOUNT command, or implicitly, by issuing the ?OPEN system call. The CLI MOUNT command syntax is:

```
MOUNT/VOLID=volid linkname operator-message
```

where:

- o linkname is the name of the link entry associated with the tape's filename
- o operator-message is a text string, which usually instructs the operator to mount the tape
- o volid is the 6-character volume identifier (See "Volume Labels" for the volid character set.)

The CLI MOUNT command creates links for both labeled and unlabeled tapes. When you issue the CLI MOUNT command against a labeled tape, EXEC passes the message string to the operator and creates a link entry for the filename in your initial working directory.

The link resolves to @LMT:volid when you open, read, write, or close that tape volume. Note that EXEC creates the link entry in your initial working directory, not in the directory from which you issued the CLI MOUNT command.

When you perform primitive I/O or issue CLI commands against the labeled tape volume, you can substitute the tape's filename for @LMT:volid. After you read or write to a tape file that you opened with the CLI MOUNT command, use the CLI DISMOUNT command to tell the operator to remove the tape from the tape drive.

You can also mount a labeled tape with the CLI DUMP command, or any CLI command that accesses @LMT:volid. When you use this method, EXEC checks to see if the tape is already mounted. If it is not, EXEC directs the operator to mount it. The syntax of the CLI DUMP command is:

```
DUMP @LMT:volid:filename
```

Each time you issue the CLI DUMP command, EXEC directs the operator to mount and then dismount the tape. Thus, the CLI MOUNT command is usually the more efficient method.

If you mount the labeled tape with the ?OPEN system call, offset ?IFNP points to the name of the tape volume, which must be in the following form:

```
@LMT:volid:filename
```

AOS/VS does not create a link when you use this method, but it does tell the operator to mount the labeled tape volume specified in the pathname. When you close that tape file with the ?CLOSE system call, AOS/VS directs the operator to dismount the labeled tape volume.

Mounting a labeled tape explicitly with the CLI MOUNT command is the most efficient way to perform I/O on more than one labeled tape file, because AOS/VS does not need to rewind and reposition the tape for each I/O sequence or direct the operator to mount and dismount the tape for each ?OPEN and ?CLOSE system call. However, the ?OPEN system call is useful because it gives you the option of creating and opening the tape file at the same time.

When you read or write to a labeled tape, refer to the tape by one of the following pathnames:

```
@LMT:volid:filename
```

where:

- @LMT is the generic filename for a labeled tape
- volid is the volume identifier number
- filename is the name of the file you wish to access

```
:UDD:username:linkname:filename
```

where:

- UDD is the name of the user directory
- username is your username
- linkname is the name of the link entry created by EXEC when the tape was mounted
- filename is the name of the tape file

You do not need to cite a specific tape unit number for either of these formats. Use the second format if your current working directory is not :UDD:username.

EXEC creates the LMT entry and assigns it file type ?FGLT, the file type for labeled tapes. The filename you choose must consist of at least 1 and not more than 17 characters from the same character set you used for volid.

Because not all characters in this set conform to the character set for filenames, you cannot pass all labeled tape filenames through the CLI. (Instead, you must write your own programs, using the I/O system calls, to perform I/O on these labeled tapes.)

File I/O on Unlabeled Magnetic Tapes

To use a magnetic tape unit, you must first open it. To do this, specify the number of the tape unit and the position of the file on the tape (its file number) in the following form:

```
@MTBx:y
```

where:

- x is the number of the tape unit
- y is the file number

Magnetic tape files are numbered sequentially, starting with 0. Thus, the pathname @MTB0:2 specifies the third file on tape unit 0. If you do not specify a file number, AOS/VS automatically opens the first file (file 0) on the tape.

If you use block I/O system calls to access a magnetic tape, you can specify the file number after you issue ?RDB or ?WRB system calls against the tape.

If you issue the CLI command MOUNT to signal the operator to mount a magnetic tape, use the linkname you used in the MOUNT command when you perform I/O against the file. For example, if you issue the following CLI command, you would be using the linkname TAPE1 to open, read, write, or close that file:

```
MOUNT TAPE1 operator_message
```

In this case, AOS/VS would find TAPE1 in your initial working directory.

File I/O Sample Programs

The following program, RITE, opens the console and the disk file FILE. Then, RITE asks you to type lines of text at your console keyboard, and writes each line to FILE. When you type "RD," RITE reads the lines back from FILE and displays them on your console.

RITE uses ?OPEN, ?READ, ?WRITE, and ?SPOS system calls.

```
                        .TITLE  RITE
                        .ENT    RITE
                        .NREL

;Open console (@CONSOLE) and file for input and output.

RITE:   ?OPEN   CON                     ;Open console (CON) for I/O.
        WBR     ERROR                   ;Report error and quit.

        ?OPEN   FILE                    ;Open or create disk file
                                        ;named FILE.
        WBR     ERROR                   ;Quit.

;Write greeting and put byte pointer to I/O buffer in packet.

     ?WRITE   CON                     ;Display message on console.
     WBR      ERROR                   ;?WRITE error return.
     XLEFB    0,BUF*2                 ;Get byte pointer to I/O
                                      ;buffer.
     XWSTA    0,CON+?IBAD             ;Put in I/O packet.

;Read line, check for terminator, and then write to file.

        NLDAI   'RD',0                  ;Put RD terminator in AC0.

LOOP:   ?READ   CON                     ;Read a line.
        WBR     ERROR                   ;Quit.

        XNLDA   1,BUF                   ;Get first word of buffer.
        WSNE    0,1                     ;Did user type RD?
        WBR     SPOS                    ;Yes.  Do ?SPOS.

        ?WRITE  FILE                    ;No.  Write line to FILE.
        WBR     ERROR                   ;Quit.
        WBR     LOOP                    ;Get next line from user.
```

RITE Program (Cont.)

```
   ;Set position at beginning of file.

   SPOS:    NLDAI    0,1                       ;Get 0 in AC1.
            XWSTA    1,FILE+?IRNH              ;Put in record number word.
            XNLDA    2,FILE+?ISTI              ;Get file's specifications.
            WMOV     2,0                       ;Save old specifications in
                                               ;AC0.
            WIORI    ?IPST,2                   ;Add ?IPST specification.
            XNSTA    2,FILE+?ISTI              ;Put in file specifications.

            ?SPOS    FILE                      ;Position at beginning of
                                               ;FILE.
            WBR      ERROR                     ;Quit.
            XNSTA    0,FILE+?ISTI              ;Restore old specifications.


   ;Read lines back from FILE and display on console.

   LOOP1:   ?READ    FILE                      ;Read from FILE into buffer.
            WBR      EOF                       ;Try to handle the error.

            ?WRITE   CON                       ;Display line on console.
            WBR      ERROR                     ;Quit.

            WBR      LOOP1                     ;Read/write another line.

   EOF:     NLDAI    EREOF,2                   ;Error code for end-of-file
                                               ;(EOF) is EREOF.
            WSEQ     0,2                       ;Was it an EOF?
            WBR      ERROR                     ;No. Quit.

   ;Close the file.

   CLOSE:   ?CLOSE   CON                       ;Close console.
            WBR      ERROR                     ;Quit.

            ?CLOSE   FILE                      ;Close FILE.
            WBR      ERROR                     ;Quit.
            WSUB     2,2                       ;Set flags for normal return.
            WBR      BYE                       ;Take good return.

   ;Process error and/or return here.

   ERROR:   WLDAI    ?RFEC!?RFCF!?RFER,2       ;Error flags: Error code is
                                               ;in AC0 (?RFEC), message is in
                                               ;CLI format (?RFCF), and
                                               ;caller should handle this as
                                               ;an error (?RFER).
```

```
BYE:    ?RETURN                          ;Return to CLI.
        WBR     ERROR                    ;?RETURN error return

;Open and I/O packet for console.

CON:    .BLK    ?IBLT                    ;Allocate enough space for
                                         ;packet.

        .LOC    CON+?ISTI                ;File specifications.
        .WORD   ?ICRF!?RTDS!?OFIO        ;Change format to data-
                                         ;sensitive records and open
                                         ;for input and output.

        .LOC    CON+?IMRS
        .WORD   -1                       ;Default physical block size
                                         ;to 2K bytes.

        .LOC    CON+?IBAD                ;Byte pointer to record I/O
        .DWORD  ITEXT*2                  ;buffer.

        .LOC    CON+?IRCL
        .WORD   120.                     ;Record length is 120
                                         ;characters.

        .LOC    CON+?IFNP                ;Byte pointer to pathname.
        .DWORD  CONS*2

        .LOC    CON+?IDEL                ;Delimiter table address.
        .DWORD  -1                       ;Use default delimiters: null,
                                         ;NEW LINE, form feed, and
                                         ;carriage return (default is
                                         ;-1).
        .LOC    CON+?IBLT                ;End of packet.

;Filename, buffer, and messages.

CONS:   .TXT "@CONSOLE"                  ;Use generic name.

BUF:    .BLK 60.                         ;Allocate enough space for
                                         ;buffer.

ITEXT:  .TXT "I write lines to file FILE.  Type RD[NL] to read lines
            back and stop.<12>"
        .NOLOC 0                         ;Resume listing all.
```

RITE Program (Cont.)

```
;?OPEN and I/O packet for FILE.  You can omit those entries that you
;want to set to 0.

FILE:   .BLK    ?IBLT                       ;Allocate enough space for
                                            ;packet.

        .LOC    FILE+?ICH
        .WORD   0                           ;AOS/VS assigns channel
                                            ;number.

        .LOC    FILE+?ISTI                  ;File specifications.
        .WORD   ?OFCR!?OFCE!?ICRF!?RTDS!?OFIO   ;Delete file and then
                                            ;recreate it (?OFCR!?OFCE),
                                            ;change format (?ICRF) to
                                            ;data-sensitive records
                                            ;(?RDTS), and open for input
                                            ;and output (?OFIO).

        .LOC    FILE+?ISTO
        .WORD   0                           ;Default to ?FUDF, user data
                                            ;file.

        .LOC    FILE+?IMRS
        .WORD   -1                          ;Default physical block size
                                            ;to 2K bytes.

        .LOC    FILE+?IBAD                  ;Byte pointer to record I/O
        .DWORD  BUF*2                       ;buffer.

        .LOC    FILE+?IRES                  ;Density mode (for magnetic
                                            ;tapes only).
        .WORD   0                           ;Default it.

        .LOC    FILE+?IRCL
        .WORD   120.                        ;Record length is 120
                                            ;characters.

        .LOC    FILE+?IRLR                  ;Number of bytes transferred.
        .WORD   0                           ;Only ?READ and ?WRITE use
                                            ;this.

        .LOC    FILE+?IRNW                  ;Reserved.
        .WORD   0                           ;Set to 0.

        .LOC    FILE+?IRNH                  ;Record number.
        .DWORD  0                           ;Only ?READ and ?WRITE use
                                            ;this.
```

RITE Program (Cont.)

```
        .LOC    FILE+?IFNP              ;Byte pointer to pathname.
        .DWORD  FNAME*2

        .LOC    FILE+?IDEL              ;Delimiter table address.
        .DWORD  -1                      ;Use default delimiters: null,
                                        ;NEW LINE, form feed, and
                                        ;carriage return (default is
                                        ;-1).

        .LOC    FILE+?IBLT              ;End of packet.

FNAME:  .TXT    "FILE"                  ;Disk filename.

        .END    RITE                    ;End of RITE program.
```

Block I/O Sample Program

The block I/O sample program, DLIST lists all filenames in a directory and prints them on the line printer. DLIST uses the CLI ?GTMES mechanism (see Chapter 11) to get the directory name as well as using ?GOPEN to open the directory. Also, DLIST uses the ?OPEN, ?READ/?WRITE, ?GNFN, and ?SEND system calls. To execute DLIST, type:

```
X program_name directory_name
```

```
                .TITLE  DLIST
                .ENT    DLIST
                .NREL

;Get the directory name, open it, and open the line printer queue.

DLIST:  ?GTMES  CLIMSG                  ;Get directory name.
        WBR     ERROR                   ;?GTMES error return.
        LLEFB   0,DIRNAME*2             ;Get byte pointer to directory
                                        ;name.
        NLDAI   -1,1                    ;Specify that AOS/VS assign
                                        ;channel number for ?GOPEN.

        ?GOPEN  DIR                     ;Open the directory.
        WBR     ERROR                   ;?GOPEN error return.

        ?OPEN   LINEP                   ;Open the line printer queue.
        WBR     ERROR                   ;?OPEN error return.

;Use ?GNFN to get next name and write to line printer.

        XNLDA   1,DIR+?ICH              ;Keep channel number in AC1.

NEXT:   ?GNFN   GNAME                   ;Put filename in ?GNFN buffer.
        WBR     EOF                     ;?GNFN error return.

        ?WRITE  LINEP                   ;Output contents of ?GNFN
                                        ;buffer (filename) on line
                                        ;printer.
        WBR     ERROR                   ;?WRITE error return.
        XLEFB   2,NL*2                  ;Get address of NEW LINE
                                        ;character.
        XWSTA   2,LINEP+?IBAD           ;Put address of NEW LINE
                                        ;character in line printer
                                        ;buffer.
```

DLIST Program (Cont.)

```
        ?WRITE  LINEP                     ;Output contents of buffer
                                          ;(address of NEW LINE
                                          ;character) on line printer.
        WBR     ERROR                     ;?WRITE error return.
        XLEFB   2,FNAME*2                 ;Get byte pointer to filename
                                          ;buffer.
        XWSTA   2,LINEP+?IBAD             ;Restore buffer address.
        WBR     NEXT                      ;Get another filename.

EOF:    NLDAI   EREOF,2                   ;Is error code EREOF
                                          ;(end-of-file)?
        WSEQ    0,2                       ;Yes.  Skip this instruction.
        WBR     ERROR                     ;No.  Try to handle the error.

;Finished with filenames.  Get ?SEND parameters and issue ?SEND.

        XLEFB   0,CONS*2                  ;Byte pointer to console name.
        XLEFB   1,TMSG*2                  ;Byte pointer to ?SEND message.
        WLDAI   (CLIMSG-TMSG)*2!1S22,2    ;Message length and byte
                                          ;pointer flag.

        ?SEND                             ;Send message to console.
        WBR     ERROR                     ;?SEND error return.
        WSUB    2,2                       ;Done.  Set flags for ?SEND
                                          ;normal return.
        WBR     BYE                       ;Goodbye.

ERROR:  NLDAI   ?RFEC!?RFCF!?RFER,2       ;Error flags: Error code is in
                                          ;AC0 (?RFEC), message is in
                                          ;CLI format (?RFCF), and
                                          ;should handle this as an
                                          ;error (?RFER).

BYE:    ?RETURN                           ;Return to CLI.
        WBR     ERROR                     ;?RETURN error return

NL:     .TXT    "<12>"                    ;Put each name on a new line.

;?SEND console name and message.  A .NOLOC 1 follows.

CONS:   .TXT    "@CONSOLE"                ;Use generic name.

TMSG:   .TXT    "All filenames written to line printer.  Bye."
        .NOLOC  0                         ;Resume listing all.
```

```
;?GTMES packet to get directory name from CLI.

CLIMSG: .BLK    ?GTLN                   ;Allocate enough space for
                                        ;packet.

        .LOC    CLIMSG+?GREQ            ;Request type.
        .WORD   ?GARG                   ;Put argument in ?GRES only.

        .LOC    CLIMSG+?GNUM
        .WORD   1                       ;Argument 1 is directory name
                                        ;(argument 0 is program name).

        .LOC    CLIMSG+?GRES            ;Byte pointer to receive
                                        ;buffer.
        .DWORD  DIRNAME*2               ;Byte pointer to directory
                                        ;name buffer (DIRNAME).

        .LOC    CLIMSG+GTLN             ;End of packet.

DIRNAME:.BLK    50.                     ;Directory name buffer.

;?GOPEN packet (needed for directory).

DIR:    .BLK    ?OPLT                   ;Allocate enough space for
                                        ;packet.
        .LOC    DIR+?OPLT               ;End of packet.

;?GNFN packet to get next filename.

GNAME:  .BLK    ?NFLN                   ;Allocate enough space for
                                        ;packet.

        .LOC    GNAME+?NFKY             ;AOS/VS uses this after first
                                        ;call.
        .DWORD  0                       ;Set to 0 for first call.

        .LOC    GNAME+?NFNM

        .DWORD  FNAME*2                 ;Byte pointer to filename
                                        ;receive buffer.

        .LOC    GNAME+?NFTP
        .DWORD  -1                      ;There is no template (default
                                        ;is -1).

        .LOC    GNAME+?NFLN             ;End of packet.

FNAME:  .BLK    16.                     ;Area of receive filenames.
```

DLIST Program (Cont.)

```
   ;?OPEN and I/O packet for line-printer output file.

   LINEP:  .BLK    ?IBLT                    ;Allocate enough space for
                                            ;packet.

           .LOC    FILE+?ICH
           .WORD   0                        ;AOS/VS assigns channel number.
           .LOC    FILE+?ISTI               ;File specifications.
           .WORD   ?ICRF!?RTDS!?OFOT        ;Change format (?ICRF) to
                                            ;data-sensitive records
                                            ;(?RDTS), and open for input
                                            ;and output (?OFIO).

           .LOC    FILE+?ISTO
           .WORD   0                        ;Default file type to ?FUDF,
                                            ;user data file.

           .LOC    FILE+?IMRS
           .WORD   -1                       ;Default physical block size
                                            ;to 2K bytes.

           .LOC    FILE+?IBAD
           .DWORD  FNAME*2                  ;Byte pointer to record I/O
                                            ;buffer.

           .LOC    FILE+?IRES               ;Density mode (for magnetic
                                            ;tapes only).
           .WORD   0                        ;Default to density mode set
                                            ;during VSGEN procedure.

           .LOC    FILE+?IRCL
           .WORD   136.                     ;Record length is 136
                                            ;characters.

           .LOC    FILE+?IRLR               ;Number of bytes transferred.
           .WORD   0                        ;Only ?READ and ?WRITE use
                                            ;this.

           .LOC    FILE+?IRNW               ;Reserved.
           .WORD   0                        ;Set to 0.

           .LOC    FILE+?IRNH               ;Record number.
           .DWORD  0                        ;Only ?READ and ?WRITE use
                                            ;this.

           .LOC    FILE+?IFNP
           .DWORD  LPTNM*2                  ;Byte pointer to pathname.
```

DLIST Program (Cont.)

```
        .LOC    FILE+?IDEL              ;Delimiter table address.
        .DWORD  -1                      ;Use default delimiters: null,
                                        ;NEW LINE, form feed, and
                                        ;carriage return (default is
                                        ;-1).

        .LOC    FILE+?IBLT              ;End of packet.

LPTNM:  .TXT    "@LPT"                  ;Printer queue filename.

        .END    DLIST                   ;End of DLIST program.
```

End of Chapter

CHAPTER 6
TASKS

The system calls that allow you to initiate and manage tasks are:

?DFRSCH	Disables scheduling and indicates prior state of scheduling.
?DQTSK	Dequeues a task or tasks previously queued.
?DRSCH	Disables task scheduling.
?ERSCH	Enables task scheduling.
?IDGOTO	Redirects a task.
?IDKIL	Kills a task specified by its TID.
?IDPRI	Changes the priority of a task specified by its TID.
?IDRDY	Readies a task specified by its TID.
?IDSTAT	Returns task statistic flag. (16-bit processes only)
?IDSUS	Suspends a task specified by its TID.
?IFPU	Initializes the floating-point status registers.
?IQTSK	Creates a queued task manager (for ?TASK queuing).
?KILAD	Defines a kill-processing routine for a task.
?KILL	Kills the calling task.
?MYTID	Returns the TID of the calling task.
?PRI	Changes the priority of the calling task.
?PRKIL	Kills all tasks of a given priority.
?PRRDY	Readies all tasks of a given priority.
?PRSUS	Suspends all tasks of a given priority.
?REC	Receives an intertask message.
?RECNW	Receives an intertask message without waiting.
?SUS	Suspends the calling task.
?TASK	Initiates one or more tasks.
?TIDSTAT	Returns the status of a task specified by its TID. (32-bit processes only)
?TLOCK	Protects a task from being redirected.
?TRCON	Reads a task message from a process console.
?TUNLOCK	Revokes redirection protection for the current task in the current ring.
?UIDSTAT	Returns the status of a task and an unambiguous identifier.

(Cont.)

WDELAY	Suspends a task for a specified time.
XMT	Transmits an intertask message.
XMTW	Transmits an intertask message and waits for its receipt.

AOS/VS includes many system calls that allow you to manage tasks and a multitasking environment. Before you can use these system calls, however, you must understand what tasks are, what multitasking is, and how to manage tasks and the multitasking environment. Therefore, this chapter is divided into the following sections:

o The first two sections define tasks, multitasking, and the AOS/VS task-protection models. (See "Task Concepts" and "Task-Protection Schemes.")

o The third section describes how to identify tasks. (See "Task Identifiers and Priority Numbers.")

o The fourth section describes how to initiate tasks. (See "Task Initiation.")

o The remaining sections describe stacks (including inner-ring stacks), how AOS/VS schedules tasks, how you can redirect tasks, how you kill tasks, console-to-task and task-to-task communication, and the registers that allow you to manipulate floating-point numbers. (See "Stack Space Allocation and Stack Definition," "Inner-Ring Stack Support," "Task Scheduling," "Task Redirection," "Task Termination," "Console-to-Task Communication," "Task-to-Task Communication," and "MV/8000 Floating-Point Registers.")

o The last section contains sample programs. These sample programs use many of the system calls described elsewhere in this chapter. (See "Sample Programs.")

Task Concepts

A task is a path through a process. It is an asynchronously controllable entity to which the CPU is allocated for a specific time. A task can only execute code within the bounds of the address space allocated to its process. (See Chapter 3 for more information on processes.)

Each process consists of one or more tasks, which execute asynchronously. You can design your code so that several tasks execute a single re-entrant sequence of instructions, or you can create a distinct instruction path for each task.

You combine program files with other information to define processes. A task is the basic element of a process. Initially, each process has only one task associated with it. However, unlike processes, tasks within a process only exist until you kill them either explicitly or implicitly. (See "Task Termination" in this chapter.)

If you are familiar with high-level languages such as BASIC or FORTRAN, you are probably familiar with single-task programs. Single-task programs display one path that connects all branches of logic, no matter how complex. Multitasking is a programming technique that allows up to 32 tasks within a single process to execute.

As a programming technique, multitasking offers several advantages, including:

o Parallelism

  Multitasking is a straightforward way to handle complex parallel events within one program. Thus, it can be useful for time-out and alarm routines, and overlapped I/O. Multitasking gives a program the flexibility to respond to external asynchronous events.

o Efficiency

  While one task is suspended, perhaps on an I/O operation, another task can be executing. Each task has a priority level, and AOS/VS schedules tasks based on their relative priorities. The AOS/VS multitasking scheduling facility provides efficient CPU and memory use, especially in an environment with heavy memory contention and devices of varying speeds.

You can design your code so that several tasks execute one re-entrant instruction sequence, or you can create a different instruction path for each task.

Task-Protection Schemes

The AOS/VS protection model prevents tasks executing in an outer ring from interfering with tasks executing in critical inner-ring code paths. AOS/VS uses two classes of protection mechanisms to protect tasks executing in one ring from interference by tasks executing in other rings:

- Ring maximization

  Under this protection scheme, AOS/VS considers a task that is executing in a user ring to be less privileged than another task that is executing in a lower user ring. For all system calls, AOS/VS uses the ring-maximization protection scheme when it validates user-supplied channels, word pointers, or byte pointers.

  This means that a channel opened by a system call issued from one user ring cannot be passed as input to a system call issued from a higher user ring. Also, system calls issued from one user ring cannot be passed as input pointers to lower-ring memory locations.

  The ring-maximization protection scheme parallels the hierarchical protection scheme of the MV-series memory-management hardware.

- Ring specification

  The ring-specification protection scheme protects tasks executing in one user ring from interference by tasks executing in any other user rings. The connection-management facility and the IPC facility use the ring-specification protection scheme in the following ways:

  - The connection-management facility considers connections to be between pairs of process identifier (PID)/ring tandems.

- The IPC facility now requires a ring field as well as a PID and a local port number field as part of each global port.

  All IPC messages are sent to specific rings within a destination process. Within the destination process, only tasks that issue IPC receive request system calls from the specified ring can receive IPC messages sent to that ring. In this way, interprocess communications paths are secured from both malicious and accidental interference by tasks issuing IPC receive requests from other rings within the same process.

(See Chapter 2 for information on the ring structure.)

Task Identifiers and Priority Numbers

When you create a task, you should assign it a task identifier (TID) in the range from 1 through 32. In addition to providing a simple way for you to keep track of each task's actions, several system calls require a TID as input.

If you do not assign each task a TID, AOS/VS assigns the initial task TID 1, but assumes that every other task is TID 0. Although permissible, this is not advisable. Tasks that share TID 0 cannot issue ?IDSTAT, ?IDPRI, ?IDRDY, ?IDSUS, and ?TIDSTAT system calls.

In addition to the TIDs that you supply, AOS/VS assigns a unique TID to each task in the system. Therefore, even though each initial task is TID 1 within its own process, it also has a unique TID. This system-assigned unique TID allows you to index into multiple-task databases.

To find out what the unique TID for a particular task is, issue the ?UIDSTAT system call. The ?UIDSTAT system call returns the unique TID and the contents of the task's status word.

Priority numbers are values AOS/VS uses to determine the order in which tasks execute. Priority numbers range from 0 (the highest priority) through 255 (the lowest priority). AOS/VS assigns the initial task (TID 1) priority 0, the highest priority.

To find out the priority and TID of a calling task, issue the ?MYTID system call. If you want to use system calls that require a TID or priority level as an input parameter, you can use the ?MYTID system call to get this information.

Task Initiation

The Link utility lets you specify the maximum number of tasks in a process, up to a limit of 32 tasks. Each process is initialized when AOS/VS begins to execute that process's initial task. To initiate other tasks, any executing task can issue the ?TASK system call.

The ?TASK system call requires a packet. This packet allows you to specify several characteristics for the new task, including its TID and its priority.

You can influence task scheduling by assigning a priority level to a task. If you do not assign a priority, AOS/VS assigns the new task the same priority level as the calling task (the task that issued the ?TASK system call).

You can use the ?TASK system call to initiate one or more tasks immediately, or you can use it to initiate a task at a later time. Therefore, there are two versions of the ?TASK packet:

- The standard packet, which initiates a task.

- The extended packet, which initiates a task at a particular time and at particular intervals. This is called queued task creation.

When you issue a ?TASK system call that specifies a starting PC within Ring 7, AOS/VS passes control to the ?UTSK task-initiation routine, which places the address of a task-kill routine in AC3 and then returns control to the ?TASK system call. (?UTSK is in the user runtime library URT32.LB.)

You can tailor a task-initiation routine to your own application. For example, you may want to assign system resources to each newly initiated task. To use a tailored task-initiation routine, you must assign the new routine the label ?UTSK and then link it with your progam. If you do not do this, AOS/VS passes control to the default ?UTSK routine, which immediately returns control to the ?TASK system call. In addition, if your tailored ?UTSK task-initiation routine pushes anything onto the stack, it must also pop it off the stack before exiting from the routine. Otherwise, if it leaves anything on the stack, the calling task may not return to the proper address in your program.

To abort the ?TASK system call while your ?UTSK task-initiation routine is executing, load AC0 with an error code and return to the address in AC3 (the address of the task-initiation error return). If you do not want to abort the ?TASK system call, increment the value of AC3 by 1 and return to the address in AC3 (the address of the task initiation normal return). This not only causes the ?UTSK task-initiation routine to return successfully, but also causes the ?TASK system call to continue normally.

To use the queued task creation option, you must use the extended ?TASK packet, and you must issue the ?IQTSK system call before you issue the ?TASK system call. The ?IQTSK system call creates an additional task, the queued task manager, which handles the initiation queue. (The queued task manager is one of the 32 possible tasks in your program.) The ?DQTSK system call removes one or more ?TASK packets from the queued task manager's initiation queue.

Stack Space Allocation and Stack Definition

Every task that uses the AOS/VS system calls must have a unique stack. A stack is a block of consecutive memory locations that AOS/VS sets aside for task-specific information.

The stack works by a push-down/pop-up mechanism; that is, you store information by "pushing" it onto the stack, and retrieve information by "popping" it off the stack. The 'Principles of Operation 32-Bit ECLIPSE Systems' manual explains stacks in detail and describes the assembly language instructions for the push and pop functions.

The Link utility allocates the stack for the initial task when you link your program. By default, Link sets up a stack of 60 words for the initial task. You can specify an alternate size by using the appropriate function switch in the Link command line.

You must allocate stack space and define the stacks for all other tasks within the ?TASK packet(s). The stack parameters in the ?TASK packet include the stack base, or starting address of the stack, the stack size, and the address of the stack fault handler.

The stack fault handler is a routine that takes control when there is a stack fault. You can define your own stack fault handler or you can use the AOS/VS default stack fault handler. To specify the default stack fault handler, set the stack fault handler parameter to -1.

A stack base value of -1 means that you will allocate the stack at a later time (that is, after task initiation). If you choose this option, you must allocate the stack before the newly initiated task issues any system calls. You must allocate a stack of at least 30 double words (60 words).

Inner-Ring Stacks

A task that tries to enter an inner ring via an LCALL instruction cannot succeed unless there is a 32-bit stack (called a wide stack) already defined in the target ring for that task. When you load a segment image into an inner ring, inner-ring stacks must be initialized for all tasks that may want to enter that ring. This section describes the rules that govern the inner-ring stack initialization that AOS/VS performs when you issue the ?RINGLD system call. (See Chapter 3 for information on loading a program file into a specific ring with ?RINGLD.)

Every process begins executing in Ring 7. You can specify the Ring 7 stack for the initial task of the process either when you link or after the initial task begins to execute.

The ?RINGLD system call initializes inner-ring wide stacks on behalf of all possible tasks in an inner ring. You can specify the size of these initial stacks at one of the following times:

o When you compile your program.

  To do this, the compiler initializes locations 20 through 27 (the wide-stack parameters) of the process image. Then, at ?RINGLD time, AOS/VS partitions the region delimited by the stack base and the stack limit into separate stacks of equal size for all of the tasks in the process.

o When you link your program into a program file.

  To do this, you must specify the following in your Link command line:

```
/STACK=n
```

  where n = (number of tasks) * (stack size per task)

  Link allocates n words at the end of your unshared area. At ?RINGLD time, AOS/VS partitions this n-word region into separate stacks for each task in the process. Although n can be as few as 12 double words, we recommend that you allocate at least 60 double words per task for n.

If you specify the segment image's initial stack size when you link, AOS/VS uses that size to override any stack size that you may have specified at compile time.

When you link an inner-ring segment image, you should also specify a value for the /TASKS= switch. The number that you choose must be greater than or equal to the number of possible tasks specified for the (Ring 7) process image. (Note that a general-purpose local server should be linked for 32 tasks.)

When you issue ?RINGLD, AOS/VS performs the following steps:

1. AOS/VS loads the segment image into the inner ring for which it was linked.

2. AOS/VS initializes wide stacks in the specified ring for all tasks of the process.

   AOS/VS gets the size of the total available stack region from locations 20 through 27 (the wide-stack parameters) of the ring. Then, AOS/VS divides the region into equal-sized wide stacks for each possible task in the process. The size of each stack is the size that was implicitly set at either compile or Link time.

Typically, AOS/VS performs the following steps to initialize the inner-ring stacks:

1. AOS/VS sets the frame pointer, the stack pointer, and the stack base to the start of the task's stack region.

2. AOS/VS sets the stack limit to the end of the stack region minus 2 frames.

3. AOS/VS sets the stack overflow handler address to the address that you specified in Page 0 of the segment image.

It is possible to force AOS/VS to initialize a single common inner-ring stack for all tasks in the process. To do this, set the stack pointer within the segment image so that it contains the same value as the stack limit. Then, at ?RINGLD time, AOS/VS initializes all the stacks within the inner ring so that they have the same stack pointer, frame pointer, stack base, and stack limit.

?TASK system calls can be issued from any loaded user ring. If a task in an inner ring issues a ?TASK system call, it can initiate a task in that ring or in any higher, loaded user ring. It can specify new wide-stack parameters for the new task. Offsets ?DSTB, ?DSFLT, and ?DSSZ of the ?TASK packet allow the caller to initialize new stack parameters for the task in the ring specified by the new task's initial PC (offset ?DPC).

The ?TASK system call causes AOS/VS to reset the wide stacks for the new task in all user rings lower than the ring specified in ?DPC. AOS/VS resets wide stacks by resetting the stack pointer and the frame pointer to the stack base. This ensures that tasks can re-use the same stack sequentially several times in a ?TASK/?KILL/?TASK/?KILL sequence.

Once a new task has been initiated, it is free to allocate a new wide stack for itself at any time. However, it is the responsibility of the task to recycle the old wide-stack memory, if the process wishes to re-use the memory.

Task Scheduling

AOS/VS schedules tasks according to a strict priority scheduling algorithm applied at the task level.

After a process's initial task begins to execute under AOS/VS, you can change its task priority at any time by issuing either the ?PRI or the ?IDPRI system call.

To change a process's own priority, you can issue the ?PRIPR system call. However, if you want to change the priority of another process, the calling process must be in Superprocess mode.

Tasks pass through several different states while a process is executing. A task passes from the inactive to the active state when you initiate it with the ?TASK system call. After a task is active, it can become ready or suspended. Figure 6-1 illustrates the task states and the system calls that affect them.

AOS/VS reschedules tasks under the following circumstances:

- When the task that is executing becomes suspended.
- When a suspended task of a higher priority than the task that is currently executing becomes ready to run.
- When there is more than one highest priority-level task that is ready to run and a round-robin interval has elapsed. (You specify the round-robin interval during the system-generation procedure.)

```
 ____________________________________________________________________
|                                                                    |
|                          RTN, ?KILL                                 |
|                                                                    |
|         _____________________________________________________      |
|         |                                                    |     |
|   ______V_____          ___________                    _____|_____ |
|  |            | ?TASK  |  Active  |  Task Scheduler   |  Active  | |
|  |  Inactive  |------->|  Ready   |------------------>|Executing | |
|  |____________|        |__________|                   |__________| |
|                   ?IDRDY ^    | ?IDSUS                 ?SUS      |  |
|                   ?PRRDY |    | ?PRSUS                 ?IDSUS    |  |
|                   ?XMT   |    |                        ?PRSUS    |  |
|                          |    |                        ?REC      |  |
|                          |    |       ___________      ?XMTW     |  |
|                          |    |----->|  Active   |               |  |
|                          |___________|Suspended  |<--------------|  |
|                                      |___________|                  |
|____________________________________________________________________|
```

Figure 6-1. Task States

To disable scheduling, you can issue either the ?DRSCH system call, which does not return an indication of the prior state of scheduling, or you can issue the ?DFRSCH system call, which does. Both the ?DRSCH and the ?DFRSCH system calls are very dangerous in that they can disrupt the entire multitasking environment. Therefore, do not use these system calls unless you are very certain that they are precisely what you need.

To re-enable scheduling after you have disabled it with a ?DRSCH or a ?DFRSCH system call, issue ?ERSCH. (See "Critical Region Locking/Unlocking" for more information on the ?DRSCH and the ?DFRSCH system calls.)

Task Suspension

Several different events, including some system calls, will suspend an active task. To explicitly suspend a task, issue one of the following system calls: ?SUS, ?IDSUS, or ?PRSUS. Certain other system calls suspend the calling task while they perform their functions. System calls of this kind include the I/O system calls ?READ and ?WRITE, system calls to acquire system resources, and system calls that depend on another task's response, such as the ?XMTW and ?REC system calls.

Tasks compete for all system resources (including the CPU). Only "ready" tasks can compete for the CPU. A task is ready if it is not waiting for some event to complete (that is, suspend). (See "Task Readying" for more information on readying tasks.) If a task is not ready, then it is suspended.

A task becomes suspended when it:

- Is part of a process that the ?BLKPR system call has blocked. To do this, the ?BLKPR system call suspends all tasks within the process. (See Chapter 3 for more information on blocked processes.)

- Issues an explicit request to suspend itself or another task within the same process (via the ?IDSUS and ?SUS system calls).

- Issues an explicit request to wait for a message from another task within the same process (via the ?REC and ?XMTW system calls).

- Issues certain (most) system calls. A system call is usually a request to use some system resource.

If every task in a process is suspended, then that process is blocked. To block a process (that is, suspend every task), you must issue the ?BLKPR system call. When you have explicitly blocked a process with the ?BLKPR system call, you must issue the ?UBLPR system call to unblock that process.

The ?WDELAY system call suspends a task for a specific amount of time. This allows you to synchronize tasks or to temporarily suspend a task until some asynchronous event has completed.

Task Readying

A task remains suspended until the event that caused the suspension completes or until the suspended task is "readied" by AOS/VS or by another task.

Tasks become ready when:

- A task that was suspended by a ?BLKPR system call against its process is explicitly unblocked by the ?UBLPR system call and the task is not suspended for any other reason.

  The ?BLKPR and ?UBLPR system calls work together. Therefore, the ?UBLPR system call can only unblock processes that were blocked by the ?BLKPR system call.

- A task issues an ?IDRDY or a ?PRRDY system call to explicitly request that AOS/VS ready another task. (The ?IDRDY system call readies a task of a given TID and the ?PRRDY system call readies all tasks of a given priority.)

  In this case, the task that is being readied must have been previously suspended by a ?SUS, ?IDSUS, or ?PRSUS system call. In addition, the task that is being readied must belong to the same process as the task that issues the ?IDRDY system call.

- A message for which the task was explicitly requested to wait becomes available. In this case, the task only becomes ready when the message is from another task within the same process.

- A system resource becomes available after an implicit wait for that system resource during a system call.

- A task issues a task-kill system call (?IDKIL or ?PRKIL) or a redirection system call (?IDGOTO) against a suspended task. (Before AOS/VS executes the system call, it automatically readies the target task.)

Task Redirection

To redirect a task's activity without killing it, you must issue the ?IDGOTO system call. The ?IDGOTO system call stops the task's current activity (or readies the task, if the task was suspended) and then directs the task to a new location. The task begins executing at the new location as soon as it regains control of the CPU. The task's priority remains the same.

Typically, you use ?IDGOTO to interrupt a task after a CTRL-C CTRL-A console interrupt sequence. (A CTRL-C CTRL-A sequence interrupts console output. For details about this function, see the description of ?IDGOTO in Chapter 13.)

Inner-Ring Task Redirection Protection

Tasks executing in critical sections of an inner ring cannot tolerate being redirected by tasks executing in outer rings. However, task redirection is a common method of responding to external events. In fact, typing a CTRL-C CTRL-A console interrupt sequence frequently causes an ?IDGOTO system call to perform task redirection on the main task(s) of a process. Therefore, to solve this problem, AOS/VS provides you with the ?TLOCK and ?TUNLOCK system calls, which allow you to control whether a task can be redirected by a task-redirection system call. (The task-redirection system calls are ?IDGOTO, ?IDKIL, ?PRKIL, ?IDSUS, and ?PRSUS.)

The ?TLOCK system call allows a task that is executing in an inner ring to lock itself against task-redirection system calls issued by another task that is executing in a higher ring of the same process. The ?TUNLOCK system call unlocks a previously locked task.

A task can issue a ?TLOCK system call to protect itself from being redirected by any task that is in a higher ring or, optionally, in the same ring. The ring-maximization protection scheme governs which tasks can and cannot redirect a task. (In other words, only a task-redirection system call that originates from the same ring or in a lower ring can redirect a locked task.)

If a task issues a task-redirection system call, but the task it wants to redirect (the target task) is locked, the calling task waits until the target task issues enough ?TUNLOCK system calls to unlock the rings that are lower than the ring in which the calling task resides.

If a task issues a ?PRKIL or a ?PRSUS system call whose input priority specifies more than one protected task, AOS/VS makes a note of all tasks of that priority when the ?PRKIL or ?PRSUS system call occurred. If the redirecting task must wait because one or more target tasks are locked, the task will only wait until all the noted locked tasks issue enough ?TUNLOCK system calls to allow the redirection to occur. If a redirecting task specifies more than one task, the redirections may occur separately (depending on whether one or more of the target tasks are locked). However, in this case, the task-redirection system call will not complete until all the specified tasks have been redirected.

As input to the ?TLOCK system call, you can specify a double-word mailbox in AC2, if you want AOS/VS to inform your protected task when another task is trying to redirect it. AOS/VS will set a nonzero flag in this mailbox if another task's redirection request is waiting.

To protect a task from being redirected by another task within the same ring, set the ?TMYRING flag in AC0 when you issue the ?TLOCK system call.

If a task in an inner ring is redirected to a higher ring, then AOS/VS resets the stack pointer and frame pointer for each affected inner ring to the stack base of that ring on behalf of all loaded user rings that are less than or equal to the redirected higher ring. This means that if a task in Ring 5 is redirected to Ring 7, AOS/VS resets the task's stack and frame pointers for Rings 5 and 6.

Task Termination

You can kill (terminate) a task explicitly or implicitly. To explicitly kill a task, issue one of the following system calls:

?IDKIL	Kills a task of a certain TID.
?PRKIL	Kills all tasks of a certain priority.
?KILL	Kills the calling task.

To kill a task implicitly, begin the new task with a WSSVS or WSSVR (wide-save) instruction, and end it with a WRTN (wide-return) instruction. As AOS/VS executes the initial wide-save instruction, it saves the contents of AC3 as the return address for the task. At this point, AC3 contains the address of the task-kill routine (placed in AC3 during task initiation). When AOS/VS executes the WRTN, it passes control to the return address in AC3; that is, the task-kill routine.

Because killing a task does not guarantee an orderly release of its user-related resources, you may want to define a kill-processing routine for this purpose (for example, to close the task's currently open channels).

You can define either a unique kill-processing routine for each task or a general kill-processing routine for all tasks within a process. ?KILAD, which you issue after task initiation, defines a unique kill-processing routine that is then invoked when you issue ?IDKIL or ?PRKIL. If you define a general kill-processing routine, assign the routine the label ?UKIL and link it with your program. You can use both ?KILAD and user-defined ?UKIL kill-processing routines within the same program.

If there is no user-defined ?UKIL routine to kill a task, AOS/VS uses the dummy ?UKIL routine in URT32.LB. This routine returns control to AOS/VS, which then kills the task. ?UKIL kill processing is only invoked on behalf of tasks that initiated processing within Ring 7 (that is, tasks whose initial PCs are Ring 7 addresses).

Task Creation and Termination Detection

Typically, a local server needs to maintain accurate task-specific databases. Therefore, to keep those task-specific databases accurate, a local server must be able to keep track of when tasks are created and when they terminate. This section describes how AOS/VS helps an inner-ring server to detect when a task is created and when it is terminated.

All active tasks have distinct Unique Storage Position (USP) pointers associated with Rings 4 through 6. Tasks within 32-bit processes also have a USP pointer associated with Ring 7. A double-word pointer at location ?USP within a ring specifies the USP pointer for a given task within the ring. The USP pointer allows tasks to keep track of task-specific databases associated with a particular ring.

When a process issues a ?TASK system call to create a task, AOS/VS initializes all the USP pointers associated with that task to zero. When a customer issues LCALL to enter a local server, the local server can examine the USP pointer to that inner ring. The local server can interpret a zero USP pointer to mean that this is the task's first visit to the local server. In this case, the local server can initialize any task-specific databases for that initially entering task.

AOS/VS uniquely identifies every task within a process to aid in identifying task-specific databases with their tasks. The ?UIDSTAT system call returns the unique TID associated with a given task.

When a task terminates, AOS/VS serially invokes a ?UKIL postprocessor for each loaded user ring whose ring number is less than or equal to the ring specified by the task's initial PC. Local servers can use the ?UKIL postprocessor to update or deallocate task-specific databases, as appropriate. The ?UKIL routine should not issue system calls.

Several ?UKIL postprocessors (one per ring) can be associated with a process. However, only one ?UTSK postprocessor can be associated with a process. AOS/VS only invokes a ?UTSK postprocessor on behalf of tasks that are to be executed in Ring 7. The ?UTSK postprocessor must reside in Ring 7.

Console-to-Task Communication

AOS/VS allows you to pass a message from your console to individual tasks in a multitasking environment.

The ?TRCON system call creates a message-management system task on your behalf, which parses each message from you and transmits that message to the proper calling task.

Task-to-Task Communication

AOS/VS provides an intertask communications facility that you can use to synchronize tasks or pass messages among them. The following system calls allow tasks to communicate with one another:

?XMT	Transmits an intertask message.
?XMTW	Transmits an intertask message and awaits its reception.
?REC	Receives an intertask message; suspends the ?REC caller if there is no message currently available.
?RECNW	Receives an intertask message; does not suspend the ?REC caller if there is no message currently available.

Tasks deposit messages in and retrieve them from 32-bit locations called mailboxes. Before you send a message with an ?XMT or an ?XMTW system call, you must initialize the appropriate mailbox to zero.

Timing is a factor for both the ?XMTW and the ?REC system call. If a sending task issues an ?XMTW system call before another task issues a complementary receive, AOS/VS suspends the sender until the receive occurs. Likewise, if a task issues an ?REC system call against an empty mailbox (the sender has not transmitted the message yet), AOS/VS suspends the receiver until the transmission occurs.

The ?XMT and ?RECNW system calls maintain the calling task in the ready state, regardless of the timing of the transmit and receive sequence. If a task issues an ?RECNW system call against an empty mailbox, the system call fails, and AOS/VS returns an error code to AC0.

You can use the ?XMT and ?XMTW system calls to "broadcast" a message; that is, to send the message to all tasks currently waiting for the message. If you do not select the broadcast option and more than one task is waiting for the message, AOS/VS sends the message to the receiver with the highest priority.

Critical Region Locking/Unlocking

You can use the intertask communications system calls to lock or unlock a critical region. A critical region is a procedure or database that all tasks share, but that is available to only one task at a time. To protect a critical region, you must define a mailbox to synchronize task execution within the critical region. A task gains control of a critical region by issuing a successful receive against that mailbox. The procedure for locking and unlocking a critical region is as follows:

- First, a task initializes the locking facility, either by setting the mailbox to a nonzero value or by issuing the ?XMT system call "without broadcast" from the initializing task to the mailbox. (The ?XMT system call message may specify the address of the critical region.)

- Second, a task locks (gains exclusive control of) the critical region by issuing an ?REC system call against the mailbox. AOS/VS suspends other tasks that issue subsequent ?REC system calls against the mailbox.

Once a task has locked a critical region, it remains locked until the task issues another ?XMT system call to unlock it. If more than one task is waiting for control of a critical region (that is, more than one task was suspended by a ?REC system call to the mailbox), the second ?XMT system call readies the highest priority receiver, which then gains control of the critical region.

You can also lock a critical region implicitly by issuing a ?DRSCH system call, which disables all task scheduling in the calling process, or a ?DFRSCH system call, which not only disables all task scheduling in the calling process, but also returns an indication of the prior state of scheduling. If you use a ?DRSCH or a ?DFRSCH system call to lock a critical region, you should use a ?ERSCH system call to unlock it. However, ?DRSCH and ?DFRSCH system calls can be dangerous because they disable all multitask scheduling for the calling process.

Unless absolutely necessary, you should avoid using the ?DRSCH and the ?DFRSCH system calls. Although there may be times when you need to issue one of these system calls, such as to control a "race" condition between two tasks that are competing for the same critical region, you must use them with discretion. Disabling task scheduling, even briefly, can disrupt the entire multitasking environment.

The ?ERSCH system call re-enables multitask scheduling for the calling process.

MV/8000 Floating-Point Registers

The MV/8000 hardware has five registers that allow you to manipulate floating-point numbers:

o Four floating-point registers, FAC0, FAC1, FAC2, and FAC3.

o One floating-point status register, FPSR, which records information about the current state of the MV/8000 floating-point processor.

Before you can use any of the MV/8000 floating-point instructions from a task, you must issue ?IFPU to initialize the floating-point status register. To obtain accurate results for floating-point arithmetic, you must do this even for single-task programs.

Multitasking Sample Programs

The initial task of the following program, NEWTSK, creates a new task that has a priority of 1 and a TID of 2. The initial task opens the console, creates the new task, announces its death, gets its priority, and kills itself. Then, the new task takes control, writes a message, and returns to the CLI. (The last task cannot kill itself with a ?TASK system call.)

NEWTSK uses the ?TASK, ?MYTID, and ?IDKIL system calls.

```
                        .TITLE  NEWTSK
                        .ENT    NEWTSK
                        .TSK    2

;Open console (CON), create a new task, and kill self.

NEWTSK: ?OPEN   CON                     ;Open console (CON) for I/O.
        WBR     ERROR                   ;?OPEN error return.

        ?TASK   TPKT                    ;Create new task, TID 2, with
                                        ;priority of 1.
        WBR     ERROR                   ;?TASK error return.
        ?WRITE  CON                     ;Display termination message
                                        ;on console.
        WBR     ERROR                   ;?WRITE error return.
        ?MYTID                          ;Get TID in AC0 and priority
                                        ;in AC1.
        WBR     ERROR                   ;?MYTID error return.
        WMOV    0,1                     ;Move TID into AC1
        ?IDKIL                          ;and die.
        WBR     ERROR                   ;?IDKIL error return.

;New task is now the only task.

NTSK:   XLEFB   0,NMSG*2                ;Get byte pointer to message.
        XWSTA   0,CON+?IBAD             ;Put message in I/O packet.
        ?WRITE  CON                     ;Display message on console.
        WBR     ERROR                   ;?WRITE error return.
        WSUB    2,2                     ;Set AC2 for normal return.
        WBR     BYE                     ;Go and return.

;Error handler.

ERROR:  NLDAI   ?RFEC!?RFCF!?RFER,2     ;Error flags: Error code is in
                                        ;AC0 (?RFEC), message is in
                                        ;CLI format (?RFCF), and caller
                                        ;should handle this as an error
                                        ;(?RFER).
```

NEWTSK Program (Cont.)

```
BYE:    ?RETURN                           ;Return to CLI.
        WBR     ERROR                     ;?RETURN error return.

;?OPEN and I/O packet for console.

CON:    .BLK    ?IBLT                     ;Allocate enough space for
                                          ;packet.
        .LOC    CON+?ISTI                 ;File specifications.
        .WORD   ?ICRF!?RTDS!?OFIO         ;Change format to data-
                                          ;sensitive records and open
                                          ;for input and output.

        .LOC    CON+?IMRS
        .WORD   -1                        ;Default physical block size
                                          ;to 2K bytes.

        .LOC    CON+?IBAD
        .DWORD  ITEXT*2                   ;Byte pointer to record I/O
                                          ;buffer.

        .LOC    CON+?IRCL
        .WORD   120.                      ;Record length is 120
                                          ;characters.

        .LOC    CON+?IFNP
        .DWORD  CONS*2                    ;Byte pointer to pathname.

        .LOC    CON+?IDEL                 ;Delimiter table address.
        .DWORD  -1                        ;Use default delimiters: null,
                                          ;NEW LINE, form feed, and
                                          ;carriage return (default is
                                          ;-1).

        .LOC    CON+?IBLT                 ;End of packet.

;Filename and messages.  A .NOLOC 1 follows.

CONS:   .TXT    "@CONSOLE"                ;Use generic name.

ITEXT:  .TXT    "I'm the default task.  I have opened the console and
                I'm about to ?IDKIL myself.<12>"

NMSG:   .TXT    "I'm the new task.  I am about to ?RETURN.<12>"

        .NOLOC  0
```

```
;?TASK packet for new task.

TPKT:   .BLK    ?DSLTH              ;Allocate enough space for the
                                    ;standard packet.

        .LOC    TPKT+?DLNK
        .WORD   1                   ;Set to 1 for standard packet.

        .LOC    TPKT+?DLNL          ;Reserved.
        .WORD   0                   ;Set to 0.

        .LOC    TPKT+?DLNKB         ;Reserved.
        .DWORD  0                   ;Set to 0.

        .LOC    TPKT+?DPRI
        .WORD   1                   ;Assign priority 1 to the new
                                    ;task (default is 0, which
                                    ;assigns the new task the same
                                    ;priority as the caller).

        .LOC    TPKT+?DID
        .WORD   2                   ;Assign TID 2 to the new
                                    ;task (default is 0, which
                                    ;does not assign a TID to
                                    ;the new task).
        .LOC    TPKT+?DPC
        .DWORD  NTSK                ;Task's starting address is
                                    ;NTSK.

        .LOC    TPKT+?DAC2
        .DWORD  0                   ;There is no message for the
                                    ;new task.

        .LOC    TPKT+?DSTB
        .DWORD  STACK               ;Stack base address is STACK.

        .LOC    TPKT+?DSFLT         ;Stack fault handler address.
        .WORD   -1                  ;Use default stack fault
                                    ;handler in URT32.LB (default
                                    ;is -1).
        .LOC    TPKT+?DSSZ
        .DWORD  60.                 ;Stack size is 60 words.

        .LOC    TPKT+?DFLGS         ;Task flag word.
        .WORD   0                   ;Set to 0.
```

NEWTSK Program (Cont.)

```
        .LOC    TPKT+?DRES              ;Reserved.
        .WORD   0                       ;Set to 0.

        .LOC    TPKT+?DNUM
        .WORD   1                       ;Create one task.

        .LOC    TPKT+?DSLTH             ;End of packet.

STACK:  .BLK    60.                     ;60-word stack for new task.

        .END    NEWTSK                  ;End of NEWTSK program.
```

The following program, BOOMER, is a fast, two-task copy program that uses ?IXMT and ?REC system calls to synchronize ?READ and ?WRITE system calls. BOOMER copies an existing input file to an output file.

BOOMER uses the ?TASK, ?XMTW, ?REC, ?KILL, ?IXMT, ?READ, and ?WRITE system calls.

```
                              .TITLE  BOOMER
                              .ENT    BOOMER
                              .TSK    2
                              .NREL   1

;Initial task uses ?GTMES to get output filename (second argument) and
;opens it.  Repeats ?GTMES to get input filename (first argument) and
;opens it.  Creates output task.

BOOMER: ?GTMES  GPKT                     ;Get input filename.
        WBR     ERROR                    ;?GTMES error return.
        LLEFB   0,FNAME*2                ;Get byte address of filename
                                         ;that ?GTMES returns.
        LWSTA   0,INPUT+?IFNP            ;Put in input I/O packet.

        ?OPEN   INPUT                    ;Open INPUT file.
        WBR     ERROR                    ;?OPEN error return.
        NLDAI   1,0                      ;Get 1 in AC0.
        LNSTA   0,GPKT+?GNUM             ;Specify argument 1.

        ?GTMES  GPKT                     ;Get output filename.
        WBR     ERROR                    ;?GTMES error return.
        LLEFB   0,FNAME*2                ;Get byte address of filename
                                         ;that ?GTMES returns.
        LWSTA   0,OUTPUT+?IFNP           ;Put in output I/O packet.

        ?OPEN   OUTPUT                   ;Open OUTPUT file.
        WBR     ERROR                    ;?OPEN error return.

        ?TASK   TPKT                     ;Create output task.
        WBR     ERROR                    ;?TASK error return.

;Loop reads into BUF1, transmits it to output task, reads into BUF2,
;and transmits it to output task.  Message for output task is buffer
;address.

READER: ?READ   INPUT                    ;Read buffer from INPUT file.
        WBR     ERROR                    ;?READ error return.
        LLEF    0,MAILBOX                ;Get message address.
        LWLDA   1,INPUT+?IBAD            ;Message is buffer address.

        ?XMTW                            ;Wake up output task.
        WBR     ERROR                    ;?XMTW error return.
```

BOOMER Program (Cont.)

```
;Swap buffer byte pointers for next read.

        LLEFB   0,BUF1*2                ;Get byte pointer to BUF1.
        LLEFB   2,BUF2*2                ;Get byte pointer to BUF2.
        WSNE    1,2                     ;Was BUF1 used for last read?
        WMOV    0,2                     ;No. Make BUF1 current buffer.
        LWSTA   2,INPUT+?IBAD           ;Yes. Put byte pointer to
                                        ;current buffer into input
                                        ;packet.
        WBR     READER                  ;Read into current buffer.

;On end-of-file condition, get number of characters to read from input
;packet and make this number the buffer length for the last ?XMT.

EOF?:   NLDAI   EREOF,1                 ;Was error code "end-of-file"
                                        ;(EREOF)?
        WSEQ    0,1                     ;Yes.
        WBR     ERROR                   ;No.  Try to handle the error.
        LLEF    0,MAILBOX               ;Get address of message.
        LWLDA   1,INPUT+?IBAD           ;Message is byte pointer to
                                        ;buffer.
        WMOV    1,2                     ;Copy to AC2 for indexing.
        NLDAI   -1,3                    ;Put -1 in AC3.
        WLSH    3,2                     ;Make byte pointer to buffer
                                        ;a word pointer.
        LNLDA   3,INPUT+?IRLR           ;Get number of characters read
                                        ;from input I/O packet.
        LWSTA   3,-2,2                  ;Make buffer length (AC2-2)
                                        ;the number of characters
                                        ;read.

        ?XMTW                           ;Send last buffer.
        WBR     ERROR                   ;?XMTW error return.

        ?KILL                           ;Input is done; output task
                                        ;will return to CLI.

;Error handler.

ERROR:  WLDAI   ?RFEC!?RFCF!?RFER,2     ;Error flags: Error code is in
                                        ;AC0 (?RFEC), message is in
                                        ;CLI format (?RFCF), and caller
                                        ;should handle this as an error
                                        ;(?RFER).

        ?RETURN                         ;Return to CLI.
        WBR     ERROR                   ;?RETURN error return.
```

BOOMER Program (Cont.)

```
;Output task does the writing:

WRITER: LLEF     0,MAILBOX                  ;Get message address.
        ?REC                                ;Wait for message.
        WBR      ERROR                      ;?REC error return.
        LWSTA    1,OUTPUT+?IBAD             ;Got message, which was byte
                                            ;pointer to buffer.  Put in
                                            ;I/O packet.
        WMOV     1,2                        ;Copy to AC2 for indexing.
        NLDAI    -1,3                       ;Put -1 in AC3.
        WLSH     3,2                        ;Make byte pointer into word
                                            ;pointer.
        XWLDA    0,-2,2                     ;Get buffer length left by
                                            ;input task (original length,
                                            ;unless task hit end of file).
        LNSTA    0,OUTPUT+?IRCL             ;Make this maximum receive
                                            ;length in I/O packet.

        ?WRITE   OUTPUT                     ;Write buffer to OUTPUT file.
        WBR      ERROR                      ;?WRITE error return.
        WLDAI    BUFLGTH,1                  ;Get original buffer length.
        WSNE     0,1                        ;Is current buffer length same
                                            ;as original buffer length?
        WBR      WRITER                     ;Yes.  Get another buffer.
        WSUB     2,2                        ;No.  Done.  Set for normal
                                            ;return.

        ?RETURN                             ;Return to CLI.
        WBR      ERROR                      ;?RETURN error return.

;Buffers, message, packets in unshared code.

        .NREL

;Buffer declarations.

BUFLGTH = 16384.                            ;Need to change only this
        BUFLGTH                             ;for residual characters after
                                            ;end of file.
BUF1:   .BLK     (BUFLGTH+1)/2              ;Size of BUF1
        BUFLGTH                             ;for residual characters after
                                            ;end of file.
BUF2:   .BLK     (BUFLGTH+1)/2              ;Size of BUF2.

;Mailbox for message.

MAILBOX: 0
```

BOOMER Program (Cont.)

```
    ;?GTMES packet to get input and output filenames.

    GPKT:   .BLK     ?GTLN                  ;Allocate enough space for
                                            ;packet.

            .LOC     GPKT+?GREQ             ;Request type.
            .WORD    ?GARG                  ;Put argument in ?GRES only.

            .LOC     GPKT+?GNUM
            .WORD    2                      ;Argument 2 is input filename.

            .LOC     GPKT+?GRES

            .DWORD   FNAME*2                ;Byte pointer to receive
                                            ;buffer.

            .LOC     GPKT+?GTLN             ;End of packet.

    ;?OPEN and I/O packet for input task.

    INPUT:  .BLK     ?IBLT                  ;Allocate enough space for
                                            ;packet.

            .LOC     INPUT+?ISTI            ;File specifications.
            .WORD    ?ICRF!?RTDY!?OFIN      ;Change format to dynamic-
                                            ;length records and open for
                                            ;input only.

            .LOC     INPUT+?IMRS
            .WORD    -1                     ;Default physical block size
                                            ;to 2K bytes.

            .LOC     INPUT+?IBAD
            .DWORD   BUF1*2                 ;Byte pointer to record I/O
                                            ;buffer.

            .LOC     INPUT+?IRCL
            .WORD    BUFLGTH                ;Record length is BUFLGTH.

            .LOC     INPUT+?IRLR
            .WORD    0                      ;Set to 0 (used by ?READ and
                                            ;?WRITE only).

            .LOC     INPUT+?IFNP
            .DWORD   FNAME*2                ;Byte pointer to pathname.
```

```
        .LOC    INPUT+?IDEL
        .DWORD  -1                      ;Use default delimiters: null,
                                        ;NEW LINE, form feed, and
                                        ;carriage return (default is
                                        ;-1).

        .LOC    INPUT+?IBLT             ;End of packet.

;?TASK packet for output task (minimum packet).

TPKT:   .BLK    ?DSLTH                  ;Allocate enough space for the
                                        ;standard packet.

        .LOC    TPKT+?DLNK
        .WORD   1                       ;Set to 1 for standard packet.

        .LOC    TPKT+?DLNL              ;Reserved.
        .WORD   0                       ;Set to 0.

        .LOC    TPKT+?DLNKB             ;Reserved
        .DWORD  0                       ;Set to 0.

        .LOC    TPKT+?DPRI
        .WORD   1                       ;Assign priority 1 to the new
                                        ;task (default is 0, which
                                        ;assigns the new task the same
                                        ;priority as the caller).
        .LOC    TPKT+?DID
        .WORD   2                       ;Assign TID 2 to the new
                                        ;task (default is 0, which
                                        ;does not assign a TID to
                                        ;the new task).
        .LOC    TPKT+?DPC
        .DWORD  WRITER                  ;Task's starting address is
                                        ;WRITER.

        .LOC    TPKT+?DAC2
        .DWORD  0                       ;There is no message for the
                                        ;new task.

        .LOC    TPKT+?DSTB
        .DWORD  STACK                   ;Stack base address is STACK.

        .LOC    TPKT+?DSFLT
        .WORD   -1                      ;Use default stack fault
                                        ;handler in URT32.LB (default
                                        ;is -1).
```

BOOMER Program (Cont.)

```
          .LOC     TPKT+?DSSZ
          .DWORD   60.                        ;Stack size is 60 words.

          .LOC     TPKT+?DFLGS                ;Task flag word.
          .WORD    0                          ;Set to 0.

          .LOC     TPKT+?DRES                 ;Reserved.
          .WORD    0                          ;Set to 0.

          .LOC     TPKT+?DNUM
          .WORD    1                          ;Create one task.

          .LOC     TPKT+?DSLTH                ;End of packet.

 ;?OPEN and I/O packet for output task.

 OUTPUT:  .BLK     ?IBLT                      ;Allocate enough space for
                                              ;packet.

          .LOC     OUTPUT+?ISTI               ;File specifications.
          .WORD    ?OFCR!?OFCE!?ICRF!?RTDY!?OFIO   ;Delete file, recreate
                                              ;file, change format to
                                              ;dynamic-length records, and
                                              ;open for input and output.

          .LOC     OUTPUT+?IMRS               ;Physical block size (in
                                              ;bytes).
          .WORD    -1                         ;Block size is 2K bytes
                                              ;default is -1).

          .LOC     OUTPUT+?IBAD
          .DWORD   BUF1*2                     ;Byte pointer to record I/O
                                              ;buffer.

          .LOC     OUTPUT+?IRCL
          .WORD    BUFLGTH                    ;Record length is BUFLGTH.

          .LOC     OUTPUT+?IRLR
          .WORD    0                          ;AOS/VS returns characters
                                              ;transferred (used by ?READ
                                              ;and ?WRITE only).

          .LOC     OUTPUT+?IFNP
          .DWORD   FNAME*2                    ;Byte pointer to pathname.
```

BOOMER Program (Cont.)

```
        .LOC    OUTPUT+?IDEL
        .DWORD  -1                      ;Use default delimiters: null,
                                        ;NEW LINE, form feed, and
                                        ;carriage return (default is
                                        ;-1).

        .LOC    OUTPUT+?IBLT            ;End of packet.

FNAME:  .BLK    (?MXPL+1)/2             ;Filename buffer.  System
                                        ;limit for number of
                                        ;characters.

STACK:  .BLK    60.                     ;60-word task stack.

        .END    BOOMER                  ;End of BOOMER program.
```

End of Chapter

CHAPTER 7
THE INTERPROCESS COMMUNICATIONS (IPC) FACILITY

The IPC system calls are:

?GCPN	Returns the global port number of the target process's console.
?GPORT	Returns the PID associated with a global port number.
?ILKUP	Returns a global port number.
?IMERGE	Modifies a ring field within a global port number.
?IREC	Receives an IPC message.
?ISEND	Sends an IPC message.
?ISPLIT	Finds the owner of a port (including its ring number).
?IS.R	Sends and then receives an IPC message.
?TPORT	Translates a local port number to its global equivalent.

AOS/VS allows processes to communicate with each other through the Interprocess Communications (IPC) facility, which allows you to:

- o Transmit variable-length free-form messages from one process to another.

- o Synchronize processes during execution.

You can use the IPC facility to pass arguments from a father process to a son process and return the results to the father before the son terminates. If there is a delay between the father's receive request and the son's message, AOS/VS pends the father process until the son process responds, thereby synchronizing the two processes. AOS/VS uses the IPC facility to send messages to father processes to notify them of their sons' terminations.

The following primitive system calls allow you to send and/or receive IPC messages:

?ISEND	Sends an IPC message.
?IREC	Receives an IPC message.
?IS.R	Sends and then receives an IPC message.

For each of these system calls, you must supply a header (packet) that includes the origin and destination of the message, its length, its address, and other information about the connection.

During each IPC transmission, portions of the sender's header overwrite portions of the receiver's header. In fact, some transmissions consist solely of passing header information from the sender to the receiver.

To use the primitive IPC system calls, ?ISEND and ?IS.R, the calling process must have privilege ?PVIP, which is one of the optional privileges you can specify when you create a process with the ?PROC system call. (See Chapter 3 for information on creating processes.)

If the calling process does not have the ?PVIP privilege, it must use the IPC facility as a standard peripheral device, which it can then access by device-independent I/O techniques. (See Chapter 5 for information on how to do this.) Also, you can use the connection-management facility, which is described in Chapter 8, to establish communications between processes. (Note that if a process is a declared customer under the connection-management facility, it does not need the ?PVIP privilege to issue the ?IS.R system call.)

Sending Messages Between IPC Ports

AOS/VS sends IPC messages between ports. Ports are full-duplex communications paths that a process identifies by port numbers. There are two types of port numbers:

o Local port numbers

  Local port numbers are values that the IPC caller (either the sender or the receiver) defines to identify its own ports.

o Global port numbers

  Global port numbers uniquely identify each port currently in use system wide. Global port numbers are made up of a process's PID, its local port number, and its ring number. When a process refers to its local port in an IPC system call, AOS/VS translates the local port number to its global equivalent. The ?TPORT system call performs this translation.

When a process sends an IPC message, it defines a local port number for the connection, then it specifies that port number and the destination's global port number in the IPC header. The receiving process issues a complementary receive system call and, like the sender, defines its own local port number and specifies the sender's global port number. If the port specifications on both ends match (including the target ring), AOS/VS sends the message.

NOTE: Only a specific task in the target ring can receive the IPC message. Therefore, it is very important that you specify the target ring. This prevents a task in one ring from intercepting an message intended for a task that is executing in another ring.

A process must use a global port number to refer to another process's port. However, because global port numbers depend on the system environment, they frequently change during subsequent process execution. To circumvent this problem, potential IPC users can issue the ?CREATE system call to create IPC files, which serve as ports. Then, these same users can define the local port numbers before they issue IPC system calls. As AOS/VS executes the ?CREATE system call, it translates the local port numbers into global port numbers. Potential senders and receivers can then issue ?ILKUP system calls against the IPC file to determine its global port number.

When you issue the ?CREATE system call to create an IPC file, AOS/VS saves the number of the ring from which the system call was issued in the new IPC file. The global port number, which ?ILKUP returns, incorporates this same ring number. AOS/VS interprets all global port numbers as containing ring fields.

The ?ISEND and ?IS.R system calls interpret ring fields (within global port numbers) as follows:

Offset ?IDPH (the global port number) must always contain a valid user ring number. The ring number specifies the ring to which the message will be sent. However, the caller must have appropriate privileges to send a message to that ring within that particular process.

The ?IREC system call interprets ring fields (within global port numbers) as follows:

> Offset ?IOPH (the global port number) can contain either a valid user ring number or a zero ring number. A nonzero ring number indicates that ?IREC returns a message only from sends issued from the specified origin ring within the specified origin process. A zero ring number indicates that ?IREC will return a message from any ring within the specified origin process that sends a message destined for the ?IREC caller's ring. (You can use the ?IMERGE system call to construct a global port number with a zero ring field.)

When you include ring fields as part of global port numbers, the ?IREC port-matching rules are affected in that if the receiver specifies a nonzero ring field in an otherwise zero global header, a ring-specific global receive takes precedence after explicit matches.

To identify the PID that is associated with a particular global port number, you must issue the ?GPORT system call. Conversely, if you know the name of the PID of a console's associated process, you can identify its console port number by issuing the ?GCPN system call.

The ?ISPLIT system call extracts the ring field from a global port number, while the ?IMERGE system call permits both 16- and 32-bit users to modify the ring field within a global port number.

Typical IPC System Call Sequence

The following steps describe a typical IPC sequence:

1. The sending process uses the ?CREATE system call to create an IPC file entry (type ?FIPC) in its working directory. This file entry serves as the origin port for the message. (See Chapter 4 for a description of the ?CREATE system call.)

2. The sending process issues the ?ISEND system call and specifies the following in the header: its own local port number, the receiver's global port number, the length and address of the message buffer, and, optionally, system and user flags.

3. (optional) The receiving process issues the ?ILKUP system call to determine the sender's global port number.

4. The receiving process issues the ?IREC system call and specifies the following in the ?IREC header: its own local port number, the sender's global port number, and, optionally, user flags.

Note that this sequence assumes that the sender issues the ?ISEND system call before the receiver issues the complementary ?IREC system call. In fact, the send and receive system calls need not be sequential. If there is no outstanding message for a receiver, AOS/VS either suspends the receiving task until you issue the ?ISEND system call, or returns an error (an option in the ?IREC headers). Similarly, if there is no ?IREC system call for an ?ISEND system call, AOS/VS either stores the message in the memory buffers or returns an error to the sender (an option in the ?ISEND header).

Send and Receive Headers

The ?ISEND and ?IREC headers consist of ?IPLTH words. The ?IS.R header is identical to the ?ISEND header, except that it contains an extension for receive information, because the ?IS.R system call performs both send and receive functions. The ?IS.R header consists of ?IPRLTH words. Figure 7-1 shows the structures of the IPC headers, and Table 7-1 describes each header offset.

As Table 7-1 shows, the sender specifies the receiver's global port number in offset ?IDPH. When AOS/VS transmits the message, it translates this value to a local port number for the receiver and places it in offset ?IDPN of the receive header.

Similarly, the receiver specifies the sender's global port number in offset ?IOPH. AOS/VS translates this to a local port number during the transmission and records it in offset ?IOPN in the send header.

Offset ?ILTH in the send header contains the length of the IPC message, and offset ?IPTR points to the start of the message in the sender's logical address space. Within the receive header, these same offsets describe the size of the receive buffer and its starting address, respectively. AOS/VS copies the contents of these offsets from the send header to the receive header during the transmission.

If you set ?ILTH to 0 in the send header, you can use offset ?IPTR to send data directly to the header, rather than to a buffer. However, you must set up both the send and receive headers in advance.

?ISEND HEADER

	0 – 15	16 – 31	
?ISFL	System flags	User flags	?IUFL
?IDPH	Destination port number		
?IOPN	Origin port number	Message length (in words)	?ILTH
?IPTR	Message buffer address		

?IPLTH = packet length

?IREC HEADER

	0 – 15	16 – 31	
?ISFL	System flags	User flags	?IUFL
?IOPH	Destination port number		
?IDPN	Origin port number	Message length (in words)	?ILTH
?IPTR	Message buffer address		

?IPLTH = packet length

?IS.R HEADER

	0 – 15	16 – 31	
?ISFL	System flags	User flags	?IUFL
?IDPH	Destination port number		
?IOPN	Origin port number	Message length (in words)	?ILTH
?IPTR	Message buffer address		
?IRSV	Reserved (Set to 0.)	Receive buffer length	?IRLT
?IRPT	Address of receive buffer		

?IPRLTH = packet length

Figure 7-1. Structure of IPC Send and Receive Headers

Table 7-1. Contents of IPC Send and Receive Headers*

Send Header		Receive Header	
Offset	Contents	Offset	Contents
?ISFL	System flags.	?ISFL	System flags.
?IUFL	User flags.	?IUFL	User flags (copied from send header).
?IDPH (double word)	Destination port number.	?IOPH (double word)	Origin port number.
?IOPN	Origin port number.	?IDPN	Destination port number (translated from send header).
?ILTH	Length of message in words.	?ILTH	Length of message buffer words (copied from send header).
?IPTR (double word)	Address of message buffer.	?IPTR (double word)	Address of message buffer.
?IS.R Extension			
?IRSV	Reserved. (Set to 0.)		
?IRLT	Length of the receive buffer.		
?IRPT (double word)	Address of receive buffer.		

* There is no default unless otherwise specified.

System and User Flags

In addition to the origin, destination, and message parameters, the headers for the ?ISEND and ?IREC system calls contain a system flag word (?ISFL) and a user flag word (?IUFL). Table 7-2 describes the optional contents of ?ISFL in the ?ISEND and ?IREC headers.

Table 7-2. Contents of System Flag Word (Offset ?ISFL)

?ISEND Header		?IREC Header	
Flag	Description	Flag	Description
?IFSTM	Loop the message (send the message back to the sender).	?IFRFM	Receive a looped message (sent by this process to itself).
?IFNSP	Do not buffer the message; signal an error if there is no ready receiver.	?IFSOV	Buffer the message if the receive buffer is too small.
		?IFBNK	Signal an error if there is no spooled message for this receiver.
		?IFRING	Contains the sender's ring field (returned by AOS/VS).
		?IFPR	Indicates .PR file type of sender: 0 if sender is a 32-bit process; 1 if sender is a 16-bit process (returned by AOS/VS).

A process can "loop" a message (send a message to itself). To do this, the process must perform the following steps:

1. Issue an ?ISEND system call.

2. Issue an ?IREC system call.

3. Set bit ?IFSTM in the ?ISEND header.

4. Set bit ?IFRFM in the ?IREC header.

Usually, a process loops a message for testing purposes. A processor does not need to specify the origin and destination ports in the headers for a looped message.

Bit ?IFNSP in the ?ISEND header directs AOS/VS to signal an error if there is no outstanding receiver for the sender's message.

Within the ?IREC headers, bit ?IFSOV directs AOS/VS to store the IPC message in the memory buffers if the receive buffer is too small to accommodate it. If the receiver does not set this bit and the receive buffer is too small, AOS/VS transmits as much of the message as possible and discards the overflow.

A receiver can set bit ?IFNBK to direct AOS/VS to return an error if there is no outstanding message for it. Otherwise, AOS/VS suspends the receiving task until the message is sent.

Bits ?IFRING and ?IFPR in the receive header provide the receiver with information about the sending process, such as the sender's ring field (?IFRING) and program type (?IFPR). AOS/VS controls these flag bits; the receiving process cannot set them.

User Flag Word

The user flag word, offset ?IUFL, serves two purposes:

- AOS/VS copies the contents of offset ?IUFL from the send header to the receive header during a transmission. Therefore, if senders and receivers set up the two headers properly, they can use offset ?IUFL to pass information.

- AOS/VS uses offset ?IUFL to pass termination and obituary messages when a process terminates or breaks a connection with another process. (For complete details about ?IUFL termination codes, see "Process Termination Messages in a Customer/Server Relationship" in this chapter.)

Process Termination Messages in a Customer/Server Relationship

In a customer/server relationship, when a process terminates or breaks a connection with another process, AOS/VS uses the IPC facility to send an obituary message to the process with which it was connected. For a process to receive an obituary message, it must first issue ?IREC and set offsets ?IOPH in the ?IREC header to global port number ?SPTM, which is the predefined origin port for obituary messages.

Various codes in offset ?IUFL of the receive header describe the reason for the termination, and the program type of the terminated process. Figure 7-2 shows the structure of offset ?IUFL.

```
 ______________________________________________________________
|                                                              |
|     0               4 5            7 8                 15    |
|     |----------------|--------------|---------------------|  |
|     | ?RETURN        |  TERMINATION |     PROCESS ID      |  |
|     |  FLAGS         |     FIELD    |                     |  |
|     |----------------|--------------|---------------------|  |
|______________________________________________________________|
```

Figure 7-2. Structure of Offset ?IUFL

Bits 0 through 4 in offset ?IUFL are reserved for codes that AOS/VS sends to a process to indicate why a process terminated. The right byte of offset ?IUFL always contains the PID of the terminated or disconnected process.

The termination field in offset ?IUFL can contain any of the codes listed in Table 7-3, depending on the reason for the termination.

All of the termination codes are unique, whether they appear in the ?IUFL termination field or in the first word of the termination message.

Termination Messages for 16-Bit Processes

When a 16-bit process terminates by issuing a ?RETURN system call, the system returns flag ?TSELF to the termination field in offset ?IUFL, and copies one or more of the codes listed in Table 7-4 to the ?IUFL return field.

Table 7-3. Process Termination Codes in Offset ?IUFL for ?IREC and ?ISEND Headers

Code	Meaning
?TSELF	Either a 16-bit process terminated itself with a ?TERM or a ?RETURN system call or a 32-bit process terminated itself with a ?RETURN system call.
?TRAP	A user trap terminated a 16-bit process; Word 5 of the IPC message to the father describes the trap.
?TCIN	An abort console interrupt (CTRL-C CTRL-B sequence) terminated a process.
?TAOS	AOS/VS terminated a process because of an error; offset ?IPTR in the IPC header contains the error code.
?TBCX	A process broke a connection that was established via the connection-management system calls. (See Chapter 8 for information on the connection-management facility.)
?TCCX	The connection still exists, but the process chained. (See Chapter 8 for information on the connection-management facility.)
?TEXT	Indicates an extended termination code; the extended code appears in offset 0 (first word) of the IPC message. A termination code of ?TEXT means that the actual termination code is a right-justified 16-bit code in the first word of the termination message in the receive buffer. The following list describes these extended termination codes.

Extended Code	Meaning
?T32T	A process terminated itself with a ?TERM or a ?RETURN system call.
?TR32	A process terminated because of a user trap; Word 10 of the termination message describes the trap.
?TABR	Task abort notification to a server process. This involves customer/server relationship, ?IDGOTO, ?IS.R and ?IREC.

Table 7-4. Termination Codes for 16-Bit Processes

Code	Meaning
?RFCF	The termination message is in CLI format (the CLI is the father).
?RFEC	AC0 contains the error code.
?RFWA	A warning condition caused the termination.
?RFER	An error condition caused the termination.
?RFAB	An abort condition caused the termination.

The ?RETURN caller specifies the termination message sent to the CLI. (See the description of the ?RETURN system call in Chapter 13.) AOS/VS precedes the message with the following 2-word header:

Word 0	Contains the message length in bytes
Word 1	Contains the error code (the ?RETURN caller's input to AC0)

The message text follows this header. If there is no message, AOS/VS sends only the header.

If the father is not the CLI (that is, ?RFCF is not set), AOS/VS copies codes ?RFEC, ?RFWA, ?RFER, or ?RFAB to the ?RETURN field for whatever interpretation the father and son processes previously agreed on.

When a 16-bit process terminates itself with a ?TERM system call, AOS/VS returns either the termination message specified by the process or, if the process did not specify a message, one of the termination codes. AOS/VS sends the termination message directly to the father's receive buffer. It sends the termination code to the ?IUFL termination field in the father's receive buffer, and sets the ?IUFL ?RETURN flags field to 0.

If the 16-bit process terminated because of a user trap, AOS/VS sets the father's ?IUFL termination field to ?TRAP, and sends the father one of the 6-word termination messages listed in Table 7-5.

Table 7-5. ?TRAP Termination Messages for 16-Bit Processes

Word	Contents
0	AC0 contents at the time of the trap.
1	AC1 contents at the time of the trap.
2	AC2 contents at the time of the trap.
3	AC3 contents at the time of the trap.
4	Bit 0, carry; Bits 1 through 15, program counter value.
5	The following flag bits, which describe the trap: Bit 0=0 Trap occurred while control was in the user context. Bit 0=1 Trap occurred while control was in the operating system. Bit 12=1 Process tried to write into a write-protected area. Bit 13=1 Memory map validity error. (The process tried to refer to an address outside the user context.) Bit 14=1 Defer error. (The process tried to use more than 16 levels of indirection in an address reference.) Bit 15=1 Process tried to issue a machine-level I/O instruction without issuing the ?DEBL system call. (See Chapter 10.)

If the 16-bit process terminated because of an abort console interrupt (a CTRL-C CTRL-B sequence) or a ?TERM system call issued by a superior process, AOS/VS returns the proper code to the father's ?IUFL termination field (?TCIN or ?TSUP), but does not send a message.

Termination Messages for 32-Bit Processes

When a 32-bit process terminates because of a ?RETURN system call, a ?TERM system call, or a user trap, AOS/VS sets the ?IREC header's ?IUFL termination field to the ?TEXT code, places the appropriate termination code (?T32T or ?TR32) in the first word of the termination message, and sets the ?IUFL ?RETURN flags field to 0.

If the process terminated on a ?RETURN or a ?TERM system call, rather than a user trap, the termination message contains the following:

Word 0	?T32T (the extended termination code)
Word 1	Byte length of the message
Words 2 and 3	Error code
Word 4	Start of message text (in CLI format)

Words 2 and 3 contain the error code (if any) that the process specified when it issued the ?RETURN system call.

Word 4 contains the termination message (if any) that the process specified when it issued the ?RETURN system call. The entire termination message is ?TPLN words long.

If the process terminated because of a user trap, AOS/VS sends one of the termination messages listed in Table 7-6.

For more information on the MV/8000 ring architecture, refer to the 'Principles of Operation 32-Bit ECLIPSE Systems' manual.

?ISEND and ?IREC System Call Logic

The flowcharts in Figures 7-3 and 7-4 show the sequence of operations for the ?ISEND and ?IREC system calls, respectively.

Table 7-6. ?TEXT Code Termination Messages Sent on 32-Bit Process User Trap

Word	Contents
0	?TR32 (the extended termination code).
1 and 2	AC0 contents.
3 and 4	AC1 contents.
5 and 6	AC2 contents.
7 and 8	AC3 contents.
9	Bit 0, carry; Bits 1 through 15, high-order bits of program counter.
10	Low-order bits of program counter.
11	The following flag bits, which describe the trap: Bit 0=0 Trap occurred while control was in the user context. Bit 0=1 Trap occurred while control was in the operating system. Bit 3=1 A node time-out occurred. (This is a hardware error.) Bit 4=1 Process tried to execute a privileged instruction. Bit 5=1 Process tried to return to an inner ring from a subroutine call. (This is a violation of the ring structure.) Bit 6=1 Process tried to issue a subroutine call to an outer ring. (This is a violation of the ring structure.) Bit 7=1 Gate protection error. (This is a violation of the ring structure.)

Table 7-6. ?TEXT Code Termination Messages Sent on 32-Bit Process User Trap (Cont.)

Word	Contents	
11 Cont.)	Bit 8=1	Process tried to reference an address in an inner ring. (This is a violation of the ring structure.)
	Bit 9=1	Process tried to read a read-protected page.
	Bit 10=1	Process tried to execute data in an execute-protected area.
	Bit 12=1	Process tried to write into a write-protected area.
	Bit 13=1	Memory map validity error. (The process tried to refer to an address outside the user context.)
	Bit 14=1	Defer error. (The process tried to use more than 16 levels of indirection in an address reference.)
	Bit 15=1	Process tried to issue a machine-level I/O instruction without issuing the ?DEBL system call. (See Chapter 10.)

```
_____________________________________________________________________
| ?ISEND System Call                *****                           |
|                                 * START *                         |
|                                   *****                           |
|                                     |                             |
|                                     /\                            |
|                                   /Proc\ No     /---\             |
|                                 <  has   >---->|ERROR|            |
|                                  \?PVIP/        \---/             |
|                                    \?/                            |
|                                     |Yes                          |
|                                     /\           /\               |
|                            Yes    /    \ No    /Orig\ No   /---\  |
|                           |-----<?IFSTM= >--><port no.>-->|ERROR| |
|                           |      \  1  /      \legal/      \---/  |
|              ***          |        \?/          \?/Yes            |
|             * 1 *        /\          ____________|____________    |
|              ***        /    \        | Translate origin port |    |
|               | No   /?IFRFM=1\ Yes  | no. to global port no.|    |
|               |----<on outstand.>--| |_______________________|    |
|               V      \ ?IREC  /    |             |                |
|               |        \ ?  /      |             /\               |
|               ^          \/        |           /Dest\ No   /---\  |
|             /   \                  |         <  port >--->|ERROR| |
| /---\ Yes /       \                |          \exists      \---/  |
||ERROR|<--<?IFNSP=1 >               |            \?/               |
| \---/     \       /                |             |Yes             |
|             \   /                  |             /\               |
|              \/                    |          /Rec.\              |
|               |No                  |        /request \ No  ***    |
|        _______|________            |       <matched by>-->* 1 *   |
|        | Spool the    |            |        \correct /     ***    |
|        | Message      |            |          \proc./             |
|        |______________|            |            \?/               |
|               |                    |------------>|Yes             |
|               |                   _______________|_______________ |
|               |                   | Do "move message" sequence |  |
|               |                   |____________________________|  |
|               |_____________________________________|             |
|                                |                                  |
|                              ******                               |
|                            * RETURN *                             |
|                              ******                               |
|-------------------------------------------------------------------|
```

Figure 7-3. ?ISEND Logic Flowchart

```
 ?IREC System Call                  *******
                                   * START *
                                    *******
                                       |Yes
                                      /\            /\
                                Yes /    \ No    /Dest\ No    /---\
                |----------------><?IFRFM= >--><port no.>-->|ERROR|
                |                  \  1  /      \legal/      \---/
               /\                    \?/          \?/Yes
             /Msg \                                 |
           /in spool\ Yes            ________________|________________
         < file with  >------|       | Translate dest. port  |
           \?IFSTM=1/        |       | no. to global port no.|
             \ ?  /          |       |_______________________|
               \/            |                    |
               |No           |                   / \
               |             |                 / Orig\ No   /---\
               |             |               <port=0 or>-->|ERROR|
               |             |                 \exists      \---/
               |             |                   \?/
               |             |                    |Yes
               |             |                   /\
               |             |                 /Read\ Yes  /---\
               |             |               <  into  >-->|ERROR|
               |             |                \shared/     \---/
               |             |                 \area?
               |             |                   \/
               |             |                    |No
               |             |                   /\
               |             |                 /    \
               |             |            Yes/ Matching
               |             |<-------------<message in >
               |   __________|__________     \  spool  /
               |  |Do "move message" seq.|     \file /
               |  |__________|___________|       \?/
               |             |                    |No
               |           ******                 |
              /\         * RETURN *               |
 /---\  Yes /    \         ******                 |
|ERROR|<--<?IFNBK= >------------------------------|
 \---/      \  1  /
              \?/No
               |
    | Suspend the caller |
    |____________________|
               |
             /---\
            |ERROR|
             \---/
```

Figure 7-4. ?IREC Logic Flowchart

IPC Sample Programs

The following programs, SPEAK and HEAR, illustrate interprocess communications with the IPC system calls ?ILKUP, ?IREC, and ?ISEND.

Program SPEAK uses routine SON (see "Processes and Memory Sample Programs) to execute program HEAR. HEAR issues an ?IREC system call to receive a message from SPEAK. Then, SPEAK issues ?ISEND to send the message to HEAR. HEAR and SPEAK both use the ?ILKUP system call to discover the other's port number.

```
                        .TITLE  HEAR
                        .EXTL   SON
                        .NREL

;Open console (CON) for input and output.  (See Chapter 6 for more
;information on ?OPEN.)

HEAR:   ?OPEN   CON                     ;Open console (CON) for I/O.
        WBR     .ERROR                  ;Report error and quit.

        ?WRITE  CON                     ;Display message on console
                                        ;(byte pointer is already in
                                        ;I/O packet).
        WBR     .ERROR                  ;?WRITE error return.

;Start the SON process to run SPEAK.PR.

        XLEFB   0,SPEAK*2               ;Get byte pointer to filename.
        XJSR    @.SON                   ;SON creates process.

;SPEAK is running.  Create IPC entry for receive.

        XLEFB   0,PORTR*2               ;Byte pointer to port name.
        ?CREATE IPCEN                   ;Create IPC entry PORTR.
        WBR     .ERROR                  ;?CREATE error return.
        XLEFB   0,MES1*2                ;Byte pointer to message.
        XWSTA   0,CON+?IBAD             ;Put in I/O packet.
        ?WRITE  CON                     ;Write message to console.
        WBR     .ERROR                  ;Try to handle the error.

;See if SPEAK's entry and its ports have been created.

GETOP:  XLEFB   0,PORTS*2               ;Byte pointer to port name.
        ?ILKUP                          ;Get port number from AC1.
        WBR     TEST                    ;Try to handle the error.
        XLEFB   0,MES2*2                ;Get byte pointer.
        XWSTA   0,CON+?IBAD             ;Put in I/O packet.
```

HEAR Program (Cont.)

```
          ?WRITE  CON                    ;Display success message on
                                         ;console.
          WBR     .ERROR                 ;Try to handle the error.
          XWSTA   1,RHDR+?IOPH           ;Put origin port number in
                                         ;record header (note wide
                                         ;storage).
          NLDAI   1,0                    ;Generate 1.
          XNSTA   ORHDR+?IDPN            ;Put destination port 1 in
                                         ;record header (note narrow
                                         ;storage).

          ?IREC   RHDR                   ;Receive SPEAK message.
          WBR     .ERROR                 ;?IREC error message.
          XLEFB   1,MSBUF*2              ;Message received.  Get byte
                                         ;pointer to message buffer.
          WLDAI   ?RFCF!100.,2           ;Put flag and 100 words for
                                         ;message in AC2.

          ?RETURN                        ;Return to CLI with message.
          WBR     .ERROR                 ;Try to handle the error.

  ;?ILKUP error.  Check for error code ERFDE (file does not exist) and
  ;delay if present.

  TEST:   WLDAI   ERFDE,1                ;Put ERFDE number in AC1.
          WSEQ    0,1                    ;Skip if error code is ERFDE.
          WBR     .ERROR                 ;Try to handle the error.
          XLEFB   1,MES3*2               ;Get byte pointer to message.
          XWSTA   0,CON+?IBAD            ;Put in I/O packet.

          ?WRITE  CON                    ;Display message on console.
          WBR     .ERROR                 ;Try to handle the error.
          WLDAI   5000.,0                ;5 seconds.

          ?WDELAY                        ;Wait for 5 seconds.
          WBR     .ERROR                 ;Try to handle the error.
          WBR     GETOP                  ;Do ?ILKUP again.

  ;Error instruction, byte pointer, filename, and port name.

  .ERROR: XJMP    ERROR                  ;To error handler.

  .SON:   SON                            ;To subroutine SON.

  SPEAK:  .TXT    "SPEAK.PR"             ;Filename of program.

  PORTR:  .TXT    "PORTR"                ;Name of receive port.

  PORTS:  .TXT    "PORTS"                ;Name of send port.
```

```
;?OPEN and I/O packet for console.

CON:    .BLK     ?IBLT                  ;Allocate enough space for
                                        ;packet.
        .LOC     CON+?ISTI              ;File specifications.
        .WORD    ?ICRF!?RTDS!?OFIO      ;Change format to data-
                                        ;sensitive records and open
                                        ;for input and output.
        .LOC     CON+?IMRS
        .WORD    -1                     ;Physical block size is 2K
                                        ;bytes.

        .LOC     CON+?IBAD              ;Byte pointer to record I/O
        .DWORD   MES*2                  ;buffer.

        .LOC     CON+?IRCL
        .WORD    120.                   ;Record length is 120
                                        ;characters.

        .LOC     CON+?IFNP              ;Byte pointer to pathname.
        .DWORD   CONS*2

        .LOC     CON+?IDEL              ;Delimiter table address.
        .DWORD   -1                     ;Use default delimiters: null,
                                        ;NEW LINE, form feed, and
                                        ;carriage return (default is
                                        ;-1).
        .LOC     CON+?IBLT              ;End of packet.

;Filename, buffer, messages.  A .NOLOC 1 follows.

CONS:   .TXT     "@CONSOLE"             ;Use generic name.

MES:    .TXT     "From HEAR--I have opened the console and I am ready
                 to call SON.<12>"

MES1:   .TXT     "From HEAR--I am back from SON. SPEAK is running. <12>
                 I created an IPC entry.<12>"

MES2:   .TXT     "From HEAR--Have ?ILKUPed the IPC port entry.<12>"

MES3:   .TXT     "From HEAR--?ILKUP error. I will wait and then try
                 again.<12>"

        .NOLOC 0
```

HEAR Program (Cont.)

```
;Header for IPC entry.

IPCEN:  .LOC    IPCEN+?CFTYP
        .WORD   ?FIPC                   ;IPC file.

        .LOC    IPCEN+?CPOR
        .WORD   1                       ;Port number is 1.

        .LOC    IPCEN+?CTIM
        .DWORD  -1                      ;Default to current time.

        .LOC    IPCEN+?CACP
        .DWORD  -1                      ;Default to current ACL.

;?IREC Receive header RHDR.

RHDR:   .LOC    RHDR+?ISFL
        .WORD   0                       ;There are no system flags.

        .LOC    RHDR+?IUFL
        .WORD   0                       ;There are no user flags.

        .LOC    RHDR+?IOPH
        .DWORD  0                       ;AOS/VS returns origin port
                                        ;number here.

        .LOC    RHDR+?IDPN
        .WORD   0                       ;AOS/VS returns destination
                                        ;port number here.

        .LOC    RHDR+?ILTH
        .WORD   100.                    ;Message buffer is 100 words.

        .LOC    RHDR+?IPTR
        .DWORD  MSBUF                   ;Message buffer address.

        .LOC    RHDR+?PLTH              ;End of ?IREC header.

MSBUF:  .BLK    101.                    ;Message buffer.
```

```
;Error handler.

ERROR:  WLDAI   ?RFEC!?RFCF!?RFER,2     ;Error flags:  Error code is
                                        ;in ACO (?RFEC), message is in
                                        ;CLI format (?RFCF), and
                                        ;caller should handle this as
                                        ;an error (?RFER).

        ?RETURN                         ;Return to CLI.
        WBR     ERROR                   ;?RETURN error return.

       .END    HEAR                    ;End of HEAR program.
```

```
;The following program, SPEAK, sends an IPC message to another
;process.  Then, SPEAK terminates itself.  SPEAK's origin port name
;is PORTS; its destination port name is PORTR.


                        .TITLE  SPEAK
                        .ENT    SPEAK
                        .NREL

;Create and IPC entry port named PORTS.

SPEAK:  XLEFB   0,PORTS*2                   ;Byte pointer to port name.
        ?CREATE IPCEN                       ;Create an IPC port.
        WBR     .ERROR                      ;Try to handle the error.

;See if PORTR, the receive port, has been created.

GETNM:  XLEFB   0,PORTR*2                   ;Byte pointer to port name.
        ?ILKUP                              ;Put port number in AC1.
        WBR     TEST                        ;Does the port exist?
        XWSTA   1,SHDR+?IDPH                ;Yes.  Put port number in send
                                            ;header.
        NLDAI   1,0                         ;No.  Generate 1.
        XNSTA   0,SHDR+?IOPN                ;Put destination port 1 in
                                            ;send header (narrow storage).
        ?ISEND  SHDR                        ;Send SPEAK message.
        WBR     .ERROR                      ;Try to handle the error.

;The message has been sent.  Wait for other process to receive message
;before terminating yourself.

        ?WLDAI  10000.,0                    ;10 seconds.
        ?WDELAY                             ;Wait for 10 seconds.
        WBR     .ERROR                      ;Try to handle the error.
        NLDAI   -1,0                        ;Get -1 to terminate yourself.
        WSUB    2,2                         ;There is no IPC message to
                                            ;the father.
        ?TERM                               ;Terminate.
        WBR     .ERROR                      ;Try to handle the error.

;?ILKUP error.  Check to see whether the error code is ERDNE (Does Not
;Exist).  If the error code is ERDNE, wait.

TEST:   WLDAI   ERFDE,1                     ;Put error code number in AC1.
        WSEQ    0,1                         ;Was error code ERFDE?
        WBR     .ERROR                      ;No.  Try to handle the error.
        WLDAI   5000.,0                     ;5 seconds.
        ?WDELAY                             ;Yes.  Wait for 5 seconds.
        WBR     .ERROR                      ;Try to handle the error.
        WBR     GETNM                       ;Do ?ILKUP again.
```

SPEAK Program (Cont.)

```
;Error instructions, pointer, filenames, and port names.

.ERROR: XJMP    ERROR                   ;To error handler.

PORTS:  .TXT    "PORTS"                 ;Name of send port.

PORTR:  .TXT    "PORTR                  ;Name of receive port.

;Header for IPC entry.  (See the description of ?CREATE in
;Chapter 13.)

IPCEN:  .LOC    IPCEN+?CFTYP
        .WORD   ?FIPC                   ;IPC file.

        .LOC    IPCEN+?CPOR
        .WORD   1                       ;Port number is 1.

        .LOC    IPCEN+?CTIM
        .DWORD  -1                      ;Default to current time.

        .LOC    IPCEN+?CACP
        .DWORD  -1                      ;Default to current ACL.

;?ISEND send header SHDR.

SHDR:   .LOC    SHDR+?ISFL
        .WORD   0                       ;There are no system flags.

        .LOC    SHDR+?IUFL
        .WORD   0                       ;There are no user flags.

        .LOC    SHDR+?IDPH
        .DWORD  0                       ;AOS/VS returns destination

        .LOC    SHDR+?IOPN
        .WORD   0                       ;AOS/VS returns origin
                                        ;port number here.

        .LOC    SHDR+?ILTH
        .WORD   100.                    ;Message buffer is 100 words.

        .LOC    SHDR+?IPTR
        .DWORD  MSBUF                   ;Message buffer address.

        .LOC    SHDR+?PLTH              ;End of ?ISEND header.
```

```
;Message that we want to send.  A .NOLOC 1 follows.

MSBUF:  .TXT    "Hello.  This is your son speaking.  As you read <12>
                these words, I am terminating and so are you.<12>"

        .NOLOC  0                       ;Resume listing everything.

;Error handler.

ERROR:  WLDAI   ?RFEC!?RFCF!?RFER,2     ;Error flags:  Error code is
                                        ;in AC0 (?RFEC), message is in
                                        ;CLI format (?RFCF), and
                                        ;caller should handle this as
                                        ;an error (?RFER).
        ?RETURN                         ;Return to CLI.
        WBR     ERROR                   ;?RETURN error return.

        .END    SPEAK                   ;End of SPEAK program.
```

End of Chapter

CHAPTER 8
CONNECTION MANAGEMENT

The system calls that allow you to perform connection management are:

?CON	Becomes a customer of a specified server.
?CTERM	Terminates a customer process.
?DCON	Breaks a connection (disconnects) in Ring 7.
?DRCON	Breaks a connection (disconnects) in a specified ring.
?MBFC	Moves bytes from a customer's buffer.
?MBTC	Moves bytes to a customer's buffer.
?PCNX	Passes a connection from one server to another in Ring 7.
?PRCNX	Passes a connection from one server to another in a specified ring.
?RESIGN	Resigns as a server.
?SERVE	Becomes a server process.
?SIGNL	Signals another task.
?SIGWT	Signals another task and then waits for a signal.
?VCUST	Verifies a customer in Ring 7.
?VRCUST	Verifies a customer in a specified ring.
?WTSIG	Waits for a signal from another task or process.

AOS/VS allows you to establish a customer/server relationship (called a connection) between processes, and then use the server process to perform certain functions on behalf of its customers. Typically, a server process performs general routines that customer processes can access. For instance, you can create a server process to build files or perform I/O.

Connection management allows servers to move bytes to and from their customers' buffers.

Connection Creation

To make a connection between two processes you must define one process as the server and the other as the customer. To do this, issue the ?SERVE system call to define the calling process as a server, and issue the ?CON system call to define a customer and establish the logical connection between the customer and an existing server. Figure 8-1 shows a server process with connections to three customer processes.

```
 ____________________________________________________________________
|                                                                    |
|     _______________________________________                        |
|     |               Process A             |---> Issues ?SERVE to   |
|     |_____________________________________|     become a server    |
|        ^              ^              ^                             |
|        | ?CON         | ?CON         | ?CON                        |
|        | (become a    | (become a    | (become a                   |
|        | customer     | customer     | customer                    |
|        | of A)        | of A)        | of A)                       |
|   _____V______   _____V______   _____V______                       |
|   | Process B |  | Process C |  | Process D |                      |
|   |___________|  |___________|  |___________|                      |
|____________________________________________________________________|
```

Figure 8-1. Model Customer/Server Configuration

AOS/VS maintains a connection table, which manages exchanges between customers and servers. When a customer makes a connection (via the ?CON system call) with a declared server, AOS/VS writes an entry in the connection table that specifies the PID of the server, the PID of the customer, and the customer's ring field. Each ?CON system call generates one connection-table entry.

A process can act as a server for other processes and can also act as a customer of other servers as long as it issues the appropriate number of ?SERVE and ?CON system calls. A process that acts as both a server and a customer is called a multilevel connection. Figure 8-2 shows a multilevel connection, where process A is the server of processes B, C, and D, and a customer of process X. Multilevel connections let you set up intermediate servers for some functions, and one or more superior servers for other functions.

```
 ______________________________________________________________________
|                                                                      |
|      ___________________________________                             |
|     |            Process X              |---> Issues ?SERVE to       |
|     |___________________________________|     become a server        |
|                      ^                                               |
|                      |                                               |
|                      V                                               |
|       _______________________________                                |
|       |             Process A        |-----> Issues ?CON (to         |
|       |______________________________|       connect with X)         |
|       ^              ^               ^       and ?SERVE              |
|       |              |               |                               |
|   ____V_____     ____V_______    ____V_______                        |
|   | Process B |  | Process C |   | Process D |--> Processes B, C,    |
|   |___________|  |___________|   |___________|    and D issue ?CON   |
|                                                   (to connect with   |
|                                                   Process A)         |
|______________________________________________________________________|
```

Figure 8-2. Multilevel Customer/Server Configuration

You can also make a double connection between two processes. A double connection allows each process to act as either the customer or the server of the other, depending on the action to be performed. As Figure 8-3 illustrates, a double connection requires two ?SERVE system calls and two ?CON system calls. AOS/VS creates two connection-table entries, one for each ?CON system call.

```
 ______________________________________________________________________
|                                                                      |
|   _______________                    _______________                 |
|  |   Process A   | ?SERVE           |   Process B   | ?SERVE         |
|  |               |                  |               |                |
|  |               | ?CON ----------->|               |                |
|  |               |(become customer of B)            |                |
|  |               |                  |               |                |
|  |               | <-----------?CON |               |                |
|  |_______________|(become customer of A) |___________|                |
|                                                                      |
|______________________________________________________________________|
```

Figure 8-3. Double Connection

Server Process

Once a process has server status (established with the ?SERVE system call), it can issue the following system calls:

?CTERM	Terminates a customer.
?MBFC	Moves bytes from a customer's buffer.
?MBTC	Moves bytes to a customer's buffer.
?PCNX	Passes a connection from one server to another in Ring 7.
?PRCNX	Passes a connection from one server to another in a specified ring.
?RESIGN	Resigns as a server.
?VCUST	Verifies a customer in Ring 7.
?VRCUST	Verifies a customer in a specified ring.

The ?CTERM system call terminates a customer process. The ?RESIGN system call signals AOS/VS that the caller has resigned as a server.

The ?MBTC and ?MBFC system calls allow the server to move bytes to or from a customer's logical address space. However, before AOS/VS executes either of these system calls, it checks the connection table to make sure that there is a valid connection between the two processes, and that the customer's buffer is in the ring defined at connect time, which must be in the caller's ring or in a higher ring. Also, there must be enough space at the destination for the data to reside entirely within the specified destination ring.

The ?PCNX system call passes a customer/server connection from one server to another in Ring 7 and directs AOS/VS to revise the connection-table entry accordingly. The ?PRCNX system call is similar to the ?PCNX system call, except the ?PRCNX system call is not restricted to Ring 7. Both the ?PCNX and the ?PRCNX system calls are useful for passing a valid customer from a dispatching server to a specialized server process.

The ?VCUST system call determines whether a target process in Ring 7 is a customer of the ?VCUST caller. The ?VRCUST system call is similar to the ?VCUST system call, except the ?VRCUST target process need not be in Ring 7. If the ?VCUST or the ?VRCUST target process is not a customer, AOS/VS takes the error return and passes error code ERCDE to AC0. If the connection between the two has been broken, the system call fails on error code ERCBK.

Typically, server processes communicate with their customers via the IPC system calls ?SEND, ?IREC, and ?IS.R. However, they can also use the fast interprocess communication system calls, ?SIGNL, ?WTSIG, and

?SIGWT, to communicate with their customers. (See "Fast Interprocess Synchronization" in this chapter for more information on the ?SIGNL, ?WTSIG, and ?SIGWT system calls.)

Connection Termination

AOS/VS breaks the customer/server connection when a process traps or when the process issues one of the following calls:

?CTERM	Terminates a customer (a server-only system call).
?DCON	Breaks a connection in Ring 7.
?DRCON	Breaks a connection in a specified ring.
?RESIGN	Resigns as a server (a server-only system call).
?TERM	Terminates a process (self-terminates).

Notice that the ?CTERM system call is a server-only system call. The ?DCON, ?DRCON, and ?TERM system calls are available to both servers and customers. (See Chapter 3 for information on terminating processes with the ?TERM system call.)

When AOS/VS detects a broken connection, it sets a flag bit in the appropriate connection table entry. For AOS/VS to actually clear the entry, however, it must receive disconnects from both the customer and the server. For example, a customer could issue a ?DCON system call to break its connection with the server, but the PIDs of both processes will remain in the connection table until the server issues a ?DCON, ?RESIGN, or ?TERM (self-termination) system call.

You should issue disconnects from both processes as soon as a connection has served its purpose. This keeps the connection-table entries within the maximum range and allows AOS/VS to reassign the PIDs. (The maximum number of connections allowed under AOS/VS is revision dependent.)

Obituary Messages

When a customer or server disconnects, AOS/VS sends the other process an obituary message. An obituary message is a zero-length IPC message. A customer can suppress the obituary message by setting bit ?COBIT in AC1 when it issues the ?CON system call.

To receive an obituary message, a process must issue the ?IREC system call; it must specify 0 and ?SPTM in ?IREC offsets ?IOPH and ?IOPL, respectively (origin port), and 0 in offset ?IDPN (destination port). AOS/VS returns the obituary message as termination code ?TBCX in offset ?IUFL of the ?IREC header.

Inner-Ring Connection Management

Segment images that are loaded into different user rings within the sample process often have very different aims and identities. Therefore, the connection-management facility identifies all connections as being between ordered pairs of PID/ring-within-PID tandems (called PID/ring tandems). A ring within a process can be connected as a customer (and/or as a server) with multiple rings that are within another process or processes.

Although multiple ?CON system calls that connect the same ordered pairs of PID/ring tandems are legal, they will result in only a single connection. However, connections between rings that are within the same process are illegal.

For a server, the move bytes to and from customer privilege is limited to only those rings in the customer that are higher than or equal to the lowest ring that issued a ?CON system call to create a connection to the server.

Every IPC message (obituary message, chain, etc.) issued by the connection-management facility, is sent to the ring from which the ?CON or ?SERVE system call was issued. The system flag word of the IPC header holds a field, ?IFRING, that contains the ring number of the segment image that caused the system to generate the message.

If a server is concurrently connected to multiple rings within the customer, AOS/VS indicates the status of those connections with a single IPC message. This prevents the race conditions that might occur if AOS/VS issued multiple messages.

For 32-bit receivers, flag bits are returned in the ?IPTL word of the IPC header. For 16-bit receivers (that is, tasks in Ring 7 of a 16-bit process), the flag bits are returned in the ?IPTR word of the IPC header. The flag bits include both a single "explicit disconnect" flag and a bit map that contains the connection status of the various inner rings.

The explicit disconnect flag expands the information that the "connection broken" (?TBCX) termination message contains when it is going to a server on a customer process termination. If the explicit disconnect bit is set in the connection broken termination message, then one of the rings of the customer process issued a ?DCON or a ?DRCON system call to break the connection. If the explicit disconnect bit is not set, then one of the following caused the broken connection:

o A customer process terminated.

o A customer process chained, but it did not have a connection in its Ring 7.

NOTE: A connection broken (?TBVC) rather than a "customer chained" (?TCCX) termination message describes this special case of a customer process chain, but it is also valid for a server process chain. All other types of process chain events cause customer chained messages (?TCCX), because Rings 4 through 6 are "unloaded" when Ring 7 chains. (Effectively, Rings 4 through 6 terminate on a Ring 7 chain.)

The meaning of individual bits within the bit map depends upon the type of event being signaled:

o When a customer is chained, bits set in the bit map indicate which rings were connected before the chain. In this case, AOS/VS automatically preserves the connections to Ring 7 and 3, providing they existed before the chain.

o When a connection is broken, if the explicit disconnect bit is set, then the bits set in the bit map indicate rings to which there are remaining connections. If the explicit disconnect bit is clear, then the bits set in the bit map indicate which rings were connected before the termination or chain.

The following parameters characterize the bit flags:

o	?CXMBM	Word mask that allows you to extract both the explicit disconnect flag and the connection bit map.
o	?CXMED	Bit mask for the explicit disconnect flag.

o	?CXBBMO	Bit position of the explict disconect flag. (The explicit disconnect flag immediately precedes the connection bit map portion of the ?IPTL or ?IPTR word.)
o	?CXBVED	Position of the explicit disconnect bit within the extracted word.

NOTE: ?CXBBMO + N defines the position of the bit that corresponds to Ring N within the bit map. (Rings 1 through 7 are mapped in the bit map.) You can point to the explicit disconnect bit as if it were the first bit in the bit map (that is, N = 0).

Fast Interprocess Synchronization

Frequently, identical local servers loaded into different processes will use a common shared memory file for global synchronization. AOS/VS includes a fast interprocess sychronization facility that common local servers can use to pend and unpend tasks, depending on the state of databases in that shared memory.

The fast interprocess synchronization mechanism, which uses the ?SIGNL, ?WTSIG, and ?SIGWT system calls, provides you with another way of synchronizing processes. Unlike the IPC system calls, the fast interprocess synchronization system calls do not move any data. Instead, they allow a task within a process to send and receive task-specific signals to and from the same or another process. Because they do not move data, ?SIGNL, ?WTSIG and ?SIGWT are very fast, and they require very little system overhead.

When a task issues a ?SIGNL or a ?SIGWT system call, the target does not have to be waiting to receive the signal. Instead, AOS/VS remembers the task-specific target. A subsequent ?WTSIG or ?SIGWT system call issued by the target task causes the target task to proceed immediately. A ?WTSIG system call, however, will pend the caller if a signal for the task is not outstanding.

Unlike the IPC system call ?IS.R, the ?SIGWT system call does not force the calling task to wait for a signal from the same task that it signaled. Any signal that specifies the pended task will wake up that task.

No privileges are necessary to issue the ?SIGNL, ?WTSIG, or ?SIGWT system calls.

End of Chapter

CHAPTER 9
BINARY SYNCHRONOUS COMMUNICATIONS (BSC)

This chapter describes the following binary synchronous communications (BSC) system calls:

?SDBL	Disables a BSC line.
?SDPOL	Defines a polling list or a poll address/select address pair.
?SDRT	Disables a relative console.
?SEBL	Enables a BSC line.
?SERT	Re-enables a relative console.
?SGES	Gets BSC error statistics.
?SRCV	Receives data or a control sequence over a BSC line.
?SSND	Sends data or a control sequence over a BSC line.

AOS/VS supports binary synchronous communications (BSC) over dedicated or switched communications lines. This chapter describes the system calls you need to implement BSC communications. This chapter is not a tutorial. In fact, this chapter assumes that you are familiar with BSC protocol and the rules governing BSC.

To help you understand this chapter, you must be familiar with the following terms:

o Station

  A station is the origin (sender) or destination (receiver) of data over a BSC line.

o Dedicated communications line

  A dedicated communications line continuously connects two or more stations, regardless of the amount of time the line is actually in use. This type of line is dedicated to serving specific local and remote stations.

- Switched communications line

  A switched communications line is one on which you use dialing procedures to establish a connection between the local and remote stations.

BSC Concepts

Before you use any of the BSC system calls, make sure that your system manager or operator has created a GSMGR (global synchronous manager) process. This process acquaints AOS/VS with the synchronous communications hardware that you specified during the system-generation dialog. If this process does not exist, any BSC system calls you issue will cause an error return. (Refer to the 'Managing AOS/VS' manual for information on creating the GSMGR process.)

AOS/VS recognizes each BSC line by the device name @SLNx, where x represents the line number. When you enable a BSC line (with the ?SEBL system call), you must supply the @SLN designator with the correct line number. However, it is not necessary to specify whether the enabled line is dedicated or switched.

AOS/VS assigns a channel number to each enabled BSC line and returns this value in the ?SEBL packet. Unlike disk files, you cannot open a BSC line on more than one channel.

To send data over an enabled BSC line, a station issues the ?SSND system call. To receive data, a station issues the ?SRCV system call. BSC protocol distinguishes between Send Initial and Send Continue system calls, and between Receive Initial and Receive Continue system calls. A system call is an Initial system call if it opens a communications session.

The ?SSND and ?SRCV system calls depend upon timing and upon the interaction of the sending and receiving stations. When AOS/VS encounters timing errors or inappropriate responses to the ?SSND and ?SRCV system calls, it begins error-recovery procedures. You must view these error-recovery procedures in the context of the send and receive system calls. (See "BSC Error-Recovery Procedures" in this chapter.)

To disable a BSC line, issue the ?SDBL system call.

Line Configurations

There are two types of BSC line configurations:

o Point-to-point

  On a point-to-point line, each station bids for the line; that is, asks to use it. Only two stations can be on a point-to-point line.

o Multipoint (also called Multidrop)

  On a multipoint line, stations do not bid for the line. Instead, one station (called the control station) has complete control over the activities of the other stations (called the tributary stations) on the line. Therefore, no contention occurs between stations. Usually, a multipoint line connects one local station with more than one remote station. However, it can connect as few as two stations.

If both stations on a point-to-point line bid for the line at the same time, the line is under contention. Contention occurs when one point-to-point station bids for a line and the other station, in response, also bids for the line. When you enable a point-to-point line, you must designate your computer as either the primary station or the secondary station. AOS/VS favors the primary station over the secondary station in the following way when contention occurs:

o If your station is the primary station, AOS/VS automatically follows your bid with another bid sequence. The secondary station should acknowledge this additional bid sequence.

o If your station is the secondary station, AOS/VS gives you an error return. To receive the primary station's bid sequence, you must issue an ?SRCV Receive Initial system call.

Unlike the secondary stations on a point-to-point line, the tributary stations on a multipoint line are completely subservient to the control station. The following restrictions apply to tributary stations:

o Tributary stations can only send data to and receive data from the control station.

o A particular tributary station can send or receive data over the line only when the control station specifically requests that it do so.

o Tributary stations cannot communicate directly with one another.

When you enable a BSC line with the ?SEBL system call, you must specify whether it is a point-to-point line or a multipoint line. Also, you must use the ?SEBL system call to specify whether your station is a primary station, a secondary station, a control station, or a tributary station.

Figure 9-1 shows the difference between a point-to-point line configuration and a multipoint line configuration.

```
 ______________________________________________________________________
| Point-to-point Communications Line                                   |
|                                                                      |
|        ---------------                  ---------------              |
|        |     A       |<--------------->|     B       |               |
|        |_____________|                 |_____________|               |
|         Station A                       Station B                    |
|         (primary)                       (secondary)                  |
|                                                                      |
| Multipoint Communications Line                                       |
|                                                                      |
|        ---------------                                               |
|        |     D       |<----------------------------------|           |
|        |_____________|         |            |            |           |
|         Control station D       V            V            V           |
|                             -------      -------      -------         |
|                             |  E  |      |  F  |      |  G  |         |
|                             |_____|      |_____|      |_____|         |
|                              Tributary stations E, F, and G           |
|______________________________________________________________________|
```

Figure 9-1. Point-to-Point/Multipoint Line Configurations

Multipoint Line Selection and Polling

To manage the activity on a multipoint line, the control station performs two operations:

o Polls

  This means that the control station contacts its tributary stations to see if any of them has data to send to it. There are two types of polls: general and specific.

In a general poll, the control station contacts each of its tributaries in round-robin fashion, and accepts the first positive response.

In a specific poll, the control station contacts a single tributary to solicit data.

- Selects

  The control station selects by contacting a specific tributary to see if it is ready to receive data from the control station.

Each tributary station on a multipiont line must have two unique identifiers for polling and selecting to occur: a poll address and a select address. If your computer is a tributary station, you must define its poll address and select address by issuing the ?SDPOL system call.

If your computer is a control station, you must issue the ?SDPOL system call before polling or selecting to define a polling list. A polling list is a series of contiguous words that contains each tributary station's poll address and device address. The device address points to the peripheral device from which the control station will request data when it polls that tributary.

To perform polling, a control station issues the ?SRCV system call ("receive data or control characters") to specify whether the system call is a Receive Initial or a Receive Continue system call and whether the operation is general polling or specific polling.

In its first general poll, the control station starts with the poll and device address entry at the top of the polling list (the lowest relative console number), and sends this entry down the BSC line. Each tributary station recognizes its own poll address; it responds to the poll only if the entry matches its poll address. If the poll address sent by the control station does not match a tributary station's poll address, the tributary station ignores it.

A general poll ends when a tributary station responds to its poll address by sending data to the control station. If there is no response to a particular poll address entry, the control station continues to poll until it reaches the end of the polling list. At that point, AOS/VS takes the error return from the control station's ?SRCV system call, and passes error code EREPL (end of polling list) to ACO.

As we mentioned, general polling is a round-robin operation. This means that when a general poll ends in a positive response, the next general poll begins with the next relative console on the polling list (that is, the tributary station immediately following the previous respondent). Specific polling, that is, polling one and only one tributary station, is a way to break out of the round-robin method of general polling.

Relative Consoles

AOS/VS assigns a relative console number to each tributary station, based on that station's position on the polling list.

The first time you enable a multipoint line (with the ?SEBL system call) and define its polling list (with the ?SDPOL system call), AOS/VS enables all relative consoles on the list for polling. To disable a relative console without redefining the polling list, issue the ?SDRT system call. To re-enable a relative console, issue the ?SERT system call.

When you disable a relative console, it does not affect the corresponding tributary station; it simply means that the control station ignores that tributary station when it performs general polling, until you subsequently re-enable the relative console or define a new polling list.

BSC Protocol

The logic behind data transmissions over a BSC line is BSC protocol. Briefly, BSC protocol is a set of rules governing:

- The initialization of communications over a BSC line.
- The orderly exchange of data over a BSC line.
- The termination of communications over a BSC line.

These objectives are accomplished in part by the protocol's data-link control characters, which are synchronization characters that both the sending and the receiving stations recognize. Data transmissions over a BSC line typically consist of text, header information (optional), and data-link control characters, which delimit various portions of the data block and control its transmission.

None of the BSC system calls require data-link control characters as input. AOS/VS provides the required control characters when you send text or header information over a BSC line, and removes them when you receive the information. However, because several of the system call descriptions refer to the data-link control characters, Table 9-1 defines the control characters that we mention in this chapter.

Table 9-1. BSC Protocol Data-Link Control Characters (DLCC)

Character	Description
ACK0 ACK1	Affirmative Acknowledgment Positive replies, sent in alternating sequence, to indicate that the receiver has accepted the previous block without error, and is ready to accept the next block of the transmission. ACK0 is also a positive response to a line bid (ENQ) for a point-to-point line and to a selection sequence on a multipoint line.
BCC	Block Check Character A value generated by the transmitting station and sent with each data block to validate the block's contents. The receiving station generates its own BCC. If the two values match, the block is accepted as error-free. A BCC follows every ITB, ETB, and ETX character. If you transmit in the ASCII code set, the BCC is a longitudinal redundancy check (LRC). For the EBCDIC code set, the BCC is a cyclical redundancy check (CRC).
DLE	Data-Link Escape The first character in a 2-character sequence used to signal the beginning or end of transparent text mode. The sequence DLE STX signals the beginning of transparent text mode. The sequence DLE ETB or DLE ETX signals the end of transparent text mode.

Table 9-1. BSC Protocol Data-Link Control Characters (DLCC)(Cont.)

Character	Description
DLE EOT	Data-Link Escape, End-of-Transmission Signals a line disconnect for a switched line. The sending or receiving station usually transmits this sequence when all message exchanges are finished.
ENQ	Enquiry Sent by a station on a point-to-point line to bid for the line (for transmission of data). Sent by the control station on a multipoint line to signal the end of a polling or selecting sequence. A transmitting station can also send ENQ to ask the receiver to repeat a response if the original response was garbled or not received when expected.
EOT	End-of-Transmission Signals the end of a message transmission (consisting of one or more separately transmitted data blocks), and resets the receiving station. On a multipoint line, a polled station sends an EOT to indicate that it has nothing to send back to the control station. EOT can also serve as an abort signal to indicate a system or transmission malfunction.
ETB	End-of-Transmission Block
ETX	End-of-Text Signal the end of a data block that began with an SOH or an STX. Both the ETB and the ETX characters reverse the direction of the transmission. When a station receives an ETB or an ETX, it replies with a control character that indicates its status (that is, ACK0, ACK1, NAK, WACK, or RVI). An ETB terminates every text block except the last.

Table 9-1. BSC Protocol Data-Link Control Characters (DLCC)(Cont.)

Character	Description
ETX (Cont.)	An ETX implies an end-of-file condition; thus, it terminates the last block of text in a message.
ITB	End-of-Intermediate-Transmission Block Separates records within a block and/or delimits field boundaries for error checking. ITB does not reverse the direction of the transmission.
NAK	Negative Acknowledgment Sent by the receiving station to indicate that it is not ready to receive, or to request that an erroneous block be transmitted again.
RVI	Reverse Interrupt A positive response used instead of ACK0 or ACK1; signals that the receiver must interrupt the transmission to send the transmitting station a high-priority message. The transmitting station treats an RVI as a positive acknowledgment and, in response, transmits all the data that prevents it from becoming a receiving station. The transmitting station can perform more than one block transmission to empty all its buffers. On a multipoint line, a control station can send an RVI after it receives data from a tributary station, to indicate that it wants to communicate with a different tributary station.
SOH	Start of Header Signals the start of header information (ancillary information within a block).
STX	Start of Text Signals the beginning of the text (and terminates the header information).

Table 9-1. BSC Protocol Data-Link Control Characters (DLCC)(Cont.)

Character	Description
SYN	Synchronization Character Establishes and maintains character synchronization; also serves as filler in the absence of data or control characters. Each transmission must begin with at least two contiguous SYN characters.
TTD	Temporary Text Delay A 2-character sequence that consits of STX ENQ, which the transmitting station sends to retain the line without immediately sending data. The receiving station responds with a NAK. The TTD/NAK sequence can repeat, if the transmitter needs additional delays.
WACK	Wait-Before-Transmit Positive Acknowledgment A positive acknowledgment that the receiver sends; signals that the receiver is temporarily unable to receive. (A receiver can send WACK as a response to a line bid on a point-to-point line or a selection sequence on a multipoint line, or as a response to data.) A receiving station can send more than one WACK until it is ready to receive. The transmitting station can respond with ENQ, EOT, or DLE EOT.

Note that BSC protocol supports transparent text mode. Transparent text mode causes AOS/VS to treat most control characters as bit patterns without control significance. The exceptions are DLE STX, which signals the end of transparent text mode, and DLE ETB or DLE ETX, which signal the end of transparent text mode. If you are sending data that may match the bit patterns of the control characters, you should send it in transparent text mode.

BSC Error-Recovery Procedures

When AOS/VS receives an inappropriate response to an ?SSND or ?SRCV system call, or does not receive a response within the time-out interval that you specify in offset ?STOV, it enters its BSC error-recovery procedures. The error-recovery procedures differ, depending on which operation was underway when the error occurred. In addition, AOS/VS's action within each recovery procedure depends on the cause of the error.

In most cases, AOS/VS responds to a BSC error by trying the particular procedure again, repeatedly if necessary, until its retry count is exhausted. The retry count is a system-maintained variable, which you cannot control.

Table 9-2 describes the error-recovery procedures for the various types of send and receive system calls.

Table 9-2. BSC Error-Recovery Procedures

Call Type	AOS/VS Action
Send Initial	
Time-out NAK or inappropriate response	Resend ENQ, unless retry count exceeded. If retry count exceeded, take error return to ?SSND system call. (Possible errors in AC0 are ERTOF, ERNAK, ERUNI.)
ENQ	If calling station is the primary, resend ENQ. If calling station is the secondary, take error return to ?SSND system call. (Error in AC0 is ERCTN.)
Send Continue	
Time-out or inappropriate response	Send ENQ, unless retry count exceeded. If retry count exceeded, take error return to ?SSND system call. (Possible errors in AC0 are ERTOF, ERUNI.)
NAK	Resend data, unless retry count exceeded. If retry count exceeded, take error return to ?SSND system call. (Error in AC0 is ERNAK.)

Table 9-2. BSC Error-Recovery Procedures (Cont.)

Call Type	AOS/VS Action
Receive Initial (point-to-point and multipoint tributary station)	
Time-out or inappropriate response	Retry receive initial, unless retry count exceeded. Take error return to ?SRCV system call. (Possible errors in AC0 are ERTOF, ERUNI.)
Receive Continue	
Time-out or inappropriate response	If retry count exceeded, take error return to ?SRCV (Possible errors in AC0 are ERTOF, ERUNI.) Otherwise, await ENQ from sender (assuming that the sender will issue a time-out and send an ENQ).
ENQ	Resend last response and attempt receive continue, unless retry count exceeded. If retry count exceeded, take error return to ?SRCV system call. (Error in AC0 is ERENQ.)
CRC (block check) error	Send NAK and attempt receive continue, unless retry count exceeded. If retry count exceeded, take error return to ?SRCV system call. (Error in AC0 is ERCRC.)
Receive Initial (multipoint control station)	
Time-out or inappropriate response	Retry receive initial, unless retry count for particular relative console exceeded. If retry count exceeded, take error return to ?SRCV system call (Possible errors in AC0 are ERTOF, ERUNI.)
EOT	If a polled console responds with EOT, step to the next relative console on the polling list and continue polling. If end of the polling list is reached, take error return to ?SRCV system call. (Error in AC0 is EREPL.)

To get BSC error-recovery statistics, issue the ?SGES system call. The ?SGES system call returns the number of block-check errors, the number of time-outs, and the total number of negative acknowledgment (NAK) characters received in response to send operations.

BSC Implementation

Figures 9-2 through 9-8 illustrate how AOS/VS implements BSC protocol using the BSC system calls. Before you read this section, refer to the system call descriptions for the ?SEBL, ?SSND, and ?SRCV system calls in Chapter 13 and to the definitions of the BSC data-link control characters in Table 9-1.

Remember, you cannot issue a send initial system call from a multipoint tributary station, and that receive continue system calls from multipoint stations and from point-to-point stations are identical.

You will notice that each figure has three columns. The first column represents the system calls that you issue, with their normal and error returns. The second column illustrates AOS/VS's actions, and the third column shows the remote station's actions.

```
 ____________________________________________________________________
|     User Task     #       AOS/VS          #      Remote Site      |
|===================#=======================#=======================|
| ________          #                 SYN   #                       |
| | DATA |---------------------------> : ------------------------|  |
| |______|          #       ^         ENQ   #  |    |     |   |  |  |
| ?SSND Initial     #       |         PAD   # NAK   |    ENQ  | ACK |
|                   #     Retry        |    #  | Inapp.   |   |  |  |
|                   #       |          V    #  |    |     |   |  |  |
|                   #  _____|_____ Time-out #  |    |     |   |  |  |
|                   #  | ERROR  |      |    #  V    V     V   |  |  |
|                   #  |RECOVERY|<-----|------------------|   |  |  |
|                   #  |________|           #                 |  |  |
|                   #       |               #                 |  |  |
|                   #       V               #      EOT        V  |  |
| Error Return <-------------------------------- DLE EOT -----|  |  |
|                   # |     _____________________________________|  |
|                   # |     ^          |    #                       |
|                   # |     |         SYN   #                       |
|                   # |     |          :    #                       |
|                   # |     |        [DLE]  #                       |
|                   # |     |         STX   #                       |
|                   # |     |         SOH   #                       |
|                   # |     |         TEXT  #                       |
|                   # |     |        [DLE]  #                       |
|                   # |     |         ETX   #                       |
|                   # |   Retry       ETB   #                       |
|                   # |     |         CRC   #                       |
|                   # |     |         CRC   #                       |
|                   # |     |         PAD   #                       |
|                   # |     |          |___________________________ |
|                   # |     |          |    #  |   |    |    |   |  |
|                   # |     |          |    #  |   | RVI     | ACK0 |
|                   # |     |          V    # NAK  | EOT     | ACK1 |
|                   # |     |      Time-out #  |   | DLE/EOT |   |  |
|                   # |     |          |    #  |   |    |   SYN  |  |
|                   # |_____|_____     |    #  |   |    |    :   |  |
|                   # || ERROR  |      |    #  | Inapp. |  [DLE] |  |
|                   # ||RECOVERY|<-----|-------|---|    |   STX  |  |
|                   # ||________|           #           |   SOH  |  |
|                   # |     |               #           |  TEXT  |  |
|                   # |     |               #           |  [DLE] |  |
|                   # |     |               #           |   ETX  |  |
|                   # |     |               #           |   ETB  |  |
|                   # ^     |               #           |   CRC  |  |
|                   # |     |               #           |   CRC  V  |
|                   # |     |               #           |   PAD  |  |
|                   # |_____V___________________________V____|   |  |
| Normal Return <------------------------------------------------|  |
|___________________#_______________________#_______________________|
```

Figure 9-2. ?SSND System Call, Initial, Point-to-Point

```
|     User Task     #        AOS/VS          #      Remote Site      |
|===================#========================#=======================|
| ________          #                        #                       |
| | DATA |_______________________________    #                       |
| |______|          #        ^          |    #                       |
| ?SSND Continue    #        |         SYN   #                       |
|                   #        |          :    #                       |
|                   #        |        [DLE]  #                       |
|                   #        |         STX   #                       |
|                   #        |         SOH   #                       |
|                   #        |        TEXT   #                       |
|                   #     Retry       [DLE]  #                       |
|                   #        |         ETX   #                       |
|                   #        |         ETB   #                       |
|                   #        |         CRC   #                       |
|                   #        |         CRC   #                       |
|                   #        |         PAD   #                       |
|                   #        |          |___________________________ |
|                   #        |          |    #  |   |    |    |   |  |
|                   #        |          |    #  |   | RVI     | ACK0 |
|                   #        |          V    # NAK  | EOT     | ACK1 |
|                   #        |     Time-out  #  |   | DLE/EOT |   |  |
|                   #        |          |    #  | Inapp. |   SYN  |  |
|                   # _______|_____     |    #  |   |    |    :   |  |
|                   # |   ERROR   |     V    #  V   |    |  [DLE] |  |
|                   # | RECOVERY  |<----------------|    |   STX  |  |
|                   # |___________|          #           |   SOH  |  |
|                   #        |               #           |  TEXT  |  |
|                   #        |               #           |  [DLE] |  |
|                   #        |               #           |   ETX  |  |
|                   #        |               #           |   ETB  |  |
|                   #        |               #           |   CRC  |  |
|                   #        |               #           |   CRC  |  |
|                   #        |               #           |   PAD  |  |
|                   #        |               #           |    |   |  |
|                   #        V               #           V    V   |  |
| Error Return <----------------------------------------------|   |  |
|                   #                        #                    V  |
| Normal Return <-------------------------------------------------|  |
|___________________#________________________#_______________________|
```

Figure 9-3. ?SSND System Call, Continue, Point-to-Point

```
_____________________________________________________________________
|     User Task     #        AOS/VS         #      Remote Site      |
|===================#=======================#=======================|
| ?SRCV Initial-----#-------------->Wait    #      SYN       Bid    |
|                   #       ^        Bid    #       :        SYN    |
|                   #       |         |     #    Inappro-     :     |
|                   #     Retry    Time-out #    priate      ENQ    |
|                   # ______|______   |     #       |         |     |
| ?SRCV Initial     #  |  ERROR   |   V     #       |         |     |
| Error Return <-------| RECOVERY |<----------------|         V     |
|                   #  |__________|         #                 |     |
|                   #                       #                 |     |
| ?SRCV Continue <--------------------------------------------|     |
|                   #       |               #                       |
|                   #       V               #                       |
|                   #     ACK0              #                       |
|                   #     ACK1              #                       |
|                   #       |               #                       |
|                   #       V____________________________________   |
|                   #       ^         |     #  |   |    |   |   |   |
|                   #       |      Time-out #  |  ENQ   |   |  SYN  |
|                   #     ENQ         |     #  |   |  CRC  |   :   |
|                   #     NAK         |     #Inapp.|  Error | [DLE] |
|                   #       |         |     #  |   |    |   |  STX  |
|                   # ______V______   |     #  |   |    |   |  SOH  |
|                   #  |  ERROR   |   V     #  V   V    V   V TEXT  |
|                   #  | RECOVERY |<--------------------|   | [DLE] |
|                   #  |__________|         #               |  ETB  |
| ?SRCV Continue    #                       #      RVI      |  ETX  |
| Error Return  <--------------------------------  EOT    --|  CRC  |
|                   #                       #    DLE/EOT       CRC  |
|                   #                       #                   |   |
| ?SRCV Continue <----------------------------------------------|   |
| Normal Return     #                       #                       |
|___________________#_______________________#_______________________|
```

Figure 9-4. ?SRCV System Call, Initial and Continue, Point-to-Point

```
 ____________________________________________________________________
|                   #        AOS/VS          #      Remote Site      |
|     User Task     #  (Control Station)     #      (Tributary)      |
|===================#========================#=======================|
| --------          #                        #                       |
|                      ________________                              |
| | DATA |----------->|Select Operation|_________________________    |
| |______|          # |________________|     # |   |   |   |   |     |
| ?SSND Initial     #         ^      |       # NAK |  WACK |   |     |
|                   #         |   Time-out   # |  RVI  | ACK0  |     |
|                   #      Retry     |       # |   |   |   | Inapp.  |
|                   #      ___|______|__     # V   V   V   V   |     |
|                   #       |  ERROR   |     # |   |   |   |   |     |
|                   #       | RECOVERY |<------|---|---|---|---|     |
|                   #       |----|-----|     # |   |   |   |         |
|                   #            V           # V   V   V   |         |
|  Error Return <--------------------------------------|   |         |
|                   #                        #             |         |
| --------          #                        #             |         |
| |      |------------------><-----------------------------|         |
| |______|          #       |                #                       |
| ?SSND Continue    #      SYN               #                       |
|                   #       :                #                       |
|                   #     [DLE]              #                       |
|                   #      STX               #                       |
|                   #      SOH               #                       |
|                   #     TEXT               #                       |
|                   #     [DLE]              #                       |
|                   #      ETX               #                       |
|                   #      ETB               #                       |
|                   #      CRC               #                       |
|                   #      CRC               #                       |
|                   #       |_____________________________________   |
|                   #       |          ^     # |   |    |    |   |   |
|                   #       V          |     #NAK  | RVI     | ACK0  |
|                   #   Time-out    Retry    # |   | EOT     | ACK1  |
|                   #       |          |     # |   | DLE/EOT |   |   |
|                   #       |          |     # |   |    |   SYN  |   |
|                   #_______|_____     |     # | Inapp. |    :   |   |
|                   # |  ERROR   |-----|     # |   |    |  [DLE] |   |
|                   # | RECOVERY |           # V   |    |   STX  |   |
|                   # |__________|<----------------|    |   SOH  |   |
|                   #       |                #          |  TEXT  |   |
|                   #       |                #          |  [DLE] |   |
|                   #       |                #          |   ETX  |   |
|                   #       |                #          |   ETB  |   |
|                   #       |                #          |   CRC  |   |
|                   #       |                #          |   CRC  |   |
|                   #       V                #          V    |   |   |
| Error Return <---------------------------------------------|   |   |
| Normal Return <------------------------------------------------|   |
|___________________#________________________#_______________________|
```

Figure 9-5. ?SSND System Call, Multipoint Control Station

```
_____________________________________________________________________
|                   #        AOS/VS         #      Remote Site      |
|     User Task     #   (Control Station)   #      (Tributary)      |
|===================#=======================#=======================|
|                   #   __________________  #                       |
| ?SRCV Initial-------->| Poll Operation |________________________  |
|                   #   |________________|  #  |        |       |   |
|                   #       ^        |      # EOT     Inapp.   SYN  |
|                   #       |        |      #  |        |       :   |
|                   #       |     Time-out  #  |        |     [DLE] |
|                   #     Retry      |      #  |        |      STX  |
|                   #       |        |      #  |        |      SOH  |
|                   #  _____|________V___   #  V        V     TEXT  |
| Error Return <-------| ERROR RECOVERY |<--------------|     [DLE] |
|                   #  |________________|   #                  ETX  |
|                   #                       #                  ETB  |
|                   #                       #                  CRC  |
|                   #                       #                  CRC  |
|                   #                       #                   |   |
| Normal Return <-----------------------------------------------|   |
|                   #                       #                       |
| ?SRCV Continue-------------               #                       |
|                   #       |               #                       |
|                   #       V               #                       |
|                   #      ACK0             #                       |
|                   #      ACK1             #                       |
|                   #      NAK              #                       |
|                   #      RVI              #                       |
|                   #       |               #                       |
|                   #       |               #                       |
|                   #       V_____________________________________  |
|                   #       ^         |     #  |   |    |   |   |   |
|                   #       |      Time-out #  |  ENQ   |   |  SYN  |
|                   #      NAK        |     #  |   |   CRC  |   :   |
|                   #       |         |     #Inapp.|  Error | [DLE] |
|                   #       |         |     #  |   |    |   |  STX  |
|                   #  _____V_____    |     #  |   |    |   |  SOH  |
|                   #  |  ERROR   |   V     #  V   V    V   | TEXT  |
|                   #  | RECOVERY |<--------------------|   | [DLE] |
|                   #  |__________|         #               |  ETB  |
|                   #                       #       RVI     |  ETX  |
| Error Return <---------------------------------   EOT   --|  CRC  |
|                   #                       #     DLE/EOT      CRC  |
|                   #                       #                   |   |
| Normal Return <-----------------------------------------------|   |
|                   #                       #                       |
|___________________#_______________________#_______________________|
```

Figure 9-6. ?SRCV System Call, Multipoint Control Station

```
_____________________________________________________________________________
|                  #        AOS/VS          #      Remote Site             |
|    User Task     #  (Control Station)     #      (Tributary)             |
|==================#========================#==============================|
|                  #                        #              SYN             |
|                  #                        #               :              |
|                  #                        #              EOT             |
|                  #   ____________________ #               |              |
| ?SRCV Initial--------->| Wait for Poll or| #  SYN     |---o---|          |
|                  #    | Select Address  | #   :       |       |          |
|                  #    |_________________| # Inapp.  Poll    Select        |
|                  #         ^        |     #   |     Addr.   Addr.         |
|                  #         |     Time-out #   |     Seq.    Seq.          |
|                  #      Retry       |     #   |      :       :            |
|                  #         |        |     #   V     ENQ     ENQ           |
|                  #  _______|________V___  #   |      |       |            |
| Error Return <-------| ERROR RECOVERY |<-------|     |       |            |
|                  #   |________________|   #          |       |            |
|                  #                        #          |       |            |
|                  #       _______________  #          |       |            |
|                  #       | ?SPLR Set   |  #          V       |            |
|                  #  |---| to 1        |<-------------|       |            |
|                  #  |   |_____________|  #                  |            |
|                  #  V                     #                  |            |
| Normal Return <------o                     #                  |            |
|                  #  ^                     #                  |            |
|                  #  |    _______________  #                  V            |
|                  #  |---| ?SSLR Set   |<----------------------|            |
|                  #      | to 1        |   #                               |
|                  #      |_____________|   #                               |
|__________________#________________________#______________________________|
|                                                                          |
| NOTE:  Use ?SRCV continue if the station was selected; use ?SSND         |
|        if the station was polled.                                        |
|__________________________________________________________________________|
```

Figure 9-7. ?SRCV System Call, Multipoint Tributary Station

```
 ______________________________________________________________________
|      User Task      #           AOS/VS         #       Remote Site      |
|===================#======================#========================|
| ?SEBL------------------------>Assert Data    #                        |
|                   #    |    Terminal Ready #                        |
|                   #    |    (wait for      #                        |
|                   #    |    dataset ready) #                        |
|                   #    V          |         #                        |
| Error Return <-----Time-out      |         #                        |
|                   #              |<----------Dataset Ready Asserted|
| Normal Return <-------------------         #                        |
|                   #                        #                        |
|___________________#________________________#________________________|
```

Figure 9-8. ?SEBL System Call, Point-to-Point

End of Chapter

CHAPTER 10
USER DEVICE SUPPORT

The system calls that support user devices are:

?DDIS	Disables access to all devices.
?DEBL	Enables access to all devices.
?IDEF	Defines a user device.
?IMSG	Receives an interrupt message.
?IRMV	Removes a user device.
?IXIT	Exits from an interrupt service routine.
?IXMT	Transmits an interrupt message.
?LEFD	Disables LEF mode.
?LEFE	Enables LEF mode.
?LEFS	Returns the current LEF mode status.
?STMAP	Sets the data channel map.

AOS/VS supports a wide variety of user devices, such as magnetic tape drives, disk drives, and line printers, which you usually define during the system-generation procedure. (Refer to the 'Managing AOS/VS' manual for information on the system-generation dialog.) However, a process that has the ?PVDV privilege can define and enable devices at execution time.

Devices that you define and enable during the system-generation procedure are called system-defined devices. Devices that a process with the ?PVDV privilege defines and enables at execution time are called user-defined devices. This chapter describes those system calls that allow you to use both system- and user-defined devices.

AOS/VS supports a maximum of 64 user (that is, system-defined and/or user-defined) devices per system. You can use any device code in the range from 1 through 63, as long as you do not use codes that are already in use. (Note that AOS/VS reserves many device codes for its own use.)

To introduce a user-defined device to AOS/VS at execution time, you must issue the ?IDEF system call. As input to the ?IDEF system call, you must supply:

- o A device code for the new device.
- o The address of the device control table (DCT) you defined for the new device.

The DCT specifies the address of the user-defined device's interrupt service routine. (See Figure 10-1.)

```
           0                          15 16                         31
           |---------------------------|---------------------------|
 ?UDVMX    |  Pointer to system database for task that issues      |
           |  ?IMSG                                                |
           |-------------------------------------------------------|
 ?UDVIS    |           Address of interrupt service routine        |
           |-------------------------------------------------------|
 ?UDVBX    |     Mailbox for message sent via ?IXMT and ?IMSG      |
           |-------------------------------------------------------|
 ?UDDRS    |        Address of user-defined power-failure/auto-    |
           |        restart routine or -1                          |
           |-------------------------------------------------------|
 ?UDVMS    |  Interrupt service mask   |  Reserved (Set to -1.)    |?UDRS
           |___________________________|___________________________|
            ?UDLN = DCT length
```

Figure 10-1. Device Control Table (DCT)

The DCT is ?UDLN words long. Note that AOS/VS returns most of the DCT parameters as output to the ?IDEF system call. However, you must perform the following steps:

1. Supply the address of the interrupt service routine (offset ?UDVIS).
2. Supply the address of the power-failure/auto-restart routine or, if you do not want to use such a routine, -1 (offset ?UDDRS).
3. Provide the interrupt service mask (offset ?UDVMS).

To remove a user device, issue the ?IRMV system call.

?IDEF System Call Options

When you issue the ?IDEF system call, you can select any of the following options:

- Burst multiplexor (BMC) I/O.
- Data channel (DCH) map A.
- DCH map B, C, or D.
- Neither BMC nor DCH I/O.

You can select either burst multiplexor (BMC) I/O or data channel (DCH) I/O for a user-defined device by selecting certain options when you issue the ?IDEF system call. (In general, your choice depends on the device's design.)

If you want to use the BMC map or the DCH B, C, or D maps, you must use an extended map table. However, you can define (issue the ?IDEF system call against) a device that is on DCH map A without using an extended map table. To do this, you must specify that you do not want to use the extended map table in the accumulators when you issue the ?IDEF system call. This option, does not allow you to specify the first acceptable map slot. Instead, you can only specify how many map slots your application needs.

However, if you do not want to use either DCH or BMC I/O, you must specify this option in the accumulators when you issue ?IDEF. Also, if you do not want to use either DCH or BMC I/O, you do not have to use the extended map table.

Burst multiplexor I/O requires program control only at the beginning of each block transfer. Therefore, BMC I/O is generally faster than DCH I/O. Typically, the BMC rate is about half the memory rate, although the exact transfer rate varies from implementation to implementation. Note that not all user-defined devices have BMC hardware.

If you use the extended form of ?IDEF, you can select specific DCH or BMC map slots. Each MV/8000 DCH map consists of 32 map slots, numbered 0 through 37 (octal). The MV/8000 BMC map consists of 1024 map slots, numbered 0 through 1777 (octal).

Each map slot (DCH and BMC) addresses 1K memory words. The hardware uses these map slots to map data from the device to memory during data transfers.

To select a particular DCH map or the BMC option, you must perform the following steps:

1. Set up a map definition table in your logical address space. (Figure 10-2 shows the structure of a map definition table and its entries. See Table 10-1 for a detailed description of the contents of each map definition table entry.

2. Issue the ?IDEF system call.

When you issue the ?IDEF system call, AOS/VS allocates--but does not initialize--map slots. To initialize these map slots, you must issue the ?STMAP system call.

If you issued the ?IDEF system call with the DCH map-A-only option, then you can issue only one ?STMAP system call to initialize the map slots allocated on DCH map A. However, if you issued the ?IDEF system call with the extended-map-table option, you can issue more than one ?STMAP system call. Each ?STMAP system call, in turn, initializes a different group of map slots. (The map definition table entries define each group of map slots.)

When you issue the ?STMAP system call, you can initialize part of a group of map slots that is defined in a map definition table entry. For example, if entry 2 has allocated 10 map slots on the BMC, an ?STMAP system call only initializes 5 of the 10 map slots.

(For a detailed description of BMC I/O, DCH I/O, and the DCH maps, refer to the 'Principles of Operation 32-Bit ECLIPSE® Systems' manual.)

```
 ______________________________________________________________________
|                                                                      |
|                                =================================}    |
|                                ||              Word 1 - First Acce{ |
|                                ||---------------------------------} |
|                                ||Offset|          Contents        { |
|                                ||======|==========================} |
|                                ||?UDID | The format is:           { |
| ========================      ||      |                          } |
| |  ------------------- |     ||      | [Map specifier] +        { |
| |  |          | Word 1 | |     ||      | [first acceptable slot]{ |
| |  | Entry 1  |--------| |=====>||      |                          } |
| |  |          | Word 2 | |     ||      | The following are        { |
| |  |-------------------| |     ||      | sample entries:          } |
| ========================      ||      |                        {   |
|    |          | Word 1 |        ||      |   ?UDDC+10               { |
|    | Entry 2  |--------|        ||      |   ?UDDB+2                } |
|    |          | Word 2 |        ||      |   ?UDBM+322            {   |
|    |-------------------|        ||      |                      {     |
|    |          | Word 1 |        ||                              }    |
|    | Entry 3  |--------|        ||     (See Table 10-1.)      {     |
|    |          | Word 2 |        ||                            }      |
|    |-------------------|        ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~        |
|    |          | Word 1 |                                             |
|    | Entry 4  |--------|                                             |
|    |          | Word 2 |                                             |
|    |-------------------|                                             |
|    |          | Word 1 |                                             |
|    | Entry 5  |--------|                                             |
|    |          | Word 2 |                                             |
|    |-------------------|                                             |
|    |          | Word 1 |                                             |
|    | Entry 6  |--------|                                             |
|    |          | Word 2 |                                             |
|    |-------------------|                                             |
|    |          | Word 1 |                                             |
|    | Entry 7  |--------|                                             |
|    |          | Word 2 |                                             |
|    |-------------------|                                             |
|    |          | Word 1 |                                             |
|    | Entry 8  |--------|                                             |
|    | (max.)*  | Word 2 |                                             |
|    ---------------------                                             |
|    * If there are fewer than eight                                   |
|      2-word entries, the first                                       |
|      word following the last valid                                   |
|      entry must be -1.                                               |
|______________________________________________________________________|
```

Figure 10-2. Structure of Map Definition Table

Table 10-1. Contents of Map Definition Table Entry

Word 1 - First Acceptable Map Slot

Offset	Contents	Comments
?UDID	The format is: [Map specifier] + [first acceptable slot] The following are sample entries: ?UDDC+10 ?UDDB+2 ?UDBM+322	Map specifier must be one of the following: o ?UDBM, which selects the BMC map o ?UDDA, which selects the DCH A map o ?UDDB, which selects the DCH B map o ?UDDC, which selects the DCH C map o ?UDDD, which selects the DCH D map First acceptable slot must be: o From 0 through 1777 (octal) if your map specifier is ?UDBM o From 0 through 37 (octal) if your map specifier is ?UDDA, ?UDDB, ?UDDC, or ?UDDD AOS/VS allocates the first contiguous group of slots on the map, starting with the first acceptable slot on the map that you selected. Then, AOS/VS returns to you the first slot that it allocated in ?UDID.

Word 2 - Number of Map Slots Requested

Offset	Contents	Comments
?UDNS	Number of map slots requested in range from 0 through 37 (octal)	The sum of the first acceptable slot plus the number of slots cannot be larger than the size of the map that you requested; that is, 37 (octal) for DCH or 2000 (octal) for BMC.

NOTE: If AOS/VS cannot allocate all entries, then it does not allocate any entries.

User Interrupt Service

To define a user device to AOS/VS, you must issue the ?IDEF system call. Each device, such as a disk, is programmed to do a particular job. When a device starts doing its job, the CPU and AOS/VS ignore that device. As soon as the device completes its job, it signals the CPU that it is done. This signal is called an interrupt.

When the CPU detects an interrupt, it stops doing whatever it is doing, so that it can "service" the interrupt. Servicing an interrupt means that AOS/VS passes control to the appropriate interrupt service routine. To do this, AOS/VS must pass a vector through the interrupt vector table, which is a hardware-defined index.

The interrupt vector table contains an entry for each device. Each entry points to a DCT, which contains the address of the interrupt service routine that will service a particular interrupt.

The ?IDEF system call directs AOS/VS to build a system DCT and enter it in the interrupt vector table. Conversely, the ?IRMV system call clears the device's DCT entry from the interrupt vector table.

The device's DCT also contains the current interrupt service mask. The current interrupt service mask is a value that specifies the devices that can interrupt the user-defined device.

Before AOS/VS transfers control to an interrupt service routine, it performs the following steps:

1. Loads AC2 with the address of the interrupting device's DCT.

2. Loads AC0 with the current interrupt service mask.

3. Takes the current interrupt service mask and inclusively ORs it with the interrupt service mask in the DCT.

4. Saves the current load effective address (LEF) state. LEF mode is a CPU state that allows AOS/VS to correctly interpret MV/8000 LEF instructions. (See "LEF Mode" in this chapter for information on LEF mode.)

5. Turns LEF mode off.

The inclusive-OR operation establishes which devices, if any, can interrupt the interrupt service routine that is currently executing. AOS/VS restores LEF mode when you issue an ?IXIT system call to dismiss your interrupt.

A process in an interrupt service routine can issue only three system calls:

- ?IXMT, which sends a message to a task outside the interrupt service routine.
- ?SIGNL, which signals a task within your own or another process.
- ?IXIT, which exits from an interrupt service routine.

AOS/VS does not save the state of the MV/8000 floating-point registers when a process enters an interrupt service routine. If necessary (for example, if you want to use floating-point instructions), you can save the state of the floating-point registers when the interrupt service routine receives control and restore that state before you issue the ?IXIT system call.

User Stacks

Each user task has its own user stack. A user stack is a data structure to which the contents of certain Page 0 hardware locations point. The contents of these hardware locations are called stack pointers. Whenever a task runs, AOS/VS must first load that task's stack pointers into hardware locations octal 20 through 26. This allows the task to use its stack.

When a user-defined device interrupt handler receives control at interrupt level, the stack that AOS/VS loads into the Ring 7 hardware registers is the stack of the last user task that was running. AOS/VS does not set up a stack for your interrupt service routine. Therefore, to use a stack, you must set up your own.

Before you issue the ?IXIT system call to exit from the interrupt service routine, you must perform the following steps:

1. Save the current hardware stack pointers.

2. Restore the current hardware stack pointers to the hardware.

NOTE: If you use the stack that is already loaded in Ring 7, you must also restore that stack to the way it was when you first received control (that is, before you issued the ?IXIT system call).

Communicating from an Interrupt Service Routine

Multitasking halts when a device interrupt occurs. However, an interrupt service routine can communicate with an outside task by issuing the ?IXMT system call. The ?IXMT system call transmits a message of up to 32 bits from the interrupt routine to a specific receiving task outside the sending routine. There is a location in the system DCT that serves as a mailbox for the message. The external task receives the message by issuing a ?IMSG system call against the DCT associated with the interrupt routine.

You can issue ?IXMT and ?IMSG system calls in any order. If the ?IMSG system call occurs before the ?IXMT system call, AOS/VS suspends the receiving task until the ?IXMT system call occurs. If the ?IXMT system call occurs first, AOS/VS posts the message in the mailbox until the receiving task issues the ?IMSG system call.

You cannot use the ?IXMT system call to broadcast a message.

Enabling and Disabling Access to All Devices

Processes can issue I/O instructions from their tasks to all system and user devices. When a process issues a ?DEBL system call, AOS/VS enables device I/O and disables LEF mode, which allows tasks within the calling process to issue I/O instructions. Note that the I/O enable and LEF mode states are process wide, and therefore, affect all tasks.

The ?DEBL and ?DDIS system calls work in parallel with the LEF mode system calls ?LEFE, ?LEFD, and ?LEFS. Table 10-2 summarizes the functions of the LEF mode and device access calls. (See "LEF Mode" in this chapter for more information on LEF mode.)

Table 10-2. LEF Mode and Device Access System Call Functions Summary

System Call	Function
?DEBL	Enables I/O, disables LEF mode.
?DDIS	Disables I/O, but does not re-enable LEF mode.
?LEFE	Enables LEF mode, disables I/O.
?LEFD	Disables LEF mode, but does not enable I/O.
?LEFS	Returns the current LEF mode status.

No device I/O can occur while the CPU is in LEF mode, because LEF instructions and I/O instructions use the same bit patterns. Similarly, when LEF mode is disabled, AOS/VS executes LEF instructions as if they were I/O instructions. Thus, the deciding factor for executing LEF and I/O instructions is the state of the CPU; that is, whether it is in LEF mode or I/O mode.

Note that the ?DDIS system call, which disables I/O mode, does not automatically re-enable LEF mode. To disable I/O mode and re-enable LEF mode, you must issue the ?LEFE system call. Also, the ?LEFD system call, which disables LEF mode, does not automatically re-enable I/O mode. To perform these two functions, you must issue the ?DEBL system call.

LEF Mode

LEF mode (load-effective-address mode) is the CPU state that protects the I/O devices from unauthorized access. I/O instructions and LEF instructions use the same bit patterns. AOS/VS decides how to interpret these instructions by checking the LEF mode state and the state of the complementary I/O mode.

LEF mode and I/O mode are mutually exclusive. When the CPU is in LEF mode, all I/O instructions execute as LEF instructions; therefore, I/O cannot take place in this state. Conversely, the CPU must be in LEF mode to execute LEF instructions properly.

AOS/VS provides the following system calls to check and alter LEF mode:

?LEFD	Disables LEF mode.
?LEFE	Enables LEF mode.
?LEFS	Returns the current LEF mode status.

Each process begins with LEF mode enabled. AOS/VS disables LEF mode when a process enters a user device routine, and restores LEF mode when the process exits from that routine.

Power-Failure/Auto-restart Routine

If you specify an extended DCT within the ?IDEF system call--provided you have the necessary battery backup hardware--AOS/VS will restart your user devices after a power failure. The DCT extension (offset ?UDDRS) points to a power-failure/auto-restart routine. When a power failure occurs, AOS/VS transfers control to the auto-restart routine, with the DCT address in AC2, and the current system mask in AC0.

AOS/VS checks to see if there are any user-defined devices that have associated power-failure/restart routines if auto-restart is enabled. (Refer to the 'Managing AOS/VS' manual.)

(During the auto-restart routine, AOS/VS enables interrupts and masks out all devices. This allows AOS/VS to recognize only power-failure interrupts.) AOS/VS transfers control to the auto-restart routine with a system mask of -1, which cannot be changed. The states of both the devices and the data channel map are undetermined after a power failure.

End of Chapter

CHAPTER 11
MISCELLANEOUS SYSTEM CALL FUNCTIONS

The system calls whose functions fall outside the specific topics addressed in this manual are:

?BNAME	Determines whether processname/queuename is on local or remote host.
?CDAY	Converts a scalar date value.
?CTOD	Converts a scalar time value.
?DEBUG	Calls the debugger utility.
?ENQUE	Queues a file entry.
?ERMSG	Reads the error message file.
?EXEC	Requests a service from EXEC.
?FDAY	Converts the date to a scalar value.
?FEDFUNC	Interfaces to AOS/VS File Editor (FED) utility.
?FTOD	Converts the time of day to a scalar value.
?GBIAS	Gets the current bias factor values.
?GDAY	Gets the current date.
?GHRZ	Gets the frequency of the system clock.
?GSID	Gets the system identifier.
?GTMES	Gets a CLI messages.
?GTNAM	Returns symbol closest in value to specified input value.
?GTOD	Gets the time of day.
?GTSVL	Gets the value of a user symbol.
?GVPID	Gets the virtual PID of a process.
?HNAME	Gets a host name or host identifier.
?INTWT	Defines a console interrupt task.
?ITIME	Returns the AOS/VS-format internal time.
?KINTR	Simulates keyboard interrupt sequences.
?KIOFF	Disables control-character console interrupts.
?KION	Re-enables control-character console interrupts.
?KWAIT	Waits for a console interrupt.
?LOGCALLS	Logs system calls.
?LOGEV	Enters an event in the system log file.
?ODIS	Disables console interrupts.
?OEBL	Enables console interrupts.

(Cont.)	
?RNAME	Determines whether a pathname contains a reference to a remote host.
?SBIAS	Sets the bias factors.
?SDAY	Sets the system calendar.
?SINFO	Gets selected information about the current operating system.
?STOD	Sets the system clock.
?TPID	Translates a PID.
?VALAD	Validates a logical address.
?WDELAY	Suspends a task for a specified time.

This chapter describes those system calls that fall into the "miscellaneous" category, either because their functions extend beyond the broad categories covered by the other chapters or because they apply to more than one of the categories treated in the other chapters.

Several of the system calls described in this chapter examine and/or change system features, such as the error message file. Other system calls return system information or perform general functions for the calling process.

Console Interrupts

As stated previously, you can control or suspend printing at your console by typing certain keyboard control characters or control sequences.

A CTRL-C CTRL-A sequence interrupts printing at your console. However, this sequence works only if you have already issued a ?INTWT system call. The ?INTWT system call defines an interrupt processing task that monitors the console keyboard for CTRL-C CTRL-A sequences. When AOS/VS detects a CTRL-C CTRL-A sequence, it readies the interrupt processing task and passes control to the ?INTWT system call's normal return. AOS/VS ignores subsequent CTRL-C CTRL-A sequences until you re-issue the ?INTWT system call. After you issue the ?INTWT system call, you can used the ?OEBL system call to re-enable console interrupts.

By default, AOS/VS enables CTRL-C CTRL-A interrupts when a program starts to execute. However, the ?ODIS system call lets you override this default and disable console interrupts that were caused by the ?OEBL, ?INTWT, or ?CHAIN system calls.

A CTRL-C CTRL-B sequence terminates the current process, whether or not you have defined an interrupt task with the ?INTWT system call. A CTRL-C CTRL-E sequence terminates the current process and creates a break file of its state. (See the description of break files in Chapter 3.)

The ?KINTR, ?KWAIT, ?KIOFF, and ?KION system calls allow you to control interrupts on virtual consoles. The ?KINTR system call allows virtual consoles to handle console interrupts as if they were real consoles. If you want a process to handle console interrupts in nontraditional ways, you can issue the ?KWAIT system call. Then, to prevent a process from being interrupted by a control sequence, you can issue the ?KIOFF system call. To re-enable interrupts that were disabled by the ?KIOFF system call, you can issue the ?KION system call.

Clock/Calendar System Calls

AOS/VS maintains a 24-hour clock and a calendar. During the system-generation procedure, you can set the clock to any one of several real-time frequencies. (Refer to the 'Managing AOS/VS' manual for more information on the system-generation procedure.)

Depending on your application, you may need to know the real-time frequency of the system clock while your program is executing. The ?GHRZ system call returns this information to AC0 as a code in the range from 0 through 4. Each digit of the code corresponds to a specific frequency. (See the description of the ?GHRZ system call in Chapter 13 for details.)

The system clock expresses the current time in seconds, minutes, and hours; the values for seconds and minutes range from 0 through 59, and the value for the hour ranges from 0 (midnight) through 23 (11 p.m.). You can issue the ?ITIME system call to get an AOS/VS-format timestamp.

The system calendar expresses the current date as day, month, and year. To determine the year, the system calendar subtracts the base year 1900 from the current year and converts the result to octal. The notation for 1980, for example, is 120 octal.

The system calls ?STOD and ?SDAY set the system clock and calendar, respectively. The ?GTOD and ?GDAY system calls return the current time and date, respectively.

In some cases, such as in the ?FSTAT packet, AOS/VS returns the time and/or date as a scalar value. In scalar notation, the current time equals the number of biseconds that have elapsed since midnight. The date equals the number of days that have elapsed since 31 December 1967. The ?CTOD system call converts a scalar time to seconds, minutes, and hours. The ?CDAY system call converts a scalar date to month, day, and year. To convert time and date back to scalar values, issue the ?FTOD and ?FDAY system calls, respectively.

Error Message File

The system file ERMES contains all the error codes, their corresponding mnemonics, and their text messages. There are 20000 (octal) groups of error codes for AOS/VS (including user programs). Data General Corporation reserves code groups 0 through 77 (octal) and 200 through 7777 (octal) for the system. You can define the remaining groups, numbered 100 through 177 (octal) and 10000 through 17777 (octal).

The error codes are 32-bit unsigned values. Each error code contains two fields: a group field and an error code field. If an error occurs when a system call is executing, AOS/VS returns the error code value to AC0. Each error is associated with a unique text string.

The ?ERMSG system call returns the text string associated with a particular error code. Before you issue the ?ERMSG system call, you must specify the error code in AC0.

To add error codes to the ERMES file, you must obtain its source version, allocate an unused code group (or add to an existing code group), and insert you own series of codes and descriptive messages. You can also create a new error message file that is structured like ERMES, but has different contents. (See the description of the ?ERMSG system call in Chapter 13 for information on the ERMES file structure.)

Program Information/Control System Calls

The ?DEBUG system call allows you to transfer control to the Debugger utility while your program is executing. By including the ?DEBUG system call in your program, you can set up predefined breakpoints for testing purposes. Another way to call the Debugger utility is to choose the ?PFDB option in offset ?PFLG of the ?PROC packet. (See the description of the ?PROC system call in Chapter 13.)

You can use the ?DEBUG system call to examine or modify inner-ring user contexts. The user debugger does not base its protection logic upon the ring-maximization protection scheme. Instead, all access is based upon the ACLs of the inner-ring segment image. (See Chapter 3 for information on the ring-maximization protection scheme.)

To examine a user ring, the caller to have Read access to the segment image file. Also, the caller must have Write access to the segment image file to permit any modification (including setting breakpoints) of the user ring. (To set breakpoints in any user ring, you must always have Write access to Ring 7.) (See Chapters 2 and 3 for information on segment image files.)

System Information

AOS/VS maintains a special accounting file, :SYSLOG, with the special file type ?FLOG. You can log messages into :SYSLOG. The ?LOGEV system call accesses this file. The ?LOGEV system call writes an event code and, optionally, a message to the log file. A process must be in Superuser mode to issue the ?LOGEV system call.

For more information about the system log file, refer to the following manuals: 'Managing AOS/VS' and the 'Command Line Interpreter (CLI) User's Manual (AOS And AOS/VS)'.

Between each major release, AOS/VS may undergo several revisions. Therefore, it is important to know your system's revision number and its memory configuration. The ?SINFO system call allows you to get this information.

The ?GSID system call lets you find out what system you are on. When your system is part of a network, it is very easy to lose track of where you are. Then, you can use the ?BNAME system call to find out whether a particular process or queue is on a local host or on a remote host. Again, this is useful if you are on a network.

Utility Interfaces

In general, the EXEC utility manages queues and magnetic tape units. Because the EXEC utility can perform many functions for you, you must issue the ?EXEC system call to tell it what to do. Specifically, the ?EXEC system call directs the EXEC utility to perform one of the following functions on behalf of a calling process:

- o Assign or deassign a logical name to a tape unit or an uninitialized disk that you want to treat as a whole unit (i.e., a non-LD disk) and issue an operator mount or dismount message.
- o Mount labeled or unlabeled magnetic tapes.
- o Dismount labeled or unlabeled magnetic tapes.
- o Place a request into a queue.
- o Hold, unhold, or cancel a queue request.
- o Provide an report on the status of the EXEC utility.

The ?EXEC system call requires a packet. Therefore, to direct the EXEC utility to perform one of these functions, you must specify in offset ?XRFNC (the first offset) what you want the EXEC utility to do. Although each function requires a unique packet, the ?XRFNC offset is common to all packets.

If your system is not running the EXEC utility, you can still queue files for spooled output by issuing the ?ENQUE system call.

The ?FEDFUNC system call is similar to the ?EXEC system call in that it provides you with a simple-to-use interface to a utility. Instead of providing an interface to the EXEC utility, however, the ?FEDFUNC system call provides an interface to the File Editor (FED) utility.

The ?FEDFUNC system call directs the FED utility to perform one of the following functions on behalf of the calling process:

- o Change the radix.
- o Open a symbol table file.
- o Evaluate a FED string.

Like the ?EXEC system call, the ?FEDFUNC system call also requires a unique packet for each function. In addition, you must define the function you want the FED utility to perform for you in the first offset, ?FRFNC, which is common to all packets.

Bias Factors

The ?GBIAS system call lets you determine your system's maximum and minimum bias factors at runtime. You can use the ?GBIAS system call with the ?SBIAS system call, which sets the bias factor values.

CLI Messages

When you use a CLI command to create a new process, the CLI sends an edited version of that CLI command to the new process in the form of an initial IPC message. The ?GTMES system call allows you to access the initial IPC message. Depending on your input specifications, you can use ?GTMES to get a specific argument in the CLI command line and to determine which switches, if any, modify it. The ?GTMES system call also returns the message that another father process sends when it creates a son with the ?PROC system call.

Symbols

The ?GTNAM system call lets you refer to the system-defined symbol table (.ST) file without knowing its contents. This means that if you do not know the symbol for a particular value, you can issue the ?GTNAM system call to search the .ST file for the symbol that is closest in value to the value you supplied in AC0. The ?GTSVL system call is similar to the ?GTNAM system call, but it allows you to refer to a particular program's user-defined .ST file without knowing its contents, instead of the system-defined .ST file.

Host Information

To find a host name or host identifier (host ID), you can issue the ?HNAME system call. Then, depending on what information you supplied as input, the ?HNAME system call returns the missing information.

Some system calls require a virtual PID as input. The ?GVPID system call translates a host ID and a PID into a virtual PID for use in these system calls. Conversely, to break down a virtual PID into its components, a host ID and a PID, you can issue the ?TPID system call.

To detect references to remote hosts in pathnames, you can issue the ?RNAME system call.

Address/Access Validation

To verify that a particular address is valid or that a caller has the proper access privilege for a particular address, you can issue the ?VALAD system call.

Sample Program

The following program, TIMEOUT, uses the ?GTMES system call to get a number from your console command line, then it delays itself for the number of seconds that you specified. The programs BOOMER (see Chapter 6) and DLIST (see Chapter 5) also contain examples of the ?GTMES system call.

```
                    .TITLE  TIMEOUT
                    .ENT    TIMEOUT
                    .NREL

;Use the ?GTMES system call to get the number of seconds that you
;typed at your console.  Then, put the ASCII value of that number of
;seconds in AC2, and put its binary value in AC1.

TIMEOUT:  ?GTMES  CLIMSG                  ;Get the number of seconds
                                          ;from the console.
          WBR     ERROR                   ;If there is an error,
                                          ;process it.

;Check to make sure that the number you typed, which is returned in
;AC1, is between 0 and 20 (decimal):

        WCLM    1,1                       ;If the number you typed is
                                          ;not in the following range:
                0                         ;lower limit of 0
                20.                       ;and upper limit of 20,
        WBR     BADVAL                    ;exit and print the "illegal
                                          ;delay" message on the console.

        NLDAI   1000.,0                   ;Put 1000. in AC0.
        WMUL    1,0                       ;Get the number of
                                          ;milliseconds in AC0.

        ?WDELAY                           ;Delay for the number of
                                          ;seconds that you specified.
        WBR     ERROR                     ;If there is an error,
                                          ;process it.

        WSUB    2,2                       ;Set for good return.
        WBR     BYE                       ;Goodbye.

ERROR:  NLDAI   ?RFEC!?RFCF!?RFER,2       ;Error flags:  Error code is
                                          ;in AC0 (?RFEC), message is
                                          ;in CLI format (?RFCF), and
                                          ;father process should handle
                                          ;this as an error (?RFER).
```

TIMEOUT Program (Cont.)

```
   BYE:     ?RETURN                             ;Return to CLI.
            WBR      ERROR                      ;Return error.

   BADVAL:  XLEFB    1,BMSG*2                   ;Set up a byte pointer to the
                                                ;"illegal delay" message.
            NLDAI    (CLIMSG-BMSG)*2!?RFCF,2      ;LEN + FLGS.
            WBR      BYE                        ;Done.  Print message and
                                                ;depart.

   BMSG:    .TXT     "Delay specified is outside legal range (0 - 20.)"

   ;?GTMES packet to get number of secs from CLI:

   CLIMSG:  .BLK     ?GTLN                      ;Allocate enough space for
                                                ;packet.

            .LOC     CLIMSG+?GREQ               ;Request type.
            .WORD    ?GARG                      ;Copy the argument specified
                                                ;in offset ?GNUM, Argument 1,
                                                ;into offset ?GRES.

            .LOC     CLIMSG+?GNUM               ;Argument number.
            .WORD    1                          ;Argument 1 is the number of
                                                ;seconds to delay (Argument 0
                                                ;is the name of the program).

            .LOC     CLIMSG+?GRES               ;Byte pointer to buffer that
                                                ;receives the results.
            .DWORD   -1                         ;No buffer is necessary.

            .LOC     CLIMSG+?GTLN               ;End of packet.
            .WORD    0                          ;Use default values for other
                                                ;offsets.

   .END     TIMEOUT                             ;End of TIMEOUT program.
```

End of Chapter

CHAPTER 12
16-BIT PROCESSES

The system calls that are unique to 16-bit processes are:

?DELAY	Suspends a 16-bit task for a specified interval.
?GCRB	Gets the base of the current resource.
?IDSTAT	Returns task status word.
?IESS	Initializes an extended state save (ESS) area.
?IHIST	Starts a histogram for a 16-bit process.
?KCALL	Keeps the calling resource and acquires a new resource.
?OVEX	Releases an overlay and returns.
?OVKIL	Exits from n overlay and kill the calling task.
?OVLOD	Loads and goes to an overlay.
?OVREL	Releases an overlay area.
?RCALL	Releases one resource and acquires a new one.
?RCHAIN	Chains to a new procedure.
?SERMSG	Returns text for associated error code.
?UNWIND	Unwinds the stack and restores the previous environment.
?WALKBACK	Returns information about previous frames in the stack.

AOS/VS allows you to execute 16-bit programs in addition to 32-bit programs. These can be programs you developed under AOS/VS or under the Advanced Operating System (AOS); in the latter case, you must relink to execute the programs under AOS/VS. In some cases, reassembling or recompiling AOS programs may also be necessary.

Memory Modification with Disk Images

Sixteen-bit programs have a more restricted address space than 32-bit programs (32K words or less). Therefore, to augment a 16-bit process's logical address space, you must call in shared or unshared overlays. There are two types of system calls for this purpose: resource system calls, which automatically load and release overlay procedures, and primitive overlay system calls.

Most 16-bit applications use the resource system calls, because they simplify resource management. These system calls let you postpone the decision to include the resources in your root program or in one or more overlay areas until link time.

The primitive overlay system calls give you greater control of overlays, but to use them, you must be willing to explicitly load, release, and control overlays.

Overlay Concepts

Overlays are useful in a small logical address space because they allow you to re-use the same portion of memory, called an overlay area, for different portions of code. In general, at link time, you define two or more overlays for each overlay area. The Link utility classifies the elements of a 16-bit program into two resource types:

- The root, which is the memory-resident portion of the program.

- Overlays.

Link reserves space in the .PR file for overlays, but diverts the actual overlay code to an overlay (.OL) file. As your program calls overlays during execution, AOS/VS reads them from the overlay areas in the .OL file to the designated overlay areas in memory.

You can define as many as 63 overlay areas per program. Each overlay area can accommodate a maximum of 511 separate overlays. An overlay area can consist of either shared or unshared overlays, but not both. Link builds shared overlay areas in multiples of 1K words and builds unshared overlays in multiples of 256 words.

Normally, the basic size of an overlay area equals the size of its largest overlay, plus any padding Link provides to fill out the area to a multiple of 1K or 256 words. As a result, AOS/VS reads only one overlay into the overlay area at that time. (See Figure 12-1.)

```
 ______________________________________________________________
|                                                              |
|                 Memory                .OL File                |
|                                    ______________  \          |
|              |\\\\\\\\\\\\\|   ____|            | |          |
|              |\\\\\\\\\\\\\|  |    | Overlay 0  | |          |
|              |_____________|  |    |            | |          |
|            / |             |  |    |------------| | Overlay  |
|   Overlay  |  |             |<---'  | Overlay 1  | | Area 0   |
|   Area     |  |             |       |------------| |          |
|            \ |-------------|       | Overlay 2  | |          |
|              |\\\\\\\\\\\\\|       |            | |          |
|              |-------------|       |------------| /          |
|                                   (Overlay area 1)            |
|                                           :                   |
|                                           :                   |
|                                                              |
|______________________________________________________________|
```

Figure 12-1. Basic Overlay Area Equals Size of Largest Overlay

You can increase an overlay area to a multiple of its basic size at link time. This results in a total overlay area that can simultaneously accommodate more than one overlay of the basic size. During execution, AOS/VS can place these overlays into any of the basic areas within the total overlay area.

Therefore, overlays destined for a multiple-overlay area must be position-independent; that is, you must write them so that all internal procedure references are relative to some point in the same overlay. Figure 12-2 shows an overlay area with a total size that is double its basic size.

As Figure 12-2 shows, doubling the basic size of the overlay area in memory allows AOS/VS to simultaneously read in two overlays of the basic size (Overlay 0 and Overlay 2). Note that AOS/VS could also fit both Overlay 1 (which is smaller than the basic size) and Overlay 2, or both Overlay 0 and Overlay 1 in the total overlay area.

Usually, you use special overlay designators to define object modules as overlays in the Link command line. Link assigns a number to each overlay area and to each of the overlays that make up that overlay area. Link bases these numbers on the order in which the overlay areas and the individual overlays appear in the Link command line.

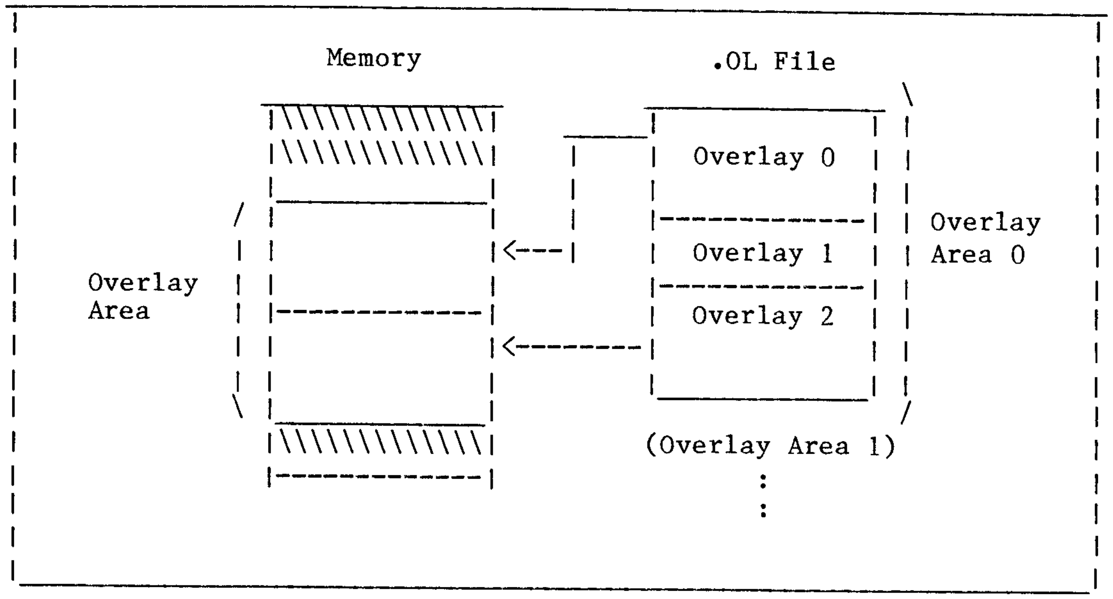

Figure 12-2. Multiple Overlay Area (total area = basic size * 2)

For example, to link six object modules, A, B, C, E, and F, to form the program file A.PR, the command line is:

```
X LINK A B !*C D!E F*!
```

The overlay designators (!*, !, and *!) define one overlay area (Overlay Area 0) with two distinct overlays: modules C and D make up Overlay 0, while modules E and F make up Overlay 1.

You can use the pseudo-ops .ENTO and .EXTN to refer symbolically to overlay areas and overlays. (For more information on .ENTO and .EXTN, refer to the "AOS/VS Macroassembler Reference Manual." Also, for more information on how Link handles overlays, refer to the "AOS/VS Link and Library File Editor (LFE) User's Manual.")

Resource System Calls

AOS/VS provides access to overlays and other procedures through a generalized procedure/system call mechanism implemented by the ?RCALL, ?KCALL, and ?RCHAIN system calls. You must define the procedure you want to call with the .PENT (procedure entry) pseudo-op. For the ?RCALL and ?KCALL system calls, the calling procedure must begin with an ?RSAVE macro instruction and end with RTN. (The ?RCHAIN caller must begin with ?RSAVE. Only the last procedure in the chain should end with a RTN.)

The ?RCALL system call releases the calling resource and then loads the new resource. Because the calling resource is released on an ?RCALL, you can load the new resource into the caller's memory area. Thus, AOS/VS preserves the state of the ?RCALL caller so that it can reload the caller, if necessary, after it executes the new procedure.

The ?KCALL system call loads a new resource and transfers control to its entry point. Unlike the ?RCALL system call, the ?KCALL system call does not release the calling resource. Use the ?KCALL system call carefully, however, because if you use it indiscriminately, you can cause a resource deadlock.

A resource deadlock occurs when every task that requires overlays is suspended waiting for overlay areas to become available. (If the overlays have issued ?KCALL system calls, AOS/VS cannot release their overlay areas.) A resource deadlock can occur if an overlay ?KCALLs another overlay to the same basic (non-multiple) overlay area. Therefore, we recommend that you use the ?RCALL system call instead of the ?KCALL system call to load procedures.

The ?RCHAIN system call releases the calling resource and acquires the new resource before it leaves the calling procedure. Typically, you use the ?RCHAIN system call to join resources that you split into small sequential pieces. Only procedures within resources that have been ?RCALLed or ?KCALLed can issue the ?RCHAIN system call.

In effect, the ?RCHAIN system call allows you to chain from an ?RCALLed or ?KCALLed procedure to a new procedure. After AOS/VS executes the new procedure, it returns control to the original procedure, not to the ?RCHAIN caller.

Link resolves each resource system call and, if necessary, binds the appropriate resource handler routines into the program file. These routines, which are part of the runtime library URT16.LB, load the called procedures as they are needed at execution time.

If a resource deadlock or error occurs while AOS/VS is executing a resource system call, it transfers control to an error-processing module with the entry ?BOMB. Unless you write your own error-handling routine with a ?BOMB entry, AOS/VS uses the default routine in URT16.LB. The default routine terminates the calling process and passes the appropriate error code to the caller's father.

If you use your own ?BOMB routine, AOS/VS transfers control to that routine, and supplies the following error-handling information:

- o An error code in AC0.
- o The procedure descriptor entry in AC1.
- o The fault address on the stack.

Procedure Entries

Usually, you pass procedure entries as arguments to the resource system calls; for example, ?RCALL procedure entry. As an alternative, you can pass procedure entry descriptors on the stack. AOS/VS then pops the procedure entry descriptor off the stack before you execute the resource system call.

The .PTARG pseudo-op translates the name of a procedure to a procedure entry descriptor. Figure 12-3 shows a sample ?RCALL sequence that uses the descriptor method.

```
NAME:    .PTARG   FIRST            ;Define a procedure entry
              .                    ;descriptor for FIRST.
              .
              .
         LDA      0,NAME           ;Load AC0 with the descriptor.
         PSH      0,0              ;Push the descriptor onto the
                                   ;stack.
         LDA      0,ARG1           ;Pass ARG1,
         LDA      1,ARG2           ;ARG2, and
         LDA      2,ARG3           ;ARG3 to FIRST.
         ?RCALL                    ;Pop procedure entry descriptor
                                   ;off the stack, release calling
                                   ;resource, and acquire resource
                                   ;that contains FIRST.
```

Figure 12-3. Passing a Procedure Entry Descriptor via the Stack

Alternate Return from Resources

After AOS/VS executes a new procedure, it returns control to the word that immediately follows the resource system call. To return control to the second word after the resource system call, issue the following instruction sequence from the calling procedure:

```
ISZ ?ORTN,3
RTN
```

The first statement (ISZ ?ORTN,3) increments the caller's return address. The RTN instruction returns to the incremented address.

System Management of Resource System Calls

The ?RCALL, ?KCALL, and ?RCHAIN system calls require two extra words on the user stack. These words must be located between the caller's return block and the called procedure's temporary variables. Do not alter these words, because AOS/VS uses them to store information from the called procedure.

The ?RSAVE macro instruction reserves these two extra words. Therefore, all procedures that issue ?RCALL or ?KCALL system calls must begin with an ?RSAVE instruction and end with a RTN instruction. Every procedure that the ?RCHAIN system call acquired must also begin with an ?RSAVE instruction, but only the last procedure in the chain should end with RTN.

Figure 12-4 shows the contents of the stack after the execution of an ?RSAVE instruction. Also, Figure 12-4 lists the parametric names of the stack locations. Note that these parameters apply to the stack for 16-bit programs (standard ECLIPSE stack), not the stack for 32-bit programs (the MV/8000 stack). The user parameter file, PARU16, defines these and the other 16-bit parameters. For more information on stacks, refer to the 'Principles of Operation 32-Bit ECLIPSE Systems' manual.

```
                -----------------------------------  \
       ?OAC0   |           Caller's AC0            | |
               |-----------------------------------| |
       ?OAC1   |           Caller's AC1            | |
               |-----------------------------------| | Caller's
       ?OAC2   |           Caller's AC2            | > return
               |-----------------------------------| | block
       ?OFP    |      Caller's frame pointer       | |
               |-----------------------------------| |
 (Also ?ORTN   | Old Carry       |       Old PC    | |
 New FP)----->|                 |(program counter) /
               |-----------------|-----------------| \
       ?VRTN   |        First reserved word        | | Virtual
               |-----------------------------------| > return
       ?DESC   |        Second reserved word       | | information
               |-----------------------------------| /
       ?TMP    |              Temp 1               | \
               |-----------------------------------| |
               |              Temp 2               | | Called
               |-----------------------------------| > procedure's
               |              Temp 3               | | temporary
               |-----------------------------------| | area
               |              Temp 4               | |
               |-----------------------------------| /
 New SP------>|                                   |
               |___________________________________|

       KEY:  ?ORTN = Caller's carry and return address
             FP    = Stack frame pointer
             SP    = Stack pointer
```

Figure 12-4. Resource System Call Stack after ?RSAVE System Call

Runtime Relocatability Requirements

If they are part of multiple overlay areas, the overlays you call with the ?RCALL, ?KCALL, and ?RCHAIN system calls must be position-independent, because AOS/VS may reload them into different portions of the overlay area after it executes the resource system call.

Moreover, overlays within multiple areas must be runtime relocatable to issue ?RCALL system calls. This means that you cannot issue any assembly language reference to a fixed address before the ?RCALL system call, because the address could be invalid when the calling overlay is reloaded. The ECLIPSE instructions subject to this restriction are JSR, EJSR, LEF, ELEF, PSHJ, POPJ, XOP, and PSHR. Figure 12-5 shows a JSR instruction whose return value will be invalid if the procedure that issues the ?RCALL system call is reloaded into a different memory area.

```
                .
                .
                .
     C:       JSR A            ;Jump to Subroutine A.
                .              ;Return address is C+1.
                .
                .
     A:       ?RSAVE           ;Save the return address of C.
                .
                .
                .
              ?RCALL B         ;Release C, acquire B, and go to
                .              ;target procedure in B.
                .
                .
              RTN              ;Return to C+1.
```

Figure 12-5. Invalid Return Address from ?RCALL System Call

The return address from the ?RCALL system call in Figure 12-5 is "C+1", which is the first word that follows the JSR A instruction. The return address will be invalid, however, if AOS/VS relocated procedure A after it executed procedure B. If procedure A issued a ?KCALL system call to B instead, the return would be valid, because the ?KCALL system call keeps the calling resource (A) before it acquires the new resource and transfers control to its correct entry point (B).

Primitive Overlay System Calls

As an alternative to the resource system calls, you can use the primitive overlay system calls, ?OVLOD, ?OVREL, ?OVEX, and ?OVKIL, to call and release overlays. These system calls give you greater control over the overlay environment, but require you to explicitly load and release the overlays. Because the resource system calls manage overlays automatically, you should use them rather than using the primitive overlay system calls.

To use the primitive overlay system calls, you must define each overlay with the .ENTO (overlay entry) pseudo-op. The system maintains an overlay use count (OUC) for every memory-resident overlay. The OUC specifies the number of tasks currently using the overlay. When the OUC value reaches 0, the overlay area is freed for use by another overlay.

As long as any task is using an overlay (that is, OUC is not 0), no other overlay can be loaded into the same basic overlay area. This is true even if another high-priority task issues an overlay load request in the meantime. If the overlay area is a multiple of its basic size, however, another task can use any free basic area in the total area.

The ?OVLOD system call loads an overlay and passes control either to the beginning of that overlay or to some offset within it. In addition, your input to the ?OVLOD system call determines whether the loading is conditional or unconditional.

If you specify unconditional loading, AOS/VS loads the overlay that you request, even if it is already resident. If you specify conditional loading, AOS/VS first checks whether the target overlay is already in the overlay area. If it is, AOS/VS does not load the overlay, but simply increments the overlay's OUC. If the overlay is not resident, AOS/VS loads it into the overlay area and sets its OUC to 1.

To release an overlay that was loaded with the ?OVLOD system call, you must use one of the following release system calls:

o ?OVREL

  The ?OVREL system call decrements the overlay's OUC and frees the overlay area if the OUC equals 0. Note that you cannot issue ?OVREL from the overlay you want to release. Instead, issue ?OVREL from some point outside that overlay.

- ?OVEX

  The ?OVEX system call decrements the overlay's OUC, frees the overlay area if the OUC equals 0 and transfers control to a specific nonoverlay address. You can use the ?OVEX system call to return from a subroutine within an overlay.

- ?OVKIL

  The ?OVKIL system call decrements the overlay's OUC, releases the overlay area if the OUC equals 0 and kills the calling task.

Extended State Save Area

AOS/VS allows each 16-bit process to set up an extended state save area (ESS) for each task in the unshared portion of the logical address space. The ESS area holds task-specific information, such as the value of the program counter and its carry bit, and the current contents of the accumulators. However, you can use the ESS area to store any information you feel is relevant to a task.

Before you can use an ESS area, you must initialize it with the ?IESS system call. Input to the ?IESS system call includes the starting address of the ESS (in the unshared area of your logical address space), and a pointer to a block of page zero locations in your logical address space. When AOS/VS schedules a 16-bit task, it copies the ESS information to the designated page zero area. When rescheduling occurs, AOS/VS transfers the ESS information back to the ESS block in the unshared area of your logical address space.

End of Chapter

INDEX

A

Aborting process and generating console interrupt with CTRL-C CTRL-B, 5-21
Aborting ?TASK while ?UTSK task-initiation routine is executing, 6-6 (See also, "Tasks".)
Accepting next character as literal with CTRL-P, 5-20 (See also, "Control characters".)
Access,
- Controls on file, 3-9
- Coordinating, to common resource, 2-7
- Disabling device with ?DDIS, 13-65f
- Enabling device with ?DEBL, 13-69f
- Permitting to protected shared file (See "?PMTPF system call".)
- Privileges (See "Access privileges".)
- Shared (See "Shared access".)

Access control list (ACL), 2-7, 3-12, 4-15ff
- Changing default with ?DACL, 4-15, 13-59f
- Defining with ?CREATE, 4-15
- Definition of, 4-15
- Examining default with ?DACL, 4-15, 13-59f
- Format, 13-60
- Getting for file entry with ?GACL, 13-149f
- Getting for specific file and username, 13-205f
- Getting with ?GACL, 4-15
- Setting default with ?DACL, 4-15, 13-59f
- Setting for files or directories with ?SACL, 4-15, 13-507f
- Specifications, 13-43
  - Format, 13-508
- Templates, 4-16
  - Asterisk (*), 4-16
  - Plus sign (+), 4-16
  - Minus sign (-), 4-16

Access control privileges, getting with ?GTACP, 13-205f
Access control specifications (See "Access control list (ACL)".)
Access field (ANSI-standard labeled magnetic tapes), 5-25 (See also, "Labeled magnetic tape".)

Access privileges, 2-8, 3-7, 3-12, 4-13f (See also, "File access".)
    File, 2-8, 3-7, 3-12
    Append (?FACA), 4-13ff, 13-60, 13-206, 13-508
    Execute (?FACE), 4-13ff, 13-60, 13-79, 13-206, 13-508
    Owner (?FACO), 4-13ff, 4-17, 13-60, 13-306, 13-508f
    Read (?FACR), 4-13ff, 13-60, 13-79, 13-138, 13-149, 13-206, 13-508
    Verifying caller's, 11-8
    Write (?FACW), 3-19, 4-13ff, 5-5, 13-60, 13-90, 13-149, 13-206, 13-508
Accessing
    all devices, 10-9f (See also, "User device support".)
    directories, 4-7f
    files, 4-13f (See also, "File access" and "Access privileges".)
    files outside current working directory with ?DIR, 13-79f
    initial IPC messages with ?GTMES, 11-7
    protected files, 13-389ff
    user devices, 3-10 (See also, "Devices".)
Accounting file, :SYSLOG, 11-5
ACK0 (positive acknowledgment), 9-7f, 13-539ff, 13-590, 13-607f (See also, "Data-link control characters (DLCC)".)
ACK1 (positive acknowledgment), 13-539ff, 13-590, 13-607f (See also, "Data-link control characters (DLCC)".)
Acknowledgments,
    Negative (See "Negative acknowledgments".)
    Positive (See "Positive acknowledgments".)
ACL (access control list) (See "Access control list (ACL)".)
Acquiring resource and releasing old one with ?RCALL, 13-441f
Address, validating logical with ?VALAD, 13-669f
Address, verifying validity of with ?VALAD, 11-8
Address of file elements, 4-2
Address range, monitoring in calling or other process, 13-679ff
Address space,
    Logical (See "Logical address space".)
    Virtual, 2-4, 3-3f
        Illustration of, 2-5
Advantages of multitasking, 6-3f
?ALLOCATE system call, 5-1, 5-7, 13-7f
Allocating
    blocks for specific data elements, 5-7 (See also, "Disk blocks".)
    disk blocks, 5-1, 13-7f (See also, "Disk blocks".)
    stack space, 6-7f (See also, "Stacks".)
Allocation, controlling disk space, 4-19
ALM (Asynchronous Line Multiplexor), 5-10 (See also, "Devices".)
Alternate return from resource system calls, 12-7
ANSI-standard
    format, 5-22, 5-28f, 5-31f
    terminals, 13-158 (See also, "Consoles".)

AOS operating system,
  Format labeled magnetic tapes, 5-24
  Program files (file type ?FPRG), 4-5ff, 5-5 (See also, "Program files".)
AOS-format internal time, getting, 13-295
AOS/VS operating system,
  Establishing interface between unsupported device and (See "?IDEF system call".)
  File structure, 4-1
  Program files (file type ?FPRV), 4-5ff, 5-5 (See also, "Program files".)
  Task-protection model, 6-4f (See also, "Tasks".)
?APND mask (in offset ?ISTI), 5-6
Append (?FACA) access, 4-13ff, 13-60, 13-206, 13-508
Append option, 3-19, 5-6
Array, external gate, 3-20
Array structure for 16-bit processes (See "?IHIST system call".)
ASCII character set, 5-23
ASCII code, 5-5, 5-9, 5-17, 13-542, 13-537f
Assembly language instructions,
  DIA, 5-8
  DIB, 5-8
  DIC, 5-8
Assembly language source files, 4-9 (See also, "Files".)
?ASSIGN system call, 3-67, 5-1, 5-18, 13-9f
Assigning
  device to process for record I/O, 5-1, 13-9f (See also, "Record input/output (I/O)".)
  son higher priority, 3-10 (See also, "Processes".)
  Superprocess privilege, 3-13 (See also, "Processes".)
Assignment, breaking file's channel, 13-160 (See also, "Consoles".)
Asterisk (*) template, 4-16 (See also, "Access control list (ACL)".)
Asynchronous Line Multiplexor (ALM), 5-10 (See also, "Devices".)
At sign (@) pathname prefix, 4-10, 5-9, 5-11 (See also, "Pathnames".)
Attribute, permanent (See "Permanent attribute".)
Attributes transferred to new program by ?CHAIN, 3-19
Auto-answer modems, 5-15f (See also, "Modems (full-duplex)".)
  Operating sequence, 5-15f
Auto-restart/power-failure routine, 10-2, 10-10f

B

/B specifications string switch, 13-94
Bad blocks, 5-8
Base, getting current resource, 13-163f

BASIC, 6-3
Basic overlay area, illustration of, 12-3 (See also, "Sixteen-bit processes".)
Batch process information (See "?LOGEV system call".)
BCC (block check character), 9-7, 13-538f, 13-605 (See also, "Data-link control characters (DLCC)".)
Becoming customer of specified process, 13-33ff
Becoming server with ?SERVE, 13-555f (See also, "Connection-management facility".)
Beginning control sequence with CTRL-C, 5-20 (See also, "Control characters".)
Bias factors, 3-5, 11-7
    Getting current values for, 13-151f
    Setting with ?SBIAS, 13-511f
Binary mode, 5-18f
Binary synchronous communications (BSC), Chapter 9
    Concepts, 9-2
    Definition of terms, 9-1f
        Dedicated communications line, 9-1
        Station, 9-1
        Switched communications line, 9-2
    Disabling line with ?SDBL, 9-2, 13-523
    Enabling line with ?SEBL, 9-2, 13-536ff
    Error-recovery procedures, 9-2, 9-11ff, 13-591, 13-607f
    Error-recovery statistics, 9-13
    Getting error statistics with ?SGES, 13-557ff
    Illustration of point-to-point/multipoint line configuration, 9-4
    Implementation, 9-13ff
        Illustration of, 9-14ff
    Line, 13-357
    Line configurations, 9-3ff
        Multipoint, 9-3ff
        Point-to-point, 9-3ff
    Protocol, 9-1 9-6ff, 13-538, 13-590
    Receiving data or control sequences over lines with ?SRCV, 9-2 13-585ff
    Sending data over enabled line with ?SSND, 9-2
    Sending text over line, 9-7
    System calls, 9-1 (See individual system call entries for additional references.)
        ?SDBL, 9-1
        ?SDPOL, 9-1
        ?SDRT, 9-1
        ?SEBL, 9-1
        ?SERT, 9-1
        ?SGES, 9-1
        ?SRCV, 9-1
        ?SSND, 9-1

Binding pages to working set (See "?WIRE system call".)
Bit masks, Returned on ?SRCV system calls, 13-593
Bit masks for ACL specifications, 4-15 (See also, "Access control list (ACL)".)
- ?FACA, 4-15
- ?FACE, 4-15
- ?FACO, 4-15
- ?FACR, 4-15
- ?FACW, 4-15
- Combining (See "?CREATE system call", "?DACL system call", or "?SACL system call".)

Bits,
- Flag, 8-6
- Privilege (?PROC system call), 13-419
- Severity, 13-488f

?BLKPR system call, 3-1, 3-6, 3-12, 3-14f, 6-12, 13-619, 13-659f (See also, Blocking Processes".)
Block, time (See "?CREATE system call, Time block".)
Block check (CRC) error, 9-12 (See also, "Binary synchronous communications (BSC), Error-recovery procedures".)
Block check character (BCC), 13-538f, 13-605 (See also, "Data-link control characters (DLCC)".)
Block count, 5-6
Block input/output (I/O), 5-2f, 5-6f, 13-357 (See also, "File input/output (I/O)".)
- Closing file previously opened for, 13-159f
- Definition of, 5-6
- Differences between physical block I/O and, 5-7f
- Opening file for, 13-185ff
- Packet, 13-220
- Performing on magnetic tapes, 13-447ff
- Performing on MCAs, 13-449f
- Physical (See "Physical block input/output (I/O)".)
- Reading/writing, 13-445ff
- System calls, 5-3, 5-38

Blocking processes, 3-6, 3-14f (See also, "?BLKPR system call".)
- Definition of blocked process, 6-12
- Voluntarily, 3-14
- When it occurs, 3-14f

Blocks, releasing disk, 13-478 (See also, "Disk blocks".)
?BNAME system call, 11-5, 13-13f
?BOMB routine, 12-5f, 13-163
Bootstraps, disk, 4-4
BRAN program, 13-346

Break files, 3-17ff
    Contents, 3-18
    Creating after terminating process with ?BRKFL (See "?BRKFL system call".)
    Creating for every process trap, 3-18
    Creating for specified user ring, 3-19
    Default pathname of, 3-19
    Enabling, 13-89ff (See also, "?ENBRK system call" and "?MDUMP system call".)
    Examining, 3-17
    Terminating processes and creating, 3-18f
    Words copied to, 3-18
Breaking
    connection in inner ring, 13-85f
    connection in Ring 7, 13-63f
    customer/server connection, 13-53f (See also, "Connection-management facility".)
    file's channel assignment, 13-160 (See also, "Channels".)
?BRKFL system call, 3-1, 3-13, 13-15ff, 13-19, 13-54, 13-330, 13-346 13-485, 13-619 (See also, "Break files".)
Broadcast option, 13-693, 13-696
Broadcasting messages with ?XMT and ?XMTW, 6-18
BSC (binary synchronous communications) line (See "Binary synchronous communications (BSC)".)
Buffer, emptying type-ahead and echoing ^C^C on console with CTRL-C CTRL-C, 5-21 (See also, "Control sequences".)
Burst multiplexor (BMC) I/O ?IDEF option, 10-3f
Burst multiplexor channel (BMC) ?IDEF option, 13-235ff (See also, "?IDEF system call".)
Bypassing retries for disk errors, 5-7
Bytes,
    Moving from customer's buffer, 8-4, 13-337ff
    Moving to customer's buffer, 8-4, 13-341ff

C

?C8BT (bit control) characteristic, 13-154, 13-157, 13-515
?CACP offset, 13-41ff, 13-47f
Calendar, system (See "System calendar".)
Calling Debugger utility with ?DEBUG, 13-71

Calling process,
    Getting full process name of, 3-7 (See also, "?PNAME system call".)
    Getting PID of with ?PNAME, 3-7 (See also, "?PNAME system call".)
    Setting search list for calling process with ?SLIST, 13-569f
    Terminating and passing termination message to father with ?RETURN, 13-487ff
Calling task,
    Changing priority of, 13-403f
    Getting priority and TID of, 13-351
    Killing, 13-307f
    Suspending with ?SUS, 13-621
Calls,
    Receive continue (See "Receive continue calls".)
    Receive initial (See "Receive initial calls".)
    Send continue (See "Send continue calls".)
    Send initial (See "Send initial calls".)
    System (See Chapter 13.)
Canceling, holding, unholding queue requests, 13-117ff
Card readers, 5-10, 5-17f (See also, "Character devices".)
    (file type ???), 4-7
Carriage control, file type of FORTRAN, 4-7
Causes of process trapping, 3-17 (See also, "Processes".)
?CCPS offset, 13-47f
CD modem flag, 5-15, 5-17
?CDAY system call, 13-127
?CDEH offset, 13-47
?CDEL offset, 13-47f
?CDT0 (device type) characteristic, 13-155, 13-157, 13-516
?CDT2 characteristic, 13-157
?CEB0 (echo-mode control) characteristic, 13-155f, 13-515
?CEOL (carriage-return/line-feed control) characteristic, 13-154, 13-515
?CESC (interrupt control) characteristic, 13-155, 13-516
?CFF (form-feed control) characteristic, 13-155, 13-515
?CFKT (delimiter control) characteristic, 13-155, 13-516
?CFTYP offset 4-19ff, 13-41ff, 13-138
?CGNAM system call, 4-1, 4-10
Chain, LRU (least recently used), 13-501
?CHAIN system call, 3-1, 11-3, 13-23ff, 13-338, 13-353
    Attributes transferred to new program, 3-19
    Linking programs together with, 3-19f
Chaining customer processes, 8-7

Chaining to new procedure with ?RCHAIN, 13-443f
Chains,
  Free memory, 2-4
  LRU (least recently used), 2-6, 13-501
Changing
  calling task priority with ?PRI, 13-403f
  number of unshared memory pages (See "?MEMI system call" and see also, "Pages".)
  number of unshared pages in logical address space, 13-349f
  priority of tasks specified by TIDs with ?IDPRI, 13-245f
  process priority, 3-10
  process priority with ?PRIPR, 13-15, 13-405ff (See also, "?PRIPR system call".)
  process type, 13-57f
  process type with ?CTYPE, 3-10 (See also, "?CTYPE system call".)
  radix using FED utility (See "?FEDFUNC system call".)
  state of another process with Superprocess mode, 3-12 (See also, "Processes".)
  working directory, 4-8 (See also, "Working directory".)
  working directory with ?DIR, 13-79f
Channel assignment, breaking file's, 13-160 (See also, "Channels".)
Channel numbers (See "Channels".)
Channels, 5-3, 9-2, 13-186
  Breaking file's channel assignment, 13-160
  Closing, 13-29ff
  Data (See "Data channels".)
  Definition of, 5-3
  Disassociating channel number from file, 5-3
  Numbers, 4-10, 9-2
  Opening, 13-357ff
Character devices, 4-2, 5-13ff
  Assignment of, 5-18
  Card readers, 5-17f
  Characteristics of, 5-13f
    Defining, 5-14
    Getting, 5-14
    Overriding, 5-14
    Consoles, 4-2
  Characteristics words, 13-515f (See also, "Characteristics words".)
  Deassigning with ?DEASSIGN, 13-67f
  Definition of, 5-13
  Extended characteristics of, 5-14
    ?XIFC, 5-14
    ?XOFC, 5-14
  Getting extended characteristics of, 13-167ff
  Reading device characteristics, 13-153ff
  Setting extended characteristics, 13-545ff
  Setting characteristics with ?SCHR, 13-513ff
  Text mode, 5-13

Character I/O, managing with PMGR, 3-8 (See also, "Input/output (I/O)".)
Character sets,
ASCII, 5-23
EBCDIC, 5-23f, 13-537f, 13-542
Characteristics,
Character device (See "Character devices".)
Extended character device (See "Character devices".)
Setting character device's, 13-513ff
Characteristics words (See also, "Character devices".)
?C8BT (bit control), 13-515
?CDTO (device type), 13-516
?CEBO (echo-mode control), 13-515
?CEOL (carriage-return/line-feed control), 13-515
?CESC (interrupt control), 13-516
?CFF (form-feed control), 13-515
?CFKT (delimiter control), 13-516
?CLT (blanks control), 13-515
?CMOD (modem control), 5-16, 13-516
?CMRI (monitor ring indicator), 5-16, 13-516
?CNAS (non-ANSI-standard device), 13-515
?CNNL, 5-18, 13-157
?CNRM (message control), 13-516
?COTT (escape-key control), 13-515
?CPBN (column control), 13-516
?CPM (page mode), 13-516
?CRAC (rubout control), 13-515
?CRAF (rubout control), 13-515
?CRAT (rubout control), 13-515
?CSFF (form-feed simulation), 13-515
?CST (tab simulation), 13-515
?CTO (device time-out), 13-516
?CTSP (blanks control), 5-18, 13-516
?CUCO (case control), 13-515
?CULC (case control), 13-516
?CWRP (wraparound control), 13-516
Characters,
Accepting as literal with CTRL-P, 5-20 (See also, "Control characters".)
Console control, 5-13 (See also, "Control characters".)
Control (See "Control characters".)
Data-link control, 9-6ff (See also, "Data-link control characters (DLCC)".)
Prefix, 13-76
Valid filename, 3-6, 4-8
Checking process creation parameters, steps AOS/VS takes, 3-11 (See also, "Processes".)
Checking volume ID of labeled magnetic tape, 13-27f (See also, "Labeled magnetic tape".)

Checkpointing shared memory pages, 2-8f (See also, "Pages".)
?CHFS offset, 13-44f
Circumstances under which AOS/VS reschedules tasks, 6-10 (See also, "Tasks".)
?CKVOL system call, 13-27f
Clearing, setting, or examining default ACL with ?DACL, 4-15, 13-59 (See also, "Access control list (ACL)".)
Clearing, setting, or examining execute-protection status, 13-123ff (See also, "?EXPO system call".)
CLI, 4-2, 5-14
    CODE macro, 13-99
    DISMOUNT command, 5-35
    DUMP command, 5-36
        Syntax, 5-36
    FILESTATUS command, 13-183
    Format, 13-488
    /FORMS switch, 5-19
    Getting CLI messages, 13-207ff
    INITIALIZE command, 4-17
    LABEL utility, 5-22, 13-372
    Macro files, 4-9 (See also, "Files".)
    Messages, 11-7
    MOUNT command, 5-38
        Syntax, 5-35
.CLI files, 4-9 (See also, "Files".)
CLI forms control utility (FCU), 13-453
CLI INITIALIZE command, 13-269
CLI LABEL utility, 13-319
Clock,
    Real-time, 13-295, 13-677
    System (See "System clock".)
Clock frequency, getting system, 13-171f
Clock/calendar system calls, 11-3
    ?CTOD, 11-4
    ?FDAY, 11-4
    ?FTOD, 11-4
    ?GDAY, 11-4
    ?GHRZ, 11-3
    ?GTOD, 11-4
    ?ITIME, 11-3
    ?SDAY, 11-4
    ?STOD, 11-4
?CLOSE system call, 5-1, 5-4, 5-18, 13-29ff, 13-67, 13-144, 13-160, 13-358, 13-455, 13-477
    Extended packet, 13-30
    Packet structure, 13-30

Closing
    file opened for block I/O, 13-159f
    files previously opened for shared access with ?SCLOSE, 13-519f
    open channels, 13-29ff
    shared file (See "?SCLOSE system call" and see also, "Files".)
?CLT (blanks control) characteristic, 13-154, 13-515
?CLTH offset, 13-41, 13-44, 13-46
?CMIL offset, 13-44ff, 13-47f, 13-138
?CMOD (modem control) characteristic, 5-16, 13-155, 13-157, 13-514, 13-516
?CMRI (monitor ring indicator) characteristic, 5-16, 13-155, 13-157, 13-514, 13-516
?CMRS offset, 13-44ff
?CMSH offset, 13-44f
?CNAS (non-ANSI-standard device) characteristic, 13-154, 13-154, 13-158, 13-515
?CNNL characteristic, 5-18, 13-157
?CNRM (message control) characteristic, 13-155, 13-515
?COBIT bit, 8-5
Code,
    16-bit process termination, 7-12 (See also, "Interprocess communications (IPC) facility".)
    ASCII, 5-5, 5-9, 5-17
    Error (See "Error codes".)
    Protected, 3-17
    Re-entrant 2-6 (See "Re-entrant code or data".)
    Source, 4-9, 4-15
    Termination (in offset ?IUFL for ?IREC and ?ISEND headers), 7-11 (See also, "Interprocess communications (IPC) facility".)
CODE CLI macro, 13-99
Colon (:) pathname prefix, 4-10 (See also, "Pathnames".)
Combining bit masks for ACL specifications (See "?CREATE system call", "?DACL system call", or "?SACL system call".)
Command, CLI INITIALIZE, 4-17, 13-269
Common local servers, using to pend/unpend tasks (See "Fast interprocess synchronization".)
Common resource, Coordinating access to, 2-7
Commonly used device characteristics, 13-156f
Communicating
    across data channel, 5-12f
    between console and task, 6-17 (See also, "Tasks".)
    between tasks, 6-17f (See also, "Tasks".)
    from interrupt service routine, 10-9 (See also, "User device support".)
    with customer via IPC system calls, 8-4 (See also, "Connection-management facility" and "Binary synchronous communications (BSC)".)

Communications device, using the IPC facility as, 5-21f (See also, "Interprocess communications (IPC) facility".)
Communications facility, intertask, 6-17f (See also, "Tasks".)
Communications lines,
    Dedicated, 9-1
    File type of synchronous, 4-7
    Switched, 9-1
Communications paths, full-duplex, 7-2 (See also, "Modems".)
Communications unit, file type of multiprocessor, 4-7
Complete pathname, getting with ?GNAME, 13-179f
?CON system call, 8-1ff, 8-6, 13-33ff, 13-485, 13-555
Concepts,
    Binary synchronous communications (BSC), 9-2 (See also, "Binary synchronous communications (BSC)".)
    File input/output (I/O), 5-2ff (See also, "File input/output (I/O)".)
    Overlays, 12-2ff
    Tasks and multitasking, 6-3 (See also, "Tasks".)
Conditions,
    Page-fault, 3-3
    Race, 6-19, 8-6
Conditions under which AOS/VS blocks processes, 3-14f
Conditions under which AOS/VS unblocks processes, 3-15
Configuration, illustration of model customer/server, 8-2 (See also, "Connection-management facility".)
Configurations, line, 9-3ff (See also, Binary synchronous communications (BSC)".)
@CONn, 5-10 (See also, "Devices".)
Connecting two or more stations (See "Dedicated communications line".)
Connection (See also, "Connection-management facility".)
    Breaking customer/server, 13-53f
    Breaking inner-ring, 13-85f
    Breaking Ring 7, 13-63f
    Creating, 8-2ff
    Establishing between customer and existing server, 8-2
    Passing to another server in any ring with ?PRCNX, 8-4, 13-395f
    Passing to another server in Ring 7 with ?PCNX, 8-4, 13-387f
    Terminating, 8-5
Connection table, 8-2, 8-4f
    Clearing entry from, 8-5

Connection-management facility, 6-4, Chapter 8
- Chaining customer processes, 8-7
- Creating connections, 8-2ff
- Description of, 8-1
- Double connections, 8-3
  - Illustration of, 8-3
- Identifying connections in inner rings, 8-6
- Inner-ring connection management, 8-6ff
- Managing exchanges between customers and servers, 8-2
- Model customer/server configuration, Illustration of, 8-2
- Moving bytes to/from customer's logical address space with ?MBTC or ?MBFC, 8-4
- Multilevel connections, 8-2
  - Illustration of, 8-3
- Obituary messages, 8-5f
- Passing customer/server connection to another server in Ring 7 with ?PCNX, 8-4
- Passing customer/server connection to another server with ?PRCNX, 8-4
- Server process, 8-4
- Server-only system call (?CTERM), 8-5
- Signaling server resignation with ?RESIGN, 8-4
- Status of inner-ring connections, 8-6
- System calls, 8-1 (See individual system call entries for additional references.)
  - ?CON, 8-1
  - ?CTERM, 8-1
  - ?DCON, 8-1
  - ?DRCON, 8-1
  - ?MBFC, 8-1
  - ?MBTC, 8-1
  - ?PCNX, 8-1
  - ?PRCNX, 8-1
  - ?RESIGN, 8-1
  - ?SERVE, 8-1
  - ?SIGNL, 8-1
  - ?SIGWT, 8-1
  - ?VCUST, 8-1
  - ?VRCUST, 8-1
  - ?WTSIG, 8-1
- Terminating connections, 8-5
- Terminating customer processes with ?CTERM, 8-4

Console (file type), 4-7
Console control characters (See "Control characters".)
Console control sequences (See "Control sequences".)
Console format control, 5-19
@CONSOLE generic filename, 5-11f (See also, "Generic files".)
Console input line, erasing with CTRL-U, 5-20 (See also, "Control characters".)
Console interrupts, 11-2f
  Controlling on virtual consoles, 11-3
  Defining task to handle, 13-271f
  Disabling with ?ODIS, 13-353f
  Enabling, 13-335
  Generating and aborting process with CTRL-C CTRL-B, 5-21 (See also, "Control sequences".)
  Generating with CTRL-C CTRL-A, 5-21 (See also, "Control sequences".)
  Waiting for, 13-315f
Console output,
  Emptying type-ahead buffer and echoing ^C^C with CTRL-C CTRL-C, 5-21 (See also, "Control sequences".)
  Freezing with CTRL-S, 5-20 (See also, "Control characters".)
  Suppressing with CTRL-O, 5-20 (See also, "Control characters".)
Console port number, getting with ?GCPN, 13-161f
Console-to-task communication, 6-17 (See also, "Tasks".)
Consoles,
  ANSI-standard, 13-158
  CRT display, 5-10, 5-13
  Non-ANSI-standard, 13-158
  Relative, 9-6
  relative (See "Relative consoles".)
  Sending messages to, 13-549ff
  Virtual, 11-3, 13-309
Contention,
  Memory, 3-15
  Definition of line, 9-3
Contents,
  Break file, 3-18
  Getting link entry, 13-173f
  Getting search list, 13-175f
  IPC send and receive headers, 7-7
  Map definition table entry, 13-237
  Parameter packets (See "Packet contents".)
  System flag word (offset ?ISFL), 7-8
  VOL1 volume labels, 5-28
Context,
  Logical, 2-1f
  Memory, Illustration of, 2-10
  User, 2-9ff

Context-management system calls, 2-10 (See individual system call entries for additional references.)
    ?ESFF, 2-10
    ?GSHPT, 2-10
    ?MEM, 2-10
    ?MEMI, 2-10
    ?PMTPF, 2-10
    ?RPAGE, 2-10
    ?SCLOSE, 2-10
    ?SOPEN, 2-10
    ?SOPPF, 2-10
    ?SPAGE, 2-11
    ?SSHPT, 2-11
Contiguous disk blocks, definition of, 4-2
Control,
    Console format, 5-19
    Disk space allocation, 4-19f
    File access, 3-9
    File type of FORTRAN carriage, 4-7
    Line-printer format, 5-19
    Passing to new process (See "?CHAIN system call".)
    Working set, 3-3
Control characters, 5-19f
    Accepting next character as literal with CTRL-P, 5-20
    Beginning control sequence with CTRL-C, 5-20 (See also, "Control sequences".)
    CTRL-C, 5-20
    CTRL-D, 5-20
    CTRL-O, 5-20
    CTRL-P, 5-19f
    CTRL-Q, 5-13, 5-20
    CTRL-S, 5-20
    CTRL-T, 5-20
    CTRL-U, 5-20
    CTRL-V, 5-20
    Data-link (See "Data-link control characters (DLCC)".)
    Definition of, 5-19
    Disabling CTRL-S with CTRL-Q, 5-20
    Erasing current console input line with CTRL-U, 5-20
    Freezing console output with CTRL-S, 5-20
    Function, 5-20
    Suppressing console output with CTRL-O, 5-20
    Terminating current read with end-of-file using CTRL-D,5-20
Control list, access (See "Access control list (ACL)".)
Control point directories (CPDs), 4-19f, 5-7
    Current space (CS), 4-19f
    File type ?FCPD, 4-6, 5-5, 13-43
    Illustration of, 4-20
    Maximum space (MS), 4-19f
    Setting maximum size of, 13-37f

Control privileges, getting access, 13-205f
Control sequences, 5-19, 5-21
    Beginning with CTRL-C, 5-20 (See also, "Control characters".
    CTRL-C CTRL-A, 13-242, 13-271, 13-353, 13-355
    CTRL-C CTRL-B, 7-11, 7-13, 11-3
    CTRL-C CTRL-D through CTRL-C CTRL-Z (reserved), 5-21
    CTRL-C CTRL-E, 11-3, 13-90
    Definition of, 5-20
    Echoing ^C^C on console with CTRL-C CTRL-C, 5-21
    Generating console interrupt and aborting process with CTRL-C CTRL-B, 5-21
    Generating console interrupt with CTRL-C CTRL-A, 5-21
    Receiving over BSC lines with ?SRCV, 13-585ff (See also, "Binary synchronous communications (BSC)".)
    Sending over BSC lines with ?SSND, 13-601ff (See also, "Binary synchronous communications (BSC)".)
Control station, 9-4f (See also, "Binary synchronous communications (BSC)".)
Controller status words for ?PRDB/?PWRB packet, 13-400
Controllers,
    Magnetic tape, 5-10 (See also, "Devices".)
    Multiprocessor communications adapter (MCA), 5-10 (See also, "Devices".)
Controlling console interrupts on virtual consoles, 11-3
Conventions, filename, 4-8f
Converting
    date to scalar value with ?FDAY, 13-127f
    scalar time value with ?CTOD, 13-55f
    time of day to scalar value with ?FTOD, 13-147
Coordinating
    access to common resource, 2-7
    shared-file update, 2-8f
?COTT (escape-key control) characteristic, 13-154, 13-156, 13-515
Count,
    Block, 5-6
    Overlay use, 12-10
    Use (See "Use Count".)
?CPBN (column control) characteristic, 13-155, 13-157, 13-516
CPD (control point directory) (see "Control point directories".)
?CPM (page mode) characteristic, 13-155, 13-157, 13-516
?CPMAX system call, 4-1, 4-19, 13-37f
?CPOR offset, 13-41f
CPU time, 3-3
    Setting maximum for processes, 13-420
@CRA, 5-10 (See also, "Devices".)
@CRA1, 5-10 (See also, "Devices".)
?CRAC (rubout control) characteristic, 13-154, 13-515
?CRAF (rubout control) characteristic, 13-154, 13-515

Crash, system, 5-6
?CRAT (rubout control) characteristic, 13-154, 13-515
CRC (block check) error, 9-12 (See also, "Binary synchronous
  communications (BSC)".)
?CREATE system call, 4-1f, 4-13, 4-15, 7-3f, 13-39ff, 13-55, 13-128,
  13-478, 13-508
  Directory packet, 13-43ff
  IPC entry packet, 13-41ff
  Other file types packet, 13-45ff
  Time block, 13-55, 13-138
    Packet structure, 13-43
  Valid file types, 13-40
Creating
  break files after terminating processes, 3-18f (See "?BRKFL
    system call".)
  break files for every process trap, 3-18
  break files of specified user ring, 3-19
  connections, 8-2ff
  directories, 4-4, 13-39ff
  files (See "File creation and management".)
  IPC files with ?CREATE, 7-3 (See also, "Interprocess
    communications (IPC) facility".)
  labels for magnetic tapes with ?LABEL, 13-317ff
  link entries, 4-13
  processes, 3-8ff, 13-411ff
  queued task manager with /IQTSK, 13-273f
  search list with ?SLIST, 4-8
  son processes, 3-11
    of any process type, 3-10
    of different program file type, 3-10
    with different usernames, 3-10
  unlimited number of sons, 3-10f
  user data area (UDA) with ?CRUDA, 13-51f
Creation and termination detection (tasks), 6-16 (See also, "Tasks".)
Creation options, file, 5-4, 5-21, 13-364
Creation parameters,
  Process, 3-11
  Steps AOS/VS takes to check process, 3-11
Critical regions, locking/unlocking, 6-18f (See also, "Tasks".)
Crossing from outer ring to inner ring, 3-20
CRT consoles (See "Consoles".)
?CRUDA system call, 5-1, 5-19, 13-51f, 13-453
CS (current space), 4-19f (See also, "Control point directories
  (CPDs)".)
?CSFF (form-feed simulation) characteristic, 13-154, 13-515
?CST (tab simulation) characteristic, 13-154, 13-515
?CTERM system call, 8-1, 8-4f, 13-53f, 13-485 (See also, "Disconnect
  system calls".)

?CTIM offset, 13-41ff
?CTO (device time-out) characteristic, 13-155, 13-516
?CTOD system call, 11-3, 13-55f
CTRL key, 5-19
CTRL-C control character, 5-20 (See also, "Control characters".)
CTRL-C CTRL-A control sequence, 5-21, 6-13, 13-242, 13-271, 13-353, 13-355 (See also, "Control sequences".)
CTRL-C CTRL-B control sequence, 5-21, 7-11, 7-13, 11-3, 13-31, 13-330 (See also, "Control sequences".)
CTRL-C CTRL-C control sequence, 5-21 (See also, "Control sequences".)
CTRL-C CTRL-D control sequence, 5-21 (See also, "Control sequences".)
CTRL-C CTRL-E control sequence, 11-3
CTRL-C CTRL-E control sequence, 3-18f, 11-3, 13-90 (See also, "Control sequences".)
CTRL-D control character, 5-20 (See also, "Control characters".)
CTRL-O control character, 5-20 (See also, "Control characters".)
CTRL-P control character, 5-19f (See also, "Control characters".)
CTRL-Q control character, 5-13, 5-20 (See also, "Control characters".)
CTRL-S control character, 5-20 (See also, "Control characters".)
    Disabling with CTRL-Q, 5-20
    Enabling console line to recognize, 5-14
    Enabling console line to send, 5-14
CTRL-T control character, 5-20 (See also, "Control characters".)
CTRL-U control character, 5-20 (See also, "Control characters".)
CTRL-V control character, 5-20 (See also, "Control characters".)
?CTSP (blanks control) characteristic, 5-18, 13-155 13-157, 13-516
?CTYPE system call, 3-1, 3-10, 3-13, 13-57, 13-406, 13-419, 13-619
?CUCO (case control) characteristic, 13-154, 13-515
?CULC (case control) characteristic, 13-155f, 13-516
Current bias factor values, getting with ?GBIAS, 13-151f
Current date, getting with ?GDAY, 13-165
Current LEF mode status, returning, 13-325f
Current number of undedicated pages, Returning (See "GMEM system call".)
Current operating system information, getting with ?SINFO, 13-565ff
Current resource base, getting with ?GCRB, 13-163f
Current search list, examining with ?GLIST, 4-8 (See also, "Search list".)
Current shared partition size, listing with ?GSHPT, 13-201f
Current size of shared partition, Listing (See "GSHPT system call".)
Current space (CS), 4-19f (See also, "Control point directories (CPDs)".)
Current time slice, rescheduling, 13-483f
Current unshared memory parameters, Listing with ?MEM, 13-347f
Current working directory (See "Working directory".)

Customer,
    Establishing logical connection between existing server and, 8-2
    Managing exchanges between server and, 8-2
    Servers concurrently connected to multiple rings within, 8-6
    Communicating with via IPC system calls, 8-4
    Verifying inner-ring, 13-673f
    Verifying Ring 7, 13-671f
Customer of specified process, becoming, 13-33ff
Customer processes,
    Chaining, 8-7
    Defining, 8-2
    Terminating with ?CTERM, 8-4, 13-53f
Customer/server configuration, illustration of model, 8-2 (See also, "Connection-management facility".)
Customer/server connection,
    Breaking, 13-53f
    Passing to another customer in Ring 7 with ?PCNX, 8-4
    Passing to another customer with ?PRCNX, 8-4
Customer/server relationship, process termination messages in, 7-9ff (See also, "Connection-management facility".)
?CWRP (wraparound control) characteristic, 13-155, 13-516

D

/D specifications string switch, 13-94
?DAC2 offset, 13-82, 13-628ff, 13-632f
?DACL system call, 4-1, 4-15, 13-59f, 13-149f, 13-206
?DADID system call, 3-1, 3-8
Data,
    Deleting following file pointer, 5-6
    Illustration of on magnetic tape, 5-26 (See also, "Labeled magnetic tape".)
    Receiving over BSC lines with ?SRCV, 9-2, 13-585ff (See also, "Binary synchronous communications (BSC)".)
    Re-entrant, 2-6 (See also, "Re-entrant code or data".)
    Sending over BSC lines with ?SSND, 9-2, 13-601ff (See also, "Binary synchronous communications (BSC)".)
Data channel, communicating across, 5-12f
Data channel (DCH) ?IDEF option, 13-235ff (See also, "?IDEF system call".)
Data channel (DCH) map, 10-11
    ?IDEF option, 10-3f
    Setting, 13-611ff

Data channel line printers, 5-10 (See also, "Character devices".)
Data elements, allocating blocks for specific, 5-7
Data files,
  File type of system, 4-6
  File type of user, 4-6
  User, 4-5
@DATA generic filename, 3-19, 5-11f (See also, "Generic files".)
Data set flag DSR, 13-543
Data-link control characters (DLCC), 9-6ff, 13-538, 13-587ff, 13-602, 13-604 (See also, "Binary synchronous communications (BSC)".)
  ACK0 (affirmative acknowledgment), 9-7f, 13-539ff, 13-590, 13-607f
  ACK1 (affirmative acknowledgment), 9-7f, 13-539ff, 13-590
  BCC (block check character), 9-7, 13-539, 13-605
  DLE (data-link escape), 9-7, 9-10, 13-539
  DLE ETB (data-link escape, end-of-transmission block), 13-539, 13-604f
  DLE EOT (data-link escape, end of transmission), 9-8, 5-39, 13-608
  DLE ETX (data-link escape, end of text), 13-539, 13-604f
  DLE STX (data-link escape, start of text), 13-539, 13-605
  ENQ (enquiry), 9-7f, 9-11f, 13-539f, 13-542, 13-607
  EOT (end of transmission), 9-8, 9-12, 13-540, 13-542, 13-588, 13-591, 13-604
  ETB (end-of-transmission block), 9-7f, 9-10, 13-539f, 13-605
  ETX (end of text), 9-7ff, 13-539f, 13-605
  ITB (end-of-intermediate-transmission block), 9-9, 13-539, 13-541, 13-594, 13-604
  NAK (negative acknowledgment), 9-8f, 9-11f, 13-540ff, 13-557, 13-588, 13-590, 13 608
  Receiving over enabled BSC line with ?SRCV, 13-585ff
  RVI (reverse interrupt), 9-8f, 13-540f, 13-587f, 13-590f, 13-604
  SOH (start of header), 9-8f, 13-541
  STX (start of text), 9-8f, 13-541
  SYN (synchronization character), 9-10, 13-542
  TTD (temporary text delay), 9-10, 13-542, 13-604
  WACK (wait-before-transmitting positive acknowledgment), 9-8, 9-10, 13-540, 13-588, 13-593, 13-604
Data-link escape, end-of-text (DLE ETX), 13-539, 13-604f (See also, "Data-link control characters (DLCC)".)
Data-link escape, end-of-transmission (DLE EOT), 9-8, 13-539, 13-608 (See also, "Data-link control characters (DLCC)".)
Data-link escape, end-of-transmission block (DLE ETB), 13-539, 13-604f (See also, "Data-link control characters (DLCC)".)
Data-link escape, start of text (DLE STX), 13-539, 13-605 (See also, "Data-link control characters (DLCC)".)

Data-link escape (DLE), 9-7, 9-10, 13-539 (See also, "Data-link control characters (DLCC)".)
Data-sensitive records, 5-9
Date,
  Converting to scalar value with ?FDAY, 13-127f
  Getting current with ?GDAY, 13-165
Day, Getting time of, 13-217
?DCC offset, 13-82, 13-632ff
?DCI offset, 13-82, 13-632ff
?DCON system call, 8-1, 8-5, 13-54, 13-63f, 13-85, 13-485, 13-671, 13-673
DCT (device control table) (See "Device control table (DCT)".)
?DDIS system call, 10-1, 10-10, 13-65f
Deadlock, resource, 12-5
Deadlock, system, 13-177
?DEASSIGN system call, 5-1, 5-18, 13-67f
Deassigning character device with ?DEASSIGN, 13-67f
?DEBL system call, 10-1, 10-10, 13-65, 13-69f
?DEBUG system call, 11-5, 13-71
Debugger utility, 11-5
  Calling with ?DEBUG, 13-71
Decoding PID into local PID and host ID with ?TPID, 13-647f
Decreasing or increasing number of unshared pages in Ring 7, 2-10 (See "?MEMI system call".)
Decrementing use count and releasing shared page (See "RPAGE system call".)
Dedicated communications line, 9-1 (See also, "Binary synchronous communications (BSC)".)
Dedicated memory pages, 2-8 (See also, "Pages".)
  Definition of, 2-9
Default pathname of break files, 3-19 (See also, "Break files".)
Default user ring, 2-3
Defining
  access control list (ACL) with ?CREATE, 4-15
  characteristics of character device, 5-14
  console interrupt task with ?INTWT, 13-271f
  customer process, 8-2
  kill-processing routines with ?KILAD, 13-305ff
  map slots, 13-235 (See also, "?IDEF system call".)
  partitions in NREL memory with .PART pseudo-op, 2-7
  polling list or poll address/select address pair with ?SDPOL, 13-529ff
  server process, 8-2
  shared area with assembly language pseudo-ops, 2-6f
  stacks, 6-7f (See also, "Stacks".)
  system devices during system-generation procedure, 10-1
  unique kill-processing routine with ?KILAD, 6-15
  user devices, 3-10
  user devices with ?IDEF, 13-233ff
  user devices with ?PVDV, 10-1
  working-set parameters for sons, 3-10

Definition table, map (See "Map definition table".)
Definitions (See Glossary for additional definitions.)
- Access control list (ACL), 4-15
- Block input/output (I/O), 5-6
- Blocks, 5-3
- Channels, 5-3
- Character devices, 5-13
- Contention, 9-3
- Contiguous disk blocks, 4-2
- Control characters, 5-19
- Control sequences, 5-20
- Dedicated communications line, 9-1
- Directories, 4-4
- Double connection, 8-3
- File elements, 4-2
- File input/output (I/O), 5-2
- Filenames, 4-2, 4-8
- Files, 4-2
- Global port numbers, 7-2
- Index levels, 4-2
- Labeled magnetic tapes, 5-22
- Link entries, 4-12
- Local port numbers, 7-2
- Logical address space, 2-1
- Logical disks (LDs), 4-17
- Memory-management terms, 2-1f
- Movable resource, 13-163
- Multilevel connection, 8-2
- Pathnames, 4-9
- Physical block input/output (I/O), 5-7
- PID/ring tandems, 8-6
- Polling, 9-4f
- Primary station, 9-3
- Processes, 3-2
- Programs, 3-2
- Record input/output (I/O), 5-8
- Search lists, 4-8
- Secondary station, 9-3
- Segments, 2-2
- Selecting, 9-5
- Shared pages, 2-2
- Station, 9-1
- Switched communications line, 9-2
- Tasks, 3-2
- Unshared pages, 2-2
- Unused page, 2-2
- Working directory, 4-7
- Working set, 2-2

?DELAY system call, 13-73f
?DELETE system call, 4-1, 4-13, 13-75f, 13-478, 13-508
Deleting data following file pointer with ?TRUNCATE, 5-6
Deleting file entries, 13-75f
Deleting (and creating) link entries, 4-13 (See also, "Link entries".)
Delimiter tables,
    Sample, 13-367
    Setting, 13-367f, 13-525ff
Demand paging, 2-4
Dequeuing queued tasks with ?DQTSK, 13-81ff
Description of file input/output (I/O) sample programs, 5-2 (See also, "File input/output (I/O)".)
Descriptor, procedure entry (See "Procedure entry descriptor".)
Detecting
    references to remote hosts in pathnames with ?RNAME, 11-7
    task creation and termination, 6-16 (See also, "Tasks".)
Determining
    bias factors at runtime with ?GBIAS, 11-7
    form names, 13-117
    whether pathname references remote host with ?RNAME, 13-493f
Device access,
    Disabling with ?DDIS, 13-65f
    Device access, Enabling with ?DEBL, 13-69f
Device characteristics,
    Common, 13-156f
    Reading character device, 13-153ff
Device codes, 10-1
Device control table (DCT), 10-2, 10-6, 10-9, 10-11, 13-234, 13-265, 13-299
    Illustration of, 10-2
    Length of (?UDLN), 10-2
    Packet structure for 16-bit processes, 13-235
    Packet structure for 32-bit processes, 13-234
    Removing entry from interrupt vector table with ?IRMV, 13-281f
Device I/O, 10-10 (See also, "Input/output (I/O)".)
Device interrupt, 10-9
Device names, 5-9f (See also, "Devices".)
Devices, 4-2, 5-10
    Accessing all, 10-9f (See also, "User device support".)
    Assigning to processes for record I/O, 5-1 (See also, "Record input/output (I/O)".)
    Character, 4-2, 5-13ff (See also, "Character devices".)
    Communications (See "Communications device".)
    Defining and accessing user, 3-10, 13-233ff
    Input/output (I/O), 4-10, 5-2
    Multifile, 4-2

Devices (Cont.)
    Names of, 5-9f
        @CONn, 5-10
        @CRA, 5-10
        @CRA1, 5-10
        @DKBn, 5-10
        @LMT, 5-10
        @LPB, 5-10
        @LPBn, 5-10
        @MCA, 5-10
        @MCA1, 5-10
        @MTBn, 5-10
        @PLA, 5-10
        @PLA1, 5-10
        @SLNx, 9-2
    Opening for exclusive use of one process, 5-18
    Opening for I/O, 13-357ff
    Setting time-out value for, 13-617f
    System, 4-10
    User (See "User devices".)
    User-defined, 10-3
?DFLGS offset, 13-82, 13-628f, 13-632f
?DFLRC flag, 13-629, 13-633
DFMTR utility, 5-8
?DFRSCH system call, 6-1, 6-11, 6-18f, 13-77f
DIA assembly language instruction, 5-8
Dialogue, system-generation, 4-2
DIB assembly language instruction, 5-8
DIC assembly language instruction, 5-8
?DID offset, 13-82, 13-628ff, 13-632f
Differences between
    ?GRNAME and ?GNAME, 4-12, 13-180
    fast interprocess communication and IPC, 8-8f
    physical block I/O and block I/O, 5-7f (See also, "Block input/output (I/O)".)
Different usernames, creating sons with, 3-10
?DIR system call, 4-1, 13-79f
Directories, 4-4f, 4-7f
    Accessing, 4-7f
    Changing working, 4-8, 13-79f
    Control point (See "Control point directory (CPD)".)
    Creating, 4-4, 13-39ff
    Definition of, 4-4
    Entries, 4-4
    File type of control point, 4-6
    File type of disk, 4-6
    File type of spoolable peripheral, 4-6

Directories (Cont.)
    Illustration of
        control point, 4-20
        directory structure, 4-11
        sample directory tree, 4-5
    Information (See "?FSTAT system call".)
    Levels, 4-17
    Names, 4-4
    Peripheral (See "Peripheral directory (:PER)".)
    Root, 4-10
    Setting access control list (ACL) with ?SACL, 4-15
    Setting or removing permanent attribute from, 13-509f
    Working, 4-7, 4-10
Directory, peripheral (:PER), 13-39
Directory entries, listing, 13-181ff
Disabling
    access to all devices with ?DDIS, 13-65f
    and enabling access to all devices, 10-9f (See also, "User device support".)
    BSC line with ?SDBL, 9-2, 13-523 (See also, "Binary synchronous communications (BSC)".)
    console interrupts with ?ODIS, 13-353f
    control-character console interrupts, 13-311
    CTRL-S with CTRL-Q, 5-20 (See also, "Control characters".)
    LEF mode with ?LEFD, 13-69, 13-321f
    relative consoles with ?SDRT, 13-533f
    task scheduling with ?DRSCH, 6-11, 6-18f, 13-77f, 13-87f
Disassociating channel number from file, 5-3
Disconnect flag, explicit (See "Explicit disconnect flag".)
Disconnect system calls, 13-54 (See individual system call entries for additional references.)
    ?BRKFL, 13-54
    ?CTERM, 13-54
    ?DCON, 13-54
    ?RESIGN, 13-54
    ?RETURN, 13-54
    ?TERM, 13-54
Disk blocks, 4-2, 4-19, 5-3
    Allocating, 5-1, 13-7f
    Allocating for specific data elements, 5-7
    Bad, 5-8
    Contiguous, 4-2
    Definition of, 5-3
    Length of, 5-7
    Releasing, 13-478
Disk bootstraps, 4-4
Disk directory file (file type ?FDIR), 4-6, 5-5

Disk drives, 10-1 (See also, "User device support".)
Disk errors, bypassing retries for, 5-7
Disk file structures, 4-2ff
Disk files, 4-4
    Truncating, 13-219ff
Disk images, memory modification with, 12-2
Disk size, reducing, 5-7
Disk space, 4-2
    Controlling, 4-19f
    How AOS/VS allocates, 4-2
Disk units, 5-10 (See also, "Devices".)
    File type, 4-6
Disks,
    File type of logical, 4-6
    Flushing contents of shared page to, 13-135f
    Flushing shared file memory pages to (See "?ESFF system call".)
    Logical (See "Logical disks (LDs)".)
    Moving logical pages on demand to memory from (See "Demand paging".)
    Reading/writing physical blocks on, 13-397ff
    Reducing size while using block I/O, 5-7
DISMOUNT command (CLI), 5-35
Dismounting
    labeled magnetic tapes with the CLI DISMOUNT command, 5-35
    magnetic tapes, 13-109ff
Display consoles, 5-10 (See also, "Devices".)
Displaying next page in page mode, 5-13
@DKBn, 5-10 (See also, "Devices".)
DLCC (data-link control characters) (See "Data-link control characters (DLCC)".)
DLE (data-link escape), 9-7, 9-10, 13-539 (See also, "Data-link control characters (DLCC)".)
DLE EOT (data-link escape, end-of-transmission), 9-8, 13-539, 13-608 (See also, "Data-link control characters (DLCC)".)
DLE ETB (data-link escape, end-of-transmission block), 13-539, 13-604f (See also, "Data-link control characters (DLCC)".)
DLE ETX (data-link escape, end-of-text), 13-539, 13-604f (See also, "Data-link control characters (DLCC)".)
DLE STX (data-link escape, start of text), 13-539, 13-605 (See also, "Data-link control characters (DLCC)".)
?DLNK offset, 13-82, 13-628f, 13-632f
?DLNKB offset, 13-82, 13-628f, 13-632f
?DLNKL offset, 13-82
?DNUM offset, 13-82, 13-628f, 13-632f
Double connection, definition of, 8-3
?DPC offset (?TASK system call), 6-10, 13-82, 13-628ff, 13-632f

?DPRI offset, 13-82, 13-628ff, 13-632f
?DQTSK system call, 6-1, 13-81ff, 13-631
    Extended task definition packet, 13-82
?DRCON system call, 8-1, 8-5, 13-63, 13-85f
?DRES offset, 13-82, 13-628f, 13-632f
?DRSCH system call, 6-1, 6-11, 6-18f, 13-87f, 13-101
?DSCH flag, 13-77
?DSFLT offset (?TASK system call), 6-9, 13-82, 13-628f, 13-631ff
?DSH offset, 13-82, 13-632f
?DSMS offset, 13-82, 13-632f
DSR data set (modem) flag, 5-15, 5-17, 13-543
?DSSZ offset (?TASK system call), 6-10, 13-82, 13-628ff, 13-632f
?DSTB offset (?TASK system call), 6-9, 13-82, 13-628ff, 13-632f
DTR modem flag, 5-15
Dump, memory, 3-17f
DUMP command (CLI), 5-36
    Syntax of, 5-36
Dump file default pathname format, 13-346
Dumping
    memory image from specified ring to file (See "?MDUMP system call".)
    memory image to file, 13-345f
    particular ring with ?MDUMP, 3-18
Dynamic-length records, 5-8
    Format (?RTDY), 13-330

E

EBCDIC character set, 5-23f, 13-537f, 13-542
ECC (error-correction code), 5-8
Echoing ^C^C on console and emptying type-ahead buffer with CTRL-C CTRL-C, 5-21 (See also, "Control sequences".)
Editors, text, 4-2
?EFFP offset, 13-456, 13-467ff
?EFLN offset, 13-456, 13-467ff
?EFNF offset, 13-456, 13-467ff
?EFTL offset, 13-456, 13-467ff
?EFTY offset, 13-456, 13-467ff
EJSR instruction, 12-9
?ELAC offset, 13-370, 13-372
?ELCR offset, 13-370f
?ELCT offset, 13-370f
ELEF instruction, 12-9

Elements,
    Allocating blocks for specific data, 5-7
    Illustration of file growth stages with file, 4-3
    Keeping track of file, 4-2
    Specifying size of file, 4-2
?ELFS offset, 13-370, 13-372
?ELGN offset, 13-370f
Eligible process state, 3-13f (See also, "Processes".)
?ELRE offset, 13-370f
?ELUH offset, 13-370, 13-372
?ELUT offset, 13-31, 13-370ff
?ELVL offset, 13-370f
?ELVR offset, 13-370f
Emptying type-ahead buffer and echoing ^C^C on console with CTRL-C CTRL-C, 5-21 (See also, "Control sequences".)
Enable console interrupts, 13-355
Enabled BSC line, sending data over, 9-2
Enabling
    access to all devices with ?DEBL, 13-69f
    and disabling access to all devices, 10-9f (See also, "User device support".)
    break files, 13-89ff (See also, "?ENBRK system call" and "?MDUMP system call".)
    BSC lines with ?SEBL, 9-2, 13-536ff (See also, "Binary synchronous communications (BSC)".)
    console line to recognize CTRL-S with ?XIFC, 5-14 (See also, "Character devices".)
    console line to send CTRL-S with ?XOFC, 5-14 (See also, "Character devices".)
    LEF mode with ?LEFE, 13-323f
    multitask scheduling with ?ERSCH, 13-101f
?ENBRK system call, 3-1, 3-19, 13-89ff
    Packet contents, 13-91f
    Packet structure, 13-91
End-of-file character, terminating current read with ?CTRL-D, 5-20 (See also, "Control characters".)
End-of-file labels (See "Labels".)
End-of-intermediate-transmission block (ITB), 9-9, 13-539, 13-541, 13-594, 13-604f (See also, "Data-link control characters (DLCC)".)
    Illustration of receive buffer format, 13-594 (See also, "Data-link control characters (DLCC)".)
End-of-text (ETX), 9-7ff, 13-539f, 13-605 (See also, "Data-link control characters (DLCC)".)
End-of-transmission (EOT), 9-8, 9-12, 13-540, 13-588, 13-591, 13-604 (See also, "Data-link control characters (DLCC)".)

End-of-transmission block (ETB), 9-7f, 9-10, 13-539f, 13-605 (See also, "Data-link control characters (DLCC)".)
End-of-volume, Forcing on labeled magnetic tape with ?FEOV, 13-133f
End-of-volume labels (See "Labels".)
?ENEL offset, 13-359
?ENESH offset, 13-91f
?ENET offset, 13-359, 13-363
?ENEUS offset, 13-91f
?ENFLG offset, 13-91
?ENFNP offset, 13-91f
ENQ (enquiry), 9-7f, 9-11, 13-539f, 13-607 (See also, "Data-link control characters (DLCC)".)
?ENQUE system call, 11-6, 13-93ff
    Specifications string format, 13-93
    Specifications string messages, 13-94f
    Specifications string switches, 13-94
Enquiry (ENQ), 9-7f, 9-11f, 13-539f, 13-607 (See also, "Data-link control characters (DLCC)".)
?ENSSH offset, 13-91f
?ENSUS offset, 13-91
.ENT pseudo-op, 3-20
Entering, leaving, or examining Superprocess mode with ?SUPROC, 13-619f (See also, "?SUPROC system call".)
Entering, leaving, or examining Superuser mode with ?SUSER, 13-623f (See also, "?SUSER system call".)
Entering events in system log file with ?LOGEV, 13-333ff
.ENTO (overlay error) pseudo-op, 12-4, 12-10, 13-382
Entries,
    Deleting file, 13-75f
    Directory, 4-4
    File (See "File entries".)
    File type of IPC port, 4-6
    File type of queue, 4-6
    Link, 4-12f (See also, "Link entries".)
    Listing directory, 13-181ff
    Procedure, 12-6
Entry descriptor, procedure (See "Procedure entry descriptor".)
Environment, Unwinding stacks and restoring previous, 13-663
EOT (end-of-transmission), 9-8, 9-12, 13-540, 13-588, 13-591, 13-604 (See also, "Data-link control characters (DLCC)".)
Equal sign (=) pathname prefix, 4-10 (See also, "Pathnames".)
Erasing current console input line with CTRL-U, 5-20 (See also, "Control characters".)
?ERBA offset, 13-554
?ERCH offset, 13-554
?ERCS offset, 13-554

?ERFT offset, 13-30
ERMES file, 11-4, 13-97
    Error code structure in, 13-98
?ERMSG system call, 11-4, 13-97ff, 13-553
Error, CRC (block check), 9-12 (See also, "Binary synchronous communications (BSC)".)
Error codes, 11-4 (See also, Appendix A.)
    Getting text associated with, 11-4, 13-553f
    Structure in ERMES file, 13-98
Error message file, 11-4
    Reading, 13-97ff
Error statistics, getting BSC with ?SGES, 13-557ff (See also, "Binary synchronous communications (BSC)".)
Error-correction code (ECC), 5-8
Error-recovery procedures, binary synchronous communications (BSC), 9-11ff (See also, "Binary synchronous communications (BSC)".)
Error-recovery statistics, getting BSC, 9-13 (See also, "Binary synchronous communications (BSC)".)
Errors, disk (See "Disk errors".)
?ERSCH system call, 6-1, 6-11, 6-18, 13-101f
?ESCR offset, 13-456, 13-463
?ESEP offset, 13-456, 13-463
?ESFC offset, 13-456, 13-463
?ESFF system call, 2-1, 2-8ff, 13-103f
ESS (extended state save) area (See "Extended state save (ESS) area".)
Establishing
    connection between local and remote stations, 9-2
    interface between AOS/VS and unsupported device (See "?IDEF system call".)
    logical connection between customer and existing server, 8-2
    new shared partition size (See "?SSHPT system call".)
    new shared partition with ?SSHPT, 13-597f
ETB (end-of-transmission block), 9-7f, 9-10, 13-539f, 13-605 (See also, "Data-link control characters (DLCC)".)
?ETER offset, 13-359, 13-363
?ETFL offset, 13-456
?ETFT offset, 13-194, 13-359, 13-456, 13-466
?ETLL offset, 13-456
?ETLT offset, 13-30, 13-194, 13-359, 13-456, 13-462
?ETSL offset, 13-456
?ETSP offset, 13-30, 13-194, 13-359, 13-456, 13-462
ETX (end-of-text), 9-7ff, 13-539f, 13-605 (See also, "Data-link control characters (DLCC)".)
Evaluating FED string using ?FEDFUNC, 13-130f (See also, "?FEDFUNC system call".)
Event logging format, illustration of, 13-334

Events in system log file, entering with ?LOGEV, 13-333ff
Examining
    break files, 3-17 (See also, "Break files".)
    current search list with ?GLIST, 4-8
    default ACL with ?DACL, 4-15, 13-59f (See also, "Access control list (ACL)".)
    execute-protection status with ?EXPO, 13-123ff (See also, "?EXPO system call".)
    Superprocess mode with ?SUPROC, 13-619f (See also, "?SUPROC system call".)
    Superuser mode with ?SUSER, 13-623f (See also, "?SUSER system call".)
Exchanges between customers and servers, managing, 8-2
Exclusive Open option, 5-4f
Exclusive use, Opening device for, 5-18
?EXEC system call, 11-6, 13-105ff
    Packet contents,
    Packet structure,
        ?XFDUN, 13-109
        ?XFMLT (labeled), 13-108
        ?XFMUN (unlabeled), 13-107
        ?XFXML (labeled extended), 13-108
        ?XFXUN (unlabeled extended), 13-107
    Queues, 13-110ff, 13-118
    Status information, 13-119f
EXEC utility, 5-19, 5-35ff, 11-6, 13-52, 13-454, 13-488, 13-600, 13-626
    Functions, 11-6
    Interface to, 11-6
    Obtaining status information, 13-119f
    Requesting services from, 13-105ff
Execute (?FACE) access, 4-13ff, 13-60, 13-79, 13-206, 13-508
Execute-protection status, 3-16
    Examining, clearing, or setting with ?EXPO, 13-123ff (See also, "?EXPO system call".)
Execution, scheduling another process for with ?RESCHED, 3-14 (See "?RESCHED system call".)
Execution path, redirecting task's, 13-241f
Existing server, establishing logical connection between customer and, 8-2 (See also, "Connection-management facility".)
Exiting from interrupt service routines with ?IXIT, 13-297f
Exits from overlay and kills calling task with ?OVKIL, 13-379f
Explicit disconnect flag, 8-6ff
    Bit position, 8-8
?EXPO system call, 3-1, 3-16, 13-123ff

Extended characteristics of character device (See also, "Character devices".)
  Getting, 13-167ff
  Setting, 13-545ff
Extended packet (?CLOSE system call), 13-30
Extended state save (ESS) area, 12-11
  Initializing with ?IESS, 13-253f
  Use of, 13-253
Extensions to I/O packet, 13-359
External gate array, 3-20
.EXTG pseudo-op, 3-20
.EXTN pseudo-op, 12-4
Extracting ring field from global port number, 7-4 (See also, "Interprocess communications (IPC) facility".)

F

?FACA (Append) access, 4-13ff, 13-60, 13-206, 13-508 (See also, "Access control list (ACL)".)
?FACE (Execute) access, 4-13ff, 13-60, 13-79, 13-206, 13-508 (See also, "Access control list (ACL)".)
?FACO (Owner) access, 4-13ff, 4-17, 13-60, 13-206, 13-508f (See also, "Access control list (ACL)".)
?FACR (Read) access, 4-13ff, 13-60, 13-79, 13-138, 13-149, 13-206, 13-508 (See also, "Access control list (ACL)".)
Factors, bias, 11-7 (See also, "Bias factors".)
?FACW (Write) access, 3-19, 4-13ff, 505, 13-60, 13-90, 13-149, 13-206, 13-508 (See also, "Access control list (ACL)".)
Fast interprocess communications (See "Fast interprocess synchronization".)
Fast interprocess synchronization, 8-8f (See also, "Interprocess communications (IPC) facility".)
  System calls, 8-4f (See inidividual system call entries for additional references.)
    ?SIGNL, 8-4
    ?SIGWT, 8-5
    ?WTSIG, 8-4
  Differences between IPC and, 8-8f
Father processes, 3-7 (See "Processes".)
?FBEX offset, 13-130f
?FBSTF offset, 13-130
?FCON file type, 4-7
?FCPC offset, 13-130f

?FCPD file type (control point directory file), 4-6, 5-5, 13-40, 13-141
?FCRA file type, 4-7
FCU (forms control) utility, 5-19
?FDAY system call, 11-3, 13-127f
?FDBA offset, 13-130f
?FDBL offset, 13-130f
?FDIR file type (disk directory file), 4-6, 5-5, 13-40
?FDKU file type, 4-6
FED (file editor) utility (See "File editor (FED) utility".)
?FEDFUNC system call, 11-6, 13-129ff
    Packet structures,
        Packet to change radix, 13-129
        Packet to open symbol table file, 13-130
        Packet to evaluate FED string, 13-130
    Packet contents,
        Packet to evaluate FED string, 13-131
?FEOV system call, 13-133f
?FFCC file type, 4-7
?FFLAG offset, 13-129ff
?FGFN file type, 4-6
?FGLT file type, 4-6, 5-37
Field,
    Access (ANSI-standard labeled magnetic tapes), 5-25
    Owner Name (labeled magnetic tapes), 5-28
    Ring, 6-5
    Version Number (labeled magnetic tapes), 5-28
File access, 4-13f
    Controls, 3-9
    Files outside current working directory, 13-79f
    Methods for file input/output (I/O), 5-3
    Privileges (See "Access privileges".)
File creation and management, Chapter 4
    Sample programs, 4-21ff
    System calls, 4-1 (See individual system call entries for additional references.)
        ?CGNAM, 4-1
        ?CPMAX, 4-1
        ?CREATE, 4-1
        ?DACL, 4-1
        ?DELETE, 4-1
        ?DIR, 4-1
        ?FSTAT, 4-1
        ?GACL, 4-1
        ?GLINK, 4-1
        ?GLIST, 4-1
        ?GNAME, 4-1
        ?GNFN, 4-1

File creation and management (Cont.)
    System calls (Cont.)
        ?GRNAME, 4-1
        ?GTACP, 4-1
        ?INIT, 4-1
        ?RECREATE, 4-1
        ?RELEASE, 4-1
        ?RENAME, 4-1
        ?SACL, 4-1
        ?SATR, 4-1
        ?SLIST, 4-1
File creation options, 13-357ff, 13-364f
File descriptor information, 5-6
    Flushing with ?UPDATE, 13-667f
File editor (FED) utility, 11-6, 13-129, 13-345 (See also, "?FEDFUNC system call".)
    Evaluating FED string with ?FEDFUNC, 13-130f
    Functions, 11-6
    Interfacing to with ?FEDFUNC, 11-6, 13-129ff
File elements,
    Address, 4-2
    Definition of, 4-2
    Illustration of file growth stages, 4-3
    Keeping track of, 4-2
    Specifying size of, 4-2
File entries,
    Deleting, 13-75f
    Getting access control list (ACL) for, 13-149
    Queuing, 13-93ff, 13-110ff (See also, "?EXQUE system call".)
File format specification, 5-19
File information (See "?FSTAT system call".)
File input/output (I/O), 4-2, Chapter 5
    Operation sequence, 5-3
    System calls, 5-1 (See individual system call entries for additional references.)
        ?ALLOCATE, 5-1
        ?ASSIGN, 5-1
        ?CLOSE, 5-1
        ?CRUDA, 5-1
        ?DEASSIGN, 5-1
        ?GCHR, 5-1
        ?GCLOSE, 5-1
        ?GECHR, 5-1
        ?GOPEN, 5-1
        ?GPOS, 5-1
        ?GTRUNCATE, 5-1
        ?LABEL, 5-1
        ?OPEN, 5-1

File input/output (I/O) (Cont.)
    System calls (Cont.)
        ?PRDB, 5-1
        ?PWRB, 5-1
        ?RDUDA, 5-1
        ?READ, 5-1
        ?RELEASE, 5-1
        ?SCHR, 5-1
        ?SDLM, 5-1
        ?SECHR, 5-1
        ?SEND, 5-1
        ?SPOS, 5-1
        ?STOM, 5-1
        ?TRUNCATE, 5-1
        ?UPDATE, 5-1
        ?WRITE, 5-1
        ?WRUDA, 5-1
    Concepts, 5-2ff
    Definition of, 5-2
    File access methods, 5-3
        Block I/O, 5-3 (See also, "Block input/output (I/O)".)
        Record I/O, 5-3 (See also, "Record input/output (I/O)".)
    Labeled magnetic tapes, 5-35, 13-193, 13-369ff (See also, "Labeled magnetic tapes".)
    Operation sequence, 5-4ff
    Sample programs, 5-38
        Description of, 5-2
    Unlabeled magnetic tapes, 5-37f, 13-368 (See also, "Magnetic tapes".)
File pointer, 5-6
    Deleting following data with ?TRUNCATE, 5-6
    Getting position of with ?GPOS, 5-6, 13-193ff
    Positioning with ?SPOS, 5-6, 13-195, 13-581ff
File specification packets, 5-4
File specifications word, 13-363ff
File structures,
    AOS/VS, 4-1
    Disk, 4-2ff
File trailer labels (See "Labels".)
File types, 4-4f
    AOS program file, 4-6f
    AOS/VS program file, 4-6
    Card reader, 4-7
    Console (hard-copy or video display), 4-7
    Control point directory (CPD), 4-6, 13-43
    Created with ?OPEN, 5-5, 13-365
    Creating sons of different program, 3-10
    Data channel line printer, 4-7

File types (Cont.)
  Disk directory, 4-6
  Disk unit, 4-6
  ?FCON, 4-7
  ?FCPD (control point directory file), 4-6, 5-5
  ?FCRA, 4-7
  ?FDIR (disk directory file), 4-6, 5-5
  ?FDKU, 4-6
  ?FFCC, 4-7
  ?FGFN, 4-6
  ?FGLT, 4-6, 5-37
  ?FIPC (IPC file), 4-6, 5-5, 5-21, 7-4
  ?FLDU (logical disk), 4-6, 13-141
  ?FLNK, 4-6
  ?FLOG, 11-5, 13-626
  ?FLPU, 4-7
  ?FMCU, 4-7
  ?FMTF, 4-6
  ?FMTU, 4-7
  ?FNCC, 4-7
  ?FOCC, 4-7
  FORTRAN carriage control, 4-7
  ?FPCC, 4-7
  ?FPLA, 4-7
  ?FPRG (AOS program file), 4-5ff, 5-5
  ?FPRV (AOS/VS program file), 4-5f, 5-5
  ?FQUE, 4-6
  ?FSDF, 4-6
  ?FSPR, 4-6
  ?FSTF, 4-6
  ?FSYN, 4-7
  ?FTXT (text file), 4-6, 5-5
  ?FUDF (user data file), 4-5f, 5-5
  ?FUPF, 4-6
  Generic filename, 4-6
  Generic labeled tape, 4-6
  IPC port entry, 4-6
  Link file, 4-6
  List of, 4-6f
  Logical disk, 4-6
  Magnetic tape file, 4-6
  Magnetic tape unit, 4-7
  Multiprocessor communications unit, 4-7
  Plotter, 4-7
  Queue entry, 4-6
  Standard directory, 13-43
  Spoolable peripheral directory, 4-6
  Symbol table file, 4-6

File types (Cont.)
    Synchronous communications line, 4-7
    System data file, 4-6
    Text file, 4-6
    User data file, 4-6
    User profile file, 4-6
    Valid ?CREATE system call, 13-40
File's requirements for indexes, 4-4
File-pointer position,
    Current, 13-193ff
    Setting with ?SPOS, 5-6, 13-195, 13-581ff
    Settings, 13-582
Filenames, 4-4, 4-8f
    Conventions, 4-8f
    Definition of, 4-2, 4-8
    Generic, 4-10, 5-9, 5-11f
        File type, 4-6
    Getting .PR for ring (See "?RNGPR system call".)
    Legal characters for use in, 4-8
    Valid characters, 3-6
Files,
    AOS program (See "File types".)
    AOS/VS program (See "File types".)
    Assembly language source (.SR), 4-9
    Break, 3-17f (See also, "Break files".)
    CLI macro (.CLI), 4-9
    Closing block I/O, 13-159f
    Closing shared with ?SCLOSE, 13-519f (See also, "?SCLOSE system call".)
    Control point directory (See "File types".)
    Creating, 13-39ff
    Definition of, 4-2
    Disassociating channel number from, 5-3
    Disk, 4-4 (See also, "Disk files".)
    Disk directory (See "File types".)
    Dumping memory image from specified ring to (See "?MDUMP system call".)
    Dumping memory images to, 13-345f
    Enabling break (See "?ENBRK system call".)
    Error message (ERMES), 11-4
    File type of AOS program, 4-6
    File type of AOS/VS program, 4-6
    File type of disk unit, 4-6
    File type of generic labeled tape, 4-6
    File type of Link, 4-6
    File type of logical disk, 4-6
    File type of symbol table, 4-6
    File type of system data, 4-6

Files (Cont.)
  File type of text, 4-6
  File type of user data, 4-6
  File type of user profile, 4-6
  Generic, 5-11f (See also, "Generic files".)
  Getting pathnames of generic files with ?GRNAME, 4-12
  Getting pathnames of with ?GNAME, 4-12
  Getting status information with ?FSTAT, 13-137ff
  Illustration of growth stages, 4-3
  IPC (See "File types".)
  Library (.LB), 4-9
  Linking object modules to form program, 12-4
  Loading into specific rings with ?RINGLD, 3-20
  Object (.OB), 4-9
  Opening, 13-357f
    For shared access, 13-571f
    For block I/O with ?GOPEN, 13-185ff
    Protected shared, 2-7f, 13-573ff (See also, "?SOPPF system call".)
  Overlay (.OL), 13-2
  Program (.PR), 4-5, 4-9, 6-3, 12-2 (See also, "Program file".)
  Protected shared, 2-7f, 3-20
    Opening, 2-7f, 13-573ff (See also, "?SOPPF system call".)
    Permitting access to protected shared (See "?PMTPF system call".)
  Recreating with ?RECREATE, 13-477f
  Renaming with ?RENAME, 4-8, 13-481f
  Sample creation and management programs, 4-21ff
  Setting access control list (ACL) with ?SACL, 4-15
  Setting or removing permanent attribute from, 13-509f
  Status, 3-17
  Symbol table (.ST)(See "Symbol table file".)
  System log (See "System log file".)
  Temporary (.TMP), 4-9
  Text (See "File types".)
  Truncating, 13-219ff
  Types of program, 4-5
  Unshared, 13-667
  User data, 4-5 (See "File types".)
FILESTATUS CLI command, 13-183
Finding global port number's owner, 13-287f
?FIPC file type (IPC file), 4-6, 5-5, 5-21, 7-4
First opener, 2-8
Fixed-length records, 5-8

Flag bits, 8-6
    Inner-ring connection management, 8-7f
Flag word, user, 7-9
Flags,
    Data set (DSR), 13-543
    Explicit disconnect (See "Explicit disconnect flag".)
    ?IFBNK, 7-8f
    ?IFNSP, 7-8f
    ?IFPR, 7-8f
    ?IFRFM, 7-8
    ?IFRING, 7-8f
    ?IFSOV, 7-8f
    ?IFSTM, 7-8
    ?ISELF (offset ?IUFL), 7-10
    ?OANS (?IRES offset of ?OPEN), 5-24
    ?OIBM (?IRES offset of ?OPEN), 5-24
    ?TMYRING (?TLOCK system call), 6-14
    System and user (IPC), 7-7ff
?FLDU (logical disk) file type, 4-6, 13-141
?FLEX offset, 13-130f
?FLNK file type, 4-6
Floating-point status registers, 10-8
    Initializing with ?IFPU, 6-19
Floating-point unit, initializing with ?IFPU, 13-255f
?FLOG file type, 11-5, 13-626
?FLPU file type, 4-7
?FLUSH system call, 13-135f, 13-502, 13-667
Flushing
    contents of shared page to disk with ?FLUSH, 13-135f
    file descriptor information with ?UPDATE, 13-667f
    shared file memory pages to disk wth ?ESFF, 2-8ff, 13-103f (See also, "?ESFF system call".)
?FMCU file type, 4-7
?FMTF file type, 4-6
?FMTU file type, 4-7
?FNCC file type, 4-7
?FNIR offset, 13-129
?FOCC file type, 4-7
Forcing AOS/VS to initialize common inner-ring stack, 6-9 (See also, "Inner rings".)
Forcing end-of-volume on labeled magnetic tape with ?FEOV, 13-133f
Form names, determining, 13-117
Format control for line printers, 5-19

Formats,
    Access control list (ACL) specifications, 13-508 (See also, "Access control list (ACL)".)
    ANSI-standard, 5-22, 5-28, 5-31f
    CLI, 13-488
    Controlling console, 5-19
    IBM, 5-22, 5-31f
    Illustration of event logging, 13-334
    Messages sent with ?SEND, 13-550
    Search list, 13-569
    Specifying file, 5-19
    Tailored line-printer output, 5-19
    Task messages, 13-652
Forms control utility (FCU), 13-454
/FORMS switch, CLI, 5-19
FORTRAN, 6-3
FORTRAN carriage control (file type), 4-7
?FPCC file type, 4-7
?FPLA file type, 4-7
?FPRG file type (AOS program file), 4-5ff, 5-5
?FPRV file type (AOS/VS program file), 4-5f, 5-5
?FQUE file type, 4-6
Frame pointer, 6-9, 6-15
Frames in stack, getting information about previous, 13-675f

Free memory chain, 2-4
Freezing console output with CTRL-S, 5-20 (See also, "Control characters".)
Frequency,
    Getting system clock, 13-171f
    Real-time clock, 13-677
?FRESD offset, 13-130f
?FRESS offset, 13-129
?FRFNC offset, 11-6, 13-129ff
?FSDF file type, 4-6
?FSPR file type, 4-6
?FSTAT system call, 4-1, 4-17, 13-55, 13-137ff, 13-453, 13-510
    Description of directory/remaining type packets, 13-140ff
    Description of unit file/IPC file packets, 13-138
    Packet structure,
        Directory file packet, 13-141
        IPC file packet, 13-140
        Unit file packet, 13-139
    Sample packet, 13-144
?FSTF file type, 4-6
?FSYN file type, 4-7
?FTOD system call, 11-3, 13-147
?FTXT file type (text file), 4-6, 5-5

?FUDF file type (user data file), 4-5f, 5-5
Full process name, 3-6f, 13-226
    Getting, 3-7, 13-393f (See also, "?PNAME system call".)
Full-duplex communications paths, 7-2
Full-duplex modems, 5-15ff (See also, "Modems (full-duplex)".)
Function of control characters, 5-20 (See also, "Control characters".)
?FUPF file type, 4-6

G

?GACL system call, 4-1, 4-15, 13-149f, 13-508
Gate array, external, 3-20
?GBIAS system call, 11-7, 13-151f, 13-512
?GCHR system call, 5-1, 5-14, 5-16, 5-18, 13-153ff, 13-168
?GCLOSE system call, 5-1, 5-4, 13-159f
?GCPN system call, 7-1, 7-4, 13-161f
?GCRB system call, 13-163f
?GDAY system call, 11-3, 13-165
?GECHR system call, 5-1, 5-14, 5-18, 13-167ff
Generating
    console interrupt and aborting process with CTRL-C CTRL-B, 5-21 (See also, "Control sequences".)
    console interrupt with CTRL-C CTRL-A, 5-21 (See also, "Control sequences".)
    histograms with ?WHIST, 3-16

Generic files, 3-19, 5-11f, 13-24
    @DATA, 13-24
    File type, 4-6
        Labeled magnetic tape files, 4-6
    Filenames, 4-10, 5-9, 5-11f
        @CONSOLE, 5-11f
        @DATA, 5-11f
        @INPUT, 5-11f
        @NULL, 5-11f
        @OUTPUT, 5-11f
    Getting complete pathname, 13-199f
    Getting pathnames of with ?GRNAME, 4-12
    @INPUT, 13-24
    @LIST, 13-24
    @OUTPUT, 13-24
    ?PROC packet parameters for, 5-12
    Sample pathname, 5-11

Getting
- access control list (ACL) with ?GACL, 4-15, 13-149f
- access control privileges with ?GTACP, 13-205f
- AOS-format internal time, 13-295
- BSC error statistics with ?SGES, 13-557ff (See also, "Binary synchronous communications (BSC)".)
- BSC error-recovery statistics, 9-13 (See also, "Binary synchronous communications (BSC)".)
- characteristics of character device, 5-14
- CLI message, 13-207ff
- complete pathname of generic file, 13-199f
- complete pathname with ?GNAME, 13-179f
- console port number with ?GCPN, 13-161f
- current bias factor values with ?GBIAS, 13-151f
- current date with ?GDAY, 13-165
- current file pointer position with ?GPOS, 5-6, 13-193ff
- current resource base with ?GCRB, 13-163f
- extended characteristics of a character device, 13-167ff
- file status information with ?FSTAT, 13-137ff
- full process name with ?PNAME, 3-7, 13-393f
- global port number, 13-261f
- host name or host identifier with ?HNAME, 11-7, 13-229ff
- information about current operating system with ?SINFO, 13-565ff
- information about previous frames in stack with ?WALKBACK, 13-675f
- information on processes/queues with ?BNAME, 11-5
- Link entry contents with ?GLINK, 13-173f
- number of undedicated memory pages with ?GMEM, 13-177f
- pathnames of files with ?GNAME, 4-12
- pathnames of generic files with ?GRNAME, 4-12
- PID associated with global port number using ?GPORT, 13-191f
- PID of calling process with ?PNAME, 3-7
- PID of father process with ?DADID, 3-8 (See also, "?DADID system call".
- .PR filename for ring (See "?RNGPR system call".)
- priority and TID of calling task with ?MYTID, 13-351
- process name (See "?PNAME system call".)
- process runtime statistics (See "?RUNTM system call".)
- process username with ?GUNM, 13-225f (See also, "?GUNM system call".)
- program's pathname with ?GPRNM, 13-197f
- runtime statistics for process with ?RUNTM, 13-503ff
- search list contents with ?GLIST, 13-175f
- status information for process (See "?PROC system call".)
- symbol closest in value to specified input value with ?GTNAM, 13-215f
- system clock frequency with ?GHRZ, 13-171f
- system identifier with ?GSID, 13-203f

Getting (Cont.)
- task status, 13-641, 13-661f
- text string associated with particular error code, 11-4, 13-553f
- time of day with ?GTOD, 13-217
- unambiguous task identifier, 13-661f
- username associated with PID using ?GUNM, 3-8
- username associated with simple process name with ?GUNM, 3-8
- value of user symbol with ?GTSVL, 13-223f
- virtual PID of process with ?GVPID, 13-227

?GHRZ system call, 11-3, 13-73, 13-171f, 13-677
?GLINK system call, 4-1, 4-13, 13-173f
?GLIST system call, 4-1, 4-8, 13-175, 13-569
- Buffer format, 13-175

Global IPC port ?SPTM, 13-53f
Global port numbers, 7-2ff, 13-162 (See also, "Interprocess communications (IPC) facility".)
- Definition of, 7-2
- Extracting ring field from, 7-4
- Finding owner, 13-287f
- Getting, 13-2261f
- Getting PID associated with, 13-191f
- Identifying PID associated with (with ?GPORT), 7-4
- Interpreting ring fields within, 7-4
- Modifying ring field within, 7-4, 13-263f
- Predefined origin port for obituary messages (?SPTM), 7-9
- Translating local port numbers into equivalent, 13-649

Global ports, ?SPTM, 13-53f
Global synchronous manager (GSMGR) process, 9-2
?GMEM system call, 2-1, 13-177f
?GNAME system call, 4-1, 4-10ff, 13-179f
- Differences between ?GNAME and ?GRNAME, 4-12

?GNFN system call, 4-1, 13-181ff
- Packet contents, 13-182
- Packet structure, 13-182

?GNUM offset, 13-208f, 13-213
?GOPEN system call, 5-1, 5-4, 5-7, 13-8, 13-159, 13-185ff, 13-397, 13-446, 13-448ff, 13-520
- Options, 13-188
- Packet contents,
  - IPC file packet, 13-186
  - Standard packet, 13-187
- Packet structure,
  - IPC file packet, 13-186
  - Standard packet, 13-187
- Sample packet, 13-189

?GPORT system call, 7-1, 7-4, 13-191f
?GPOS system call, 5-1, 5-6, 13-193ff
?GPRNM system call, 13-197f

Grant, Access, 2-8
?GRCH offset, 13-504
?GREQ offset, 13-208ff
?GRES offset, 13-208, 13-210f
?GRIH offset, 13-504
?GRNAME system call, 4-1, 4-12, 13-180, 13-199f
    Differences between ?GRNAME and ?GNAME, 4-12, 13-180
Growth stages, illustration of file, 4-3
GRP CLI macro, 13-99
?GRPH offset, 13-504
?GRRH offset, 13-504
?GSHPT system call, 2-1, 2-10, 13-201f, 13-598
?GSID system call, 11-5, 13-203f, 13-600
GSMGR (global synchronous manager) process, 9-2
?GSW offset, 13-208f, 13-213
?GTACP system call, 3-13, 4-1, 4-15, 13-205, 13-619
?GTMES system call, 11-7, 11-9, 13-207ff, 13-214
    Input parameters for offset ?GREQ, 13-210f
    Output from requests, 13-212f
    Packet contents, 13-209
    Packet structure, 13-208
    Request types, 13-208ff
        ?GARG, 13-208ff
        ?GCMD, 13-208ff
        ?GCNT, 13-208ff
        ?GMES, 13-208ff
        ?GSWS, 13-208ff
        ?GTSW, 13-208ff
    Sample program, 11-8
?GTNAM system call, 11-7, 13-215f, 13-224
?GTOD system call, 11-3, 13-217
?GTRUNCATE system call, 5-1, 5-7, 13-219ff
    Packet contents, 13-220
    Packet structure, 13-220
?GTSVL system call, 11-7, 13-223f
?GUNM system call, 3-1, 3-8, 13-225f
?GVPID system call, 11-7, 13-227

H

/H specifications string switch, 13-94
Handler, stack fault, 6-7f (See also, "Stacks".)

Hardware errors, 3-17
Hardware protection rings, 2-2 (See also, "Rings".)
Hash frame size, 13-140
HASP multileaving, 13-538
Header 1 labels, 5-29ff
Header 2 labels, 5-30f
Headers,
    ?IREC (See "?IREC system call".)
    ?IS.R (See "?IS.R system call".)
    ?ISEND (See "?ISEND system call".)
?HIBUF offset, 13-258
?HIEND offset, 13-258
Hierarchy,
    Illustration of process, 3-9
    System, 3-8
?HIST offset, 13-258
Histograms,
    Array structure, 13-681
    Array structure for 16-bit processes, 13-259
    Generating with ?WHIST, 3-16 (See also, "?WHIST system call".)
    Killing with ?KHIST
    Starting for 16-bit processes with ?IHIST, 13-257ff
    Starting with ?WHIST, 13-679ff
    Statistics, 13-681
    Terminating with ?KHIST, 3-16 (See also, "?KHIST system call".)
    Updated, 3-16
?HIWDS offset, 13-258
?HNAME system call, 11-7, 13-117ff
Holding, unholding, canceling queue requests, 13-117ff
Host identifier (host ID),
    Getting, 13-229ff
    Translating virtual PID into PID and, 11-7
    Translating with PID into virtual PID, 11-7
Host information, 11-7
Host name, getting, 13-229ff
Hosts,
    Determining whether pathnames reference remote, 13-493f
    Local, 11-5
    Remote, 11-5
        Detecting references to with ?RNAME, 11-7
How AOS/VS allocates disk space, controlling, 4-19
?HWBUF offset, 13-680
?HWEND offset, 13-680
?HWST offset, 13-680
?HWWDS offset, 13-680

I

?IBAD offset, 13-30, 13-194, 13-359, 13-366, 13-456, 13-459
?IBIN flag, 13-330, 13-461
IBM format, 5-22, 5-28, 5-31f
?ICH offset, 13-29f, 13-194, 13-358ff, 13-456, 13-458
?IDEF system call, 10-1f, 10-7, 10-11, 13-233ff, 13-265, 13-281
    Options, 10-3f, 13-235ff
        Burst multiplexor channel (BMC), 10-3f, 13-235ff
        Data channel (DCH) maps A through D, 10-3f, 13-235ff
        Neither BMC nor DCH, 10-3
    Contents of map definition table entry, 13-237
    Illustration of map definition table structure, 13-326
    Revoking previous ?IDEFs, 13-281f
    Sample packet, 13-239
?IDEL offset, 13-30, 13-194, 1-359, 13-363, 13-367, 13-456, 13-460
Identifiers,
    Process, 3-6f
    System (See "System identifier".)
    Task, 6-5
    Volume, 5-22
Identifying
    connections in inner rings, 8-6
    PID associated with global port number (with ?GPORT), 7-4 (See also, "Global port numbers".)
    system with ?GSID, 11-5
?IDGOTO system call, 6-1, 6-13f, 13-241f
?IDKIL system call, 6-1, 6-13ff, 13-307f, 13-243f, 13-305
?IDPH offset, 7-3, 7-5ff, 13-284f, 13-290ff
?IDPN offset, 7-5ff, 8-6, 13-276f
?IDPRI system call, 6-1, 6-5, 6-10, 13-245f, 13-403
?IDRDY system call, 6-1, 6-5, 6-13, 13-87, 13-247f, 13-427
?IDSTAT system call, 6-1, 6-5, 13-249, 13-661
?IDSUS system call, 6-1, 6-5, 6-11ff, 13-87, 13-241, 13-409, 13-425, 13-473, 13-475
?IESS system call, 12-11, 13-253f
?IFBNK flag, 7-8f
?IFNP offset, 13-30, 13-194, 13-359, 13-363, 13-456, 13-460
?IFNSP flag, 7-8f
?IFPR flag, 7-8f
?IFPU system call, 6-1, 6-19, 13-255f
?IFRFM flag, 7-8
?IFRING flag, 7-8f
?IFSOV flag, 7-8f
?IFSTM flag, 7-8

?IHIST system call, 13-257ff
    Array structure, 13-259
    Packet structure, 13-258
?ILKUP system call, 7-1, 7-3f, 13-285, 13-287f
Illustrations,
    Basic overlay area, 12-3
    BSC implementation, 9-14ff
    Control point directories (CPDs), 4-20 (See also, "Control point directories (CPDs)".)
    Device control table (DCT), 10-2
    Directory structure, 4-11
    Double connection, 8-3 (See also, "Connection-management facility".)
    Event logging format, 13-334
    File growth stages, 4-3
    Initializing a logical disk, 4-18
    Invalid return address from ?RCALL, 12-9
    ?IUFL offset structure (?IREC system call), 13-278
    ITB (end-of-intermediate-transmission block) receive buffer format, 13-594 (See also, "Data-link control characters (DLCC)".)
    Labels and data on labeled magnetic tape, 5-26
    ?LOGEV event logging format, 13-334
    Map definition table structure, 13-236
    Model customer/server configuration, 8-2 (See also, "Connection-management facility".)
    Multilevel customer/server configuration, 8-3
    Multiple overlay area, 12-4
    Passing procedure entry descriptor via the stack, 12-6
    Point-to-point/multipoint line configurations, 9-4 (See also, "Binary synchronous communications (BSC)".)
    Process hierarchy, 3-9
    Process names, 3-7
    Process spanning rings, 3-22
    Resource system call stack after ?RSAVE, 12-8
    Sample directory tree, 4-5
    Sample process tree, 3-12
    Segments and their protection rings, 2-3 (See also, "Rings".)
    Structure of map definition table, 10-5
    Structure of offset ?IUFL, 7-10
    Task states, 6-11 (See also, "Tasks".)
    Working sets in memory, 2-5
?ILTH offset, 7-5ff, 13-276f, 13-284f, 13-290f
Image,
    Memory (See "Memory image".)
    Process, 6-8
    Segment (See "Segment image".)

?IMERGE system call, 7-1, 7-4, 13-263f, 13-288
?IMRS offset, 13-30, 13-194, 13-359, 13-362, 13-368, 13-456, 13-459
?IMSG system call, 10-1, 10-9, 13-234f, 13-265f, 13-299f
Inclusive-OR operation, 10-8
Increasing or decreasing number of unshared pages in Ring 7, 2-10
    (See "?MEMI system call".)
Incrementing use count and reading shared page (See "?SPAGE system
    call".)
Index levels, 13-141
Indexes,
    Definition of, 4-2
    File's requirements for, 4-4
Indicator, monitor ring, 5-16f
Indirection, levels of, 3-17
Ineligible process state, 3-13
Influencing task scheduling, 6-6 (See also, "Tasks".)
Information,
    Current operating system, 13-565ff
    Directory (See "?FSTAT system call".)
    EXEC status, 13-119f
    File (See "?FSTAT system call".)
    File descriptor, 5-6
    File status, 13-137ff
    Flushing file descriptor, 13-667f
    Host, 11-7
    Previous frames in stack, 13-675f
    Process, 3-15, 11-5 (See also, "?PROC system call".)
    Process status, 13-429ff
    Queue, 11-5
    System, 11-5
    System calls that describe program, 11-5
    User console or batch process (See "?LOGEV system call".)
?INIT system call, 4-1, 4-17, 13-267ff, 13-479
    Packet contents, 13-268
Initial (operator) process (PID 2), 3-8f, 5-14, 5-16
Initial IPC messages,
    Accessing with ?GTMES, 11-7
    Definition of, 11-7
Initial stacks, specifying size of, 6-8f (See also, "Stacks".)
Initial task, 6-6 (See also, "Tasks".)
INITIALIZE command (CLI), 4-17, 13-269
Initialized logical disks, releasing with ?RELEASE, 13-479f
Initializing
    extended state save (ESS) area with ?IESS, 13-253f
    floating-point status register with ?IFPU, 6-19
    floating-point unit with ?IFPU, 13-255f
    logical disks with ?INIT, 13-267ff
        Illustration of, 4-18

Initiating tasks, 6-6, 13-627ff (See also, "Tasks".)
Inner rings, 3-20f, 6-16
    Breaking a connection in, 13-85f
    Identifying connections in, 8-6
    Task-redirection protection for, 6-13ff (See also, "Tasks".)
    Servers, 2-7f, 6-16
    Stacks, 6-8ff
        Forcing AOS/VS to initialize common, 6-9
Inner-ring connection management, 8-6ff
    Flag bits, 8-7f
        ?CXBBMO 8-8
        ?CXBVED, 8-8
        ?CXMBM, 8-7
        ?CXMED, 8-7
    Identifying connections in inner rings, 8-6
@INPUT generic filename, 5-11f (See also, "Generic Files".)
Input line, erasing from console with CTRL-U, 5-20 (See also, "Control characters".)
Input parameters for offset ?GREQ (?GTMES system call), 13-210f
Input/output (I/O), 13-321, 13-325, 13-357, 13-581ff
    Assigning device to process for record, 5-1
    Block, 5-2f, 5-6f, 13-445ff
        Definition of, 5-6
        System calls, 5-3
    Concepts of file, 5-2ff
    Data channel (See "Data channel ?IDEF option".)
    Device, 10-10
    Devices, 4-10, 5-2
    Differences between physical block I/O and block I/O, 5-7f (See also, "Block input/output (I/O)".)
    File (See "File input/output (I/O)".)
    Instructions, 3-17
    Managing character I/O with PMGR, 3-8
    Operation sequence for file, 5-4ff
    Packet extensions, 13-358
    Physical block, 5-2, 5-7f (See also, "Physical block I/O".)
    Record, 5-8f, 13-455ff (See also, "Record input/output (I/O)".)
        System calls, 5-3
Instructions,
    Assembly language,
        DIA, 5-8
        DIB, 5-8
        DIC, 5-8
    EJSR, 12-9
    ELEF, 12-9
    I/O, 3-17
    ISZ, 13-301, 13-441
    JSR, 12-9

Instructions (Cont.)
    LCALL, 6-16
    LEF, 12-9
    POPJ, 12-9
    Privileged, 3-17
    PSHJ, 12-9
    PSHR, 12-9
    RTN, 12-5, 12-7
    Wide-save, 6-15
    WRTN, 13-305
    XOP, 12-9
Interfaces,
    Establishing between AOS/VS and unsupported device (See "?IDEF system call".)
    Utility, 11-6
Interfacing to file editor (FED) utility with ?FEDFUNC, 13-129ff (See also, "?FEDFUNC system call".)
Interpreting ring fields (within global port number) with ?IREC, 7-4 (See also, "Global port numbers".)
Interprocess communications (IPC) facility, 3-17, 6-5, Chapter 7, 11-7, 13-93, 13-192
    Block I/O packet contents, 13-186
    Block I/O packet structure, 13-186
    Connection status messages, 8-6 (See also, "Connection-management facility".)
    Contents of send and receive headers, 7-7
    Contents of system flag word, 7-8
    Creating files with ?CREATE, 7-3
    Differences between fast interprocess synchronization and, 8-8f
    Entry packet, 13-41f
    File status information, 13-138
    File type ?FIPC, 5-5
    Files, creating with ?CREATE, 7-3
    Global port ?SPTM, 13-53f
    Illustration of user flag word structure,7-10
    Initial IPC messages, 11-7
    Looping messages, 7-8f
    Messages, 6-5, 8-6, 13-208, 13-214, 13-264
    Opening ports with ?ISEND, 13-283 (See also, "?ISEND system call".)
    Packets, 7-2
    Port entry file type, 4-6
    Process termination messages in customer/server relationship, 7-9ff (See also, "Connection-management facility".)
    Receiving messages with ?IREC, 13-275ff (See also, ?IREC system call".)
    Sample programs, 7-19ff

Interprocess communications (IPC) facility (Cont.)
    Send and receive headers, 7-5ff
        Contents of, 7-7
    Sending messages between ports, 7-2ff
    Structure of send and receive headers,7-6
    System and user flags, 7-7ff
    System calls, 7-1 (See individual system call entries for additional references.)
            Communicating with customer via, 8-4
            Privilege to issue, 3-10
        ?GCPN, 7-1
        ?GPORT, 7-1
        ?ILKUP, 7-1
        ?IMERGE, 7-1
        ?IREC, 7-1
        ?IS.R, 7-1
        ?ISEND, 7-1
        ?ISPLIT, 7-1
        ?TPORT, 7-1
    Termination messages, 13-638
    Typical system call sequence, 7-4f
    User flag word, 7-9
    Using as communications device, 5-21f
Interprocess synchronization, fast, 8-8f (See also, "Fast interprocess synchronization".)
Interrupt sequences,
    Disabling with ?KIOFF, 13-311
    Re-enabling with ?KION, 13-313
    Simulating keyboard, 13-309f
    Waiting for with ?KWAIT, 13-315f
Interrupt service mask, 10-2
Interrupt service messages, receiving with ?IMSG, 13-265f
Interrupt service routines, 10-9
    Communicating from, 10-9
    Exiting from with ?IXIT, 13-297f
    Transmitting messages from, 13-299f
Interrupt vector table, 10-7
    Removing user devices from, 13-281f
Interrupts,
    Console (See "Console interrupts".)
    Device, 10-9
    Disabling console, 13-353f
    Enabling console, 13-355
    Generating and aborting process with CTRL-C CTRL-B, 5-21 (See also, "Control sequences".)
    Generating CTRL-C CTRL-A, 5-21 (See also, "Control sequences".)
    Reverse (See "Reverse interrupt (RVI)".)
Intertask communications facility, 6-17f (See also, "Tasks".)

Intertask messages,
  Receiving with ?REC, 13-473f
  Receiving without waiting, 13-475f
  Sending and waiting for receipt with ?XMTW, 13-695f
  Sending with ?XMT, 13-693f
Introducing user-defined devices to AOS/VS at execution time, 10-2
?INTWT system call, 11-2f, 13-242, 13-271f, 13-353, 13-355
Invalid return address from ?RCALL, illustration of, 12-9
Investigating prior state of rescheduling, 13-77
?IOPH offset, 7-4ff, 8-6, 13-276ff, 13-284
?IOPL offset, 8-6
?IOPN offset, 7-5ff, 13-290ff
IPC (interprocess communications) facility (See "Interprocess communications (IPC) facility".)
?IPKL parameter, 13-29f
?IPLTH (length of ?ISEND and ?IREC headers), 7-5
?IPRLTH (length of ?IS.R header), 7-5
?IPTL word, 8-6
?IPTR offset, 7-5, 7-7, 7-11, 13-276f, 13-284f, 13-290ff
?IQTSK system call, 6-1, 6-7, 13-81f, 13-273f, 13-631
?IRCL offset, 13-30, 13-194, 13-359, 13-362, 13-366, 13-373, 13-456, 13-459
?IREC system call, 3-15, 5-21, 7-1ff, 7-5, 7-8f, 8-4, 8-6, 13-34, 13-54, 13-262, 13-275ff, 13-292, 13-350, 13-488, 13-691
  Header, 7-6ff, 7-11
    Contents, 13-277
    Structure, 13-276
  Illustration of ?IUFL offset structure, 13-278
  Logic of, 7-14, 7-17f
  Sample header, 13-279
?IRES offset, 5-24, 13-30, 13-194, 13-359, 13-456, 13-459
?IRLR offset, 13-30, 13-194, 13-359, 13-363, 13-456, 13-460
?IRLT offset, 7-6f, 13-290ff
?IRMV system call, 10-1, 13-281f
?IRNH offset, 13-30, 13-193f, 13-359, 13-363, 13-456, 13-460
?IRNW offset, 13-30, 13-359, 13-363, 13-456, 13-460
?IRPT offset, 7-6f, 13-290ff
?IRSV offset, 7-6f, 13-290ff

?IS.R system call, 3-10, 7-1ff, 7-5, 8-4, 13-34, 13-209ff, 13-419, 13-563, 13-649
    Header, 7-6f
    Sample header, 13-292
?ISEND system call, 3-10, 5-22, 7-1ff, 7-5, 7-8, 13-34, 13-262, 13-278, 13-283ff, 13-289, 13-292, 13-419, 13-561 (See also, "Interprocess communications (IPC) facility".)
    Header, 7-6ff, 7-11
        Contents, 13-284
        Structure, 13-284
    Logic of, 7-14, 7-17f
    Sample header, 13-285
?ISFL offset, 7-6f, 13-276ff, 13-284, 13-290ff
?ISPLIT system call, 7-1, 7-4, 13-287f
?ISTI offset, 5-33, 13-30, 13-194, 13-358ff, 13-363ff, 13-456ff, 13-466
    ?APND mask, 5-6
?ISTO offset, 5-5, 13-30, 13-194, 13-359, 13-361f, 13-366, 13-456, 13-459
ISZ instruction, 13-301, 13-441
ITB (end-of-intermediate-transmission block), 9-9, 13-539, 13-541, 13-594, 13-604f (See also, "Data-link control characters (DLCC)".)
    Illustration of receive buffer format, 13-594 (See also, "Data-link control characters (DLCC)".)
?ITIME system call, 11-3, 13-295
?IUFL offset, 7-6f, 7-9, 7-12, 8-6, 13-54, 13-276ff, 13-284, 13-290ff
    Illustration of structure, 7-10
    Process termination codes for ?IREC and ?ISEND headers, 7-11
?IXIT system call, 10-1, 10-8f, 13-266, 13-299
?IXMT system call, 13-234f, 13-265f, 13-297f, 13-299f

J

JSR instruction, 12-9

K

?KCALL system call, 12-4f, 12-7ff, 13-301f, 13-383, 13-442f, 13-676
Keeping track of file elements, 4-2
Kernel, monitoring, 2-4, 13-327f (See also, "?LMAP system call".)
Key, CTRL, 5-19
Keyboard interrupt sequences,
  Disabling with ?KIOFF, 13-311
  Re-enabling with ?KION, 13-313
  Simulating with ?KINTR, 13-309f
  Waiting for with ?KWAIT, 13-315f
?KHIST system call, 3-1, 13-303
?KILAD system call, 6-1, 6-15, 13-243, 13-305f
?KILL system call, 6-1, 6-15, 13-77, 13-101, 13-243, 13-305ff
Kill-processing routines,
  Defining with ?KILAD, 13-305ff
  ?UKIL, 6-15f, 13-243, 13-306f, 13-409
Killing
  calling task, 13-307f
  calling task and exiting from overlay, 13-379f
  histograms with ?KHIST, 13-303 (See also, "?KHIST system call".)
  tasks, 6-15 (See also, "Tasks".)
  tasks of specified priority with ?PRKIL, 13-409f
  tasks specified by TIDs with ?IDKIL, 13-243f
?KINTR system call, 11-3, 13-309f
?KIOFF system call, 11-3, 13-311
?KION system call, 11-3, 13-313
?KWAIT system call, 11-3, 13-315f

K

LABEL CLI utility, 13-319
?LABEL system call, 5-1, 5-22, 5-24, 13-317ff, 13-372
  Packet contents, 13-319
  Packet structure, 13-318
  Sample packet, 13-320
Label types for labeled magnetic tapes, 5-24
LABEL utility (CLI), 5-22, 13-372
Labeled magnetic tapes, 5-10, 5-22ff, 13-582 (See also, "Devices".)
  Advantages of, 5-22
  ANSI format, 5-24
  AOS format, 5-24

Labeled magnetic tapes (Cont.)
- Checking volume ID, 13-27f
- Contents of extended ?OPEN packet, 13-371ff
- Creating labels for, 13-317ff
- Definition of, 5-22
- File I/O on, 5-35
- File type of generic, 4-6
- Forcing end-of-volume with ?FEOV, 13-133f
- Formats, 5-23
- IBM format, 5-24
- Illustration of labels and data, 5-26
- Label types, 5-24ff (See also, "Labels".)
  - End-of-file labels, 5-25ff
  - End-of-volume labels, 5-25ff
  - File header labels, 5-24, 5-26f
  - Volume labels, 5-24ff
- Labeling levels, 5-23
- Mounting, 13-106ff
- Mounting explicitly with CLI MOUNT command, 5-35f
- Mounting implicitly with ?OPEN system call, 5-35f
- ?OPEN system call packet extension for, 5-34
- Structure of ?EXEC packet for ?XFDUN, 13-109
- Structure of ?EXEC packet for ?XFMLT, 13-108
- Structure of ?EXEC packet for ?XFXML (extended), 13-108
- Structure of extended ?OPEN packet, 13-370
- Volume sets, 5-23

Labeling levels for magnetic tapes, 5-23
Labeling magnetic tape with ?LABEL, 13-317ff
Labels,
- End-of-volume (EOV), end-of-file (EOF), 5-25, 5-27, 5-34
- File header, 5-29ff (See also, "Labeled magnetic tape".)
- File trailer, 5-25
- User header (UHL), 5-34
- User trailer (UTL), 5-27, 5-34
- User volume (UVL), 5-27f

Labels and data on labeled magnetic tape, illustration of, 5-26
.LB files, 4-9 (See also, "Files".)
?LBAC offset, 13-318f
?LBDV offset, 13-318f
?LBFG offset, 13-317ff
?LBOI offset, 13-318f
?LBST offset, 13-318f
?LBUV offset, 13-318f
?LBVD offset, 13-318f
LCALL instruction, 6-16
LD (See "Logical disks (LDs".)
Least recently used (LRU) chain (See "LRU chain".)

Leaving, entering, or examining Superprocess mode with ?SUPROC, 13-619f (See also, "Superprocess mode".)
Leaving, entering, or examining Superuser mode with ?SUSER, 13-623f (See also, "Superuser mode".)
LEF (load-effective address) mode (See "Load-effective address (LEF) mode".)
?LEFD system call, 10-1, 10-10f, 13-321ff (See also, "Load-effective address (LEF) mode".)
?LEFE system call, 10-1, 10-10f, 13-69, 13-323f (See also, "Load-effective address (LEF) mode".)
?LEFS system call, 10-1, 10-10f, 13-325 (See also, "Load-effective address (LEF) mode".)
Legal filename characters, 4-8
Length of ?ISEND and ?IREC headers (?IPLTH), 7-5
Length of blocks, 5-7
Levels, index, 13-141
Levels of indirection, 3-17
?LFOP flag word, 13-330
Library, user runtime (URT32.LB), 6-6
Library files, 4-9 (See also, "Files".)
Lifting task suspensions, 13-241f
Limit, stack, 6-8f
Line configurations, binary synchronous communications (BSC), 9-3ff (See also, "Binary synchronous communications (BSC)".)
Line printers, 10-1 (See also, "User device support".)
    Data channel, 5-10
    File type of data channel, 4-7
    Format control, 5-19
    Tailoring output format, 5-19
Line selection and polling, multipoint, 9-4ff
Lines,
    Dedicated communications, 9-1
    Enabling binary synchronous communications (BSC), 9-2, 13-536ff (See also, "Binary synchronous communications (BSC)".)
    Erasing current input from console with CTRL-U, 5-20 (See also, "Control characters".)
    File type of synchronous communications, 4-7
    Sending data over enabled BSC, 9-2
    Switched communications, 9-1
Link entries, 4-12f
    Creating and deleting with ?CREATE and ?DELETE, 4-13
    Definition of, 4-12
    Finding out what a link entry represents, 4-13
    Getting contents of with ?GLINK, 13-173f
Link files (file type), 4-6
Link number, 5-13

Link utility, 2-7, 3-2, 4-7, 6-6f
Link-to-link references, 4-12f, 12-2ff
Linking
  object modules to form program file, 12-4
  programs together with ?CHAIN, 3-19f
List, search (See "Search list".)
@LIST generic file, 3-19, 5-11f (See also, "Generic files".)
Listing
  current shared partition size with ?GSHPT, 13-201f
  current unshared memory parameters with ?MEM, 13-347f
  directory entries, 13-181ff
Lists,
  Access control (See "Access control list (ACL)".)
  Polling (See "Polling".)
  Search (See "Search list".)
Literal, accepting next character as with CTRL-P, 5-20 (See also, "Control characters".)
?LMAP system call, 2-1, 2-4, 13-327f
@LMT, 5-10, 5-35f (See also, "Devices".)
Load-effective address (LEF) mode, 3-17, 10-7f, 10-10f, 13-69, 13-321ff
  Disabling with ?LEFD, 13-69, 13-321f
  Enabling with ?LEFE, 13-321f
  Instructions, 12-9
  Returning current LEF mode status with ?LEFS, 13-325f
Loading and going to overlay with ?OVLOD, 13-381ff
Loading lower rings, preventing ?RINGLD system call from, 13-499f
Loading program files into specific rings with ?RINGLD, 3-20, 13-491f (See also, "?RINGLD system call".)
Local hosts, 11-5, 13-13f
Local port numbers, 4-4, 6-5
  Translating to global equivalent with ?TPORT, 7-2, 13-649
  Definition of, 7-2 (See also, "Interprocess communications (IPC) facility".)
  Range, 13-42f
Local root, 4-17, 4-20
Local servers, 2-7, 6-16
  Using common local servers to pend/unpend tasks (See "Fast interprocess synchronization".)
Locking/unlocking critical regions, 6-18f (See also, "Tasks".)
Log file, system (See "System log file".)
?LOGCALLS system call, 13-329ff
?LOGEV system call, 11-5, 13-333ff, 13-626
  Restrictions, 13-334
Logging
  messages into system log file with ?LOGEV, 11-5
  system calls, 13-329ff

Logic,
    ?IREC system call, 7-14, 7-17f
    ?ISEND system call, 7-14, 7-17f
Logical address, validating with ?VALAD, 13-669f
Logical address space, 2-1, 2-4, 3-2f, 3-17
    Changing number of unshared pages in, 13-349f
    Definition of, 2-1
    Moving bytes to/from customer's, 8-4
    Sixteen-bit programs, 12-2
    User-visible, 3-3
Logical connection between customer and existing server, establishing, 8-2 (See also, "Connection-management facility".)
Logical context, 2-1f
Logical disks (LDs), 4-17, 4-19f, 13-481, 13-565
    Definition of, 4-17
    File type, 4-6
    Illustration of initialization, 4-18
    Initializing with ?INIT, 13-267ff
    Master, 4-17
    Releasing with ?RELEASE, 4-18
    Root, 4-19f
Looping IPC messages, 7-8f
Lower rings,
    Mapping, 13-327f (See also, "?LMAP system call".)
    Stopping from being ringloaded (See "?RNGST system call".)
@LPB, 5-10 (See also, "Devices".)
LRU (least recently used) chain, 2-6, 13-501, 13-577
?LSTART flag word, 13-330

M

Macroassembler (MASM) (See "MASM (macroassembler)".)
Macros,
    CODE, 13-99
    GRP, 13-99
Magnetic tape drives (See "Magnetic tape units".)
Magnetic tape units, 10-1 (See also, "User device support".)
    Opening, 5-37
Magnetic tape units, managing with EXEC utility, 11-6

Magnetic tapes, 5-2, 13-368ff
  Controllers, 5-10 (See also, "Devices".)
  Dismounting, 13-109ff
  File type of, 4-6f
  File type of generic labeled, 4-6
  Labeled, 5-10, 5-22ff, 13-193, 13-369ff (See also, "Labeled magnetic tapes" and "Devices".)
  Opening magnetic tape unit for use with, 5-37
  Performing block I/O on, 13-448f
  Unlabeled,
    Mounting, 13-106ff
    Structure of ?EXEC packet for ?XFMUN, 13-107
    Structure of ?EXEC packet for ?XFXUN (extended), 13-107
Mailboxes, 6-17f, 13-473ff, 13-693, 13-695 (See also, "Tasks".)
Maintaining and creating files, Chapter 4
Manager, queued task, 6-7 (See also, "Tasks" and "Queued tasks".)
Managing
  and creating files, sample programs, 4-21ff (See also, "Files".)
  character I/O with PMGR, 3-8
  customer/server connections, Chapter 8 (See also, "Connection-management facility".)
  exchanges between customers and servers, 8-2
  multiprocessing environment, 3-5 (See also, "Processes".)
  queues and magnetic tape units with EXEC utility, 11-6
Manipulating system log file with ?SYLOG, 13-625f (See also, "System log file".)
Map, setting data channel, 13-611f
Map definition table, 13-235ff (See also, ?IDEF system call".)
  Contents of entry, 13-237
  Illustration of structure, 13-236
  Structure of, 10-5f
Map slots, 10-3ff
  Defining, 13-235 (See also, "?IDEF system call".)
Mapping lower rings, 13-327f (See also, "?LMAP system call".)
Mask, interrupt service, 10-2
Mask ?APND (in offset ?ISTI), 5-6
Masking process privileges, 13-418
Masks,
  Record format (See "Record format masks".)
  ACL specifications, 4-15 (See also, "Access control list (ACL)".)
MASM (macroassembler), 2-6f
Master logical disks (LDs), 4-17
Maximum CPU time for processes, setting, 13-420
Maximum size for control point directory, 13-37f

Maximum space (MS), 4-19f (See also, "Control point directories (CPDs)".)
?MBAH offset, 13-338, 13-342
?MBBC offset, 13-338, 13-342
?MBCH offset, 13-338, 13-342
?MBFC system call (See "?MBFC/?MBTC system call".)
?MBFC/?MBTC system call, 8-1, 8-4, 13-25, 13-337ff, 13-341ff
    Packet structure, 13-338, 13-342
    Sample packet, 13-339
?MBID offset, 13-338, 13-342
?MBTC system call (See "?MBFC/?MBTC system call".)
?MBTC/?MBFC system call ((See "?MBFC/?MBTC system call".)
@MCA, 5-10 (See also, "Devices".)
MCA (Multiprocessor communications adapter) (See "Multiprocessor communications adapters (MCAs)".)
@MCA1, 5-10 (See also, "Devices".)
MCAs, performing block I/O on, 13-449f
?MCOBIT bit mask, 13-33f
?MCPID bit mask, 13-33
?MCRNG bit mask, 13-33
?MDUMP system call, 3-1, 3-18, 13-345f
Mechanisms, protection (See "Protection mechanisms".)
?MEM system call, 2-1, 2-10, 13-347f
?MEMI system call, 2-1, 2-9f, 13-348ff
Memory, Chapter 2, 3-3f
    Illustration of working sets in, 2-5
    Moving logical pages on demand from disk to (See "Demand paging".)
    NREL (normal relocatable), Defining partitions in, 2-7
Memory and process sample programs, 3-23ff
Memory contention, 3-15
Memory descriptor structure for ?PSTAT system call, 13-434
Memory dumps, 3-17f
Memory image, dumping from specified ring to file with ?MDUMP, 13-345f
Memory management, Chapter 2
    Definition of terms, 2-1f
    Logical address space, 2-1 (See also, "Logical address space".)
    Logical context, 2-1 (See also, "Logical context".)
    Shared page, 2-2 (See also, "Shared page".)
    System calls, 2-1 (See individual system call entries for additional references.)
        ?ESFF, 2-1
        ?GMEM, 2-1
        ?GSHPT, 2-1
        ?LMAP, 2-1
        ?MEM, 2-1
        ?MEMI, 2-1

Memory management (Cont.)
    System calls (Cont.)
        ?PMTPF, 2-1
        ?RPAGE, 2-1
        ?SCLOSE, 2-1
        ?SOPEN, 2-1
        ?SOPPF, 2-1
        ?SPAGE, 2-1
        ?SSHPT, 2-1
    Unshared page, 2-2 (See also, "Unshared page".)
    Unused page, 2-2 (See also, "Unused page".)
    Working set, 2-2 (See also, "Working set".)
Memory modification with disk images, 12-2
Memory organization, Chapter 2
Memory pages,
    Flushing to disk (shared file), 13-103f (See also "Pages".)
    Getting number of undedicated, 13-177f
Memory parameters,
    Listing current unshared with ?MEM, 13-347f
    Saving state of, 3-17
Memory-resident processes, 2-2
/MES=message specifications string switch, 13-94
Messages,
    16-bit process termination, 7-10ff
    32-bit process termination, 7-14ff
    ?TRAP termination for 16-bit processes, 7-13
    Broadcast option, 13-693, 13-696
    Broadcasting with ?XMT and ?XMTW, 6-18
    CLI, 11-7 (See also, "CLI messages".)
    Error (See "Error Codes".)
    Format for sending with ?SEND, 13-550
    Format for task, 13-652
    Interrupt service (See "Interrupt service messages".)
    IPC, 6-5, 8-6 (See also, "Interprocess communications (IPC facility".)
        Initial, 11-7
        Sending between IPC ports, 7-2ff
        Sending IPC to itself, 7-8f
    IPC connection status, 8-6 (See also, "Connection-management facility".)
    Length of 32-bit termination, 7-14
    Looping IPC, 7-8f
    Message-manager task, 13-651f
    Obituary, 7-9, 8-5f, 13-34, 13-53f, 13-27
        Receiving with ?IREC, 8-6
    Operator, 13-106, 13-110
    Passing from console to individual tasks, 6-17 (See also, "Tasks".)

Messages (Cont.)
- Passing termination, 13-487ff
- Process termination in customer/server relationship, 7-9ff
- Receiving intertask, 13-473f
- Receiving intertask without waiting, 13-475f
- Sending from process console to task, 13-651ff
- Sending intertask and waiting for receipt with ?XMTW, 13-695f
- Sending intertask with ?XMT, 13-693f
- Sending to consoles with ?SEND, 13-549ff
- Specifications string, 13-94f
- Transmitting from interrupt service routines with ?IXMT, 13-299f

Minus sign (-) template, 4-16 (See also, "Access control list (ACL)".)

Miscellaneous system calls, Chapter 11 (See individual system call entries for additional references.)
- ?BNAME, 11-1
- ?CDAY, 11-1
- ?CTOD, 11-1
- ?DEBUG, 11-1
- ?ENQUE, 11-1
- ?ERMSG, 11-1
- ?EXEC, 11-1
- ?FDAY, 11-1
- ?FEDFUNC, 11-1
- ?FTOD, 11-1
- ?GBIAS, 11-1
- ?GDAY, 11-1
- ?GHRZ, 11-1
- ?GSID, 11-1
- ?GTMES, 11-1
- ?GTNAM, 11-1
- ?GTOD, 11-1
- ?GTSVL, 11-1
- ?GVPID, 11-1
- ?HNAME, 11-1
- ?INTWT, 11-1
- ?ITIME, 11-1
- ?KINTR, 11-1
- ?KIOFF, 11-1
- ?KION, 11-1
- ?KWAIT, 11-1
- ?LOGCALLS, 11-1
- ?LOGEV, 11-1
- ?ODIS, 11-1
- ?OEBL, 11-1
- ?RNAME, 11-1
- ?SBIAS, 11-1
- ?SDAY, 11-1

Miscellaneous system calls (Cont.)
    ?SINFO, 11-1
    ?STOD, 11-1
    ?TPID, 11-1
    ?VALAD, 11-1
    ?WDELAY, 11-1
Model, AOS/VS task-protection, 6-4f (See also, "Tasks".)
Model customer/server configuration, illustration of, 8-2 (See also, "Connection-management facility".)
Modems (full-duplex), 5-15ff
    Auto-answer, 5-15f
        Operating sequence, 5-15f
    Flags, 5-15
        CD, 5-15
        DSR, 5-15f
        DTR, 5-15f
        RTS, 5-15f
    Non-auto-answer, 5-15
        Operating sequence, 5-16f
Modes,
    Binary, 5-18f
    LEF (load effective address), 3-17, 10-7f, 10-10f, 13-69
    Page, 5-13
    Superprocess (See "Superprocess mode".)
    Superuser (See "Superuser mode".)
    Transparent text, 9-10
Modification of memory with disk images, 12-2
Modified pages, Flushing to disk, 2-8
Modified shared page, updating, 13-135f
Modifying ring field within global port number, 7-4, 13-263f (See also, "Interprocess communications (IPC) facility".)
Modularity, software, 2-2
Monitor ring indicator, 5-16f
Monitoring
    address range in calling process, 13-679ff
    address range in other process, 13-679ff
    kernel, 2-4, 13-327f (See also, "?LMAP system call".)
MOUNT command (CLI), 5-38
    Syntax of, 5-35
Mounting labeled magnetic tapes
    explicitly with CLI MOUNT command, 5-35f
    implicitly with ?OPEN system call, 5-35f
    with CLI DUMP command, 5-36
Mounting magnetic tapes, 13-106ff
Movable resources, 13-302
    Definition of, 13-163

Moving
    bytes from customer's buffer, 13-337ff
    bytes to customer's buffer, 13-341ff
    bytes to/from customer's logical address space with ?MBTC and ?MBFC, 8-4
    logical pages from disk to memory on demand (See "Demand paging".)
MS (maximum space), 4-19f (See also, "Control point directories (CPDs)".)
@MTBn, 5-10 (See also, "Devices".)
Multifile devices, 4-2
    Disks, 4-2
    Magnetic tape, 4-2
Multileaving, HASP, 13-538
Multilevel connection,
    Definition of, 8-2
    Illustration of, 8-3
Multiple overlay area, illustration of, 12-4 (See also, "Sixteen-bit processes".)
Multiple rings, servers concurrently connected to within customer, 8-6
Multiplexor, Asynchronous Line, 5-10 (See also, "Devices".)
Multipoint control stations, 13-536f, 13-591f, 13-607ff (See also, "?SRCV system call and ?SSND system call".)
Multipoint lines, 13-589
    Selection and polling, 9-4ff
Multipoint tributaries, 13-536, 13-592f (See also, "?SRCV system call".)
Multipoint/point-to-point line configurations, Illustration of, 9-4
Multiprocessing, 3-5
Multiprocessor communications adapters (MCAs), 5-12f
    Controllers, 5-10
    Pathnames, 5-13
    Performing block I/O on, 13-449f (See also, "Block input/output (I/O)".)
    Protocol, 5-12f
Multiprocessor communications unit (file type), 4-7
Multitasking, Chapter 6
    Advantages of, 6-3f
    Disrupting with ?DRSCH, 13-87f
    Enabling scheduling with ?ERSCH, 13-101f
    Sample programs, 6-19ff
MV/8000 floating-point registers, 6-19
?MYTID system call, 6-1, 6-5, 13-351

N

NAK (negative acknowledgment), 9-8, 9-11, 13-540f, 13-557, 13-588, 13-590, 13-608, (See also, "Data-link control characters (DLCC)".)
Names,
- Device, 5-9f (See also, "Devices".)
- Determining form, 13-117
- Directory, 4-4
- Full process, 3-6f, 13-226
- Getting full process, 13-393f
- Getting process (See "?PNAME system call".)
- Illustration of process, 3-7
- Simple process, 3-7f

Negative acknowledgment (NAK), 9-8, 9-11, 13-540f, 13-557, 13-588, 13-590, 13-608, (See also, "Data-link control characters (DLCC)".)
New access control list (ACL), setting with ?SACL, 13-507f (See also, "Access control list (ACL)".)
New procedure, chaining to with ?RCHAIN, 13-443f
New process, passing control to, 13-23ff
Next character, accepting as literal with CTRL-P, 5-20 (See also, "Control characters".)
?NFKY offset, 13-182
?NFLN offset, 13-182
?NFNM offset, 13-182
?NFRS offset, 13-182
?NFTP offset, 13-182
Non-ANSI-standard terminals, 13-158
Non-auto-answer modems, 5-15
- Operating sequence, 5-16f

NREL (nonrelocatable) memory, 2-7
.NREL pseudo-op, 2-6f
@NULL generic filename, 5-11f (See also, "Generic files".)
Number of undedicated pages, Returning (See "?GMEM system call".)
Number of unshared memory pages, Changing (See "?MEMI system call".)
Number of unshared pages in logical address space, changing, 13-349f
Numbers,
- Channel, 4-10
- Interpreting ring fields within global port, 7-4 (See also, "Interprocess communications (IPC) facility".)
- Link, 5-13
- Local port, 4-4
- Process priority, 3-6
- Ring, 7-2f
- Task Priority, 6-5

O

?OANS flag (?IRES offset of ?OPEN), 5-24
.OB files, 4-9 (See also, "Files".)
Obituary messages, 7-9, 8-5f, 13-34, 13-53f, 13-278
    Origin port number for (?SPTM), 7-9
    Receiving with ?IREC, 8-6
    Suppressing with bit ?COBIT, 8-5
Object files, 4-9 (See also, "Files".)
Object modules, linking to form program file, 12-4
Obtaining EXEC status information, 13-119f
?ODIS system call, 13-353ff
?OEBL system call, 11-2f, 13-271, 13-353ff
?OFCR bit (Creation option of ?OPEN system call), 5-21
?OIBM flag (?IRES offset of ?OPEN), 5-24
.OL (overlay) file, 12-2
?OPCH offset, 13-186, 13-188
?OPEH offset, 13-187
?OPEN system call, 5-1, 5-4ff, 5-15f, 5-18, 5-21f, 5-24, 5-35f, 13-8, 13-29ff, 13-48, 13-67, 13-98, 13-144, 13-193f, 13-317, 13-357ff, 13-520, 13-526, 13-582
    Extended packet,
        Contents, 13-371f
        Structure, 13-370
        Using for labeled magnetic tapes, 5-34
    File types you can create with, 5-5, 13-365
    Packet,
        Contents, 13-260ff
        Structure, 13-359
    Sample packet, 13-374ff
Opener, first, 2-8
Opening
    channels, 13-357ff
    device for exclusive use of one process, 5-18
    devices for I/O, 13-357ff
    files, 13-357ff
    file for block I/O with ?GOPEN, 13-185ff
    files for shared access, 2-6, 13-571f (See also, "?OPEN system call".)
    IPC ports for calling process with ?ISEND, 13-283 (See also, "Interprocess communications (IPC) facility".)
    magnetic tape unit, 5-37
    protected shared files, 2-7ff, 13-573ff (See also, "?SOPPF system call".)
    shared files, 2-7 (See also, "?SOPEN system call".)
    symbol table file using FED utility (See "?FEDFUNC system call".)

Operating sequence for
    Auto-answer modems, 5-15f
    Non-auto-answer modems, 5-16f
Operating system, getting information about current, 13-565ff
Operation, inclusive-OR, 10-8
Operation sequence for file input/output (I/O), 5-3ff
Operator (initial) process (PID 2), 3-8f, 5-14, 5-16, 13-514, 13-615, 13-625
Operator messages, 13-106, 13-110
?OPFC offset, 13-187
?OPFL offset, 13-187, 13-188
?OPPH offset, 13-186
Options,
    Append, 5-6
    Broadcast, 13-693, 13-696
    Creation (?OPEN system call), 5-4, 5-21
    Exclusive Open, 5-4f
    File creation, 13-257ff, 13-364f
    ?IDEF system call, 10-3 (See also, "?IDEF system call".)
?OPTY offset, 13-186f
Organization of memory, Chapter 2
Origin port number for obituary messages, ?SPTM, 7-9
Outer ring to inner ring, crossing from, 3-20
Output,
    Freezing to console with CTRL-S, 5-20 (See also, "Control characters".)
    Suppressing console with CTRL-O, 5-20 (See also, "Control characters".)
    Tailoring format of line-printer, 5-19
Output, spooled (See "Spooled output".)
@OUTPUT generic filename, 3-19, 5-11f (See also, "Generic files".)
Overlay use count (OUC), 13-377, 13-379, 13-386
Overlays, 13-163f
    Concepts of, 12-2ff
    Exiting from and killing calling task, 13-379f
    Loading and going to with ?OVLOD, 13-381ff
    .OL file, 12-2
    Primitive overlay system calls, 12-2, 12-10f (See individual system call entries for additional references".)
        ?OVEX, 12-10
        ?OVKIL, 12-10
        ?OVLOD, 12-10
        ?OVREL, 12-10
    Releasing and returning, 13-377f
    Releasing with ?OVREL, 13-385f
    Runtime relocatability requirements, 12-8f
    Use count (OUC), 12-10f (See also, "Overlay use count (OUC)".)

Overriding characteristics of character device, 5-14
Overriding process protections with ?SUPROC, 13-619f
?OVEX system call, 12-10f, 13-377f, 13-380, 13-382f, 13-386
?OVKIL system call, 12-10f, 13-378ff, 13-382f, 13-386
?OVLOD system call, 12-10, 13-377f, 13-381ff, 13-386
?OVREL system call, 12-10, 13-378, 13-380, 13-382, 13-385f
Owner (?FACO) access, 4-13ff, 4-17, 13-60, 13-206, 13-508f
Owner Name field (labeled magnetic tapes), 5-28

P

/P specifications string switch, 13-94
Packet contents (See also, individual system call entries.)
    ?EXEC packet for queue requests, 13-112ff, 13-118
    ?EXEC packet for status information, 13-120
    ?GNFN packet, 13-182
    ?GTMES packet, 13-209
    ?GTRUNCATE packet, 13-220
    ?OPEN extended packet for labeled magnetic tapes, 13-371f
    ?PSTAT system call packet, 13-431ff
    ?READ/?WRITE system call extended packet for selected field translation, 13-468
    ?READ/?WRITE system call packet, 13-458ff
    ?READ/?WRITE system call packet for screen management primitives 13-464
    ?RNGPR system call packet, 13-496
    ?TASK system call packet, 13-629
Packet structure (See individual system call entries.)
Packets,
    Extended task definition, 13-82
    File specification, 5-4
    Interprocess communications (IPC) facility, 7-2
        Entry, 13-41f
    System call (See individual system call entries.)
Page mode, Displaying next page in, 5-13
Page-fault condition, 3-3
Pages,
    Changing number of unshared memory (See "?MEMI system call".)
    Flushing to disk shared file memory (See "?ESFF system call".)
    Getting number of undedicated memory, 13-177f

Pages (Cont.)
    Memory, 2-4, 2-6f
        Dedicated, 2-9
        Shared, 2-2, 2-4, 2-6f
            Checkpointing, 2-8f
            Illustration of, 2-5
        Undedicated, 2-9
        Unshared, 2-2, 2-4, 2-6f
            Write-protected (illegal), 2-4
        Unused, 2-9
    Moving from disk to memory on demand (See "Demand paging".)
    Permanently binding to working set (See "?WIRE system call".)
    Releasing permanently wired (See "?UNWIRE system call".)
    Releasing shared and decrementing use counts (See "?RPAGE system call".)
    Returning current number of undedicated (See "?GMEM system call".)
    Shared (See "Shared pages".)
    Wired (See "Wired pages".)
    Wiring to working set, 13-683f (See also, "?WIRE system call".)
    Write-protected, 2-1
Paging, Demand, 2-4
Parameters,
    Listing current unshared memory (See "?MEM system call".)
    Process creation, 3-11
    Saving the state of memory, 3-17
    Steps AOS/VS takes to check process creation, 3-11
    Working-set (See "Working set".)
Parameters, listing current unshared memory, 13-347f
.PART pseudo-op, 2-6f
Partition,
    Establishing size of new shared (See "?SSHPT system call".)
    Listing current size of shared (See "?GSHPT system call".)
    Shared (See "Shared partitions".)
Partition size, listing current shared, 13-201f
PARU.32, 4-4, 13-206
Passing
    Control to new process, 13-23ff (See also, "?CHAIN system call".)
    Customer/server connection to another server in Ring 7 with ?PCNX, 8-4, 13-387f
    Customer/server connection to another server with ?PRCNX, 8-4, 13-395f
    Messages from console to individual tasks, 6-17 (See also, "Tasks".)
    Procedure entry descriptor via the stack, Illustration of, 12-6
    Superprocess privilege to sons, 3-13
    Superuser privilege to sons, 3-12
    Termination message to father and terminating calling process with ?RETURN, 13-487ff

Pathnames, 4-9ff
    At sign (@) prefix, 4-10
    Colon (:) prefix, 4-10
    Default break files, 3-19
    Definition of, 4-9
    Determining remote host references in, 13-493f
    Equal sign (=) prefix, 4-10
    Generic file sample, 5-11
    Getting complete for generic files, 13-199f
    Getting complete with ?GNAME, 4-12, 13-179f
    Getting program's, 13-197f
    Getting for generic files with ?GRNAME, 4-12
    Multiprocessor communications adapters (MCAs), 5-13
    Prefixes of, 4-9f
    Templates, 5-9
    Uparrow (^) prefix, 4-10
Pathnames, Detecting references to remote hosts in, 11-7
Paths, execution (See "Execution path".)
Paths, full-duplex communications, 7-2
?PBLKS offset, 13-434
?PBRK bit, 3-18
?PCAD offset, 13-398f, 13-446ff, 13-578
?PCAL offset, 13-413, 13-416, 13-420
?PCNX system call, 8-1, 8-4, 13-388f
?PCON offset, 13-413, 13-416
?PCS1 offset, 5-8
?PCS3 offset, 13-398ff
?PDFP offset, 13-413, 13-417
?PDIR offset, 13-413, 13-415
Pending/unpending tasks via common local servers (See "Fast
    interprocess synchronization".)
.PENT (procedure entry) pseudo-op, 12-4
:PER (peripheral directory), 13-339 (See "Peripheral directory
    (:PER)".)
Performing block I/O on
    magnetic tapes, 13-447ff
    MCAs, 13-449f
    physical block I/O, 13-397ff
    record I/O, 13-495ff
    shared-page read with ?SPAGE, 13-577ff
Peripheral directory (:PER), 4-10, 5-9, 13-339
    File type of spoolable, 4-6
Peripheral manager (PMGR), 3-8f
    Managing character I/O with, 3-8
Permanent attribute, 4-17
    Setting or removing for file or directory with ?SATR, 4-17,
        13-509f
Permanently binding pages to working set (See "?WIRE system call".)

Permitting access to
    protected files, 13-389ff
    protected shared files (See "?PMTPF system call".)
?PFADW offset, 13-390, 13-574
?PFFLG offset, 13-390, 13-574
?PFIH offset, 13-390, 13-574
?PFLG offset, 3-18, 13-71, 13-413ff
?PFPID offset, 13-390, 13-574
?PFRNG offset, 13-390, 13-574
Physical block input/output (I/O), 5-2, 5-7f, 13-357ff, 13-397ff
    Definition of, 5-7
    Differences between block I/O and, 5-7f
Physical blocks, reading/writing on disk, 13-445ff
PID (See "Process identifiers (PIDs)".)
PID/ring tandems, 2-8, 13-34 (See also, "Process identifiers (PIDs)".)
    Definition of, 8-6
?PIFP offset, 13-413, 13-417
?PIPC offset, 13-413, 13-415
@PLA, 5-10 (See also, "Devices".)
?PLFP offset, 13-413, 13-417
Plotters, 5-10 (See also, "Devices".)
    File type, 4-7
Plus sign (+) template, 4-16 (See also, "Access control list (ACL)".)
?PMDIS offset, 13-430, 13-433
?PMEM offset, 13-314, 13-415, 13-420
PMGR (See "Peripheral manager (PMGR)".)
?PMSDEN offset, 13-430
?PMTPF system call, 2-1, 2-7f, 2-10, 13-389ff, 13-575
    Packet contents, 13-391
    Packet structure, 13-390
?PNAME system call, 3-1, 3-7, 3-15, 13-393f
?PNM offset, 13-413, 13-415
?POFP offset, 13-413, 13-417
Point-to-point lines, 13-589
Point-to-point stations, 13-536f, 13-591, 13-607, 13-609 (See also, "?SRCV system call and ?SSND system call".)
Point-to-point/multipoint line configurations, Illustration of, 9-4
Pointer,
    File, 5-6
    Frame, 6-9, 6-15
    Positioning file, 5-6
    Stack, 6-15
Poll-address/select-address pair, defining with ?SDPOL, 13-529ff
Polling,
    Definition of, 9-4f
    General poll, 9-5f, 13-592
    Multipoint line selection and, 9-4ff
    Specific poll, 9-5f

Polling list, 13-533
    Defining with ?SDPOL, 13-529ff
POPJ instruction, 12-9
Port entry, file type of IPC, 4-6 (See also, "Interprocess communications (IPC) facility".)
Port numbers, (See also, "Interprocess communications (IPC) facility".)
    Extracting ring field from global, 7-4
    Finding owner of global, 13-287f
    Getting console, 13-161f
    Global, 13-261f, 13-162
    Identifying PID associated with global, 7-4
    Interpreting ring fields within global, 7-4
    Local, 4-4, 6-5, 13-41
    Modifying ring field within global, 7-4
    Translating from local to global with ?TPORT, 7-2, 13-649
Ports, (See also, "Interprocess communiations (IPC) facility".)
    Global, 7-2ff, 13-162 (See also, "Global port numbers".)
    Local, 4-4, 6-5, 7-2ff (See also, "Local port numbers".)
    Opening IPC with ?ISEND (See "?ISEND system call" and "Interprocess communications (IPC) facility".)
    Sending messages between IPC, 7-2ff
Position of file pointer (See also, "File pointer".)
    Changing, 5-6, 13-195
    Getting current, 5-6, 13-193ff
    Setting, 13-581ff
Positive acknowledgment (ACK0), 13-539ff, 13-590, 13-607f (See also, "Data-link control characters (DLCC)".)
Positive acknowledgment (ACK1), 13-539ff, 13-590, 13-607f (See also, "Data-link control characters (DLCC)".)
Postprocessors,
    ?UKIL, 6-16
    ?UTSK, 6-16
Power-failure/auto-restart routine, 10-2, 10-11
?PPCR offset, 3-11, 13-413, 13-416
?PPRI offset, 13-413, 13-415, 13-418
?PPRV offset, 3-9, 3-12, 13-413, 13-416, 13-418 (See also, "?PROC system call".)
.PR files, 4-9, 12-2 (See also, "Files".)
    Getting name of for ring (See "?RNGPR system call".)
?PRBB offset, 5-8, 13-399ff
?PRCL offset, 13-221, 1-398f, 13-446ff
?PRCL offset, 13-578
?PRCNX system call, 8-1, 8-4, 13-395f
?PRDB/?PRWB system calls, 5-1, 5-7, 13-397ff, 13-450
    Packet contents, 13-399
    Packet controller status words, 13-400
    Packet structure, 13-398

Pre-emptible processes, 2-6, 3-5f, 3-14
Prefix characters, 13-398
Prefixes,
  At sign (@), 5-9, 5-11
  Pathname, 4-9f
?PRES offset, 13-398, 13-446f
Preventing
  ?RINGLD system call from loading lower ring, 13-499f
  lower rings from being ringloaded (See "?RNGST system call".)
Previous environment, unwinding stacks and restoring, 13-663f
Previous frames in stack, getting information about, 13-675f
Previously wired pages, releasing (See "?UNWIRE system call".)
?PRI system call, 6-1, 6-10, 13-403f
Primary station, definition of, 9-3
Printers, data channel line, 5-10 (See also, "Devices".)
Priorities,
  Assigning sons higher, 3-10
  Changing process with ?PRIPR, 3-10, 3-15 (See also, "?PRIPR system call".)
  Changing for calling task with ?PRI, 13-403f
  Changing process, 13-405ff
  Changing task specified by TID, 13-245f
  Killing tasks of specified, 13-409f
  Readying tasks of specified, 13-425f
  Suspending tasks of specified, 13-427f
Priority and TID of calling task, getting with ?MYTID, 13-351
Priority numbers, 6-5
  Process, 3-6
?PRIPR system call, 3-1, 3-10, 3-13, 3-15, 6-10, 13-245, 13-405ff, 13-619
Privilege bits (?PROC system call), 13-206, 13-419
Privilege to issue IPC system calls, 3-10
Privileged instructions, 3-17
Privileged user, 2-4
Privileges,
  Assigning son higher priority, 3-10
  Assigning Superprocess mode, 3-13
  Changing priority of own process, 3-10
  Changing process type, 3-10
  Creating sons of any process type, 3-10
  Creating sons of different program file type, 3-10
  Creating sons with different usernames, 3-10
  Creating unlimited number of sons, 3-10f
  Defining and accessing user devices, 3-10
  Defining working-set parameters for sons, 3-10
  File access (See "Access privileges".)
  Getting access control, 13-205f
  Issuing primitive IPC system calls, 3-10

Privileges (Cont.)
    Masking process, 13-418
    ?PCSP, 13-619f
    Process, 3-3, 3-9ff
    ?PVDV, 3-10, 10-1, 13-234, 13-281
    ?PVIP, 7-2
    ?PVSU, 13-623f
    Superuser, 13-333f
    Turning on Superprocess mode, 3-10, 3-12
    Turning on Superuser mode, 3-10, 3-12
    Verifying caller's access, 11-8
?PRKIL system call, 6-1, 6-13ff, 13-305f, 13-409f, 13-427
?PRNH offset, 13-8, 13-221, 13-398f, 13-446ff, 13-578
?PRNL offset, 13-446f, 13-449
?PROC system call, 3-1, 3-8f, 3-11ff, 3-18, 3-20, 11-7, 13-58, 13-71, 13-208, 13-214, 13-411ff
    Packet, 5-12
        Contents, 13-414ff
        Structure, 13-413
    Privilege bits, 13-419
    Sample packet, 13-421ff
Procedure entries, 12-6
    Passing descriptor via stack, 12-6
    Translating procedure name to descriptor, 12-6
Procedure entry (.PENT) pseudo-op, 12-4
Procedure entry descriptor,
    Passing via stack, 12-6
    Translating procedure name to, 12-6
Procedure name, translating to procedure entry descriptor, 12-6
Procedures,
    Chaining to new, 13-443f
    Error-recovery (See "Error-recovery procedures".)
    System-generation, 4-2, 6-10, 9-2, 10-1, 13-151, 13-511, 13-521
Process, aborting and generating console interrupt with CTRL-C CTRL-B, 5-21 (See also, "Control sequences".)
Process, operator (See "Operator process".)
Process and memory sample programs, 3-23ff
Process blocking, 3-6
Process creation parameters, 3-11
Process creation parameters, steps AOS/VS takes to check, 3-11
Process hierarchy, illustration of, 3-9
Process identifier (PID), 3-6f
    Getting calling process's with ?PNAME, 3-7
    Getting father process's with ?DADID, 3-8
    Getting PID associated with global port number, 13-191f
    Getting username associated with, 3-8
    Local, 13-648
    PID/ring tandems, 6-4
        Definition of, 8-6

Process identifier (PID) (Cont.)
  Virtual (See also, "Virtual PID".)
    Getting, 13-227
    Translating into component parts with ?TPID, 13-647f
Process image, 6-8
Process information, 3-15f
Process name,
  Full, 3-6f, 13-226
  Getting calling process's, 3-7 (See also, "?PNAME system call".)
  Getting full, 13-393f
  Illustration of, 3-7
Process priorities, changing with ?PRIPR, 3-15 (See also, "?PRIPR system call".)
Process priority numbers, 3-6
Process privileges, 3-3, 3-9ff
Process protections, overriding with ?SUPROC, 13-619f
Process runtime statistics, getting (See "?RUNTM system call".)
Process scheduling, 3-14
Process states, 3-13f
  Blocked, 3-14
  Eligible, 3-13f
  Ineligible, 3-13
Process termination, 3-2f
  Reasons for, 3-8
Process termination codes in offset ?IUFL (for ?IREC and ?ISEND headers), 7-11, 8-6 (See also, "Interprocess communications (IPC) facility".)
  ?T32T extended code, 7-11
  ?TABR extended code, 7-11
  ?TAOS code, 7-11
  ?TBCX code, 7-11, 8-7, 13-54
  ?TCCX code, 7-11, 8-7
  ?TCIN code, 7-11, 7-13
  ?TEXT code, 7-11, 7-15f, 13-276, 13-278
  ?TR32 extended code, 7-11
  ?TRAP code, 7-11, 7-13
  ?TSELF code, 7-10f
Process termination messages in customer/server relationship, 7-9ff
Process trapping, 3-2, 3-8, 3-17
  Causes of, 3-17
  Creating break files at every occurrence, 3-18
Process tree, 3-11
  Illustration of, 3-12
Process types, 3-5f
  Changing, 3-10, 13-57f (See "?CTYPE system call".)
  Creating sons with any, 3-10

Process-management system calls, 3-1 (See individual system call entries for additional references.)
    ?BLKPR, 3-1
    ?BRKFL, 3-1
    ?CHAIN, 3-1
    ?CTYPE, 3-1
    ?DADID, 3-1
    ?ENBRK, 3-1
    ?EXPO, 3-1
    ?GUNM, 3-1
    ?KHIST, 3-1
    ?MDUMP, 3-1
    ?PNAME, 3-1
    ?PRIPR, 3-1
    ?PROC, 3-1
    ?PSTAT, 3-1
    ?RESCHED, 3-1
    ?RETURN, 3-1
    ?RINGLD, 3-1
    ?RNGPR, 3-1
    ?RNGST, 3-1
    ?RUNTM, 3-1
    ?SUPROC, 3-1
    ?SUSER, 3-1
    ?TERM, 3-1
    ?UBLPR, 3-1
    ?UNWIRE, 3-1
    ?WHIST, 3-1
    ?WIRE, 3-1
Processes, 6-3, 6-8
    Assigning devices to for record I/O, 5-1
    Becoming customer of specified, 13-33ff
    Blocked, 6-12
    Blocking 3-6, 3-14f (See "?BLKPR system call".)
        voluntarily, 3-14
    Chaining customer, 8-7
    Changing
        priority of other (See "?PRIPR system call".)
        priority of self, 3-10
        priority with ?PRIPR, 13-405ff (See also, "?PRIPR system call".)
        process type, 3-10
        state with Superprocess privilege, 3-12
    Conditions under which AOS/VS blocks, 3-14
    Conditions under which AOS/VS unblocks, 3-15
    Creating, 3-8ff, 13-411ff
    Creating son, 3-11
    Defining customer, 8-2
    Defining server, 8-2

Processes (Cont.)
Definition of, 3-2
Father, 3-7
Getting
name of (See "?PNAME system call".)
PID of father process with ?DADID, 3-8 (See also, "?DADID system call".)
runtime statistics on (See "?RUNTM system call".)
status information for (See "?PROC system call".)
username of with ?GUNM, 13-225f
Initial operator (PID 2), 3-8f, 5-14, 5-16
Masking privileges, 13-418
Memory-resident (See "Resident processes".)
Opening device for exclusive use of particular, 5-18
Operator (PID 2), 13-514
Passing control to new (See "?CHAIN system call".)
Pre-emptible, 3-5f, 3-14
Reasons for termination of, 3-8
Rescheduling with ?RESCHED, 3-14
Resident, 3-5f, 3-14f (See "Resident processes".)
Runtime statistics for, 13-503ff
Scheduling another for execution (See "?RESCHED system call".)
Scheduling, 3-14
Sending messages to console from, 13-549ff
Server, 8-4f
Setting maximum CPU time, 13-420
Setting search list for calling with ?SLIST, 13-569f
Setting working set size, 13-420f
Sixteen-bit (See "Sixteen-bit processes".)
Spanning rings, 3-20f
Illustration of, 3-22
States of, 3-13f
Status information, 13-429ff
Swappable, 3-5f, 3-14
System, 3-9
Terminating and creating break files, 3-18f (See also, "?BRKFL system call".)
Terminating calling and passing termination message to father with ?RETURN, 13-487ff
Terminating customer, 8-4, 13-53f
Termination messages for
16-bit, 7-10ff
32-bit, 7-14ff
Terminating with ?TERM, 13-637ff (See also, "?TERM system call".)
Trapping, 3-8, 3-17
Types of, 2-6
Pre-emptible (See "Pre-emptible processes".)
Resident (See "Resident processes".)
Swappable (See "Swappable processes".)

Processes (Cont.)
    Unblocking with ?UBLPR, 13-659f (See also, "?UBLPR system call".)
    User, 3-9
    Waiting for signal from another, 13-691f
Processname/queuename, 13-13
Profile file, file type of user, 4-6
Program control, transferring to Debugger utility with ?DEBUG, 11-5
Program control/information system calls, 11-5
Program files, 4-5, 4-9, 5-5, 6-3, 12-2 (See also, "Files".)
    Creating sons of different type, 3-10
    File type of AOS (?FPRV), 4-5ff
    File type of AOS/VS (?FPRG), 4-5ff
    Linking object modules to form, 12-4
    Loading into specific rings with ?RINGLD, 3-20, 13-491f (See also, ?RINGLD system call".)
    Types of, 4-5
Program information/control system calls, 11-5
    ?DEBUG, 11-5
Program's pathname, getting with ?GPRNM, 13-197f
Programs,
    Definition of, 3-2
    Linking together with ?CHAIN, 3-19f
    Sample (See "Sample programs".)
Protected code, 3-17
Protected files, accessing, 13-389ff
Protected shared files, 2-7f, 3-20
    Opening, 2-7f, 13-573ff (See also, "?SOPPF system call".)
    Permitting access to, 2-7f (See also, "PMTPF system call".)
Protection,
    Inner-ring task-redirection, 6-13ff (See also, "Tasks".)
    Task redirection, 13-643f
Protection mechanisms, 2-2
    Overriding process, 13-619f
Protection model, AOS/VS, 6-4f (See also, "Tasks".)
Protection rings, Hardware, 2-2 (See also, "Rings".)
Protection schemes, 6-4f (See also, "Tasks".)
    Ring maximization, 6-4 (See also, "Tasks".)
    Ring specification, 6-4f (See also, "Tasks".)
Protocol,
    Binary synchronous communications (BSC), 9-6ff, 13-538, 13-590 (See also, "Binary synchronous communications (BSC)".)
    Multiprocessor communications adapters (MCAs), 5-12f
?PRRDY system call, 6-1, 6-13, 13-87, 13-247, 13-425ff
?PRSUS system call, 6-1, 6-11, 13-3f, 13-87, 13-241, 13-409, 13-425, 13-427f, 13-473, 13-475
?PRWB system call, See ?PRDB/?PRWB system call

?PSCPL offset, 13-430, 13-432
?PSCW offset, 13-430f
?PSEN offset, 13-430f
Pseudo-ops,
    Defining shared area with, 2-6f
    .ENTO (overlay entry), 12-4, 12-10, 13-382
    .EXT, 3-20
    .EXTG, 3-20
    .EXTN, 12-4
    .NREL, 2-6f
    .PART, 2-6f
    .PENT (procedure entry), 12-4
    .PTARG, 12-6, 13-301, 13-441, 13-443
?PSEX offset, 13-430, 13-432f
?PSF4 offset, 13-430f
?PSFA offset, 13-430, 13-433
?PSFL offset, 13-434
?PSFP offset, 13-434
PSHJ instruction, 12-9
PSHR instruction, 12-9
?PSHSH offset, 13-434
?PSHST offset 5-164
?PSHSZ offset 5-164
?PSIH offset, 13-430, 13-433
?PSLFA offset, 13-430, 13-433
?PSMX offset, 13-430, 13-433
?PSNM offset, 13-413, 13-415
?PSNR offset, 13-430f
?PSNS offset, 13-430f
?PSPH offset, 13-413, 13, 415
?PSPR offset, 13-430f
?PSPRST offset, 13-434
?PSPV offset, 13-430f
?PSQF offset, 13-430f
?PSRH offset, 13-430, 13-432
?PSSL offset, 13-430, 13-433
?PSSN offset, 13-430f
?PSST offset, 13-430f
?PSTAT system call, 3-1, 3-15, 13-429ff, 13-505
    Memory descriptor structure, 13-434
    Packet contents, 13-431ff
    Packet structure, 13-430
    Sample packet, 13-435ff
?PSTI offset, 13-8, 13-398f, 13-401, 13-446ff
?PSTO offset, 13-398f, 13-446f
?PSWM offset, 13-430, 13-433
?PSWS offset, 13-430, 13-433
.PTARG pseudo-op, 12-6, 13-301, 13-441, 13-443

?PUNM offset, 13-413, 13-415
?PUSPR offset, 13-434
?PVDV privilege, 3-10, 10-1, 13-69, 13-234, 13-281
?PVEX privilege, 3-10
?PVIP privilege, 3-10, 7-2, 13-34, 13-419
?PVPC privilege, 3-10f
?PVPR privilege, 3-10
?PVSP privilege, 3-10, 3-13, 13-619f
?PVSU privilege, 3-10, 1-623f
?PVTY privilege, 3-10, 13-58, 13-419
?PVUI privilege, 3-10
?PVWM privilege, 3-10
?PVWS privilege, 3-10
?PWMI offset, 13-413, 13-417, 13-420
?PWRB system call, 5-1, 5-7, 13-397ff (See also, "?PRDB/?PWRB system call".)
?PWSS offset, 13-413, 13-415

Q

Queue entry (file type), 4-6
Queue requests, holding, unholding, canceling, 13-117ff
Queued task creation option, 6-7 (See also, "Tasks".)
Queued task manager, 6-7 (See also, "Tasks".)
Queued tasks,
  Creating manager for, 13-273f
  Dequeuing with ?DQTSK, 13-81ff
Queues,
  Managing with EXEC utility, 11-6
  Removing tasks from with ?DQTSK, 13-81ff
Queuing
  file entries, 13-93ff, 13-110ff
  files for spooled output with ?ENQUE, 11-6, 13-93ff

R

Race condition, 6-19, 8-6
Radix, changing using FED utility (See "?FEDFUNC system call".)

Range for local port numbers, 13-42
?RCALL system call, 12-4ff, 12-8f, 13-302, 13-383, 13-441ff
13-676
    Illustration of invalid return address from, 12-9
?RCHAIN system call, 12-4, 12-7f, 13-383, 13-442ff, 13-676
?RDB/?WRB system calls, 5-1, 5-4, 5-7, 13-8, 13-445ff, 13-577
    Packet contents, 13-447
    Packet structure, 13-446
    Sample packet, 13-451f
?RDUDA/?WRUDA system call, 5-1, 5-19, 13-51, 13-453f
Re-enabling
    console interrupts with ?OEBL, 11-2
    control-character console interrupts, 13-313
    relative consoles with ?SERT, 13-533f
    task scheduling, 6-11 (See also, "Tasks".)
Re-entrant code or data, 2-6, 3-2
Read,
    Performing shared-page, 13-577ff
    Terminating with end-of-file character using CTRL-D, 5-20 (See also, "Control characters".)
Read (?FACR) access, 4-13ff, 13-60, 13-79, 13-138, 13-149, 13-206, 13-508
?READ system call, 5-1, 5-4, 5-6, 5-13, 5-18, 6-11, 13-30, 13-193ff, 13-358f, 13-362f, 13-495ff, 13-582, 13-651
    Packet contents, 13-458ff
    Packet structure, 13-456
    Screen management primitives, 13-462ff
        Packet contents, 13-464
        Packet structure, 13-463
    Selected field translation, 13-466ff
        Extended packet contents, 13-468
        Extended packet structure, 13-467
    Sample packet, 13-470
Readers,
    Card, 5-10, 5-17f (See also, "Devices".)
    File type of card, 4-7
Reading
    device characteristics of character devices, 13-153ff
    error message files, 13-97ff
    physical blocks from disk, 13-397ff
    shared page and incrementing use count (See "?SPAGE system call".)
    task message from process console with ?TRCON, 13-651ff
Reading/writing block I/O, 13-445ff
Reading/writing user data area (UDA) with ?RDUDA/?WRUDA, 13-453f
Readying tasks, 6-12f (See also, "Tasks".)
    specified by TID with ?IDRDY, 13-247f
    specified by priority with ?PRRDY, 13-425f

Real-time clock, 13-295
    Frequency, 13-677
Reasons for process termination, 3-8
?REC system call, 6-1, 6-11f, 6-17f, 13-87, 13-241, 13-409, 13-473ff, 13-693ff
Receive and send IPC headers, 7-5ff (See also, "Interprocess communications (IPC) facility".)
    Structure of, 7-6
Receive continue calls, 13-590ff (See also, "?SRCV system call".)
Receive initial calls, 13-589f, 13-592 (See also, ?SRCV system call".)
Receiving
    data or control sequences over BSC lines with ?SRCV, 9-2, 13-585ff (See also, "Binary synchronous communications (BSC)".)
    interrupt service messages with ?IMSG, 13-265f
    intertask messages immediately with ?RECNW, 13-475f
    intertask messages after waiting with ?REC, 13-473f
    IPC messages with ?IREC, 13-275ff (See also, "Interprocess communications (IPC) facility".)
    obituary messages with ?IREC, 8-6
?RECNW system call, 6-1, 6-17, 13-474ff, 13-693ff
Record format masks, 13-46
    ?ORDS, 13-46
    ?ORDY, 13-46
    ?ORFX, 13-46
    ?ORVR, 13-46
Record formats, 13-366f
    Data-sensitive (?RTDS), 13-361, 13-457, 13-460
    Dynamic-length (?RTDY), 13-30, 13-361, 13-457, 13-460
    Fixed-length (?RTFX), 13-361, 13-457, 13-460
    Undefined-length (?RTUN), 13-361, 13-457, 13-460
    Variable block, variable record (?RTVB), 13-361, 13-457, 13-460
    Variable-length (?RTVR), 13-361, 13-457
Record input/output (I/O), 5-3, 5-8f, 13-455ff
    Assigning device to process for, 5-1
    Definition of, 5-8
    Record types, 5-8f
        Data-sensitive, 5-9
        Dynamic-length, 5-8
        Fixed-length, 5-8
        Variable-length, 5-9
    System calls, 5-3
Records, 5-3
    Data-sensitive, 5-9
    Dynamic-length, 5-8
    Fixed-length, 5-8
    Variable-length, 5-9

?RECREATE system call, 4-1, 13-477f
Recreating files with ?RECREATE, 13-477f
Redirecting task's execution path with ?IDGOTO, 13-241f
Redirecting tasks, 6-13 (See also, "Tasks".)
Redirection, protecting task from, 13-643ff
Redirection protection, inner-ring task, 6-13ff (See also, "Tasks".)
Reducing disk size, 5-7
References, link-to-link, 4-12f
Referencing symbol table file without knowing its format, 13-215f
Regions, locking/unlocking critical, 6-18f (See also, "Tasks".)
Registers,
    Floating-point, 10-8
    Initializing floating-point status, 6-19
Relative consoles, 9-6
    Disabling/re-enabling with ?SDRT/?SERT, 13-533f
?RELEASE system call, 4-1, 4-18, 5-1, 5-18, 13-479f
Releasing
    disk blocks, 13-478
    initialized logical disks with ?RELEASE, 13-479f
    logical disks (LDs) with ?RELEASE, 4-18
    overlay and returning, 13-377f
    overlay area with ?OVREL, 13-385f
    previously wired pages (See "?UNWIRE system call".)
    resource and acquiring new one with ?RCALL, 13-441f
    shared page and decrementing use count with ?RPAGE, 13-501f
Relocatability, requirements for runtime, 12-8f
Remote host, 13-13f
    Determining references to from pathname, 1-493f
Remote hosts, 11-5
Remote hosts, Detecting references to with ?RNAME, 11-7
Removing
    channel number from file, 5-3
    or setting permanent attribute for file or directory with ?SATR,
            13-509f
    queued tasks from queue with ?DQTSK, 13-81ff
    user devices from interrupt vector table with ?IRMV, 13-281f
?RENAME system call, 4-1, 4-8, 13-481f
Renaming
    files with ?RENAME, 4-8, 13-481f
    system log file, 13-626 (See also, "System log file".)
Requesting services from EXEC utility, 13-105f
Requirements for
    indexes by file, 4-4
    runtime relocatability, 12-8f
    volume identifier, 5-25
?RESCHED system call, 3-1, 3-14, 13-483f
Rescheduling, disabling task, 13-77f

Rescheduling
current time slice, 13-483f
tasks, 6-10 (See also, "Tasks".)
Resident processes, 2-2, 2-4, 2-6, 3-5f, 3-14f
?RESIGN system call, 8-1, 8-4f, 13-54, 13-485f
Resignation, signaling server, 8-4
Resigning as server with ?RESIGN, 13-485f
Resource base, getting current, 13-163f
Resource deadlock, 12-5
Resource system calls, 12-2, 12-4ff
Alternate return from, 12-7
Illustration of stack after ?RSAVE, 12-8
Resources,
Movable, 13-163, 13-302
Releasing and acquiring new resource with ?RCALL, 13-441f
System, 3-3, 3-15, 3-19
Restoring previous environment after unwinding stacks, 13-663f
Restrictions on ?LOGEV, 13-334 (See also, "?LOGEV system call".)
Retaining calling resource and acquiring new resource with ?KCALL, 13-301f
Retries for disk errors, bypassing, 5-7
?RETURN system call, 3-1, 3-8, 7-10ff, 7-14, 13-54, 13-278, 13-330, 13-485, 13-487ff, 13-638
Returning current
LEF mode status with ?LEFS, 13-325f
number of undedicated pages (See "?GMEM system call".)
Reverse interrupt (RVI), 9-8f, 13-540f, 13-587f, 13-590f, 13-604 (See also, "Data-link control characters (DLCC)".)
Revoking previous ?IDEF system call with ?IRMV, 13-281f
?RFAB code, 7-12
?RFCF code, 7-12
?RFEC code, 7-12
?RFER code, 7-12
?RFWA code, 7-12
Ring 0, 2-3
Ring 4, 6-16
Ring 6, 6-15f
Ring 7, 6-15ff
Ring 7, breaking a connection in, 13-63f
Ring 7 customer, verifying, 13-671f
Ring field, 6-5 (See also, "Interprocess communications (IPC) facility".)
Extracting from global port number, 7-4
Interpreting with ?IREC, 7-4
Modifying within global port number, 7-4, 13-263
Ring indicator, monitor, 5-16f
Ring number, 7-2f
Ring structure, illustration of, 3-22

Ring-maximization protection scheme, 6-4 (See also, "Tasks".)
Ring-specification protection scheme, 6-4f (See also, "Tasks".)
?RINGLD system call, 2-3, 3-1, 3-20, 6-8f, 13-434, 13-491f, 13-500, 13-630 (See also, "?RNGST system call".)
    AOS/VS actions in response to, 6-9
Ringload, stopping lower rings from a (See "?RNGST system call".)
Rings,
    Creating break files of specified user, 3-19
    Crossing from outer to inner, 3-20
    Default user (Ring 7), 2-3
    Dumping memory image to file from specified with ?MDUMP, 3-18
    Getting .PR filename for (See "?RNGPR system call".)
    Hardware protection, 2-2
    Illustration of segments and their, 2-3
    Inner (See "Inner rings".)
    Loading program file into specified, 13-491f
    Mapping lower, 13-327f
    PID/ring tandems, 6-4
    Preventing ?RINGLD system call from loading lower, 13-499f
    Processes spanning, 3-20f
    Stopping lower from being ringloaded (See "?RNGST system call".)
    Structure of, 2-2ff
    System, 2-3
    Target, 7-3
    User, 2-3, 3-3
    Verifying number of particular, 13-495ff
?RNAME system call, 13-493f
?RNGBL offset, 13-496
?RNGBP offset, 13-496
?RNGNM offset, 13-496
?RNGPL offset, 13-496
?RNGPR system call, 3-1, 3-20, 13-495ff
    Packet contents, 13-496
    Packet structure, 13-496
?RNGST system call, 3-1, 13-499f
Root directory, 4-10
Roots,
    Local, 4-17, 4-20
    Logical disk (LD), 4-19f
    System, 3-8f, 4-17
Routines,
    ?BOMB, 12-5f, 13-163
    Communicating from interrupt service, 10-9
    Interrupt service (See "Interrupt service routines".)
    Power-failure/auto-restart, 10-2, 10-10f
    ?UKIL kill-processing, 6-15, 13-243, 13-306, 13-409
    ?UTSK task-initiation, 6-6f (See also, "Tasks".)
?RPAGE system call, 2-1, 2-6, 2-10, 13-501f, 13-667

?RSAVE system call, 12-5, 12-7
?RTDS record format (data-sensitive), 13-361, 13-457, 13-460
?RTDY record format (dynamic-length), 13-330, 13-361, 13-457, 13-460
?RTFX record format (fixed-length), 13-361, 13-457, 13-460
RTN instruction, 12-5, 12-7, 13-443
RTS modem flag, 5-15
?RTUN record format (undefined-length), 13-361, 13-457, 13-460
?RTVB record format (variable-block, variable-record), 13-361,
13-457, 13-460
?RTVR record format (variable-length), 13-361, 13-457
Runtime, determining bias factors at, 11-7
Runtime library, user (URT32.LB), 6-6
Runtime relocatability requirements, 12-8f
Runtime statistics, getting for process with ?RUNTM, 13-503ff
?RUNTM system call, 3-1, 3-15f, 13-435, 13-503ff
  Packet structure, 13-504
  Sample packet, 13-505
RVI (reverse interrupt), 9-8f, 13-540f, 13-587f, 13-590f, 13-604 (See
also, "Data-link control characters (DLCC)".)

S

?SACL system call, 4-1, 4-15, 13-149f, 13-206, 13-507f
?SACP offset, 13-139ff
Sample delimiter table, 13-367
Sample directory tree, illustration of, 4-5
Sample packets,
  ?IDEF system call, 13-239
  ?OPEN system call, 13-374ff
  ?PROC system call, 13-421ff
  ?PSTAT system call, 13-435ff
  ?RDB/?WRB system call, 13-451f
  ?READ/?WRITE system call, 13-470
  ?RUNTM system call, 13-505
  ?SEBL system call, 13-543
  ?SPAGE system call, 13-579
  ?TASK system call, 13-634f
  ?WHIST system call, 13-682
Sample pathname for generic files, 5-11
Sample process tree, illustration of, 3-12

Sample programs,
    Description of file input/output (I/O), 5-2
    File creation and management, 4-21ff
    File input/output (I/O)
    IPC, 7-19ff
    Multitasking, 6-19ff
    Process and memory, 3-23ff
Sample programs, TIMEOUT, 11-9f
?SATR system call, 4-1, 4-17, 13-509f
Save area, extended state, 12-11
Saving the state of memory parameters and tables, 3-17f
?SBER offset, 13-558
?SBIAS system call, 13-151, 13-511f
?SBUL offset, 13-602, 13-606
?SBUP offset, 13-590, 13-602, 13-604, 13-606
?SBYC offset, 13-586, 13-588, 13-602, 13-604, 13-606
?SBYM offset, 13-586, 13-589, 13-602, 13-604
Scalar time value, converting with ?CTOD, 13-55f
Scalar values,
    Converting dates to with ?FDAY, 13-127f
    Converting time to with ?FTOD, 13-147
Scheduling,
    Another process for execution with ?RESCHED, 3-14
    Disabling task, 6-11, 6-18f, 13-87f (See also, "Tasks".)
    Enabling multitask, 13-101f
    Influencing task, 6-6 (See also, "Tasks".)
    Process, 3-14
    Re-enabling task, 6-11 (See also, "Tasks".)
    Task, 6-10f (See also, "Tasks".)
Scheme, memory, 3-3f
?SCHN offset, 13-536
?SCHR system call, 5-1, 5-16, 5-18, 13-513ff
?SCLOSE system call, 2-1, 2-7, 2-10, 13-509ff, 13-572, 13-519f
SCP (System Control Processor), 5-10 (See also, "Devices".)
?SCPS offset, 13-142
Screen management primitives (?READ/?WRITE system call packet),
    13-462ff
?SCSH offset, 13-139ff
?SDAD offset, 13-586, 13-589
?SDAY system call, 11-3
?SDAY system call, 13-521f, 13-615
?SDBL system call, 9-2, 13-523, 13-608
?SDCU offset, 13-139f
?SDEH offset, 13-139f, 13-142
?SDLM system call, 5-1, 13-368, 13-525ff

?SDPOL system call, 9-6, 13-529ff
?SDRT/?SERT system call, 9-6, 13-533f
Search list, 4-8, 4-10
    Creating with ?SLIST, 4-8
    Definition of, 4-8
    Examining current with ?GLIST, 4-8
    Format, 13-569
    Getting contents of with ?GLIST, 13-175f
    Setting for calling process with ?SLIST, 13-569f
?SEBL system call, 9-2, 9-4, 9-6, 9-13, 9-20, 13-523, 13-535ff
    Connect time-out, 13-543
    Packet structure, 13-536
    Sample packet, 13-543
?SECHR system call, 5-1, 5-18, 13-168, 13-545ff
Secondary station, definition of, 9-3
?SEFH offset, 13-139ff
?SEFM offset, 13-139ff
?SEFW offset, 13-139ff
Segment image, 2-8, 6-9, 8-6
Segments, 2-3, 2-7
    Definition of, 2-2
    Illustration of with their protection rings, 2-3
Select address/poll address pair, defining with ?SDPOL, 13-529ff
Selected field translation extension (?READ/?WRITE system call packet), 13-466ff
Selecting, definition of, 9-5
Selection and polling, multipoint line, 9-4ff
Send and receive headers,
    Contents of IPC, 7-7
    IPC, 7-5ff (See also, "Interprocess communications
        Structure of IPC, 7-6
Send continue calls, 13-604, 13-607ff (See also, "?SSND system call".)
Send initial calls, 13-604, 13-606ff (See also, "?SSND system call".)
?SEND system call, 5-1, 5-14, 8-4, 13-549ff
Sending
    and then receiving IPC messages, 13-289ff
    data or control sequences over BSC lines with ?SSND, 9-2, 13-601ff (See also, "Binary synchronous communications (BSC)".)
    intertask messages and waiting for receipt with ?XMTW, 13-695f
    intertask messages with ?XMT, 13-693f
    IPC messages to itself, 7-8f
    messages between IPC ports, 7-2ff
    messages to consoles with ?SEND, 13-549ff
    text over BSC line, 9-7

Sequences,
    Disabling interrupt, 13-311
    Re-enabling interrupt, 13-313
    Simulating keyboard interrupt, 13-309f
    Typical IPC system call, 7-4f (See also, "Interprocess communications (IPC) facility".)
    Waiting for interrupt, 13-315f
?SERMSG system call, 13-553f
    Packet contents, 13-554
    Packet structure, 13-554
?SERT system call (See "?SDRT/?SERT system call".)
?SERVE system call, 8-1ff, 8-6, 13-34, 13-485, 13-555f
Server process, 8-4f
    Defining, 8-2
Server-only system call (?CTERM), 8-5
Servers,
    Becoming with ?SERVE, 13-555f (See also, "Connection-management facility".)
    Establishing logical connection between customer and existing, 8-2
    Inner-ring, 2-7f, 6-16
    Local, 2-7, 6-16
    Managing exchanges between customers and, 8-2
    Passing a connection to another server, 13-387f 13-395ff
    Resigning as server with ?RESIGN, 13-485f (See also, "Connection-management facility".)
    Signaling resignation with ?RESIGN, 8-4
Servers concurrently connected to multiple rings within customer, 8-6
Service, user interrupt, 10-6f
Set, working (See "Working set".)
Setting
    access control list (ACL) for files or directories with ?SACL, 4-15, 13-507f (See also, "Access control list (ACL)".)
    bias factors with ?SBIAS, 13-511f
    character device's characteristics with ?SCHR, 13-513ff
    clearing, or examining default ACL with ?DACL, 4-15, 13-59f (See also, "Access control list (ACL)".)
    clearing, or examining execute-protection status with ?EXPO, 13-123ff (See also, "?EXPO system call".)
    data channel map, 13-611ff
    delimiter table, 13-367f, 13-525ff
    extended characteristics of character device, 13-545ff
    file-pointer position with ?SPOS, 13-581ff
    maximum CPU time for processes, 13-420
    maximum size for control point directory, 13-37f

Setting (Cont.)
- permanent attribute for file or directory with ?SATR, 4-17, 13-509f
- search list for calling process with ?SLIST, 13-569f
- system calendar with ?SDAY, 13-521f
- system clock with ?STOD, 13-615
- system identifier with ?SSID, 13-599f
- time-out value for devices, 13-617f
- working set size for processes, 13-420f

Settings, file-pointer, 13-582
Severity bits, 13-488f
?SFAH offset, 13-139ff
?SGES system call, 13-557ff
- Packet contents, 13-558
- Packet structure, 13-558
- Sample packet, 13-559

Shared access,
- Closing files previously opened for, 13-519f
- Opening files for, 2-6, 13-571f (See also, "?OPEN system call" and "Shared files".)

Shared area, Defining with assembly language pseudo-ops, 2-6
Shared file memory pages, flushing to disk, 13-103f
Shared files,
- Closing (See "?SCLOSE system call".)
- Flushing memory pages to disk (See "?ESFF system call".)
- Opening, 2-7 (See also, "?SOPEN system call".)
- Protected, 2-7f, 3-20
  - Opening, 2-7f, 13-573ff (See also, "SOPPF system call".)
  - Permitting access to (See "PMTPF system call".)

Shared pages, 2-4, 2-6f, 2-9ff
- Definition of, 2-2
- Flushing contents to disk with ?FLUSH, 13-135f
- Illustration of, 2-5
- Reading and incrementing use count for (See "?SPAGE system call".)
- Releasing and decrementing use count for (See "?RPAGE system call".)
- Updating modified, 13-135f
- Ways to use, 2-6

Shared partitions,
- Establishing new size of with ?SSHPT, 13-597f
- Listing current size of with ?GSHPT, 13-201f

Shared-page read, performing with ?SPAGE, 13-577ff
?SHFS offset, 13-140f
?SIDX offset, 13-139ff
Signal, waiting for from another task/process, 13-691f

Signaling
    another task, 13-561f
    another task and then waiting for signal, 13-563f
    server resignation with ?RESIGN, 8-4
    with fast interprocess communication system call, 8-8f
?SIGNL system call, 8-1, 8-4, 8-8, 13-561f, 13-564, 13-691
?SIGWT system call, 8-1, 8-8, 13-562ff, 13-691
?SIID offset, 13-566
?SILN offset, 13-566
?SIMM offset, 13-566
Simple process name, 3-7f
    Getting username associated with, 3-8 (See also, "?GUNM system call".)
Simulating keyboard interrupt sequences with ?KINTR, 13-309f
?SINFO system call, 13-565ff
    Packet structure, 13-566
    Sample packet, 13-567
?SIRL offset, 13-602, 13-604ff
?SIRN offset, 13-566
?SIRS offset, 13-566
Sixteen-bit processes, Chapter 12
    Illustration of basic overlay area, 12-3
    Illustration of multiple overlay area, 12-4
    Illustration of passing a procedure entry descriptor via the stack, 12-6
    Linking object modules to form program files, 12-4
    Memory modificastion with disk images, 12-2
    Overlays (See "Overlays".)
    Primitive overlay system calls (See "Overlays".)
    Resource system calls (See also, "Resource system calls" and individual system call entries for additional references".)
        ?DELAY, 12-1
        ?GCRB, 12-1
        ?IDSTAT, 12-1
        ?IESS, 12-1
        ?IHIST, 12-1
        ?KCALL, 12-1
        ?OVEX, 12-1
        ?OVKIL, 12-1
        ?OVLOD, 12-1
        ?OVREL, 12-1
        ?RCALL, 12-1
        ?RCHAIN, 12-1
        ?SERMSG, 12-1

Sixteen-bit processes (Cont.)
    Resource system calls (Cont.)
        ?UNWIND, 12-1
        ?WALKBACK, 12-1
    Starting histograms for, 13-257ff
    Termination messages for, 7-10ff
    ?TRAP termination messages, 7-13
Sixteen-bit tasks,
    Suspending for specified interval with ?DELAY, 13-73f
    Synchronizing with ?DELAY, 13-73f
Size,
    Hash frame, 13-140
    Listing current shared partition, 13-201f
    Shared partition,
        Establishing new (See "?SSHPT system call".)
        Listing (See "?GSHPT system call".)
    Specifying file-element, 4-2
    Specifying initial stack, 6-8f (See also, "Stacks".)
    Working set, 3-3
?SLAU offset, 13-141f
?SLIST system call, 4-1, 4-8, 13-176, 13-569f
@SLNx device name, 9-2
Slots, map, 10-3ff (See also, "Map slots".)
?SMCH offset, 13-413, 13-417
?SMDI offset, 13-536
?SMIL offset, 13-138, 13-141f
?SMSH offset, 13-140f
?SMSL offset, 13-141
?SNKC offset, 13-558
Software modularity, 2-2
SOH (start-of-header), 9-8f, 13-541 (See also, "Data-link control characters".)
Son processes, creating, 3-11
Sons,
    Assigning higher priority to than father, 3-10
    Creating unlimited number of, 3-10f
    Creating with any process type, 3-10
    Creating with different program file types, 3-10
    Creating with different usernames, 3-10
    Defining working-set parameters for, 3-10
    Passing Superprocess privileges to, 3-13
    Passing Superuser privileges to, 3-12
?SOPEN system call, 2-1, 2-6, 2-10, 13-135, 13-216, 13-223f, 13-519f, 13-571f, 13-577, 13-598
?SOPN offset, 13-139ff
?SOPPF system call, 2-1, 2-7f, 2-10, 13-391, 13-573ff
    Packet contents, 13-574
    Packet structure, 13-574

Source code, 4-9, 4-15
Source files, assembly language, 4-9 (See also, "Files".)
Space,
    Allocating stack, 6-7f (See also, "Stacks".)
    Controlling disk, 4-19f
    Current (CS), 4-19f (See also, "Control point directories (CPDs)".)
    Disk, 4-2 (See also, "Disk space".)
    Logical address, 3-2f
    Maximum (MS), 4-19f (See also, "Control point directories (CPDs)".)
    Virtual address, 3-3f
?SPAGE system call, 2-1, 2-6, 2-11, 13-577ff, 13-598
    Packet contents, 13-578
    Packet structure, 13-578
    Sample packet, 13-579
Spanning rings, processes, 3-20f
Specification format for access control list (ACL), 13-508 (See also, "Access control list (ACL)".)
Specifications, ACL (See "Access control list (ACL)".)
Specifications strings, 13-93ff
    Format, 13-93
    Messages, 13-94f
    Switches, 13-94
Specifications word, file, 13-363ff
Specified priority,
    Killing tasks of, 13-409f
    Readying tasks of, 13-425f
    Suspending tasks of, 13-427f
Specifying
    file formats, 5-19
    file-element size, 4-2
    size of initial stacks, 6-8f (See also, "Stacks".)
?SPNH offset, 13-138ff
Spoolable peripheral directory (file type), 4-6
Spooled output, queuing files for, 11-6
Spooled output, queuing files for, 13-93ff
?SPOS system call, 5-1, 5-6, 13-581ff
?SPTM global port number (predefined origin port for obituary messages), 7-9, 8-6
.SR files, 4-9 (See also, "Files".)
?SRCV system call, 9-2, 9-11ff, 9-16, 9-18f, 13-585ff, 13-608
    Bit masks returned, 13-593
    From multipoint control station, 13-591f
    From multipoint tributary, 13-592f
    From point-to-point station, 13-591
    Illustration of ITB receive buffer format, 13-594
    Output values, 13-593f

?SRCV system call (Cont.)
- Packet structure, 13-586
- Receive continue calls, 13-590ff
- Required input, 13-590
- Types,
  - Receive continue, 13-589ff
  - Receive initial, 13-589

?SRES offset, 13-586, 13-588, 13-602, 13-604
?SSHPT system call, 2-1, 2-6, 2-9, 2-11, 13-202, 13-597
?SSID system call, 13-203, 13-599
?SSIS offset, 13-586ff, 13-590, 13-592f, 13-602ff
?SSND system call, 9-2, 9-11, 9-13ff, 9-17, 13-601ff
- Call types, 13-607ff
- Input status word (?SSIS offset), 13-605f
- Other offsets, 13-606
- Packet contents, 13-603f
- Packet structure, 13-602
- Timing errors, 9-2

?SSTI offset, 13-536
?SSTS offset, 13-139ff
- Flags returned in, 13-143
- Structure, 13-143

?SSUP/?SBUP offset, 13-586, 13-588
.ST file, 13-216, 13-223
Stack fault handler, 6-7f (See also, "Stacks".)
Stacks, 13-301
- Allocating space, 6-7f
- Defining, 6-7f
- Forcing AOS/VS to initialize common inner-ring, 6-9
- Getting information about previous frames in, 13-675fd
- How they work, 6-7
- Illustration of passing procedure entry descriptor via the, 12-6
- Inner-ring, 6-8ff
- Limits, 6-8f
- Pointer, 6-15
- Specifying size of initial, 6-8f
- Stack fault handler, 6-7f
- Unwinding and restoring previous environment with ?UNWIND, 13-663f
- User, 10-8f, 12-7
- Wide (32 bits), 6-8ff

?STAH offset, 13-139ff
?STAL offset, 13-139ff
Standard directory file type, 13-43
Standard I/O packet structure, 13-194
Start of header (SOH), 9-8f, 13-541 (See also, "Data-link control characters (DLCC)".)

Start of text (STX), 9-8f, 13-540ff, 13-605 (See also, "Data-link control characters (DLCC)".)
Starting histograms
  for 16-bit processes with ?IHIST, 13-257ff
  for 32-bit processes with ?WHIST, 13-679ff
States,
  Illustration of task, 6-11 (See also, "Tasks".)
  Process (See "Process states".)
Stations, 9-1
  Control, 9-4f
  Primary (See "Primary station".)
  Secondary (See "Secondary station".)
  Tributary (See "Tributary station".)
  Connecting two or more (See "Dedicated communications line", 9-1
Statistics,
  Getting BSC error, 13-557ff (See also, "Binary synchronous communications (BSC)".)
  Getting BSC error-recovery, 9-13 (See also, "Binary synchronous communications (BSC)".)
  Process runtime, 13-503ff (See also, "?RUNTM system call".)
Status,
  Examining, clearing, or setting execute-protection, 13-123ff
  Execute-protection, 3-16
  Getting task, 13-641, 13-661f
  LEF mode (See "Load-effective address (LEF) mode".)
Status file, 3-17
Status information,
  File, 13-137ff
  Process, 13-429ff (See also, "?PROC system call".)
  EXEC, 13-119f
Status register, floating-point, 6-19
?STCH offset, 13-138ff
?STCL offset, 13-139ff
Steps AOS/VS takes to check process creation parameter, 3-11
?STIM offset, 13-139ff
?STMAP system call, 10-1, 10-4, 13-611ff
?STMH offset, 13-139ff
?STML offset, 13-139ff
?STOC offset, 13-536, 13-543
?STOD system call, 11-3, 13-615
?STOM system call, 5-1, 13-617f, 13-617f
Stopping lower rings from being ringloaded (See "?RNGST system call".)
?STOV offset, 13-586, 13-589, 13-591f, 13-602, 13-604
Strings, specifications (See "Specifictions strings".)

Structure,
  AOS/VS file, 4-1
  Array (See "Array structure".)
  Disk file, 4-2ff
  Error codes in ERMES file, 13-98
  ?EXEC system call packet for
    dismount function ?XFDUN, 13-109
    labeled mount function ?XFMLT, 13-108
    labeled mount function ?XFXML (extended), 13-108
    queue requests, 13-111, 13-118
    status information, 13-120
    unlabeled mount function ?XFMUN, 13-107
    unlabeled mount function ?XFXUN (extended), 13-107
  ?GNFN system call packet, 13-182
  ?GTMES system call packet, 13-208
  ?GTRUNCATE system call packet, 13-220
  Illustration of directory, 4-11
  IPC send and receive headers, 7-6
  Map definition table, 10-5f, 13-236
  ?MBFC/?MBTC system call packet, 13-338
  Offset ?IUFL, 7-10
  ?OPEN system call extended packet for labeled magnetic tapes, 13-370
  ?PSTAT system call memory descriptor packet, 13-434
  ?PSTAT system call packet, 13-430
  ?READ/?WRITE system call packet, 13-456
    For selected field translation (extended packet), 13-467
    For screen management primitives, 13-463
  Ring, 2-2ff
  ?RNGPR system call packet, 13-496
  Standard I/O packet, 13-194
  ?TASK system call extended packet, 13-632
  ?TASK system call packet, 13-628
  ?UIDSTAT system call, 13-662
?STTO offset, 13-558
STX (start of text), 9-8f, 13-540ff, 13-605 (See also, "Data-link control characters (DLCC)".)
?STYP offset, 13-138ff
Superprocess mode, 2-4, 3-12f, 3-15, 13-138ff, 13-638, 13-659f
  Assigning privilege, 3-13
  Changing state of another process with, 3-12
  Examining, entering, or leaving (See "?SUPROC system call".)
  Passing privilege to sons, 3-13
  Privilege of turning on, 3-10, 3-12
Superuser mode, 3-12f, 4-15
  Examining, entering, or leaving (See "?SUSER system call".)
  Passing privilege to sons, 3-12
  Privilege of turning on, 3-10, 3-12, 13-333f

Suppressing
    console output with CTRL-O, 5-20 (See also, "Control characters".)
    obituary messages with bit ?COBIT, 8-5
?SUPROC system call, 3-1, 3-10, 3-13, 13-419, 13-619f, 13-624
?SUS system call, 6-1, 6-11ff, 13-409, 13-425, 13-621
?SUSER system call, 3-1, 3-10, 13-420, 13-623f
Suspended tasks, 6-10 (See also, "Tasks".)
Suspending
    16-bit task for specified interval with ?DELAY, 13-73f
    calling task with ?SUS, 13-621
    tasks for specific time with ?WDELAY, 6-12, 13-677f (See also, "Tasks".)
    tasks of specified priority with ?PRSUS, 13-427f
Suspensions, lifting task, 13-241f
Swappable processes, 2-6, 3-5f, 3-14
Swapping to next reel of labeled magnetic tape, 13-133f
Switched communications lines, 9-1f
Switches,
    /B specifications string, 13-94
    /FORMS (CLI), 5-19
    /H specifications string, 13-94
    /MES=message specifications string, 13-94
    /P specifications string, 13-94
?SYLOG system call, 13-334, 13-625f
Symbol, getting value of user, 13-223f
Symbol closest in value to specified input value, getting, 13-215f
Symbol table (.ST) file, 11-7
    Accessing with ?GTNAM and ?GTSVL, 11-7
    File type, 4-6
    Opening using FED utility (See "?FEDFUNC system call".)
    Referencing without knowing its format, 13-215f
    System-defined, 11-7
    User-defined, 11-7
Symbols, 11-7
SYN (synchronization character), 9-10, 13-542 (See also, "Data-link control characters".)
Synchronization, fast interprocess (See "Fast interprocess synchronization".)
Synchronization character (SYN), 9-10, 13-542 (See also, "Data-link control characters (DLCC)".)
Synchronizing 16-bit tasks with ?DELAY, 13-73f
Synchronous communications line (file type), 4-7
Syntax,
    CLI DUMP command, 5-36
    CLI MOUNT command, 5-35
:SYSLOG system log file, 11-5, 13-625f (See also, "System log file".)

System,
    Identifying with ?GSID, 11-5
    Operating (See "Operating system".)
System and user flags (IPC), 7-7ff
System calendar, setting with ?SDAY, 13-521f
System calls, Chapter 13 (See also, individual system call entries.)
    Block input/output (I/O), 5-3
    Clock/calendar, 11-3
    Connection-management, 8-1 (See also, "Connection-management facility".)
    Context-management, 2-10 (See also, "Context-management system calls".)
    File input/output (I/O), 5-1
    Logging, 13-329ff
    Memory-management, 2-1 (See also, "Memory-management system calls".)
    Miscellaneous, 11-1
    Primitive overlays (See "Overlays".)
    Privilege to issue IPC, 3-10
    Program information/control, 11-5
    Record input/output (I/O), 5-3
    Resource, 12-4ff
    Server-only (?CTERM), 8-5
    Sixteen-bit processes, 12-1
    Typical IPC sequence, 7-4f (See also, "Interprocess communications (IPC) facility".)
System clock, 13-55, 13-165, 13-171
    Setting with ?STOD, 13-615
    Getting frequency for with ?GHRZ, 13-171f
System Control Processor (SCP), 5-10 (See also, "Devices".)
System crash, 5-6
System data file (file type), 4-6
System deadlock, 13-177
System devices, 4-10 (See also, "Devices".)
System flag word (offset ?ISFL), contents of, 7-8
System hierarchy, 3-8
System identifier, 13-566
    Getting with ?GSID, 13-203f
    Setting with ?SSID, 13-599f
System information, 11-5
System log file, :SYSLOG, 11-5, 13-625f
    Entering events in, 13-333ff
    Logging messages into with ?LOGEV, 11-5
    Manipulating with ?SYLOG, 13-625f
    Renaming, 13-626
System processes, 3-9
System resources, 3-3, 3-15, 3-19

System rings, 2-3 (See also, "Rings".)
System root, 3-8f, 4-17
System-generation procedure, 4-2, 6-10, 9-2, 10-1, 13-151, 13-511, 13-521

T

?T32T extended code, 7-11 (See also, "Process termination codes in offset ?IUFL".)
Tables,
- Connection (See "Connection table".)
- Interrupt vector (See "Interrupt vector table".)
- Map definition (See "Map definition table".)
- Sample delimiter, 13-367
- Setting delimiter, 13-367f, 13-525ff
- Structure of map definition, 10-5f

?TABR extended code, 7-11 (See also, "Process termination codes in offset ?IUFL".)
Tailoring
- ?UTSK task-initiation routine, 6-6f (See also, "Tasks".)
- format of line-printer output, 5-19

Tandem, PID/ring (See "PID/ring tandem".)
?TAOS code, 7-11 (See also, "Process termination codes in offset ?IUFL".)
Tape files, file type of generic labeled, 4-6
Tape unit, file type of magnetic, 4-7
Tapes (See also, "Magnetic tapes".)
- Controllers for labeled magnetic, 5-10 (See also, "Devices".)
- Controllers for unlabeled magnetic, 5-10 (See also, "Devices".)
- File type of magnetic, 4-6
- Forcing end-of-volume on labeled magnetic, 13-133f
- Labeled magnetic, 5-10 (See also, "Devices".)
- Magnetic (See "Magnetic Tapes".)

Target ring, 7-3
Task control blocks (TCBs), 13-249, 13-273
Task identifier (TID), 6-5, 6-15f
- Changing task priority specified by, 13-245f
- Getting for calling task with ?MYTID, 13-351
- Getting unambiguous, 13-661f
- Killing tasks specified by, 13-243f
- Readying tasks specified by, 13-247f

Task rescheduling, disabling, 13-77f

Task scheduling,
    Disabling, 6-11 (See also, "Tasks".)
    Re-enabling, 6-11 (See also, "Tasks".)
Task states, Illustration of, 6-11 (See also, "Tasks".)
?TASK system call, 6-1, 6-6f, 6-9, 6-17, 13-82, 13-237f, 13-273f 13-627ff, 13-631ff
    Aborting while ?UTSK is executing, 6-6 (See also, "Tasks".)
    Extended packet structure, 13-632
    Packet contents, 13-629
    Packet structure, 13-628
    Sample packet, 13-634f
Task-initiation routine (?UTSK), 6-6f (See also, "Tasks".)
Task-management system calls, 6-1f (See individual system call entries for additional references.)
    ?DFRSCH, 6-1
    ?DQTSK, 6-1
    ?DRSCH, 6-1
    ?ERSCH, 6-1
    ?IDGOTO, 6-1
    ?IDKIL, 6-1
    ?IDPRI, 6-1
    ?IDRDY, 6-1
    ?IDSTAT, 6-1
    ?IDSUS, 6-1
    ?IFPU, 6-1
    ?IQTSK, 6-1
    ?KILAD, 6-1
    ?KILL, 6-1
    ?MYTID, 6-1
    ?PRI, 6-1
    ?PRKIL, 6-1
    ?PRRDY, 6-1
    ?PRSUS, 6-1
    ?REC, 6-1
    ?RECNW, 6-1
    ?SUS, 6-1
    ?TASK, 6-1
    ?TIDSTAT, 6-1
    ?TLOCK, 6-1
    ?TRCON, 6-1
    ?TUNLOCK, 6-1
    ?UIDSTAT, 6-1
    ?WDELAY, 6-2
    ?XMT, 6-2
    ?XMTW, 6-2
Task-redirection protection for inner rings, 6-13ff (See also, "Tasks".)
Task-termination routine, ?UKIL (See "Kill-processing routines".)

Task-to-task communication, 6-17f (See also, "Tasks".)
Tasks, Chapter 6
    Aborting ?TASK while ?UTSK is executing, 6-6
    Changing priority by specifying TID, 13-245f
    Changing priority of calling, 13-403f
    Circumstances under which AOS/VS reschedules, 6-10
    Concepts, 6-3
    Console-to-task communication, 6-17
    Creating manager for queued, 13-273f
    Definition of, 3-2
    Dequeuing previously queued, 13-81ff
    Detecting termination and creation of, 6-16
    Disabling scheduling with ?DRSCH, 6-11, 6-18f, 13-87f
    Getting priority and TID for calling, 13-351
    Getting status of, 13-641, 13-661f
    Handling console interrupt, 13-271f
    Illustration of states, 6-11
    Influencing scheduling, 6-6
    Initial, 6-6
    Initiating, 6-6, 13-627ff
    Inner-ring task-redirection protection, 6-13ff
    Killing by specifying TID, 13-243f
    Killing calling, 13-307f
    Killing those with specified priority, 13-409f
    Lifting suspensions, 13-241f
    Locking/unlocking critical regions, 6-18f
    Message-manager, 13-651f
    Protecting from redirection, 13-643ff
    Protection schemes, 6-4f (See also, "Tasks".)
        Ring maximization, 6-4
        Ring specification, 6-4f
    Queued task creation option, 6-7
    Queued task manager, 6-7
    Re-enabling previously disabled scheduling, 6-11
    Ready, 6-12
    Readying, 6-12f (See also, "Tasks".)
        by specifying TID, 13-247f
        those with specified priority, 13-425f
    Redirecting, 6-13
    Redirecting execution path with ?IDGOTO, 13-241f
    Scheduling, 6-10f
    Sending intertask messages and waiting for receipt with ?XMTW
                13-695f
    Sending intertask messages with ?XMT, 13-695f
    Sending messages from process console to, 13-651ff
    Signaling, 13-561f, 13-562ff
    Suspended, 6-10, 6-12

Tasks (Cont.)
    Suspending
        16-bit for specified interval, 13-73f
        calling, 13-621
        for specified time with ?WDELAY, 6-12, 13-677f
        those with specified priority, 13-427f
    Synchronizing 16-bit with ?DELAY, 13-73f
    Tailoring ?UTSK task-initiation routine to your appication, 6-6
    Task-to-task communication, 6-17f
    Terminating, 6-15
    Using common local servers to pend/unpend (See "Fast interprocess synchronization".)
    Waiting for signal from another, 13-691f
?TATH offset, 13-43
?TBCX termination code, 7-11, 8-7, 13-54 (See also, "Process termination codes in offset ?IUFL".)
TCB (task control block), 3-17, 13-249
?TCCX code, 7-11, 8-7 (See also, "Process termination codes in offset ?IUFL".)
?TCIN code, 7-11, 7-13 (See "Process termination codes in offset ?IUFL".)
?TCTH offset, 13-43, 13-138
?TEFH offset, 13-220
?TEFM offset, 13-220
?TEFW offset, 13-220
Templates,
    ACL, 4-16 (See also, "Access control list (ACL)".)
    Pathname, 5-9
Temporary files, 4-9 (See also, "Files".)
Temporary text delay (TTD), 9-10, 13-542, 13-604 (See also, "Data-link control characters (DLCC)".)
?TERM system call, 3-1ff, 3-8, 3-13, 7-11ff, 8-5, 13-54, 13-330, 13-485, 13-619, 13-637ff
Terminals (See "Consoles".)
Terminating
    calling process and passing termination message to father with ?RETURN, 13-487ff (See also, "?RETURN system call".)
    calling task with ?UKIL, 13-306f
    connections, 8-5
    current read with end-of-file character using CTRL-D, 5-20 (See also, "Control characters".)
    customer processes with ?CTERM, 8-4, 13-53f (See also, "?CTERM system call".)
    histograms with ?KHIST, 3-16 (See also, "?KHIST system call".)
    process and creating break file, 3-18f (See also, "?BRKFL system call".)

Terminating (Cont.)
    processes, 3-2f, 3-8
    processes with ?TERM, 13-637ff (See also, "?TERM system call".)
    tasks, 6-15 (See also, "Tasks".)
Termination and creation detection (tasks), 6-16 (See also, "Tasks".)
Termination code ?TBCX, 13-54 (See also, "Process termination codes in offset ?IUFL".)
Termination code ?TEXT, 7-11, 7-15f, 13-276, 13-278 (See also, "Process termination codes in offset ?IUFL".)
Termination codes for 16-bit processes, 7-12 (See "Interprocess communications (IPC) facility".)
    ?RFAB code, 7-12
    ?RFCF code, 7-12
    ?RFEC code, 7-12
    ?RFER code, 7-12
    ?RFWA code, 7-12
Termination codes in offset ?IUFL for ?IREC and ?ISEND headers, process (See "Process termination codes in offset ?IUFL".)
Termination messages, 8-7, 13-638 (See also, "Process termination codes in offset ?IUFL".)
    ?TBVC, 8-7
    16-bit processes, 7-10ff
    32-bit processes, 7-14ff
    Passing to father and terminating calling process with ?RETURN, 13-487ff
Termination messages in customer/server relationship, process, 7-9ff (See also, "Connection-management facility".)
Terms, Definition of memory-management, 2-1f
Text associated with error code, getting, 13-553f
?TEXT code termination messages sent on 32-bit process user trap, 7-11, 7-15f, 13-276, 13-278 (See also, "Process termination codes in offset ?IUFL".)
Text editors, 4-2
Text files (file type ?FTXT), 4-6, 5-5
Text mode, transparent, 9-10
Text string associated with particular error code, 11-4
?TEXT termination code, 13-276, 13-278
Thirty-two-bit processes, termination messages for, 7-14ff
?TIDSTAT system call, 6-1, 6-5, 13-641, 13-661
Time,
    Converting to scalar value with ?FTOD, 13-147
    CPU, 3-3
    Getting with ?GTOD, 13-217
Time block (?CREATE system call), 13-55
Time slice, rescheduling current, 13-483f
Time stamp, 13-295

Time value, converting scalar, 13-55f
Time-out value, setting for devices, 13-617f
Time-out values, 13-592
TIMEOUT sample program, 11-9f
Timing errors (?SSND system call), 9-2
?TLOCK system call, 6-1, 6-14, 13-643ff, 13-657
?TLTH offset, 13-220
.TMP files, 4-9 (See also, "Files".)
?TMTH offset, 13-43
?TMYRING flag (?TLOCK system call), 6-14
?TPID system call, 11-7
?TPID system call, 13-647f
?TPLN termination message length (32-bit processes), 7-14
?TPORT system call, 7-1f, 13-285, 13-290, 13-649
?TR32 extended code, 7-11 (See also, "Process termination codes in offset ?IUFL".)
Transferring
    attributes to new program with ?CHAIN, 3-19
    program control to Debugger utility with ?DEBUG, 11-5
Translating
    host ID and PID into virtual PID with ?GVPID, 11-7
    local port number into global equivalent with ?TPORT, 13-649
    local port number to global equivalent with ?TPORT, 7-2 (See also, "Interprocess communications (IPC) facility".)
    procedure name to procedure entry descriptor, 12-6
    virtual PID into host ID and PID with ?TPID, 11-7, 13-647f
Transmitting intertask messages
    and waiting for receipt with ?XMTW, 13-695f
    from interrupt service routines with ?IXMT, 13-299f
    with ?XMT, 13-693f
Transparent text mode, 9-10
Trap,
    Creating break files for every process, 3-18
    Process (See "Process trapping".)
    User, 13-31
?TRAP termination messages for 16-bit processes, 7-11, 7-13
?TRCON system call, 6-1, 6-17, 13-651ff
Tree,
    Illustration of sample directory, 4-5
    Process, 3-11f
        Illustration of, 3-12
Tributary station, 9-3ff, 9-6
?TRUNCATE system call, 5-1, 5-6, 13-655f
Truncating files, 13-219ff
?TSELF code, 7-10f (See also "Process termination codes in offset ?IUFL".)
?TSUP, 7-13

TTD (temporary text delay), 9-10, 13-542, 13-604 (See also, "Data-link control characters (DLCC)".)
?TUNLOCK system call, 6-1, 6-14, 13-644, 13-657
Turning on Superprocess mode, 3-13
    Privilege of, 3-10, 3-12
Turning on Superuser mode, privilege of, 3-10, 3-12
Type, creating sons with any process, 3-10
Type-ahead buffer, emptying an echoing ^C^C on console with CTRL-C CTRL-C, 5-21 (See also, "Control sequences".)
Types,
    Changing process, 3-10 (See "?CTYPE system call".)
    File access, 4-13f (See also, "Access privileges".)
    File, 4-5
    Process, 3-5f
    Program file, 4-5
Typical IPC system call sequence, 7-4f (See also, "Interprocess communications (IPC) facility".)

U

?UBLPR system call, 3-1, 3-12, 3-15, 6-12, 13-619, 13-659f
UDA (user data area) (See "User data area (UDA)".)
?UDDRS offset, 10-2, 10-11, 13-234
?UDID offset, 13-236f
?UDLN device control table (DCT) length, 10-2
?UDNS offset, 13-237f
?UDRS offset, 10-2, 13-234
?UDVBX offset, 10-2, 13-234f
?UDVIS offset, 10-2, 13-234f
?UDVMS offset, 10-2, 13-234f
?UDVMX offset, 10-2
?UDVXM offset, 13-234f
?UIDSTAT system call, 6-1, 6-5, 6-16, 13-661f
    Packet structure, 13-662
?UKIL kill-processing routine, 6-15f, 13-243, 13-306f, 13-409 (See also, "Kill-processing routines".)
Unblocking processes with ?UBLPR, 13-659f (See also, "?UBLPR system call".)
    When it occurs, 3-15
Undedicated pages, 2-9
    Getting number of with ?GMEM, 13-177f
    Returning current number of (See "?GMEM system call".)
Unholding, holding, canceling queue requests, 13-117ff

Unique kill-processing routine, defining with ?KILAD, 6-15
Unique Storage Position (USP) pointers, 6-16
Unit mount, 13-106
Units,
    Disk, 5-10 (See also, "Devices".)
    File type of disk, 4-6
    File type of magnetic tape, 4-7
    File type of multiprocessor communications, 4-7
    Floating-point (See "Floating-point unit".)
Unlabeled magnetic tapes (See "Magnetic tapes".)
Unlocking/locking critical regions, 6-18f (See also, "Tasks".)
Unpending/pending tasks via common local servers (See "Fast interprocess synchronization".)
Unshared files, 13-667
Unshared memory pages, 2-2, 2-4, 2-6f, 2-9f (See also, "Unshared pages".)
    Changing number of in logical address space, 13-349f (See "?MEMI system call".)
Unshared memory parameters, listing current, 13-347f (See "?MEM system call".)
Unsupported device, establishing interface between AOS/VS operating system and (See "?IDEF system call".)
Unused pages, 2-2, 2-9f
?UNWIND system call, 13-663f
Unwinding stack and restoring previous environment with ?UNWIND, 13-663f
?UNWIRE system call, 3-1, 3-3, 13-665f
Unwiring previously wired pages with ?UNWIRE, 13-665f
Uparrow (^) pathname prefix, 4-10 (See also, "Pathnames".)
?UPDATE system call, 5-1, 5-6, 13-667f
Updating
    histograms, 3-16
    modified shared page, 13-135f
URT32.LB user runtime library, 6-6, 6-15
Use count, 2-4
    Overlay, 12-10f
    Reading shared page and incrementing (See "?SPAGE system call".)
    Releasing shared page and decrementing (See "?RPAGE system call".)
User, privileged, 2-4
User and system flags (IPC), 7-7ff
User console or batch process information (See "?LOGEV system call".)
User context, 2-9
User data area (UDA), 5-19, 13-51f, 13-453f, 13-477
    Creating with ?CRUDA, 13-51f
User data files, 4-5
    File type ?FUDF, 4-6, 5-5

User device support, Chapter 10 (See also, "User devices".)
- Communicating from interrupt service routine, 10-9
- Defining system devices, 10-1
- Defining user devices with ?PVDV, 10-1, 13-233ff
- Enabling and disabling access to all devices, 10-9f
- Illustration of device control table (DCT), 10-2
- Introducing devices to AOS/VS at execution time, 10-2,
- System calls, 10-1 (See individual system call entries for additional references.)
  - ?DDIS, 10-1
  - ?DEBL, 10-1
  - ?IDEF, 10-1
  - ?IMSG, 10-1
  - ?IRMV, 10-1
  - ?IXIT, 10-1
  - ?IXMT, 10-1
  - ?LEFD, 10-1
  - ?LEFE, 10-1
  - ?LEFS, 10-1
  - ?STMAP, 10-1

User devices,
- Defining and accessing, 3-10
- Defining with ?IDEF, 13-233ff
- Disk drives, 10-1
- Line printers, 10-1
- Magnetic tape drives, 10-1
- Removing from interrupt vector table with ?IRMV, 13-281f

User flag word (offset ?IUFL), 7-9 (See also, "Interprocess communications (IPC) facility".)
- Illustration of structure of, 7-10

User interrupt service, 10-6f
User processes, 3-9
User profile file (file type), 4-6
User rings, 2-3, 3-3 (See also, "Rings".)
- Creating break files of specified, 3-19
- Default, 2-3

User runtime library URT32.LB, 6-6
User stacks, 10-8f, 12-7
User symbol, getting value of with ?GTSVL, 13-223f
User trailer labels (See "Labels".)
User traps, 7-12, 13-31
User volume labels, 5-27f (See "Labels".)
User-visible logical address space, 3-3
Usernames, 3-7
- Associated with PID, getting with ?GUNM, 3-8
- Associated with simple process name, getting with ?GUNM, 3-8
- Creating sons with different, 3-10
- Getting process with ?GUNM, 13-225f

Using
    common local servers to pend/unpend tasks (See "Fast interprocess synchronization".)
    ?CREATE and ?DELETE on link entries, 4-13
    IPC facility as communications device, 5-21f (See also, "Interprocess communications (IPC) facility".)
USP (Unique Storage Position) pointers, 6-16
UST, 3-17
?UTID offset, 13-662
Utilities, 4-2
    CLI forms control (FCU), 13-453
    CLI LABEL, 5-22, 13-319, 13-372
    Debugger, 11-5, 13-71
    DFMTR, 5-8
    EXEC (See "EXEC utility".)
    FCU (forms control), 5-19
    File editor (FED) (See "File editor (FED) utility".) call".)
    Interfaces to, 11-6
    Link, 2-7, 3-2, 4-7, 6-6f, 12-2ff
Utility interfaces, 11-6
UTL (user trailer labels) (See "Labels".)
?UTPRI offset, 13-662
?UTSK task-initiation routine, 6-6f, 6-16 (See also, "Tasks".)
?UTSTAT offset, 13-662
?UUID offset, 13-662
UVL (user volume labels) (See "Labels".)

V

?VALAD system call, 11-8, 13-669f
Valid ?CREATE file types, 13-40
Valid filename characters, 3-6
Valid pathname prefixes, 4-9f
Validating logical address, 13-669f
Values,
    Converting scalar time, 13-55f
    Getting current bias factor, 13-151f
    Getting symbols closest in value to specified input, 13-215f
    Getting user symbol, 13-223f
    Time-out, 13-592
Variable-length records, 5-9
?VCUST system call, 8-1, 8-4, 13-671f

Verifying
    caller's access privileges with ?VALAD, 11-8
    customer, 13-673f
    Ring 7 customer, 13-671f
    ring number, 13-495ff
    validity of address with ?VALAD, 11-8
Version Number field (labeled magnetic tapes), 5-28
Virtual address space, 2-4, 3-3f
    Illustration of, 2-5
Virtual consoles, 13-309
    Controlling console interrupts on, 11-3
Virtual PID, 11-7
    Forming from host ID and PID with ?GVPID, 11-7
    Getting with ?GVPID, 13-227
    Translating into host ID and PID with ?TPID, 11-7, 13-647f
Volid (See "Volume identifier".)
Volume identifiers, 5-22
    Checking labeled magnetic tape, 13-27f
    Requirements for, 5-25
Volume labels,
    Contents of, 5-29
    Contents of VOL1, 5-28
Volume sets of labeled magnetic tapes, 5-23
Voluntary process blocking, 3-14
?VRCUST system call, 8-1, 8-4, 13-673f

W

WACK (wait-before-transmit positive acknowledgment), 9-8, 9-10, 13-540, 13-542, 13-588, 13-593, 13-604, 13-607 (See also, "Data-link control characters (DLCC)".)
Wait-before-transmit positive acknowledgment (WACK), 9-8, 9-10, 13-540, 13-542, 13-588, 13-593, 13-604, 13-607 (See also, "Data-link control characters (DLCC)".)
Waiting for
    console interrupt with ?KWAIT, 13-315f
    signal from another task/process, 13-691f
?WALKBACK system call, 13-675f
Ways to use shared memory pages, 2-6
?WDELAY system call, 3-15, 6-2, 6-12, 13-73, 13-677f

?WHIST system call, 3-1, 3-15, 13-258, 13-303, 13-435, 13-505, 13-680
  Packet structure, 13-680
  Sample packet, 13-682
Wide stack (32 bits), 6-8ff
Wide-save instruction, 6-15
?WIRE system call, 2-9, 3-1, 3-3, 13-666, 13-683f
Wired pages, releasing previously, 13-665f
Wiring pages to working set, 13-683f (See also, "?WIRE system call".)
Word,
  File specifications, 13-363ff
  User flag, 7-9
Words copied to break file, 3-18
Working directory, 4-7, 4-10, 13-75
  Changing, 4-8, 13-79f
  Definition of, 4-7
Working set, 2-2, 2-4, 3-3f, 3-9
  Controlling, 3-3
  Defining parameters for sons, 3-10
  Illustration of, 2-5
  Permanently binding pages to (See "?WIRE system call".)
  Setting size for processes, 13-420f
  Size of, 3-3
  Wiring pages to, 13-683f (See also, "?WIRE system call".)
?WRB system call, 5-1, 5-4, 5-7, 13-577 (See also, "?RDB/?WRB system calls".)
Write (?FACW) access, 3-19, 5-5, 13-90, 13-149, 13-508
?WRITE system call, 5-1, 5-4, 5-6, 5-13, 5-18, 5-22, 5-24, 6-11, 13-30, 13-193ff, 13-358f, 13-362f, 13-495ff, 13-582f, 13-667 (See also, "?READ/?WRITE system calls".)
Write-protected pages, 2-1, 2-4
Writing physical blocks to disk, 13-397ff
Writing/reading user data area (UDA) with ?WRUDA/?RDUDA, 13-453f
WRTN instruction, 13-305
?WRUDA system call, 5-1, 5-19, 13-51 (See also, "?RDUDA/?WRUDA system call".)
?WTSIG system call, 8-1, 8-4, 8-8, 13-561, 13-564, 13-691f

X

?XAFD offset, 13-111, 13-114, 13-117
?XAFT offset, 13-111, 13-117
?XDAT offset, 13-111f

?XDUL offset, 13-109
?XDUT offset, 13-109
?XFBP offset, 13-111, 13-114, 13-117
?XFCAN cancel function, 13-117
?XFDUN mount function (See "?EXEC system call".)
?XFGS offset, 13-111, 13-113f, 13-117
?XFHOL hold function, 13-117
?XFMLT mount function (See "?EXEC system call".)
?XFMUN mount function (See "?EXEC system call".)
?XFP1 offset, 13-118ff
?XFXML mount function (See "?EXEC system call".)
?XFXUN mount function (See "?EXEC system call".)
?XIFC characteristic, 5-14
?XLMT offset, 13-111ff
?XMLE offset, 13-108
?XMLF offset, 13-108
?XMLL offset, 13-106, 13-108
?XMLR offset, 13-108
?XMLS offset, 13-108
?XMLT offset, 13-106, 13-108
?XMLV offset, 13-108
?XMT system call, 6-2, 6-17f, 13-87, 13-425, 13-473ff, 13-693ff
    Broadcast option, 13-693
?XMTW system call, 6-2, 6-11f, 6-17f, 13-241, 13-409, 13-473ff, 13-694ff
?XMUE offset, 13-107
?XMUF offset, 13-107
?XMUL offset, 13-106f
?XMUQ offset, 13-107
?XMUR offset, 13-107
?XMUS offset, 13-107
?XMUT offset, 13-106f, 13-110
?XOFC characteristic, 5-14
XOP instruction, 12-9
?XPBP offset, 13-111, 13-114
?XPRI offset, 13-111, 13-113
?XRES offset, 13-107ff
?XRES1 offset, 13-111f
?XRFNC offset, 11-6
?XRFNC offset, 13-106f, 13-117ff
?XSEQ offset, 13-111, 13-114
?XTIM offset, 13-111f
?XTYP offset, 13-111f
?XXWOL offset, 13-111, 13-115
?XXW3 offset, 13-111, 13-116