OS/390

UNIX System Services: APAR OW43776

OS/390

UNIX System Services: APAR OW43776

Contents

About This Guide

This document supports APAR OW43776 for OS/390 UNIX System Services (OS/390 UNIX), which is available for OS/390 Version 2 Releases 8, 9, and 10. The information is based on the Release 9 library. This document is available only on the OS/390 UNIX web site at:

http://www.s390.ibm.com/unix/release/apar.html

Part 1. APAR OW43776: OS/390 UNIX System Services Library

Chapter 1. APAR OW43776: OS/390 UNIX System Services Planning

Chapter 12. Summary of Interface Changes

This section summarizes the new and changed interface components of OS/390 UNIX.

BPXPRMxx

Table 1 lists new and changed statements on the BPXPRMxx parmlib member. While Chapter 14. Customizing OS/390 UNIX has some information about certain BPXPRMxx statements, see *OS/390 MVS Initialization and Tuning Reference* for more detailed information about each statement.

Table 1. Summary of OS/390 UNIX Changes to BPXPRMxx

Statement	Release	Description	Related Support
RUNOPTS	V2R4	**New statement:** RUNOPTS specifies that the Language Environment Run-Time library services be passed to **/etc/init** as the Language Environment Run-Time options.	Language Environment
SYSCALL_COUNTS	V2R4	**New statement:** SYSCALL_COUNTS records the number of system calls.	System calls
IPCMSGBYTES	V2R7	**Changed statement:** The maximum number of bytes in a queue has been changed.	Performance
IPCMSGQMNUM	V2R7	**Changed statement:** The maximum number of messages for each message queue has been changed.	Performance
IPCSHMMPAGES	V2R7	**Changed statement:** The maximum number of pages for a shared memory segment has been changed.	Performance
MAXASSIZE	V2R8	**New function:** You can use the RACF ADDUSER and ALTUSER commands to specify user limits.	RACF
MAXCPUTIME	V2R8	**New function:** You can use the RACF ADDUSER and ALTUSER commands to specify user limits.	RACF
MAXFILEPROC	V2R8	**New function:** You can use the RACF ADDUSER and ALTUSER commands to specify user limits.	RACF
MAXMMAPAREA	V2R8	**New function:** You can use the RACF ADDUSER and ALTUSER commands to specify user limits.	RACF
MAXPROCUSER	V2R8	**New function:** You can use the RACF ADDUSER and ALTUSER commands to specify user limits.	RACF
MAXQUEDSIGS	V2R8	**New statement:** MAXQUEDSIGS specifies the number of signals that are to be concurrently queued within a single process.	Performance
MAXTHREADS	V2R8	**New function:** You can use the RACF ADDUSER and ALTUSER commands to specify user limits.	RACF

Table 1. Summary of OS/390 UNIX Changes to BPXPRMxx (continued)

Statement	Release	Description	Related Support
MOUNT	V2R9	**New keywords:** • SYSNAME(system_name) specifies the system that the mount should be performed on. • AUTOMOVE | NOAUTOMOVE specifies whether the file system is to be automatically moved to another system, which will then become the server, if the original server is brought down.	Shared HFS
ROOT	V2R9	**New keywords:** • SYSNAME(system_name) specifies the system that the mount should be performed on. • AUTOMOVE | NOAUTOMOVE specifies whether the file system is to be automatically moved to another system, which will then become the server, if the original server is brought down.	Shared HFS
SHLIBRGNSIZE()	V2R9	**New statement:** SHRLIBRGNSIZE() specifies the size of the shared library region within the system	Shared library
SHLIBMAXPAGES()	V2R9	**New statement:** SHRLIBMAXPAGES() specifies the amount of data space storage to be used for non-system shared library objects.	Shared library
SYSPLEX	V2R9	**New statement:** SYSPLEX specifies that resources be shared across the sysplex.	Shared HFS
VERSION('nnnn')	V2R9	**New statement:** VERSION('nnnn') enables multiple releases and service levels of the binaries to exist and participate in shared HFS	Shared HFS
LIMMSG(NONE|SYSTEM|ALL)	V2R10	**New statement:** LIMMSG(NONE|SYSTEM|ALL) controls the displaying of console messages that indicate when parmlib limits are reaching critical levels.	RAS Enhancements

Operator Commands

Table 2 on page 5 lists new and changed operator commands that affect OS/390 UNIX. For more information, see *OS/390 MVS System Commands*.

Table 2 on page 5 lists new and changed operator commands that affect OS/390 UNIX. For more information, see *OS/390 MVS System Commands*.

Table 2. Summary of New and Changed Operator Commands

Operator Command	Release	Description	Related Support
SETOMVS	V2R8	**New operand:** The RESET operand enables you to dynamically add the FILESYSTYPE, NETWORK, and SUBFILESYSTYPE statements to the BPXPRMxx parmlib member.	File system
	V2R9	**New operand:** The SYNTAXCHECK operand enables you to check the syntax of a BPXPRMxx parmlib member before doing an IPL.	BPXRMxx
	V2R10	**New operands:** The PID= operand dynamically changes a limit for a process. LIMMSG=NONE|SYSTEM|ALL specifies how console messages that indicate when parmlib limits are reaching critical levels are to be displayed.	RAS Enhancements
DISPLAY OMVS	V2R9	**New operands:** The CINET operand displays the network routing information for the Common INET prerouter. The PFS operand displays information about the FILESYSTYPE, SUBFILESYSTYPE, and NETWORK statements.	File system
	V2R10	**New operands:** The LIMITS keyword displays information about OS/390 UNIX System Services parmlib limits and current system usage. With the PID= keyword, LIMITS displays information for an individual process. The RESET keyword, with D OMVS,LIMITS, resets the high-water marks for system limits to 0. The BRL operand, with D OMVS,PID=, displays thread-level information for any thread that is in a byte-range lock wait.	RAS Enhancements

Chapter 14. Customizing OS/390 UNIX

Customizing the BPXPRMxx Parmlib Members

The BPXPRMxx parmlib member contains the parameters that control processing and the file system. IBM recommends that you have two BPXPRMxx parmlib members, one defining the values to be used for system setup and the other defining the file systems. Using these two members makes it easier to migrate from one release to another, especially when using the ServerPac method of installation. *OS/390 MVS Initialization and Tuning Reference* contains a complete description of the BPXPRMxx statements.

When you complete your installation activities, you have one or two BPXPRMxx members, depending on whether you used ServerPac or CBPDO:

- With ServerPac, you receive two members, as IBM recommends.
- With CBPDO, after you complete all the instructions in the *OS/390 Program Directory*, you have the one member that you copied from SYS1.SAMPLIB.

In this case, you should define a second BPXPRMxx member so that the system setup parameters are in one member and the parameters that define the file systems are in the other.

Customize these BPXPRMxx members, according to the instructions in this section and the needs of your installation. When customizing, remember to use columns 1 through 71 for data; columns 72 through 80 are ignored.

Figure 1 shows the IBM-supplied BPXPRMXX member in SYS1.SAMPLIB for the current release.

```
MAXPROCSYS(900)
   MAXPROCUSER(25)
   MAXUIDS(200)
   MAXFILEPROC(2000)
   MAXPTYS(800)
   CTRACE(CTIBPX00)
  /*STEPLIBLIST('/etc/steplib')        */
  /*USERIDALIASTABLE('/etc/tablename') */

  FILESYSTYPE   TYPE(HFS)
                ENTRYPOINT(GFUAINIT)
                PARM(' ')

/* FILESYSTYPE  TYPE(AUTOMNT)        */
/*              ENTRYPOINT(BPXTAMD)  */

/* FILESYSTYPE  TYPE(TFS)            */
/*              ENTRYPOINT(BPXTFS)   */

/* FILESYSTYPE  TYPE(NFS)            */
/*              ENTRYPOINT(GFSCINIT) */
/*              ASNAME(MVSNFSC)      */
/*              PARM('biod(6)')      */

   ROOT     FILESYSTEM('OMVS.ROOT')
            TYPE(HFS)
            MODE(RDWR)
/* MOUNT FILESYSTEM('OMVS.USER.JOE')  */
/*          TYPE(HFS)                */
/*          MODE(RDWR)               */
/*          MOUNTPOINT('/u/joe')     */
/*          NOSETUID                 */
/*          WAIT                     */
/*          SECURITY                 */

   FILESYSTYPE TYPE(UDS) ENTRYPOINT(BPXTUINT)
   NETWORK DOMAINNAME(AF_UNIX)
           DOMAINNUMBER(1)
           MAXSOCKETS(200)
           TYPE(UDS)
   FILESYSTYPE TYPE(INET) ENTRYPOINT(EZBPFINI)
   NETWORK DOMAINNAME(AF_INET)
           DOMAINNUMBER(2)
           MAXSOCKETS(2000)
           TYPE(INET)
 /* FILESYSTYPE TYPE(CINET) ENTRYPOINT(BPXTCINT) */
 /* NETWORK DOMAINNAME(AF_INET)                  */
 /*         DOMAINNUMBER(2)                      */
 /*         MAXSOCKETS(2000)                     */
 /*         TYPE(CINET)                          */
 /*         INADDRANYPORT(2000)                  */
 /*         INADDRANYCOUNT(325)                  */
```

Figure 1. BPXPRMXX Parmlib Member in SAMPLIB (Part 1 of 2)

```
/* SUBFILESYSTYPE NAME(LINET)          */
/*                TYPE(CINET)          */
/*                ENTRYPOINT(BPXTLINT) */

/* SUBFILESYSTYPE NAME(TCPIP)          */
/*                TYPE(CINET)          */
/*                ENTRYPOINT(EZBPFINI) */
/*                DEFAULT              */

/* SUBFILESYSTYPE NAME(TCPIP2)         */
/*                TYPE(CINET)          */
/*                ENTRYPOINT(EZBPFINI) */

  MAXTHREADTASKS(1000)
  MAXTHREADS(200)
/*PRIORITYPG (n,...,n)*/
/*PRIORITYGOAL (n,...,n)*/

   IPCMSGNIDS    (500)
   IPCMSGQBYTES  (2147483647)
   IPCMSGQMNUM   (10000)
   IPCSHMNIDS    (500)
   IPCSHMSPAGES  (262144)
   IPCSHMMPAGES  (25600)
   IPCSHMNSEGS   (500)
   IPCSEMNIDS    (500)
   IPCSEMNSEMS   (1000)
   IPCSEMNOPS    (25)
   MAXMMAPAREA(40960)

/* MAXFILESIZE(1000)  */

   MAXCORESIZE(4194304)
   MAXASSIZE(209715200)
   MAXCPUTIME(1000)
   MAXSHAREPAGES(131072)
   FORKCOPY(COW)
   SYSPLEX(NO)
   SUPERUSER(BPXROOT)
   TTYGROUP(TTY)
   STARTUP_PROC(OMVS)

 /* STARTUP_EXEC('Dsname(Memname)',SysoutClass) */
 /* RUNOPTS('runtime options') */

   SYSCALL_COUNTS(NO)
   MAXQUEUEDSIGS(1000)
   SHRLIBRGNSIZE(67108864)
   SHRLIBMAXPAGES(4096)
   LIMMSG(NONE)
```

Figure 1. BPXPRMXX Parmlib Member in SAMPLIB (Part 2 of 2)

Defining File Systems:

Defining System Limits: You can customize your BPXPRMxx parmlib member to provide the performance needed for the way your installation uses kernel services.

Table 3 on page 9 lists the system-wide and process-level limits that can be set in the BPXPRMxx parmlib member.

Table 3. System-Wide and Process-Level Limits

System-Wide Limits	Process-Level Limits
MAXPROCSYS	MAXPROCUSER
MAXCPUTIME	MAXFILEPROC
MAXUIDS	MAXTHREADTASKS
MAXPTYS	MAXTHREADS
MAXRTYS	MAXQUEUEDSIGS
MAXMMAPAREA	MAXFILESIZE
MAXSHAREPAGES	MAXCORESIZE
MAXASSIZE	
IPCMSGNIDS	
IPCSHMNIDS	
IPCSHMSPAGES	
IPCSEMNIDS	
IPCMSGQBYTES	
IPCMSGQMNUM	
IPCSHMMPAGES	
IPCSHMNSEGS	
IPCSEMNSEMS	
IPCSEMNOPS	
FORKCOPY	
STEPLIBLIST	
USERIDALIASTABLE	
PRIORITYPG	
PRIORITYGOAL	
SYSCALL_COUNTS	
NOARGS	
SUPERUSER	
TTYGROUP	
SHRLIBMAXPAGES	
VERSION	
LIMMSG	

CTRACE: Use the CTRACE statement to provide tracing while the kernel is starting and to avoid having to issue a TRACE operator command to set tracing options. See “CTnBPXxx Parmlib Member to Control Tracing” for information about specifying your customized component trace parmlib members.

The only way to change any CTRACE value is with the TRACE command. You cannot use the SETOMVS or SET OMVS command to change the value.

LIMMSG: Use the LIMMSG statement to control the display of console messages that indicate when parmlib limits are reaching critical levels. For more information, see “Displaying the Status of OS/390 UNIX Parmlib Limits” on page 28.

MAXASSIZE: MAXASSIZE is the maximum region size (in bytes) for an address space. You can set a system-wide limit in BPXPRMxx and then set higher limits for individual users. Use the RACF ADDUSER or ALTUSER command to specify the ASSIZEMAX limit on a per-user basis as follows:

```
ALTUSER userid OMVS(ASSIZEMAX(nnnn)
```

MAXCPUTIME: MAXCPUTIME is the time limit (in seconds) for processes that were created by **rlogind** and other daemons. You can set a system-wide limit in BPXPRMxx and then set higher limits for individual users. Use the RACF ADDUSER or ALTUSER command to specify the CPUTIMEMAX limit on a per user basis as follows:

```
ALTUSER userid OMVS(CPUTIMEMAX(nnnn))
```

MAXFILEPROC: Use MAXFILEPROC to determine the number of character-special files, **/dev/fd***xx*, that a single process can have open concurrently. You can also limit the amount of system resources available to a single user process.

When selecting a value, consider the following factors:

- For conformance to standards, set MAXFILEPROC to at least 16 to conform to the POSIX standard or at least 25 to conform to the FIPS standard.

  It is recommended that you set this value to 256.
- The minimum value of 3 supports stdin, stdout, and stderr.
- The value must be larger than 3 to support shell users. If the value is too small, the shell may issue the message “File descriptor not available.” If this message occurs, increase the MAXFILEPROC value.

A process can change the MAXFILEPROC value using the setrlimit() function. Only processes with appropriate privileges can increase their limits.

You can set a system-wide limit in BPXPRMxx and then set higher limits for individual users. Use the RACF ADDUSER or ALTUSER command to specify the FILEPROCMAX limit on a per user basis as follows:

```
ALTUSER userid OMVS(FILEPROCMAX(nnnn))
```

“Dynamically Changing Certain BPXPRMxx Parameter Values” on page 22 explains how to dynamically change the MAXFILEPROC value.

MAXMMAPAREA: For MAXMMAPAREA, you can set a system-wide limit in BPXPRMxx and then set higher limits for individual users. Use the RACF ADDUSER or ALTUSER command to specify the MMAPAREAMAX limit on a per user basis as follows:

```
ALTUSER userid OMVS(MMAPAREAMAX(nnnn))
```

MAXPROCSYS: You can manage system resources by limiting the number of processes that the system is to support. The values that you specify for MAXPROCSYS, MAXPROCUSER, and MAXUIDS are interrelated. When selecting a value for MAXPROCSYS, remember that these processes are needed:

- The initialization process (BPXOINIT)
- **/usr/sbin/init**, for starting and processing
- **exec sh** to run a shell script
- The process in which the shell script runs

Plan on one process for each daemon (for example, **inetd** and **cron**) that you start from a shell script such as **/etc/rc**. In addition, each shell user needs a minimum of three processes and possibly a few more for piping between shell commands.

Do not specify a higher value for MAXPROCSYS than your system can support because most processes use an entire MVS address space. This value will vary, depending on your environment. If you set the value too high, failures (EAGAIN) for fork or spawn might occur because WLM could not provide enough fork initiators.

"Dynamically Changing Certain BPXPRMxx Parameter Values" on page 22 explains how to dynamically change the MAXPROCYS value.

For an example of MAXPROCSYS, MAXPROCUSER, MAXRTYS, MAXPTYS, and MAXUIDS settings in BPXPRMxx, see "Tuning Process Activity" on page 39.

MAXPROCUSER: To improve performance, use MAXPROCUSER to limit user activity. For a typical shell user who starts up 1 to 3 shells, set the limit to 10.

When selecting a value, consider the following factors:

- Set MAXPROCUSER to at least 16 to conform to the POSIX standard for CHILD_MAX, or to at least 25 to conform to the FIPS standard.
- A low MAXPROCUSER value limits the number of concurrent processes that a user can run. A low value limits a user's consumption of processing time, virtual storage, and other system resources.
- Some daemons or users run without UID(0), and may create many address spaces. In these cases, give the daemon ID a high enough PROCUSERMAX value in the OMVS segment.

A user with a UID of 0 is not limited by the MAXPROCUSER value because a superuser may need to be able to log on and use kernel services to solve a problem.

Though not recommended, the security administrator can give the same OMVS UID to more than one TSO/E user ID. Therefore, the number of users can be greater than the number of UIDs that are defined. Check with the security administrator; if users share UIDs, you will need to define a greater number of processes for each user.

You can set a system-wide limit in BPXPRMxx and then set higher limits for individual users. Use the RACF ADDUSER or ALTUSER command to specify the PROCUSERMAX limit on a per-user basis as follows:

```
ALTUSER userid OMVS(PROCUSERMAX(nnnn))
```

MAXPTYS: Use MAXPTYS to manage the number of interactive shell sessions, where each interactive session requires one pseudo-TTY pair. Do not specify an arbitrarily high value for MAXPTYS. But, because each user may have more than one session, it is recommended that you allow four pseudo-TTY pairs for each user (MAXUIDS * 4). Specify a MAXPTYS value that is at least twice the MAXUIDS value.

"Dynamically Changing Certain BPXPRMxx Parameter Values" on page 22 explains how to dynamically change the MAXPTYS value.

MAXRTYS: MAXRTYS enables you to manage the number of interactive shell sessions that are accessed by Communications Server terminal support. When you specify this value, each interactive session requires one remote TTY. Avoid

specifying an arbitrarily high value for MAXRTYS. However, because each user may have more than one session, you should allow four remote TTY files for each user (MAXUIDS * 4).

The MAXRTYS value influences the configuration of Communications Server nodes and associated terminal files.

"Dynamically Changing Certain BPXPRMxx Parameter Values" on page 22 explains how to dynamically change the MAXPROCYS value.

MAXTHREADS: MAXTHREADS is the maximum number of threads that a single process can have active concurrently. If an application needs to create more than the recommended maximum in SAMPLIB, it must minimize storage allocated below the 16M line by specifying C run-time options. For information on the set_thread_limit service (BPX1STL), refer to *OS/390 UNIX System Services Programming: Assembler Callable Services Reference*.

You can set a system-wide limit in BPXPRMxx and then set higher limits for individual users by using the RACF ADDUSER or ALTUSER command to specify the THREADSMAX limit on a per user basis as follows:

```
ALTUSER userid OMVS(THREADSMAX(nnnn))
```

MAXTHREADTASKS: MAXTHREADTASKS is the maximum number of MVS tasks that a single process can have concurrently active.

A high MAXTHREADTASKS value may affect storage and performance. Each task requires additional storage for the following:

- The control blocks built by the kernel
- The control blocks and data areas required by the run-time library
- System control blocks such as the TCB and RB

MAXUIDS: MAXUIDS limits the number of active UIDs. When you select a value for MAXUIDS, consider the following factors:

- Because users are likely to run with three or more concurrent processes each, they require more system resources than typical TSO/E users.
- If the MAXUIDS value is too high relative to the MAXPROCSYS value, too many users can invoke the shell. All users may be affected, because forks might begin to fail.

  For example, if your installation can support 400 concurrent processes—MAXPROCSYS(400)—and each UID needs an average of 4 processes, then the system can support 100 users. For this operating system, specify MAXUIDS(100).

PRIORITYGOAL: If you are using your system to run a critical real-time application program, set the performance groups or service classes to meet the needs of the application program. It is difficult to run both real-time application programs and general users on the same OS/390 UNIX system. There is no mechanism to restrict any set of users from access to the nice() and setpriority() functions. For more information, see "nice(), setpriority(), and chpriority()" on page 41.

PRIORITYPG: If you are using your system to run a critical real-time application program, set the performance groups or service classes to meet the needs of the application program. It is difficult to run both real-time application programs and general users on the same OS/390 UNIX system. There is no mechanism to restrict

any set of users from access to the nice() and setpriority() functions. For more information, see “nice(), setpriority(), and chpriority()” on page 41

STEPLIBLIST: With STEPLIBLIST, programs can have temporary access to files that are not normally accessible to other users. Step libraries have many uses; one is so that selected users can test new versions of run-time libraries before the new versions are made available to everyone on the system. Customers who do not put the Language Environment run-time library SCEERUN into the linklist should put the SCEERUN data set name in this file.

If your installation runs programs that have the setuid or setgid bit turned on, only those load libraries that are found in the STEPLIBLIST sanction list are set up as step libraries in the environment that those programs will run in. Because programs with the setuid or setgid bit turned on are considered privileged programs, they must run in a controlled environment. The STEPLIBLIST sanction list provides this control by allowing those programs to use only the step libraries that are considered trusted by the installation.

IBM recommends that the pathname of the file be **/etc/steplib**. This fits in with the IBM strategy to place all customized data in the **/etc** directory.

If you do not specify a value for STEPLIBLIST, step libraries will not be set up for set-user-ID and set-group-ID executable files.

These step libraries are set up as a result of the invocation of a HFS executable file using the exec service (BPX1EXC), the attach_exec service (BPX1ATX) or spawn (BPX1SPN) service. After one of those services has been invoked, the step libraries can be propagated from the calling task’s environment. They can also be specified by using the STEPLIB environment variable that is passed to the exec service. When the exec service invokes a set-user-ID or set-group-ID executable file, only those libraries that are found in the sanctioned list are set up as step libraries in the environment that the executable file will run in.

The following is a list of formatting rules for the STEBLIBLIST file that contains the sanctioned list:

- You can include comment lines in the list. Each comment line must start with /* and end with */.
- You must follow standard MVS data set naming conventions in naming the files in the list.
- Each data set name must be fully qualified and cannot be enclosed in quotation marks.
- Each data set name must be on a line by itself, with no comments.
- You must use uppercase letters for data set names.
- You can put blanks before and after each data set name. Entirely blank lines in the list are ignored.
- You can use the * character to specify multiple files that begin with the same characters. For example, if you list SYS1.*, you are sanctioning any file that begins with SYS1. as a step library.

If the file does not follow these formatting rules, the sanctioned list is not built using the file.

You should catalog each data set listed in the file to prevent user versions of the data set from being used.

Following is a sample sanctioned list file:

```
/*********************************************************************/
/*                                                                   */
/*   Name: Sample Sanctioned List for set-user-ID and set-group-ID   */
/*         files                                                     */
/*                                                                   */
/*   Updated by:   May only be updated by OSTEPLIB TSO/E command     */
/*                                                                   */
/*   Description:  Contains a list of data set names that may        */
/*                 be used as STEPLIB libraries for SETUID           */
/*                 programs                                          */
/*                                                                   */
/*                 Wild cards may be used to specify multiple        */
/*                 data set names that have the same prefix          */
/*                 characters.                                       */
/*                                                                   */
/*********************************************************************/

/*********************************************************************/
/* Sanction all data set names beginning with CEE.SCEERUN            */
/*********************************************************************/
CEE.SCEERUN*
```

You can create or update the sanctioned list file using the OSTEPLIB command, which specifies read and execute permissions for all users (permissions 555). The sanctioned list file must be protected from update by nonprivileged users; therefore, only users with superuser authority should be given update access to it.

Because a working copy of the sanctioned list is maintained in storage, an update to the file will take effect when the next setuid(0) program is run from a process with read access to the stepliblist file.

Use the SETOMVS or SET OMVS command to dynamically change the value of STEPLIBLIST; this changes the current system settings. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

USERIDALIASTABLE: On most UNIX systems, you use lowercase IDs. With OS/390 UNIX, typically you will use the uppercase user IDs and group names specified in your security database. In some cases, however, you may want to use lowercase or mixed case names in OS/390 UNIX processing. To do that, you need to create a user ID alias table to associate lowercase or mixed case alias names with uppercase OS/390 user ID and group names.

IBM recommends that the pathname of the file be **/etc/tablename**. This fits in with the IBM strategy to place all customized data in the **/etc** directory. If a value for USERIDALIASTABLE is not specified, alias names are not used.

Using the USERIDALIASTABLE statement degrades performance slightly. The more names that you define, the greater the performance degradation. Installations are encouraged to continue using uppercase-only userids and group names defined in their security databases.

Following is a list of formatting rules for the userid alias table:

- You can include comment lines in the list. Each comment line must start with /* and end with */.
- You must follow standard MVS userid and group name naming conventions in the first column.

- You must follow XPG4 standard naming conventions in the second column.
- Do not enclose the names in quotation marks.
- Each userid or group name and associated alias name must be on a line by itself, with no comments.
- The MVS userids and group names must be located in columns 1-8 and the associated aliases must be located on the same line in columns 10-17.
- The MVS name and the alias name must be separated by 1 or more blanks.
- The tags :userids and :groups must be used to delineate between userids and group names.
  - If no tags are present in the file, then all names in the file are assumed to be userids.
  - If there are any names listed before a tag, those names are considered to be userids.
  - If a :userids tag is present, then all name lines following it and up to the next tag are considered to be userids.
  - If a :groups tag is present, then all name lines following it and up to the next tag are considered to be group names.
  - If specified, the tag must start in column 1.
  - The tag names are not case sensitive.

If the file does not follow these formatting rules, the alias name may not be recognized and various functions relating to the attempted use of the alias may fail.

Following is a sample userid and group name alias table:

```
/*************************************************************/
/*                                                           */
/*   Name: Sample user ID/group name alias table             */
/*                                                           */
/*   Description:  Contains a list of MVS user IDs and their */
/*                 associated alias names.                   */
/*                                                           */
/* Alias names may be constructed from the following characters: */
/*                                                           */
/*       A B C D E F G H I J K L M N O P Q R S T U V W X Y Z */
/*       a b c d e f g h i j k l m n o p q r s t u v w x y z */
/*       0 1 2 3 4 5 6 7 8 9 . _ -                           */
/*                                                           */
/*       The hyphen shall not be used as the first character. */
/*                                                           */
/*************************************************************/

/*************************************************************/
/* Mixed case group names                                    */
/*************************************************************/
 :Groups
 DEPTD10  DeptD10
 DEPTD20  DeptD20

/*************************************************************/
/* Non-alphanumeric alias user IDs and group names           */
/*************************************************************/
 :UserIDs
/*************************************************************/
/* Mixed case alias names                                    */
/*************************************************************/
 MYUSERID MyUserid

/*************************************************************/
/* Easier to remember alias names                            */
/*************************************************************/
 K61XDLBC Daniel

 JOEDOE   Joe_Doe
 MRDOE    Mr.Doe
 ABCD     A-B-C-D
 :groups
 DEVEL    OE-Dev
 TEST     OE_Test
```

For UUCP, you must set up userid UUCP or define uucp as an alias. Likewise, you must set up group ID UUCPG or define uucpg as an alias.

For more information, refer to Chapter 19, “Customizing the OS/390 UNIX Shell” and Chapter 22, “Configuring the UNIX-to-UNIX Copy Program (UUCP).”

The userid/group name alias table must be protected from update by non-privileged users; therefore, only users with superuser authority should be given update access to it. All users should be given read access to the file.

Once a user is logged into the system, changing the userid/group name alias table does not change the alias name immediately. Database queries, however, will yield the new alias if the userid performing the query has read/execute access to the userid/group name alias table. The table is checked every 15 minutes and refreshed if it has been changed. If a change needs to be activated sooner, you can use the SETOMVS or SET OMVS command. See “Dynamically Changing the BPXPRMxx Parameter Values” on page 21 for more information.

Chapter 23. Managing Operations

OS/390 UNIX is designed to be continually available. This chapter discusses these tasks, which are done by operators.

For information about the CANCEL, DISPLAY, MODIFY MSGRT, and TRACE operator commands, see *OS/390 MVS System Commands*.

Stopping Processes

There are three ways to stop a process:

- The operator enters a MODIFY operator command to terminate a process.
- A shell user enters the **kill** command to cancel processes.
- The operator enters a CANCEL command to stop an address space containing a process. If the address space contains multiple processes, CANCEL terminates all of the processes.

Terminating a Process with the MODIFY Command: If a process is hung, the operator can enter one of these two MODIFY console commands to terminate the process:

- To allow the signal interface routine to receive control before the process is terminated, issue:

  ```
  F BPXOINIT,TERM=pppp
  ```

  where `pppp` is the process identifier.
- Sometimes a process is not terminated when a TERM request is sent. In these cases, issue:

  ```
  F BPXOINIT,FORCE=pppp
  ```

  where `pppp` is the process identifier.

Terminating a Process with the kill Command: The best way to end a process is to issue the **kill** command. Using the DISPLAY OMVS operator command or the **ps** command, display all the active processes. Then issue the **kill** command, specifying the signal and the PID (process identifier) for the process.

Start by sending a SIGTERM signal:

```
kill -s TERM pid
```

where `pid` is the process identifier. If that does not work, try sending a SIGKILL signal:

```
kill -s KILL pid
```

where pid is the process identifier.

Terminating a Process with the CANCEL Command: An operator can cancel all processes or selected processes in an address space. To cancel all processes, use the CANCEL command. Before issuing CANCEL, display all processes running in that address space and the address space identifier by issuing:

```
DISPLAY OMVS,A=xxxx
```

If there is only one process in the address space or if you want to terminate all the processes, issue:

```
CANCEL name,A=asid
```

For example, for a user with a TSO/E userid of JOE, Figure 2 shows how to obtain the ASIDs for the user's work and then cancel the user's process that is running the **sleep 6000** shell command.

```
display omvs,u=joe
BPXO001I 17.12.23 DISPLAY OMVS 361

OMVS      ACTIVE             OMVS=(93)
USER     JOBNAME  ASID        PID       PPID  STATE   START    CT_SECS
JOE      JOE      001D          5          1  1RI   17.00.10     1.203
JOE      JOE3     001B     131076     262147  1SI   17.00.10      .111
  LATCHWAITPID=          0 CMD=sleep 6000
JOE      JOE1     0041     262147          5  1WI   17.00.10      .595
  LATCHWAITPID=          0 CMD=-sh

cancel joe3,a=1b
```

Figure 2. Console Display for a CANCEL Command

If you want to terminate one or more selected processes in an address space, but not all the processes, then use the MODIFY command as described in "Terminating a Process with the MODIFY Command" on page 17 or the **kill** command as described in "Terminating a Process with the kill Command" on page 17.

Terminating Threads with the MODIFY Command

An operator can terminate a thread, without disrupting the entire process. The syntax of the MODIFY command to terminate a thread is:

```
F BPXOINIT,{TERM}=pid[.tid]
           {FORCE}
```

where

- pid indicates the process identifier (PID) of the thread to be terminated. The PID is specified in decimal form as displayed by the D OMVS command.
- tid indicates the thread identifier (TID) of the thread to be terminated. The TID is 16 hexadecimal (0-9,A-F) characters as displayed by the following command:

  ```
  D OMVS,PID=pppppppp
  ```

- TERM= indicates the signal interface routine will be allowed to receive control before the thread is terminated.
- FORCE= indicates the signal interface routine will not be allowed to receive control before the thread is terminated.

Although abnormal termination of a thread usually causes a process to terminate, using the MODIFY command to terminate a thread will not cause the process to terminate.

You will typically want to terminate a single thread when the thread represents a single user in a server address space. Otherwise, random termination of threads can cause some processes to hang or fail.

If a thread in a process is hung, the operator can enter one of these two MODIFY console commands to terminate the thread without terminating the entire process. We recommend that you use the `TERM` keyword first, and if that does not succeed, use `FORCE`:

- To allow the signal interface routine to receive control before the thread is terminated, use:

  ```
  F BPXOINIT,TERM,PID=pppppppp.ttttttttttttttt
  ```

  where `pppppppp` is the process identifier and `ttttttttttttttt` is the thread identifier.
- To terminate the thread without allowing the signal interface routine to receive control, use:

  ```
  F BPXOINIT,FORCE,PID=pppppppp.ttttttttttttttt
  ```

  where `pppppppp` is the process identifier and `ttttttttttttttt` is the thread identifier.

Shutting Down OS/390 UNIX

This section explains how to shut down OS/390 UNIX. When you are doing a planned shutdown and will be re-IPLing the system, issue the following operator command:

```
F BPXOINIT,SHUTDOWN=FORKINIT
```

"Planned Shutdowns" describes the procedure. If you want to shut down the system as part of JES2 maintenance and do not want to re-IPL the system, use the following operator command:

```
F BPXOINIT,SHUTDOWN=FORKS
```

"Partial Shutdowns (for JES2 Maintenance)" on page 20 describes the procedure.

Planned Shutdowns: As part of a planned shutdown, you should clean up the system first before re-IPLing.

1. Use the operator SEND command to send a note to all TSO/E users telling them that the system will be shut down at a certain time. For example:

   ```
   send 'The system is being shut down in five minutes. Log off.',NOW
   ```
2. Use the **wall** command to send a similar note about the impending shutdown to all logged-on shell users. For example:

   ```
   wall The system is being shut down in five minutes. Please log off.
   ```
3. Prevent new TSO/E logons and shut down other OS/390 subsystems (such as CICS and IMS), following your usual procedures.
4. Shut down all JES initiators.
5. Unmount all NFS-mounted file systems as part of the normal shutdown process.

6. Use normal shutdown procedures to terminate all file system address spaces such as TCP/IP and DFSS. Do this after the final warning has been sent to users that the system is terminating.
7. Terminate running daemons such as **inetd**. To get a list of daemons that are running, issue, for example:

   ```
   D OMVS,U=OMVSKERN
   ```

   In this example, `OMVSKERN` is the userid that is used for the kernel and daemons. In addition, you can display all processes (most daemons will have recognizable names) by issuing:

   ```
   D OMVS,A=ALL
   ```

   Then use the `F BPXOINIT,PID=xxxxxxxx` operator command or the **kill** command to terminate those processes.
8. Terminate any remaining processes and unmount all file systems (including the root file system) by using the bpxstop tool. It is available from the tools and toys page on the OS/390 UNIX web site.

   ```
   http://www.ibm.com/s390/unix/
   ```
9. Take down JES. At this point, there may still be a number of initiators that are provided by WLM for use on fork and spawn. These initiators time out after 30 minutes on their own. To terminate the initiators, you can issue the following operator command:

   ```
   F BPXOINIT,SHUTDOWN=FORKINIT
   ```
10. After all the processes have been terminated, you can do any of the following:
    - IPL
    - Power off
    - Take down JES, restart JES, and then rebuild your environment. For example:
      - Remount any file systems that you unmounted. To do all the mounts, you must issue mount commands or construct a REXX exec or CLIST. If you are using automount for user file systems, there will be less work involved.
      - If you terminated the address spaces for TCP/IP and DFSS, you must restart these.
      - If you terminated daemons, logon to TSO as superuser and run **/etc/rc** from a shell or from the ISHELL.
      - Notify users that the system is once again available for UNIX processing.

Partial Shutdowns (for JES2 Maintenance): Before JES2 can be shut down for maintenance purposes, part of OS/390 UNIX must be shut down. This section explains how you can terminate all of the forked processes without having to re-IPL the entire system. (The kernel remains active but new forked processes are not allowed.) Use this procedure for JES2 maintenance only.

Do the partial shutdown as infrequently as possible because it is a disruptive shutdown; all the user processes that are either forked or non-local spawned are terminated.

After the forked processes have been terminated, you can terminate the colony address space. Now JES2 can be shut down for maintenance. OS/390 UNIX can

be reinitialized after JES2 has been restarted, and forked processes will start being dubbed again. The file system colonies can then be restarted manually. The following steps describe the procedure:

1. Use the operator SEND command to send a note to all TSO/E users telling them that the system will be shut down. For example:

   ```
   send 'The system is being shut down in five minutes. Please log off.'
   ```

2. Use the **wall** command to send a similar note to all logged-on shell users:

   ```
   wall The system is being shut down in five minutes.  Please log off.
   ```

3. Issue the following operator command to begin the shutdown of OS/390 UNIX.

   ```
   F BPXOINIT,SHUTDOWN=FORKS
   ```

   This terminates all forked and non-local spawned address spaces on the system. If the operator receives a success message, the shutdown can be continued.

   A failure message means that some forked processes or non-local spawned address spaces could not be terminated. Try to find these processes by issuing:

   ```
   D OMVS,A=ALL
   ```

   To terminate them, issue:

   ```
   F BPXOINIT,FORCE,PID=xxxxxxxx
   ```

   If that does not work, use the CANCEL or FORCE operator commands.

4. Terminate the file system colonies. Use normal shutdown procedures to close all file system address spaces such as Network File System Client (NFSC) and the Distributed File System Cache Manager (DFSCM).

   For NFSC, determine what the process name was used to start this colony. Use this name to cancel it. (For example, `C NFSC`.)

   For DFSCM, use the procedure in *OS/390 Distributed File Service DFS Administration Guide and Reference* to stop the DFS Cache Manager. Issue `STOP DFSCM` to stop DFSCM.

   For all other colonies, use the procedures documented in their publications.

5. Now you can do whatever corrective or maintenance actions that were needed for JES2, such as restarting it.

6. To restart OS/390 UNIX, issue the Modify (F) command.

   ```
   F BPXOINIT,RESTART=FORKS
   ```

7. Restart the file system address spaces.

   For NFSC, you have to respond to the operator message BPXF014D issued when the colony was taken down. Then reissue all the mounts.

   For DFSCM, respond to the operator message BPXF014D.

   For all other colonies, use the procedures they have documented in their product publications.

Dynamically Changing the BPXPRMxx Parameter Values

The SETOMVS command enables you to modify BPXPRMxx parmlib settings without re-IPLing. For example:

```
SETOMVS MAXTHREADTASKS=100,MAXPROCUSER=8
```

You can dynamically change process-wide limits separately for each process. For example:

```
SETOMVS PID=123,MAXFILEPROC=200
```

The SET OMVS command enables you to dynamically change the BPXPRMxx parmlib members that are in effect. Because you can have multiple BPXPRMxx definitions, you can easily reconfigure a large set of the system characteristics. You can keep the reconfiguration settings in a permanent location for later reference or reuse. A sample SET OMVS command is:

```
SET OMVS=(AA,BB)
```

If a parameter is specified more than once with different values, in the parmlib members, the first value specified is the first value that is used. For example, if you specify SET OMVS=(AA,BB) where AA has a MAXPROCUSER=10 value and BB has a MAXPROCUSER=5 value, MAXPROCUSER =10 is used.

You can use the SETOMVS RESET command to dynamically add the FILESYSTYPE, NETWORK, and SUBFILESYSTYPE statements without having to re-IPL. However, if you change the values, a re-IPL will be necessary. For more information, see “Dynamically Adding FILESYSTYPE Statements in BPXPRMxx” on page 24.

See *OS/390 MVS System Commands* for a complete description of the SET OMVS and SETOMVS commands.

You can use the SETOMVS SYNTAXCHECK operator command to check the syntax of a BPXPRMxx parmlib member before doing an IPL. (You cannot use that command to verify whether HFS datasets or mount points are valid.)

Dynamically Changing Certain BPXPRMxx Parameter Values: The MAXPROCSYS, MAXPTYS, MAXRTYS, MAXFILEPROC, IPCMSGNIDS, IPCSEMNIDS, IPCSHMNIDS, and IPCSHMSPAGES specify maximum values. You can use the SETOMVS or SET OMVS command to dynamically increase the current system setting, but if you specify a value that is too low or too high, you will get an error message. To use a value outside the range, you will need to change the specification in BPXPRMxx and re-IPL.

To avoid specifying a value that is too low or too high, you can use a formula to calculate the maximum values. The minimum value is sometimes the current setting of the parameter and sometimes lower than that, as identified in the description of each parameter. The formula for each parameter is described later in this section.

The following example shows you how to perform the calculations using the IPCMSGNIDS parameter, which determines the highest number of unique message queues in the system. To use SETOMVS IPCMSGNIDS=*xxx* to increase the current setting, you must calculate the highest number that you can specify. According to the description of IPCMSGNIDS in “IPCMSGNIDS and IPCSEMNIDS” on page 23, the formula is:

```
MIN(20000,MAX(4096,3*initial value))
```

For this example, the current value of IPCMSGNIDS is 1000; the value of IPCMSGNIDS at IPL is also 1000 (that is, 1000 is the initial value). Use the formula in the following way:

1. Compare 4096 with 3 times 1000 to find the higher number (the MAX). 4096 is the higher number.
2. Compare 20000 with 4096 to find the smaller number (the MIN). 4096 is the smaller number.

Therefore, the highest number that you can specify on SETOMVS IPCMSGNIDS is 4096. The range of numbers that you can specify is 1000 (the current value) to

4096. The correct SETOMVS command for increasing the message queue limit to the maximum (assuming a starting value of 1000) would be:

```
SETOMVS IPCMSGNIDS=4096
```

To change to a number higher than 4096 (but lower than 20000), you will have to change BPXPRMxx and re-IPL.

MAXPROCSYS: The range that you can use has a minimum value of 5; the maximum value is based on the following formula:

```
MIN(32767,MAX(4096,3*initial value)
```

The initial value is the MAXPROCSYS value that was specified during BPXPRMxx initialization. You cannot use a value less than 5. If you want to use a value greater than the current maximum (as calculated by the formula) but lower than the initial maximum (32767), you will have to change the value in BPXPRMxx and re-IPL.

MAXPTYS: The range's minimum value is 1 and the maximum is based on the following formula:

```
MIN(10000,MAX(256,2*initial value)
```

The initial value is the MAXPTYS value that was specified during BPXPRMxx initialization.

MAXRTYS: The range's minimum value is the current setting of MAXRTYS, and the maximum is based on the following formula:

```
MIN(10000,MAX(256,2*initial value)
```

The initial value is the MAXRTYS value that was specified during BPXPRMxx initialization. If you want to use a value greater than the current maximum (as calculated by the formula) but lower than the initial maximum (10000), you will have to change the value in BPXPRMxx and re-IPL.

IPCMSGNIDS and IPCSEMNIDS: The range's minimum value is the current setting of IPCMSGNIDS or IPCSEMNIDS, and the maximum is based on the following formula:

```
MIN(20000,MAX(4096,3*initial value)
```

The initial value is the value that was specified during BPXPRMxx initialization. If you want to use a value greater than the current maximum (as calculated by the formula) but lower than the initial maximum (20000), you will have to change the value in BPXPRMxx and re-IPL.

IPCSHMNIDS and IPCSHMSPAGES: The range's minimum value is the current setting of IPCMSGNIDS or IPCSHMSPAGES, and the maximum is based on the following formula:

```
MIN(20000,MAX(4096,3*initial value)
```

The initial value is the value that was specified during BPXPRMxx initialization. If you want to use a value greater than the current maximum (as calculated by the formula) but lower than the initial maximum (20000), you will have to change the value in BPXPRMxx and re-IPL.

Dynamically Switching to Different BPXPRMxx Members: Another way to dynamically reconfigure parameters is to use the SET OMVS command to change the BPXPRMxx parmlib members that are in effect. With the SET OMVS command,

you can have multiple BPXPRMxx definitions and use them to easily reconfigure a set of the OS/390 UNIX system characteristics. You can keep the reconfiguration settings in a permanent location for later reference or reuse.

For example, you could keep the system limits parameters that can be reconfigured in parmlib member BPXPRMLI. When you need to change any of the limits, edit the parmlib member and then issue SET OMVS. For example:

```
SET OMVS=(LI)
```

Changes to system limits (for example, MAXPROCSYS) take effect immediately. Changes to user limits (for example, MAXTHREADS) are set when a new user enters the system (for example, **rlogin** or a batch job). These limits persist for the length of the user connection to OS/390 UNIX.

Dynamically Adding FILESYSTYPE Statements in BPXPRMxx: Use the SETOMVS RESET command to dynamically add the FILESYSTYPE, NETWORK, and SUBFILESYSTYPE statements without having to re-IPL. If you want to change the values, you will have to edit the BPXPMRxx member that is used for IPLs. You can also dynamically add the parmlib statements currently supported by SETOMVS, such as MAXPROCSYS.

To display information about the current FILESYSTYPE, NETWORK, or SUBFILESYSTYPE statements, issue the following command:

```
DISPLAY OMVS,PFS
```

The following section shows examples of some of the more common configuration changes, adding the HFS and adding sockets. The examples discuss:

1. Activating the HFS file system for the first time.
2. Activating a single sockets file system for the first time.
3. Activating multiple sockets file systems for the first time with Common INET.
4. Adding another sockets file system to an existing common INET configuration.
5. Changing the MAXSOCKETS value.

Activating the HFS File System for the First Time: To activate the HFS file system for the first time, do the following:

1. Set up a root HFS dataset.
2. Create a temporary BPXPRM*tt* member that has the following statement:
   ```
   FILESYSTYPE TYPE(HFS) ENTRYPOINT(GFUAINIT)
   ```
3. Issue `SETOMVS RESET(tt)`.
4. From TSO or the ISHELL, do the following:
   a. Unmount the current root file system.
   b. Mount the root HFS dataset as the new root file system.
   c. Mount any additional HFS datasets as needed.
5. Add the following statements to the BPXPRMxx parmlib member used on IPL:
   a. The FILESYSTYPE statement used above.
   b. A ROOT statement for the root HFS.
   c. MOUNT statements for the additional mounts that should be done initially.

Activating a Single Sockets File System for the First Time: This example explains how to activate a single sockets file system for the first time. It uses the SecureWay

TCP/IP Socket File System for network sockets and also brings up support for local sockets. The MAXSOCKETS value used is just an example; the value that you use may be different.

1. Create a temporary BPXPRM*tt* member with the following statements:

```
/* Start Address Family AF_INET for Network Sockets /*
FILESYSTYPE TYPE(INET) ENTRYPOINT(EZBPFINI)
NETWORK TYPE(INET)    MAXSOCKETS(2000)
   DOMAINNAME(AF_INET)  DOMAINNUMBER(2)

/* Start Address Family AF_UNIX for Local Sockets  */
FILESYSTYPE TYPE(UDS)  ENTRYPOINT(BPXTUINT)
NETWORK TYPE(UDS)      MAXSOCKETS(1000)
  DOMAINNAME(AF_UNIX)  DOMAINNUMBER(1)
```

2. Issue `SETOMVS RESET(tt)`.
3. Start the TCPIP address space.
4. Add these parmlib statements to the BPXPRMxx member used on IPL.

Activating Multiple Sockets File Systems for the First Time with Common INET: This example shows how to activate multiple sockets file systems for the first time with Common INET. It starts two socket file systems, TCP/IP and AnyNet. Because they both support address family AF_INET, they are configured underneath Common INET to give applications the appearance of a single AF_INET socket file system.

Because this is an example of the initial configuration of sockets, the support for local, or AF_UNIX, sockets is also included for completeness.

1. Create a temporary BPXPRM*tt* member with the following statements:

```
/* Start Address Family AF_INET for Common INET */
FILESYSTYPE TYPE(CINET)  ENTRYPOINT(BPXTCINT)
NETWORK TYPE(CINET)    MAXSOCKETS(1000)
     DOMAINNAME(AF_INET)   DOMAINNUMBER(2)
     INADDRANYPORT(5000)  INADDRANYCOUNT(100)
/* Start TCP/IP and AnyNet under Common INET */
SUBFILESYSTYPE TYPE(CINET)  NAME(TCPIP)  ENTRYOINT(EZBPFINI) DEFAULT
SUBFILESYSTYPE TIME(CINET)  NAME(ANYNET) ENTRYPOINT(ISTOEPIT)
```

2. Issue `SETOMVS RESET(tt)`.
3. Start the TCPIP address space.
4. Start the Sockets Over SNA address space.
5. Add these parmlib statements to the BPXPRMxx member used on IPL.

The names used in the example, `TCPIP` and `ANYNET` must match those used when configuring the associated products.

Increasing the MAXSOCKETS Value: This example shuts down TCP/IP and brings it back up with a new value for MAXSOCKETS:

1. Shut down TCP/IP. For example:

   ```
   p tcpip
   ```

   Most socket programs and daemons will either terminate after TCP/IP is shut down or will tolerate a recycle of TCP/IP. There may be others that will have to be stopped manually.
2. Create a temporary BPXPRM*tt* member that has the following statements:

   ```
   NETWORK TYPE(INET) MAXSOCKETS(10000)
      DOMAINNAME(AF_INET)  DOMAINNUMBER(2)
   ```
3. Issue `SETOMVS RESET=(tt)`.

4. Restart TCP/IP. For example:`S TCPIP`.
5. Restart the socket programs and daemons, as necessary.
6. Update the MAXSOCKETS value in the BPXPRMxx member used on IPL.

Only the SecureWay Socket PFS, EZBPFINI, supports picking up a new MAXSOCKETS value when it is recycled.

The MAXSOCKETS value for a Common INET configuration can be changed with a similar procedure:

1. The TYPE() keyword of the NETWORK statement would specify the TYPE name of the Common INET PFS, which was “CINET” in the previous examples.
2. Common INET is not shut down, though, and the change takes effect in each TCP/IP stack when that stack was recycled.
3. INADDRANYPORT and INADDRANYCOUNT cannot be changed.

Adding Another Sockets File System to an Existing Common INET Configuration: This example starts a second SecureWay Sockets File System and uses names based on the previous examples.

1. Create a temporary BPXPRM*tt* member with the following statements:
   ```
   SUBFILESYSTYPE TYPE(CINET) NAME(TCPIP2) ENTRYPOINT(EZBPFINI)
   ```
2. Issue `SETOMVS RESET(tt)`.
3. Start the TCPIP2 address space.
4. Add this parmlib statement to the BPXPRMxx member used on IPL.

Tracing Events in OS/390 UNIX

To provide problem data, events are traced. When the OMVS address space is started, the trace automatically starts. The trace cannot be completely turned off.

Your installation specifies events to be traced in CTnBPXxx parmlib members. Each member should specify one or more events; keep the number of events small because tracing affects system performance. The installation can filter the events by address spaces, user IDs, and level of detail.

The CTnBPXxx member to be used when the OMVS address space is initialized is identified on the CTRACE parameter of the BPXPRMxx parmlib member. You also specify the size of the trace buffers in the CTnBPXxx member used when the system is IPLed. You can change the buffer size while OS/390 UNIX is running. The buffer can be 16KB minimum to 4MB maximum. If you need a different buffer size, change buffer size (BUFSIZE) in a CTnBPXxx member and issue:

```
TRACE CT,ON,COMP=SYSOMVS,PARM=CTnBPXxx
```

An operator starts and stops tracing events in the OS/390 UNIX system with the commands:

```
TRACE CT,ON,COMP=SYSOMVS,PARM=CTnBPXxx
TRACE CT,OFF,COMP=SYSOMVS
```

The operator can resume full tracing, with the previously used CTnBPXxx parmlib member or a different member, with the command:

```
TRACE CT,ON,COMP=SYSOMVS,PARM=CTnBPXxx
```

The PARM operand specifies the parmlib member with the tracing options.

Tracing DFSMS/MVS Events: You can also trace DFSMS/MVS events for the HFS. For example, to set up a trace, you can enter the following command:

```
TRACE CT,nnnnk,COMP=SYSSMS
R X,OPTIONS=(CALL,RRTN,CB,SUSP,EXITA,COMP=(ALL,NOIMF,NOSSF)),END
```

or:

```
TRACE CT,nnnnk,COMP=SYSSMS
R X,OPTIONS=(ENTRY,EXIT,EXITA,CB,COMP=(PFS,CDM)),END
```

Attention: SMS trace buffers are allocated in every initiator running kernel workloads. They are allocated in DREF ELSQA, which can cause a shortage of real pages.

For information about how to set up and use a trace, and for diagnosis information on interpreting a trace, see *OS/390 DFSMSdfp Diagnosis Reference*.

Re-creating Problems for IBM Service: If you are re-creating a problem for IBM service, it is generally a good idea to increase the OMVS CTRACE buffer size to 4MB. To do this, issue:

```
TRACE CT,4M,COMP=SYSOMVS,PARM=CTnBPXxx
```

with the parmlib member specifying the desired options. Alternatively, you could change the parmlib member to specify the desired buffer size. After you capture the dump for the problem, you can reset the trace buffer size to the original setting. Issue:

```
TRACE CT,xxxK,COMP=SYSOMVS
```

where *xxx*K is the size of the desired trace buffer.

Displaying the Status of the Kernel

Display information about the kernel or processes as follows:

- The operator enters a DISPLAY OMVS command to display the status of the kernel and processes.
- The operator enters the DISPLAY TRACE,COMP=SYSOMVS command to display the status of the kernel trace.
- A shell user enters the **ps** command or the PS ISHELL command to display the status of the user's processes.
- A superuser enters the **ps** command or the PS ISHELL command to display the status of all processes.

The operator displays the status for kernel services with the command:

```
DISPLAY OMVS
```

The command can be used to show information about a userid, about the parmlib members that are in effect, or about the current values of reconfigurable parmlib member settings.

To display the status of address spaces that the userid JANES is using and the processor resources used by each address space, the operator enters:

```
DISPLAY OMVS,U=JANES
```

For another example, see Figure 2 on page 18.

If the system IPLed with the specification of OMVS=(XX,YY,ZZ), the output for the D OMVS command is:

```
BPXO004I 10.17.23 DISPLAY OMVS 869
OMVS     ACTIVE  000E         OMVS=(XX,YY,ZZ)
```

The keyword OPTIONS lets you display the current configuration of the BPXPRMxx parmlib statements that are reconfigurable via the SET OMVS or SETOMVS command. The updated output from D OMVS,OPTIONS reflects any changes that resulted from a SETOMVS or a SET OMVS= operator command invocation.

In this example, when the PID option is used to obtain the thread identifiers, the output is:

```
D OMVS,PID=117440514

BPXO040I 14.16.58 DISPLAY OMVS 177
OMVS     000E ACTIVE         OMVS=(93)
USER     JOBNAME  ASID        PID       PPID STATE   START    CT_SECS
MEGA     TC1      0021  117440514  117440515 HKI   14.16.14      .170
  LATCHWAITPID=         0 CMD=ACEECACH
 THREAD_ID        TCB@     PRI_JOB  USERNAME   ACC_TIME SC  STATE
 0496146000000000 009E0438                        .050 PTJ  KU
 04961D0800000001 009D5E88                        .002 SLP JSN
 049625B000000002 009D8798                        .003 SLP JSN
 04962E5800000003 009D5090                        .012 SLP JSN
 0496370000000004 009D5228                        .011 SLP JSN
 04963FA800000005 009D5A88                        .010 SLP JSN
 0496485000000006 009D8048                        .011 SLP JSN
 049650F800000007 009D81E0                        .011 SLP JSN
 049659A000000008 009D8378                        .011 SLP JSN
 0496624800000009 009D8510                        .011 SLP JSN
 04966AF00000000A 009D8930                        .030 SLP JSN
```

You can then cancel selected threads, as shown in this example:

```
F BPXOINIT,FORCE=117440514.04962E5800000003
BPXM027I COMMAND ACCEPTED.

F BPXOINIT,TERM=117440514.0496624800000009
BPXM027I COMMAND ACCEPTED.
```

An operator displays status for the rest of the OS/390 system with the commands:

- **DISPLAY TS,LIST**: The number of time-sharing users, including the number of users
- **DISPLAY JOBS,LIST**: The number of active jobs, including the number of address spaces that were forked or that were created in other ways but requested kernel services.
- **DISPLAY A,LIST**: The combined information from the DISPLAY TS,LIST and DISPLAY JOBS,LIST commands.

Displaying the Status of OS/390 UNIX Parmlib Limits: You can display information about current system-wide parmlib limits, including current usage and high-water usage, with the DISPLAY OMVS,LIMITS command:

```
DISPLAY OMVS,L

BPXO051I 14.05.52 DISPLAY OMVS 904
 OMVS     0042 ACTIVE          OMVS=(69)
 SYSTEM WIDE LIMITS:         LIMMSG=NONE
                    CURRENT  HIGHWATER     SYSTEM
                      USAGE      USAGE      LIMIT
 MAXPROCSYS              1          4        256
 MAXUIDS                 0          0        200
```

```
MAXPTYS                0          0        256
MAXMMAPAREA            0          0        256
MAXSHAREPAGES          0         10       4096
IPCMSGNIDS             0          0        500
IPCSEMNIDS             0          0        500
IPCSHMNIDS             0          0        500
IPCSHMSPAGES           0          0     262144 *
IPCMSGQBYTES         ---          0     262144
IPCMSGQMNUM          ---          0      10000
IPCSHMMPAGES         ---          0        256
SHRLIBRGNSIZE          0          0   67108864
SHRLIBMAXPAGES         0          0       4096
```

An * displayed after a system limit indicates that the system limit was changed via a SETOMVS or SET OMVS= command.

The display output shows for each limit the current usage, high-water (peak) usage, and the system limit as specified in the BPXPRMxx parmlib member. The displayed system values may be the values as specified in the BPXPRMxx parmlib member, or they may be the modified values resulting from the SETOMVS or SET OMVS commands.

You can also use the DISPLAY OMVS,LIMITS command with the PID= operand to display information about high-water marks and current usage for an individual process. See *OS/390 UNIX System Services Command Reference*.

The high-water marks for the system limits can be reset to 0 with the D OMVS,LIMITS,RESET command. Process limit high-water marks cannot be reset.

Taking a Dump of the Kernel and User Processes

If you have a loop, hang, or wait condition in a process and need a dump for diagnosis, you need to dump several types of data:

- The kernel address space.
- Any kernel data spaces that may be associated with the problem.
- Any process address spaces that may be associated with the problem.
- Appropriate storage areas containing system control blocks (for example, SQA, CSA, RGN, TRT).

The steps are:

1. Use DISPLAY commands to display information on currently active address spaces and data spaces. (For more details on these DISPLAY commands, see *OS/390 MVS System Commands*.)
2. Allocate a sufficiently large dump data set.
3. Take the dump.
4. Review the dump completion information.

Displaying the Kernel Address Space: To find the kernel address space and associated data spaces, use `D A,OMVS`. Here is a sample output:

```
       D A,OMVS
IEE115I 12.55.47 94.208 ACTIVITY 503
 JOBS     M/S    TS USERS    SYSAS    INITS   ACTIVE/MAX VTAM
00001    00013    00002      00019    00019    00002/00050
 OMVS     OMVS     OMVS     NSW  SO  A=000E   PER=NO   SMC=000
                                     PGN=001  DMN=001  AFF=NONE
                                     CT=033.466S  ET=03.44.48
                                     WUID=STC06055 USERID=OMVSKE
                                     ADDR SPACE ASTE=0173ECC0
                                     DSPNAME=SYSZBPXU ASTE=00A35
                                     DSPNAME=SYSGFU01 ASTE=007F8
                                     DSPNAME=SYSZBPX3 ASTE=007F8
                                     DSPNAME=SYSIGWB1 ASTE=007F8
                                     DSPNAME=SYSZBPX2 ASTE=00A35
                                     DSPNAME=SYSZBPX1 ASTE=00A35
```

The display output shows the kernel address space identifier (ASID) as `A=`*nnnn* where *nnnn* is the hexadecimal ASID value. In this example, `A=000E`. The display output also shows the data space names associated with the kernel address space. The system uses these data spaces as follows:

- SYSZBPX1 for kernel data (including CTRACE buffers). The CTRACE buffers are automatically included in the dump and need not be explicitly added to a DUMP command or a SLIP trap.
- SYSZBPX2 for file system data
- SYSZBPX3 for pipes
- SYSIGWB1 for byte-range locking
- SYSGFU01 for file system adapter
- SYSZBPXU for AF_UNIX sockets
- SYSZBPXC for common INET sockets
- SYSZBPXL for local AF_INET sockets

Dump other data spaces if there is reason to believe that they contain data that could be useful in analyzing the problem.

Displaying Process Information: To display the process information for address spaces, use `D OMVS,A=ALL`. Here is a sample output:

```
    D OMVS,A=ALL

 USER     JOBNAME  ASID        PID       PPID STATE
 OMVSKERN BPXOINIT 002A          1          0 1WI
 MVS      TCPIP    002B      65538          1 MR
 DCEKERN  DCEKERN  003A     262147          1 HK
 DCEKERN  DCEKERN  003A     262148     262147 HK
 DCEKERN  DCEKERN  003A      65541     262147 HK
 DCEKERN  DCEKERN  003A      65542     262147 HF
 DCEKERN  DCEKERN  003A          7     262147 HK
 DCEKERN  DCEKERN  003A          8     262147 HK
 TS65106  TS65106  0032          9          1 1RI
 TS65106  TS65106  0032         10          9 1CI
   LATCHWAITPID=         0 CMD=-sh
```

The display output shows all of the active processes, ASIDs, process identifiers, parent process IDs, and states. Use this to obtain ASIDs of processes you wish to dump.

Displaying Global Resource Information: To display global resource serialization information to see possible latch contention, use `D GRS,C`.

This display may show latch contention, which could be the cause of the problem. You should dump the address space of the process holding the latch. If the latch is a file system latch, dump the file system data space SYSZBPX2 also.

Allocating a Sufficiently Large Dump Data Set: Because you are dumping multiple address spaces, multiple data spaces, and multiple storage data areas, you may need a much larger dump data set defined than is normally used for dumping a single address space. You should preallocate a very large SYS1.DUMPnn data set. For more information on SYS1.DUMPnn data, see the DUMPDS command in *OS/390 MVS System Commands*.

SDUMP has a limit on how much storage it allows in a single dump. It is called MAXSPACE. To determine the current value of MAXSPACE, issue the `D D,O` command. The default value is 500 megabytes. To change this value, issue:

```
CD SET,SDUMP,MAXSPACE=nnnnM
```

In a large server environment, you may need to increase MAXSPACE to 2000M (2 gigabytes) or more.

Taking the Dump: To initiate the dump, enter this command:

```
DUMP COMM=(dname)
```

where *dname* is a descriptive name for this dump. You can specify up to 100 characters for the title of the dump. The system responds and gives you a prompt ID. You reply by specifying the data to be included in the dump. If you specify the operand CONT, the system will prompt you for more input.

In the following examples of replies you can give, *rn* is the REPLY number to the prompt.

The data areas in the following reply contain system control blocks and data areas generally necessary for investigating problems:

```
R rn,SDATA=(CSA,SQA,RGN,TRT,GRSQ),CONT
```

In the next reply, `x'E'` is the OMVS address space. The other address space IDs specified are those believed to be part of the problem. You can specify up to 15 ASIDs.

```
R rn,ASID=(E,3A,32),CONT
```

This example specifies data spaces:

```
R rn,DSPNAME=('OMVS'.SYSZBPX2,'OMVS'.SYSZBPX1),END
```

The file system data space, SYSZBPX2, is useful if the hang condition appears to be due to a file system latch.

For more information on the DUMP command, particularly on specifying a large number of operands, see *OS/390 MVS System Commands*.

Reviewing Dump Completion Information: After the dump completes, you receive an IEA911E message indicating whether the dump was complete or partial. If it was partial, check the SDRSN value. If insufficient disk space is the reason, delete the dump, allocate a larger dump data set, and request the dump again.

Recovering from a Failure

The operator needs to recover if a failure occurs:

- **Kernel failure:** As a result, interactive processing in the shell and OS/390 UNIX applications fail.
- **File system type failure:** OS/390 UNIX continues processing even though the file system type is not operational. Requests to use the files in any file systems of that file system type will fail.
- **File system failure:** As a result, some files cannot be used, which may cause programs to fail.

The operator starts recovery by collecting messages and a dump, if written.

System Services Failure: If the OS/390 UNIX system fails, the operator collects problem data, which includes messages, SVC dumps, and SYS1.LOGREC records for abends and decides if re-IPL is warranted.

The work in progress when the failure occurred is lost and must be started from the beginning.

File System Type Failure: After a failure of a file system type, the system issues message BPXF014D. In response, the operator or automation corrects the problem as indicated by previous messages and then enters `R` in reply to message BPXF014D.

File System Failure: These events can be symptoms of file system failure:
- 0F4 abend
- EMVSPFSFILE return code
- EMVSPFSPERM return code
- A file becomes unrecognizable or unopenable

After a failure of a file system, the operator:
1. Restores the HFS data set with the data set from the previous level. For more information on recovering an HFS data set, see:
   - *OS/390 DFSMS Migration*
   - *OS/390 DFSMShsm Storage Administration Guide*
2. Asks a superuser to logically mount the restored HFS data set with a TSO/E MOUNT command.
3. Notifies all shell users that when they invoke the shell they will mount a backlevel file system, telling them the mount point. (Use the **wall** command to broadcast a message to all shell users.)

Files added since the back-level data set was saved must be re-created and added again.

If the physical file system owning the root fails, or if the root file system is unmounted, the operator must restore the root file system. This can be done by a superuser who is defined with a home directory of */*; (root). All work in progress when the failure occurred is lost and must be started from the beginning.

Recovery of DCE Components: Perform any necessary backup of OS/390 DCE program libraries, configurations, and optional data sets as a part of your regular installation backup and recovery procedures. See *OS/390 DCE Administration Guide* for information about DCE recovery.

Managing Interprocess Communication (IPC)

Users can invoke applications that create IPC resources and wait for IPC resources. IPC resources are not automatically released when a process terminates or a user logs off. Therefore, it is possible that an IPC user may need assistance to:

- Remove an IPC resource using the shell's **ipcrm** command
- Remove an IPC resource using the shell's **ipcrm** command to release a user from an IPC wait state

To display IPC resources and which userid owns the resource, issue the following command:

```
ipcs -w
```

To delete message queue IDs, use the **ipcrm -q** or **ipcrm -Q** command.

Another problem may occur when a user waits a long time for a resource such as semaphores or a message receive. Removing a message queue ID or semaphore ID brings any users in an IPC wait state out of the wait state. To display which users are waiting for semaphores and message queues, issue:

```
ipcs -w
```

Chapter 29. Tuning Performance

You need to take some tuning steps because you are combining MVS and UNIX. There are two tuning situations, depending on how your system is being used: as a production system or a porting system. For both, you can take important steps to improve performance and control resource consumption.

To learn how to improve performance on a porting system, read Chapter 8 of *Porting Applications to the OS/390 UNIX Platform*, GG24-4473.

These are the key areas to target when you start to tune the production system:

Adjusting Storage Size

If your system is running in an LPAR or as a VM guest, the storage size should be at least 64M.

Using DASD Cache

Place on cached DASD:

- Volumes that contain user file systems. To get the responsiveness that UNIX users are accustomed to, place these volumes on cached DASD that has SET DASD FAST WRITE on. Do this because the hierarchical file system hardens all file system data to disk synchronously. (Data is stored on disk synchronously on certain writes, and any remaining data is stored to disk synchronously on close. That is, the close does not return to the user until the file has been completely stored on disk.)

  Note: To avoid DASD contention, the user file systems should be distributed among multiple control units and DASDs. Too many user file systems on one volume can negatively affect I/O performance.
- The RACF data base.

Improving Performance of Run-Time Routines

When C programs (including the shell and utilities) are run, they frequently use routines from the Language Environment run-time library, which come from the SCEERUN data set. On average, about 4 MB of the run-time library are loaded into memory for every address space running a Language Environment-enabled program, and copied on every fork. If you have 200 address spaces running, this uses 800 MB of pageable storage. It also increases your paging rates or reduces the amount of work that the system can support. For information about the effect of putting modules into the LPA, see *OS/390 MVS Initialization and Tuning Guide*.

The following sections describe how you can reduce this overhead and improve performance.

Placing SCEERUN in the Link Pack Area

Because the SCEERUN data set has many modules that are not reentrant, you cannot place the entire data set in the Link Pack Area (LPALSTxx parmlib). However, as of OS/390 Release 6, there is a new SCEELPA data set that contains a subset of the SCEERUN modules—those that are re-entrant, reside above the line, and are heavily used by OS/390 UNIX System Services. (For more information, see *OS/390 Language Environment Customization*.)

If you put the SCEERUN data set in the link list (LNKLSTxx), you can place the new SCEELPA data set in LPA list. Doing this will improve performance.

You can also add additional modules to the LPA, using the Modify Link Pack Area (MLPA=) option at IPL. You can also use the Dynamic LPA capability (SET PROG=). Using the Dynamic LPA method avoids the performance degradation that occurs with the use of MLPA.

The RUNOPTS parameter in the BPXPRMxx parmlib member specifies the Language Environment run-time options that is to be passed to **/etc/init** when using RTLS.

Placing SCEERUN in the Link List

If you choose not to put any modules from SCEERUN in the LPA, you can still put SCEERUN in the link list. This will not perform as well as having modules in LPA, but can still benefit from reduced input/output due to management by LLA and VLF.

Managing the Run-Time Library with RTLS

Some installations cannot put the current level of the Language Environment run-time library into the LINKLIST because older Language Environment levels are needed to run key production applications. This means that key run-time library routines cannot be put in the LPA for better performance. In addition, you cannot put the SCEELPA data set as part of the LPALSTxx.

The answer to this problem is Run-Time Library Services. RTLS enables installations to use more than one level of the run-time library on the same system without using STEPLIBs. They can put key run-time library modules from more than one level of Language Environment into common storage for shared access.

See *OS/390 Language Environment Customization* for information on using RTLS on your system. You will need to set up some FACILITY profiles, as documented in the CSVRTLxx description in *OS/390 MVS Initialization and Tuning Reference*.

After you set up RTLS, you only need to set up RUNOPTS in the BPXPRMxx member for most OS/390 UNIX environments. (Customizing the BPXPRMxx member is discussed in "Customizing the BPXPRMxx Parmlib Members" on page 5.

For OS/390 UNIX users to use RTLS, you must also specify RTLS(ON), LIBRARY, and, optionally, VERSION run-time options in the RUNOPTS parameter of the BPXPRMxx parmlib member. For example:

```
RUNOPTS(RTLS(ON) LIBRARY(xxxxxxxx) VERSION(yyyyyyyy)
```

where *xxxxxxxx* is the library name and *yyyyyyyy* is the version name assigned in the CSVRTLxx parmlib member for the current level of Language Environment (for example, CEE.SCEERUN).

Managing the Run-Time Library in STEPLIBs

If you decide not to put the run-time library in the link list or RTLS, then you must set up the appropriate STEPLIB for each application that needs to load modules from SCEERUN. Although this method always uses additional virtual storage, you can improve performance by defining the SCEERUN data set to LLA. This reduces the I/O that is needed to load the run-time modules.

Improving Compiler Performance

This section discusses how you can improve compiler performance by placing the C/C++ compiler and Program Management Binder in the LPA. As of OS/390 Release 7, you no longer have to make any changes to the **c89**, **cc**, **cxx**, and **c++** utilities.

Putting Compiler Load Modules into LPA

On systems where application development is the primary activity, performance may be improved if you put CBC.SCBCCMP in the LPALST concatenation. All compiler modules run above the line and they consume just over 42 MB in total.

Place the program binder in LPA:

- From SYS1.LINKLIB:

Module	Location
IEFIB600 (alias IEFXB603)	44K below the line
IEWBLINK	2K below the line

IEWBLINK has these aliases:
alias HEWL
alias HEWLDRGO
alias HEWLH096
alias HEWLOAD
alias HEWLOADR
alias IEWBLDGO
alias IEWBLOAD
alias IEWBLODI
alias IEWBODEF
alias IEWL
alias IEWLDRGO
alias IEWLOAD
alias IEWLOADI
alias EWLOADR
alias LINKEDIT
alias LOADER

- From CEE.SCEERUN:

Module	Location
EDCRNLIB (alias EDCRNLST)	Above the line

Caching RACF User and Group Information in VLF

Caching UIDs and GIDs improves performance for commands such as **ls -l**, which must convert UID numbers to user IDs and GID numbers to RACF group names. RACF allows you to cache UID and GID information in Virtual Lookaside Facility (VLF). Add the following VLF options to the COFVLFxx member of SYS1.PARMLIB to enable the caching:

```
CLASS NAME(IRRUMAP)
  EMAJ(UMAP)
CLASS NAME(IRRGMAP)
  EMAJ(GMAP)
CLASS NAME(IRRSMAP)
  EMAJ(SMAP)
```

For details about these VLF and the other VLF classes that are used by RACF, see *OS/390 SecureWay Security Server RACF System Programmer's Guide*.

Start VLF, specifying the updated member (in this example, COFVLF33 member) with an operator command:

```
START VLF,SUB=MSTR,NN=33
```

Because VLF is started after RACF and OMVS, you may get a message from RACF during the IPL saying that running without VLF will cause slower performance. If VLF is being started, you can ignore this message.

For information about updating the VLF parmlib member COFVLFxx, see "COFVLFxx Parmlib Member to Activate RACF Classes" in Chapter 14, "Customizing OS/390 UNIX".

Checking the Owning UIDs and GIDs on Files

Ensure that all files in your file system have a valid owning UID and GID. If you restore files from an archive and accidentally keeps a UID and GID from another

system that are not valid on the system, it can create problems that affect response time. For example, suppose that there is an invalid UID associated with a file. When you use a utility that checks the UID (such as **ls -l**), RACF searches the entire database for the UID. To prevent this search, IBM provides APAR OW23748 for OS/390 Releases 2 and 3.

For customers running Release 3 or higher, APAR OW30858 introduces the UNIXMAP class to prevent the search. When a file is transferred through NFS, an owning UID of 0 is changed to -2. This is often done with the client's user's identity within the RPC because NFS does not want to assume that a superuser on one system is also a superuser on another system. OS/390 UNIX does not support a UID of -2. As a consequence, you get files on your system with an owning UID of 4294967294. Be sure to change this to a valid owning UID.

Moving HFS Executables into the Link Pack Area

Some executables in the HFS may be commonly used by many concurrent users, or they may be loaded and deleted frequently during normal production. Such executables are performance sensitive, and they may be good candidates for inclusion in the LPA. Moving such programs to the LPA can reduce storage consumption, reduce DASD I/O activity for loads, and reduce the storage copied on each fork().

One thing to consider when you analyze which HFS executables belong in LPA is that modules with the sticky bit on are not eligible for local spawn(). If your executable is normally invoked by spawn(), either by the shell or by another application, turning on the sticky bit forces spawn() processing to execute the program in a spawned child address space. In cases where local spawn() would be used if the sticky bit were not on, this reduces the benefit of loading the executable from the LPA.

To move an executable in the HFS into the LPA, do the following steps:

1. If the executable or DLL name is less than 8 characters excluding the extension (such as longname.dll):
   a. Bind the executable or DLL into a PDS (for example, LONGNAME)
   b. For the executable or DLL in the HFS, turn on the sticky bit. For example:
      ```
      chmod +t longname.dll
      ```
   c. If the executable or DLL name has invalid characters, then do a symbolic link such as:
      ```
      ln -s longname long+name
      ```
2. If the executable or DLL name is more than 8 characters long, excluding the extension (for example, reallylonglongname.dll:
   a. Bind the executable or DLL into a PDS (for example, REALLY)
   b. Create an external link for the name. For example:
      ```
      ln -e REALLY reallylonglongname.dll
      ```

To bind the executable or DLL into a PDS, you can use the following sample JCL:

```
//PUTINLPA  JOB  MSGLEVEL=(1,1)
//*                                                              *
//* INLMOD DD STATEMENT SPECIFIES THE DIRECTORY THAT CONTAINS  *
//* THE PROGRAM.                                                 *
//*                                                              *
//* THE INCLUDE STATEMENT SPECIFIES THE NAME OF THE FILE TO    *
//* RUN FROM THE LPA.                                            *
//*                                                              *
//* THE NAME STATEMENT SPECIFIES THE FILE NAME BUT IN          *
//* UPPERCASE. THIS MUST BE SAME AS THE FILE NAME.               *
//*                                                              *
//LINK     EXEC PGM=IEWL,REGION=100M,
//  PARM='LIST,XREF,LET,RENT,REUS,AMODE=31,RMODE=ANY,CASE=MIXED'
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(10,10))
//SYSPRINT DD SYSOUT=*
//INLMOD   DD PATH='/bin/'
//SYSLMOD  DD DSN=OECMD.LPALIB,DISP=SHR
//SYSLIN   DD *
         INCLUDE INLMOD(myprog)
         ENTRY   CEESTART
         NAME    MYPROG(R)
/*
```

Figure 3. Job for Placing a Program in the LPA

Note: You should use an SMP/E usermod to link any IBM-supplied programs from an HFS into another library. (For example, in order to load it into LPA.) Doing so automatically keeps the two copies of the module at the same level when service is installed. It also provides a record of modifications to your systems. See *OS/390 SMP/E User's Guide* for more information about SMP/E usermods.

Also, not all modules are eligible for LPA. Modules placed in LPA must be both reentrant and executable. For more information, see *OS/390 MVS Initialization and Tuning Reference*.

Tuning Limits in Parmlib

This section contains information that may be helpful in tuning your OS/390 UNIX environment. It provides guidelines that should prove to be generally helpful. However, because each installation is unique, some of the recommendations may not be appropriate for your system.

For more information, refer to these books:

- *OS/390 MVS Planning: Workload Management*
- *OS/390 MVS Initialization and Tuning Guide*
- *OS/390 MVS Initialization and Tuning Reference* for parmlib members
- *RMF User's Guide* for RMF monitoring
- *RMF Report Analysis* for RMF reports

Monitoring Parmlib Limits

You can monitor the status of OS/390 UNIX system and process limits with the D OMVS, LIMITS operator command and console messages that indicate when limits are reaching critical levels.

You can then use SET OMVS or SETOMVS to change certain system limits dynamically, or SETOMVS with PID= to change a process-level limit for a specific process. See *OS/390 MVS System Commands*.

The LIMMSG(NONE|SYSTEM|ALL) statement in the BPXPRMxx parmlib member controls message activity for limits checking. You can specify whether no console messages are to be displayed when any of the parmlib limits have been reached (NONE); console messages are to be displayed for all processes that reach system limits and for certain process limits (SYSTEM); or console messages are to be displayed for all the system limits and the process limits (ALL).

The LIMMSG options (SYSTEM|ALL|NONE) can be changed with the SETOMVS LIMMSG command. The LIMMSG value appears in the D OMVS,O display.

If the LIMMSG statement is specified with SYSTEM or ALL, a warning console message appears whenever a limit reaches 85%, 90%, 95%, and 100%; identifying the process that has reached the limit. As the limit reaches the next limit level, the prior message is removed from the console and a new message is displayed indicating the new limit level that has been reached. When the limit falls below the 85% threshold, a message is issued indicating that the resource shortage has been relieved.

Changing from LIMMSG(ALL) or LIMMSG(SYSTEM) to LIMMSG(NONE) with the SETOMVS command stops any further monitoring of resources. However, existing outstanding messages are not deleted from the screen for a process until the limit is relieved for that process.

Note: When LIMMSG(ALL) is in effect, a large number of messages can be issued. This option is best suited for use during the initial configuration of a system, when the installation has not yet determined the optimal settings for the OS/390 UNIX parmlib limits.

Tuning Process Activity

OS/390 UNIX provides the system programmer with a number of controls that monitor and tune the use of system resources by users. This section focuses on the following fields in the BPXPRMxx statements:

- MAXUIDS
- MAXPTYS
- MAXRTYS
- MAXPROCUSER
- MAXPROCSYS

Initial Rules of Thumb:

1. Assume that each user will consume up to double the system resources required for a TSO/E user.
2. Assume that at most 4 PTYs or RTYs will be required per average user.
3. Assume that the starting point for maximum processes per user is 25.
4. Assume that 4 concurrent processes will be required by the average active user.
5. Assume that 5 processes will be required for various daemons.
6. Assume that 3 concurrent address spaces will be required by the average active user. This number will be high if your users are running with the _BPX_SHAREAS environment variable set to YES or REUSE.

If you have a few users who need a large number of processes, you should set the process limits for these users by using the PROCUSERMAX keyword in the OMVS segment.

Example: Assume that your system supports 600 TSO/E users and has enough capacity for 20 additional users. Rather than adding more TSO/E work, you want to allow TSO/E users to access OS/390 UNIX. You have no other OS/390 UNIX work on your system at this time.

In this example, in BPXPRMxx, the initial settings might be:

```
MAXUIDS(20)
MAXPTYS(80)
MAXRTYS(80)
MAXPROCUSER(25)
MAXPROCSYS(85)
```

MAXUIDS

20 - If you allow 20 current TSO/E users to access the OS/390 UNIX system, each of them could consume twice the resource they normally used for TSO/E. This would require all your remaining system resources.

MAXPTYS

80 - Assume that 4 PTYs are needed per user. Users can login with multiple sessions at the same time.

MAXRTYS

80 - Assume that 4 RTYs are needed per user. Users can login with multiple sessions at the same time.

MAXPROCUSER

25 - This should normally be a reasonable starting point. Some users may require more processes, depending on the work they are doing. This value can be set only on a system-wide basis.

MAXPROCSYS

85 - Assume that you need 4 processes per user and 5 processes for daemons. (20 users * 4) + 5 daemons = 85 processes.

Controlling Use of ESQA

A number of services use base OS/390 functions that uses ESQA storage. Much of this storage is fixed, consuming main memory rather than only virtual storage. Installations having constraints on virtual storage or main memory can control the amount of ESQA storage used by the following services:

- Shared Memory
- Memory map files
- ptrace
- fork (copy-on-write)

The following BPXPRMxx parmlib statements are the primary means of controlling consumption by UNIX services:

- MAXSHAREPAGES controls the maximum number of shared pages to be used for fork, shared memory, memory map files, and ptrace. ESQA storage is required for each shared page.
- FORKCOPY determines whether fork should use copy-on-write support. Copy-on-write support should normally reduce the cost of fork by removing the need to copy all the parent's virtual storage to the child address space. However, on systems with storage constraints, the benefit of copy-on-write may be outweighed by the impact on ESQA storage.

  Follow these guidelines:

  – If the run-time library is in the link pack area, specify FORKCOPY(COPY).
  – If the run-time library is not in the link pack area, specify FORKCOPY(COW).

Other statements in the BPXPRMxx parmlib member provide more detailed control of how shared memory, and memory map files can be used.

For more detail on statements in the BPXPRMxx parmlib member, see “Customizing the BPXPRMxx Parmlib Members” on page 5. For more detail on ESQA and other storage requirements for MVS, see “Evaluating Virtual Storage Needs” in Chapter 14, “Customizing OS/390 UNIX”.

nice(), setpriority(), and chpriority()

Making Sure that the Sticky Bit for the OS/390 Shell Is On

Improving the OS/390 Shell Performance

Improving Performance on POSIX by Using Medium-Weight Processes

Improving Performance of Security Checking

OMVS Command and TSO/E Response Time

Part 2. APAR OW43776: OS/390 MVS Library

Chapter 2. APAR OW43776: OS/390 MVS System Commands

Displaying OS/390 UNIX System Services Status

The MVS operator can use the DISPLAY command to obtain:

- OS/390 UNIX System Services status information (for example, active or terminating)
- Hierarchical file system (HFS) information
- OS/390 UNIX System Services process information for address spaces
- The current setting for all OS/390 UNIX System Services parmlib statements
- Information about multiple parmlib members
- Information about each physical file system that is currently part of the OS/390 UNIX System Services configuration
- Routing information from the Common Inet Pre-Router routing tables.
- Information about OS/390 UNIX System Services parmlib limits, including current system-wide and process limits, their high-water marks, and current usage.
- Thread-level information for any thread that is in a byte-range lock wait.

You can use this command to display address space information for a user who has a process that is hung. You can also use the information returned from this command to determine how many address spaces a given TSO/E user ID is using, whether an address space is using too many resources, and whether a user's process is waiting for an OpenMVS kernel function to complete.

The syntax for the DISPLAY OMVS command is:

```
D OMVS[{,SUMMARY|S}            ]
      |,{ASID|A}=ALL
      |,{ASID|A}=asid
      |,U=userid
      |,{PID}=processid[,BRL]
      |,{FILE|F[,CAPS|C]}
      |,{VSERVER|V}
      |,{PFS|P}
      |,{CINET|CI}=All|TPname
      |,{OPTIONS|O}
      |,{LIMITS|L[,PID=ProcessId][,RESET]}

[,L={a|cc|cca|name|name-a}]
```

SUMMARY *or* **S**
Displays status of OpenMVS processes, file systems, and servers (for example, active or terminating) and the BPXPRMxx parmlib member specified during initialization or specified by the SET OMVS= OS/390 UNIX System Services command.

ASID= *or* **A=ALL**
Displays process information for all OS/390 UNIX System Services address spaces.

ASID= *or* **A=asid**
Displays process information for the specified hexadecimal address space ID (ASID). If the specified ASID is not an OS/390 UNIX System Services address space, an error message is issued.

U=userid
Displays process information for all processes associated with the specified

TSO/E user ID. Use this operand when a user requests that a hung process be canceled. You can display all processes owned by the user and find the address space ID (ASID) of the process that needs to be canceled. Then use the CANCEL command to cancel the address space.

PID=processid
Displays thread information for the processid that is specified in decimal numbers. In a sysplex environment, the D OMVS,PID= command must always be issued from the system on which the specified process is running.

FILE *or* **F**
Displays a list of HFS file systems that OS/390 UNIX System Services is currently using and the status of each HFS.

VSERVER *or* **V**
Displays process information for all processes that have been defined as servers that use the virtual file system (VFS) callable services API.

CAPS *or* **C**
Displays variable data containing lowercase letters in uppercase.

CINET = *or* **CI = ALL|tpname**
Displays the Common Inet routing information for all of the active transport providers in use by the Common Inet Pre-Router. The transport providers were specified with the SUBFILESYSTYPE statements in the BPXPRMxx profile or specified with the SETOMVS command. The network routing information was specified in the appropriate data set for the transport provider. When the name (*tpname*) of an active transport provider is specified, the command displays the Common Inet routing information for that specific transport provider.

OPTIONS *or* **O**
Displays the current settings of the options that

(a) were set during initialization in the parmlib member BPXPRMxx or by a SET OMVS or SETOMVS command after initialization, and that

(b) can be altered dynamically via a SET OMVS or SETOMVS command.

PFS *or* **P = Physical File System**
Displays information about each physical file system that is currently part of the OS/390 UNIX System Services configuration. The physical file systems were specified in the BPXPRMxx profile, or with the SETOMVS command, or are an internal part of OS/390 Unix System Services.

LIMITS *or* **L**
Displays information about current OS/390 UNIX System Services parmlib limits, their high-water marks, and current system usage. When the PID= keyword is specified, LIMITS displays high-water marks and current usage for an individual process.

RESET
Resets the high-water mark for a system limit to 0.

BRL
Displays thread-level information for any thread that is in a byte-range lock wait. This operand can only be specified with PID=.

Example 1

To display process information for all OS/390 UNIX System Services address spaces, enter:

```
DISPLAY OMVS,A=ALL
```

OS/390 UNIX System Services status information (OMVS ACTIVE) appears before the process information.

```
BPXO040I 14.31.40 DISPLAY OMVS 018
OMVS     000E ACTIVE          OMVS=(93)
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
IBMUSER  BPXOINIT 0013          1          0 MKI   11.02.40       .037
  LATCHWAITPID=         0 CMD=BPXPINPR
  SERVER=Init Process                     AF=    0 MF=65535 TYPE=FILE
MEGA     MEGA     001A   16777218          1 1RI   11.18.17       .634
  LATCHWAITPID=         0 CMD=OMVS
MEGA     MEGA     001A   16777219   16777218 1CI   11.18.25       .634
  LATCHWAITPID=         0 CMD=sh -L
```

Example 2

To display OS/390 UNIX System Services process information on all OS/390 UNIX System Services address spaces owned by user ID MEGA, enter:

```
DISPLAY OMVS,U=MEGA
```

OS/390 UNIX System Services status information (OMVS ACTIVE) appears before the process information.

```
BPXO040I 14.34.15 DISPLAY OMVS 021
OMVS     000E ACTIVE          OMVS=(93)
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
MEGA     MEGA     001A   16777218          1 1RI   11.18.17       .634
  LATCHWAITPID=         0 CMD=OMVS
MEGA     MEGA     001A   16777219   16777218 1CI   11.18.25       .634
  LATCHWAITPID=         0 CMD=sh -L
```

Example 3

To display OS/390 UNIX System Services process information for the address space with ASID equal to 001A, enter:

```
DISPLAY OMVS,ASID=1A
```

OS/390 UNIX System Services status information (OMVS ACTIVE) appears before the process information.

```
BPXO040I 14.36.04 DISPLAY OMVS 024
OMVS     000E ACTIVE          OMVS=(93)
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
MEGA     MEGA     001A   16777218          1 1RI   11.18.17       .634
  LATCHWAITPID=         0 CMD=OMVS
MEGA     MEGA     001A   16777219   16777218 1CI   11.18.25       .634
  LATCHWAITPID=         0 CMD=sh -L
```

Example 4

To display detailed file system information on currently mounted files, enter:

```
DISPLAY OMVS,FILE
```

OS/390 UNIX System Services status information (OMVS ACTIVE) appears before the file system information.

```
00  BPXO0451 12.28.28 DISPLAY OMVS 011
    OMVS     000E ACTIVE          OMVS=(66)
    TYPENAME   DEVICE ----------STATUS----------- MODE
    HFS             4 ACTIVE                     READ
      NAME=POSIX.USR.LPP
      PATH=/usr/lpp
      MOUNT PARM=SYNC(60)
```

```
    OWNER=SYSTEM2   AUTOMOVE=Y   CLIENT
    QSYSTEM=system1 QJOBNAME=FRED   QPID=34567
  HFS             3 ACTIVE                      READ
    NAME=POSIX.HFS.NLS
    PATH=/usr/ib/nls
    OWNER=SYSTEM2   AUTOMOVE=Y   CLIENT=Y
  HFS             2 ACTIVE                      READ
    NAME=POSIX.HFS.MAN
    PATH=/usr/man
    OWNER=SYSTEM3   AUTOMOVE=Y   CLIENT=Y
  HFS             1 ACTIVE                      RDWR
    NAME=POSIX.HFS.FS
    PATH=/
    OWNER=          AUTOMOVE=N   CLIENT=N
```

Example 5

To display process information for all processes that have been defined as a server, enter:

```
DISPLAY OMVS,V
```

OS/390 UNIX System Services status information (`OMVS ACTIVE`) appears before the file system information.

```
BPXO040I 14.38.46 DISPLAY OMVS 030
OMVS     000E ACTIVE          OMVS=(93)
USER     JOBNAME  ASID        PID       PPID STATE   START    CT_SECSS
IBMUSER  BPXOINIT 0013          1          0 MKI   11.02.40      .0373
  LATCHWAITPID=         0 CMD=BPXPINPR
  SERVER=Init Process                     AF=    0 MF=65535 TYPE=FILE
```

Example 6

To display all options set during initilization by the parmlib member BPXPRMxx or with the SET command, enter:

```
DISPLAY OMVS,O
```

```
d omvs,o

BPXO043I 10.26.49 DISPLAY OMVS 007
OMVS     000E ACTIVE          OMVS=(69)
OS/390 UNIX CURRENT CONFIGURATION SETTINGS:
MAXPROCSYS      =        256    MAXPROCUSER     =         16
MAXFILEPROC     =        256    MAXFILESIZE     = NOLIMIT
MAXCPUTIME      =       1000    MAXUIDS         =        200
MAXPTYS         =        256
MAXMMAPAREA     =        256    MAXASSIZE       =   41943040
MAXTHREADS      =        200    MAXTHREADTASKS  =         50
MAXCORESIZE     =    4194304    MAXSHAREPAGES   =       4096
IPCMSGQBYTES    =     262144    IPCMSGQMNUM     =      10000
IPCMSGNIDS      =        500    IPCSEMNIDS      =        500
IPCSEMNOPS      =         25    IPCSEMNSEMS     =         25
IPCSHMMPAGES    =        256    IPCSHMNIDS      =        500
IPCSHMNSEGS     =         10    IPCSHMSPAGES    =     262144
SUPERUSER       = BPXROOT       FORKCOPY        = COW
STEPLIBLIST     =
USERIDALIASTABLE=
PRIORITYPG VALUES: NONE
PRIORITYGOAL VALUES: NONE
MAXQUEUEDSIGS   =       1000    SHRLIBRGNSIZE   =   67108864
SHRLIBMAXPAGES  =       4096    VERSION         = /           SSYSCALL COUNTS  = NO            TTYGR
SYSPLEX         = NO            BRLM SERVER     = N/A
LIMMSG          = NONE
```

Note: The SYSPLEX (YES) option indicates the system is in a sysplex and is using the shared HFS capability. You cannot dynamically change the SYSPLEX parameter through SETOMVS or SET OMVS. For more information, see the chapter on Shared HFS in *OS/390 UNIX System Services Planning*.

Example 7

To display the thread information for the processid 1, enter:

```
DISPLAY OMVS,PID=1
```

```
BPXO040I 11.13.40 DISPLAY OMVS 971
OMVS     000E ACTIVE          OMVS=(93)
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
IBMUSER  BPXOINIT 0013          1          0 MKI   11.02.40       .037
  LATCHWAITPID=         0 CMD=BPXPINPR
SERVER=Init Process                    AF=    0 MF=65535 TYPE=FILE
THREAD_ID        TCB@     PRI_JOB  USERNAME   ACC_TIME SC  STATE
04B9267800000000 009DEA70 OMVS                  .028 WAT  W
04B92F2000000001 009DE8D8                       .003 VRT  Y
04B937C800000002 009DE278 OMVS                  .002 KIN  K
```

Example 8

To display information about each physical file system that is currently part of the OS/390 UNIX System Services configuration when the physical file systems are specified in the BPXPRMxx profile, enter:

```
D OMVS,P
```

```
BPXO046I 14.35.38 DISPLAY OMVS 092
OMVS     000E ACTIVE          OMVS=(33)
PFS CONFIGURATION INFORMATION
 PFS TYPE     DESCRIPTION            ENTRY      MAXSOCK   OPNSOCK   HIGHUSED
  TCP          SOCKETS AF_INET        EZBPFINI    50000       244       8146
  UDS          SOCKETS AF_UNIX        BPXTUINT       64         6         10
  HFS          LOCAL FILE SYSTEM      GFUAINIT
  BPXFTCLN    CLEANUP DAEMON       BPXFTCLN
  BPXFTSYN    SYNC DAEMON          BPXFTSYN
  BPXFPINT    PIPE                 BPXFPINT
  BPXFCSIN    CHAR SPECIAL         BPXFCSIN
  NFS         REMOTE FILE SYSTEM   GFSCINIT

 PFS NAME    DESCRIPTION          ENTRY     STATUS   FLAGS
  TCP41       SOCKETS              EZBPFINI  ACT      CD
  TCP42       SOCKETS              EZBPFINI  ACT
  TCP43       SOCKETS              EZBPFINI  INACT    SD
  TCP44       SOCKETS              EZBPFINI  INACT

PFS PARM INFORMATION
 HFS        SYNCDEFAULT(60) FIXED(50) VIRTUAL(100)
            CURRENT VALUES: FIXED(55) VIRTUAL(100)
 NFS        biod(6)
```

The information displayed is:

PFS TYPE
For each FILESYSTYPE statement, the data specified with the TYPE operand is displayed.

PFS DESCRIPTION
A brief description of the physical file system.

ENTRY
The name of the load module specified with the ENTRYPOINT operand on the FILESYSTYPE or SUBFILESYSTYPE statements.

MAXSOCK
This is the MAXSOCKETS operand of a NETWORK statement for a sockets physical file system. It specifies the maximum number of sockets that can be open at one time for the address family.

OPNSOCK
OPEN SOCKETS: The number of sockets that are currently opened for this sockets physical file system.

HIGHUSED
The highest number of sockets that have been in use at one time for each of the configured address families.

PFS NAME
For each SUBFILESYSTYPE statement, the transport provider specified with the NAME operand is displayed.

STATUS
The status of each PFS specified with the SUBFILESYSTYPE statement: ACT = ACTIVE, INACT = INACTIVE.

FLAGS
Additional information for each PFS that was defined with the SUBFILESYSTYPE statement:

CD Current Default transport provider. The system is currently using this PFS as the default transport provider although it wasn't specified as the default with the SUBFILESYSTYPE statement.

SD Specified Default transport provider. This PFS was specified as the default transport provider with the SUBFILESYSTYPE statement. Currently, however, it is not being used as the default.

SC Specified is Current default transport provider. This PFS was specified as the default transport provider with the SUBFILESYSTYPE statement and the system is currently using it as the default.

PARM INFORMATION
Data specified with the PARM operand on the FILESYSTYPE or SUBFILESYSTYPE statements is displayed. For the HFS, in addition to the IPL settings specified with PARM, the current settings for the FIXED and VIRTUAL PARMs are displayed.

Notes:

1. Although you may specify up to 1024 bytes of parameter information in the BPXPRMxx profile, only the first 165 bytes of parameter information is displayed.
2. If a dash ('-') should appear as the first character for any PFS name, it means the PFS is dead.

Example 9

To display the Common Inet routing information when there are three active transport providers:

```
DISPLAY OMVS,CINET=ALLBPXO0nnI 17:12:37 DISPLAY OMVS nn
OMVS     000E ACTIVE          OMVS=(ZD)
HOME INTERFACE INFORMATION
```

```
 TP NAME    HOME ADDRESS      FLAGS
  TCP41      127.116.117.233   DRS
  TCP42      127.116.118.234
  TCP43      127.116.119.235

HOST ROUTE INFORMATION
 TP NAME    HOST DESTINATION
  TCP41      127.117.193.234
  TCP41      127.117.194.234
  TCP42      127.117.195.234

NETWORK ROUTE INFORMATION
 TP NAME    NET DESTINATION   NET MASK          METRIC
  TCP41      127.111.000.000   255.255.000.000    10
  TCP42      127.113.000.000   255.255.000.000     0
  TCP41      197.119.119.000   255.255.255.000     F
  TCP43      009.000.000.000   255.000.000.000     F
```

The information displayed is:

TP NAME
The name of the transport provider for which the information is being displayed.

HOME ADDRESS
The internet protocol (IP) address of the transport provider.

HOST DESTINATION
When a transport provider is connected to a host, the host IP address is displayed.

NET DESTINATION
When a transport provider supplies network routing information to the Common Inet Pre-Router, the network destination address is the IP address of a network that can be accessed through the transport provider.

NET MASK
A mask that is applied to destination IP addresses to separate the network number from the host number.

METRIC
When selecting a route, if two transport providers can access the same route, the Common Inet Pre-Router selects the route with the best metric. The higher the number, the better the metric. The metric 255 = a direct connection

FLAGS
DRS = Default Routes Supported: When the Common Inet Pre-Router cannot find a specified IP address in its routing tables, it passes the request to a transport provider that supports default routes. If no transport provider supports default routes, the request is rejected with **ENETUNREACH**.

Note: When the cinet is not installed, similar routing information can be obtained by using the **netstat TC** *tpname* **gate** command or the **onetstat -p** *tpname* **-r** command.

Example 10

To display information about current system-wide parmlib limits, enter:

```
DISPLAY OMVS,L
 BPXO051I 14.05.52 DISPLAY OMVS 904
 OMVS     0042 ACTIVE          OMVS=(69)
 SYSTEM WIDE LIMITS:         LIMMSG=SYSTEM
                   CURRENT  HIGHWATER     SYSTEM
                     USAGE      USAGE      LIMIT
```

```
MAXPROCSYS              1          4        256
MAXUIDS                 0          0        200
MAXPTYS                 0          0        256
MAXMMAPAREA             0          0        256
MAXSHAREPAGES           0         10       4096
IPCMSGNIDS              0          0        500
IPCSEMNIDS              0          0        500
IPCSHMNIDS              0          0        500
IPCSHMSPAGES            0          0     262144 *
IPCMSGQBYTES          ---          0     262144
IPCMSGQMNUM           ---          0      10000
IPCSHMMPAGES          ---          0        256
SHRLIBRGNSIZE           0          0   67108864
SHRLIBMAXPAGES          0          0       4096
```

An * displayed after a system limit indicates that the system limit was changed via a SETOMVS or SET OMVS= command.

Note: Although IPCMSGQBYTES, IPCSMSGQMNUM, and IPCSHMMPAGES are displayed in the output of the D OMVS,L command, these resources are not monitored and no resource messages are issued.

Example 11

To display information about current parmlib limits for a process with a PID of 33554434, enter:

```
DISPLAY OMVS,L,PID=33554434
```

```
d omvs,l,pid=33554434
BPXO051I 14.06.49 DISPLAY OMVS 907
OMVS     0042 ACTIVE          OMVS=(69)
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
WELLIE1  WELLIE1  001C    33554434         1 IRI   14.04.38       .015
  LATCHWAITPID=         0  CMD=EXEC
PROCESS LIMITS:         LIMMSG=SYSTEM
                   CURRENT  HIGHWATER    PROCESS
                     USAGE      USAGE      LIMIT
MAXFILEPROC              0          1        256,1000
MAXFILESIZE            ---        ---    NOLIMIT
MAXPROCUSER              1          4         16
MAXQUEUEDSIGS            0          0       1000
MAXTHREADS               0          0        200
MAXTHREADTASKS           0          0         50
IPCSHMNSEGS              0          0         10
MAXCORESIZE            ---        ---    4194304,NOLIMIT
```

An * displayed after a process limit indicates that the limit was changed, either directly, with a SETOMVS,PID= command; or indirectly, by a global change of this value with a SETOMVS command.

The values displayed are in the same units as the values used in the SETOMVS command. For example, MAXFILESIZE is displayed in units of 4KB.

Notes:

1. Although MAXFILESIZE and MAXCORESIZE are displayed in the output, their current and high-water usage are not monitored, and no resource messages are issued for these resources.
2. The MAXPROCUSER limit is based on UID, as opposed to PID, value. The current and high-water usage values reflect all values for all processes that have the same UID as the UID for the specified PID.

3. For UID=0, there is no limit on MAXPROCUSER. When the **PID=** value in the DISPLAY command is for a process with UID=0, the process limit appears as unlimited. For example:

```
MAXPROCUSER            4         11         NOLIMIT
```

4. MAXCORESIZE, MAXFILESIZE, and MAXFILEPROC each have hard and soft limits. (See the documentation for the C-RTL function **setrlimit()** in *OS/390 C/C++ Run-Time Library Reference*.) When the hard and soft limits are the same, only one value is displayed. When the limits are different, both values are displayed: first the soft limit and then the hard limit, separated by a comma.

   In the preceding example, MAXFILEPROC has a hard limit of 100 and a soft limit of 256. For MAXFILESIZE, the soft limit is equal to the hard limit and is unlimited. For MAXCORESIZE, the soft limit is 4 194 304 and the hard limit is unlimited.

Example 12

If the SETOMVS command is issued to change the value of MAXFILEPROC to 256, the information displayed is:

```
                 CURRENT  HIGHWATER    PROCESS
                   USAGE      USAGE      LIMIT
MAXFILEPROC            0          0        256 *
    .
    .
    .
```

If the process changes its soft limit for MAXFILEPROC to 100 (using the **setrlimit()** function), the information displayed is:

```
                 CURRENT  HIGHWATER    PROCESS
                   USAGE      USAGE      LIMIT
MAXFILEPROC            0          0        100,256
    .
    .
    .
```

Example 13

To display thread-level information for any thread that is in a byte-range lock wait. enter:

```
D OMVS,PID=16777219,BRL
```

```
BPXO040I 13.50.54 DISPLAY OMVS 042
OMVS     000E ACTIVE          OMVS=(99)
USER     JOBNAME  ASID        PID       PPID STATE   START     CT_SECS
WELLIE0  WELLIE0  0015   16777219   16777218 1CI   14.11.53       .703
  LATCHWAITPID=         0 CMD=sh -L
 THREAD_ID        TCB@     PRI_JOB  USERNAME   ACC_TIME SC  STATE
 250640E000000002 009C8550 OMVS                    .124 RED  C
 BRLWAIT DEV=00000001 INO=0000002E FILE=/u/john/filenam+ PID=12345678
```

The information displayed is:

FILE
Up to 16 characters of the filename of the file that is being locked. It the filename has more than 16 characters, the first 15 are displayed, followed by a plus sign (+).

PID
The process ID of another process that is blocking this process from obtaining

the lock. Usually this is the owner (or one of the owners) of a lock on the same range, but sometimes it is another process that is also waiting.

INO
The inode number of the file, as shown by **ls -i**.

DEV
The device number of the file's mounted file system.

SETOMVS Command

Use the SETOMVS command to change dynamically the options that OS/390 UNIX System Services currently is using. These options are originally set in the BPXPRMxx parmlib member at the time of initially program loading (IPL'ing) the system. For more information on the BPXPRMxx parmlib member, see *OS/390 UNIX System Services Planning*.

Changes to all of the system-wide limits take effect immediately. When a process limit is updated, all processes that are using the system-wide process limit have their limits updated. All process limit changes take effect immediately, except for those processes with a user-defined process limit (defined in the OMVS segment or set with a SETOMVS PID= command). An exception is MAXASSIZE and MAXCPUTIME, which are not changed for active processes.

Note: If a process-level limit is lowered with the SETOMVS command, there may be some processes that will immediately hit 100% usage. Depending on the process limit specified and what the process is doing, this could cause failure for some processes.

Syntax

The complete syntax for the SETOMVS command is:

SETOMVS	SETOMVS EXTENSIONS (sysplex exclusive)
SETOMVS [FORKCOPY=(COPY|COW)] [,IPCSEMNIDS=ipcsemnids] [,IPCSEMNOPS=ipcsemnops] [,IPCSEMNSEMS=ipcsemnsems] [,IPCMSGQBYTES=ipcmsgqbytes] [,IPCMSGNIDS=ipcmsgnids] [,IPCSHMMPAGES=ipcshmmpages] [,IPCSHMNIDS=ipcshmnids] [,IPCSHMNSEGS=ipcshmnsegs] [,IPCSHMSPAGES=ipcshmspages] [,IPCMSGQMNUM=ipcmsgqmnum] [,MAXASSIZE=maxassize] [,MAXCORESIZE=maxcoresize] [,MAXCPUTIME=maxcputime] [,MAXFILEPROC=maxfileproc] [,MAXFILESIZE=(maxfilesize|NOLIMIT)] [,MAXMMAPAREA=maxmmaparea] [,MAXPROCSYS=maxprocsys] [,MAXPROCUSER=maxprocuser] [,MAXPTYS=maxptys] [,MAXRTYS=maxrtys] [,MAXSHAREPAGES=maxsharepages] [,MAXTHREADS=maxthreads] [,MAXTHREADTASKS=maxthreadtasks] [,MAXUIDS=maxuids] [,PID=pid,processlimitname=newvalue] [,PRIORITYGOAL=(n) | NONE] [,PRIORITYPG=(n) | NONE] ; [,STEPLIBLIST='stepliblist'] [,SUPERUSER=superuser] [,SYNTAXCHECK='parmlibmember'] [,TTYGROUP=ttygroup] [,USERIDALIASTABLE=useridaliastable] [,VERSION='string'] [,LIMMSG=[NONE|SYSTEM|ALL]]	SETOMVS FILESYS ,FILESYSTEM=filesystem ,AUTOMOVE=YES|NO ,SYSNAME=sysname|* or SETOMVS FILESYS ,FILESYSTEM=filesystem ,AUTOMOVE=YES|NO or SETOMVS FILESYS ,FILESYSTEM=filesystem ,SYSNAME=sysname|* or SETOMVS FILESYS ,MOUNTPOINT=mountpoint ,AUTOMOVE=YES|NO ,SYSNAME=sysname|* or SETOMVS FILESYS ,MOUNTPOINT=mountpoint ,AUTOMOVE=YES|NO or SETOMVS FILESYS ,MOUNTPOINT=mountpoint ,SYSNAME=sysname|* or SETOMVS FILESYS ,FROMSYS=sysname ,SYSNAME=sysname|* **Note: FILESYSTEM**, **MOUNTPOINT**, and **FROMSYS** are mutually exclusive parameters. When you specify **FILESYS**, you must supply one of these three parameters.

Parameters

AUTOMOVE=YES|NO, FILESYS=filesys, FILESYSTEM=filesystem, FROMSYS=sysname, MOUNTPOINT=mountpoint, SYSNAME=sysname|*, and VERSION='nnnn', which are described in this section, are parameters that are used in a sysplex environment where systems are exploiting shared HFS. For more information on shared HFS in a sysplex, see *OS/390 UNIX System Services Planning*.

The parameters are:

AUTOMOVE=YES|NO
The AUTOMOVE|NOAUTOMOVE parameters apply only in a sysplex where systems are participating in shared HFS. The AUTOMOVE and NOAUTOMOVE parameters indicate what happens if the system that owns a file system goes down. AUTOMOVE indicates that ownership of the file system automatically changes to another system participating in shared HFS. NOAUTOMOVE indicates that ownership of the file system is not moved if the owning system goes down; as a result, the file system becomes inaccessible. AUTOMOVE is the default.

Note: **AUTOMOVE** is not allowed when moving multiple filesystems. Also, in OS/390 R9 and later, to ensure that the root is always available, use the default for AUTOMOVE.

FILESYS=filesys
In a sysplex environment, this parameter alerts the parser that commands which change mount attributes are forthcoming.

For examples on the use of this parameter when making move or change requests, see *OS/390 UNIX System Services Planning*.

FILESYSTEM=filesystem
In a sysplex environment, **FILESYSTEM** is the 45 character alphanumeric field that denotes the name of the filesystem to be changed or moved. This filesystem name may be in the following form: 'OMVS.USER.JOE'. **FILESYSTEM**, **MOUNTPOINT**, and **FROMSYS** are mutually exclusive parameters.

For examples on the use of this parameter when making move or change requests, see *OS/390 UNIX System Services Planning*.

FROMSYS=sysname
In a sysplex environment, this parameter indicates the system where all the filesystems will be moved from. The filesystems will be moved to the system identified by the **sysname** keyword. **FILESYSTEM**, **MOUNTPOINT**, and **FROMSYS** are mutually exclusive parameters.

MOUNTPOINT=mountpoint
In a sysplex environment, MOUNTPOINT is the mountpoint specification. For example:

```
'/usr/d1'
```

It is case sensitive. This is the mountpoint where the filesystem is mounted. If specified, the filesystem associated with this mountpoint will be moved or changed. **FILESYSTEM**, **MOUNTPOINT**, and **FROMSYS** are mutually exclusive parameters.

For examples on the use of this parameter when making move or change requests, see *OS/390 UNIX System Services Planning*.

FORKCOPY = (COPY|COW)
Specifies how user storage is copied from the parent process to the child process during a **fork()** system call.

If you specify **FORKCOPY(COW)**, all **fork()** calls are processed in copy-on-write (COW) mode if the suppression-on-protection hardware feature is available. Before the storage is modified, both the parent and child processes refer to the same view of the data. The parent storage is copied to the child as soon as storage is modified, either by the parent or the child.

Using copy-on-write causes the system to use the extended system queue area (ESQA) to manage page sharing.

If you specify **FORKCOPY(COPY)**, **fork()** immediately copies the parent storage to the child, regardless of whether the suppression-on-protection feature is available. Use this option to avoid any additional ESQA use in support of **fork()**.

Follow these guidelines:

- If the run-time library is in the link pack area, specify **FORKCOPY(COPY)**.
- If the run-time library is not in the link pack area, specify **FORKCOPY(COW)**.

If you do not specify **FORKCOPY**, the default is **FORKCOPY(COW)**.

IPCSEMNIDS = ipcsemnids
Specifies the maximum number of unique semaphore sets in the system. The range is from 1 to 20 000. The default is 500.

IPCSEMNOPS = ipcsemnops
Specifies the maximum number of operations for each semaphore operation call. The range is from 0 to 32 767. The default is 25. This is a system-wide limit.

IPCSEMNSEMS = ipcsemnsems
Specifies the maximum number of semaphores for each semaphor set. The range is from 0 to 32 767. The default is 25.

IPCMSGQBYTES = ipcmsgqbytes
Specifies the maximum number of bytes in a single message queue. The range is from 0 to 1 048 576. The default is 262 144.

IPCMSGNIDS = ipcmsgnids
Specifies the maximum number of unique message queues in the system. The range is from 1 to 20 000. The default is 500.

IPCSHMMPAGES = ipcshmmpages
Specifies the maximum number of pages for a shared memory segment. The range is from 1 to 25 600. The default is 256.

IPCSHMNIDS = ipcshmnids
Specifies the maximum number of unique shared memory segments in the system. The range is from 1 to 20 000. The default is 500.

IPCSHMNSEGS = ipcshmnsegs
Specifies the maximum number of shared memory segments attached for each address space. The range is from 0 to 1 000. The default is 10.

IPCSHMSPAGES = ipcshmspages
Specifies the maximum number of pages for shared memory segments in the system. The range is from 0 to 2 621 440. The default is 262 144.

IPCMSGQMNUM = ipcmsqgmnum
Specifies the maximum number of messages for each message queue in the system. The range is from 0 to 20 000. The default is 10 000.

LIMMSG=(NONE|SYSTEM|ALL)
Specifies how console messages that indicate when system parmlib limits are reaching critical levels are to be displayed:

NONE No console messages are to be displayed when any of the parmlib limits have been reached.

SYSTEM
Console messages are to be displayed for all processes that reach system limits. In addition, messages are to be displayed for each process limit of a process if:
- The process limit or limits are defined in the OMVS segment of the owning User ID
- The process limit or limits have been changed with a SETOMVS PID=*pid*,*process_limit*

ALL Console messages are to be displayed for the system limits and for the process limits, regardless of which process reaches a process limit.

Default: NONE

MAXASSIZE = maxassize
Specifies the RLIMIT_AS hard limit resource value that processes receive when they are dubbed a process. RLIMIT_AS indicates the address space region size. The soft limit is obtained from MVS. If the soft limit value from MVS is greater than the MAXASSIZE value, the hard limit is set to the soft limit.

This value is also used when processes are initiated by a daemon process using an **exec** after **setuid()**. In this case, both the RLIMIT_AS hard and soft limit values are set to the MAXASSIZE value.

Refer to the description of **setrlimit()** in *OS/390 UNIX System Services Programming: Assembler Callable Services Reference* for more information about RLIMIT_AS.

The range is from 10 485 760 (10MB) to 2 147 483 647; the default is 41 943 040 (40MB).

MAXCORESIZE = maxcoresize
Specifies the RLIMIT_CORE soft and hard limit resource values that processes receive when they are dubbed a process. RLIMIT_CORE indicates the maximum core dump file size (in bytes) that a process can create. Also, it specifies the limit when they are initiated by a daemon process using an **exec** after **setuid()**.

Refer to the description of **setrlimit()** in *OS/390 UNIX System Services Programming: Assembler Callable Services Reference* for more information about RLIMIT_CORE.

The range is from 0 to 2 147 483 647; the default is 4 194 304 (4MB).

MAXCPUTIME = maxcputime
Specifies the RLIMIT_CPU hard limit resource values that processes receive when they are dubbed a process. RLIMIT_CPU indicates the CPU time that a process is allowed to use, in seconds. The soft limit is obtained from MVS. If the soft limit value from MVS is greater than the MAXCPUTIME value, the hard limit is set to the soft limit. This value is also used when processes are initiated by a daemon process using an **exec** after **setuid()**. In this case, both the RLIMIT_CPU hard and soft limit values are set to the MAXCPUTIME value.

Refer to the description of **setrlimit()** in *OS/390 UNIX System Services Programming: Assembler Callable Services Reference* for more information about RLIMIT_CPU.

The range is from 7 to 2 147 483 647. The default is 1 000.

Specifying a value of 2 147 483 647 indicates unlimited CPU time.

MAXFILEPROC = maxfileproc
Specifies the maximum number of files that a single user is allowed to have concurrently active or allocated. The range is 3 to 65 535.

MAXFILESIZE = (maxfilesize | NOLIMIT)
Specifies the RLIMIT_FSIZE soft and hard limit resource values that processes

receive when they are dubbed a process. RLIMIT_FSIZE indicates the maximum file size (in 4KB increments) that a process can create. Also, it specifies the limit when they are initiated by a daemon process using an **exec** after **setuid()**.

The range is from 0 to 524 228 If you specify 0, no files will be created by the process. Omitting this statement or specifying NOLIMIT indicates an unlimited file size.

MAXMMAPAREA = maxmmaparea
Specifies the maximum amount of data space storage (in pages) that can be allocated for memory mappings of HFS files. Storage is not allocated until memory mappings are active.

The range is from 1 to 16 777 216. The default is 4 096.

MAXPROCSYS = maxprocsys
Specifies the maximum number of processes that OS/390 UNIX System Services will allow to be active at the same time. The range is 5 to 32 767; the default and the value in BPXPRMXX is 200.

MAXPROCUSER = maxprocuser
Specifies the maximum number of processes that a single OMVS user ID (UID) is allowed to have active at the same time, regardless of how the process became an OS/390 UNIX process. The range is 3 to 32 767;

MAXPTYS = maxptys
Specifies the maximum number of pseudo-TTY (pseudoterminal) sessions that can be active at the same time. The range is 1 to 10 000; the default and the value in BPXPRMXX is 256.

MAXPTYS lets you manage the number of interactive shell sessions. When you specify this value, each interactive session requires one pseudo-TTY pair. You should avoid specifying an arbitrarily high value for MAXPTYS. However, because each interactive user may have more than one session, we recommend that you allow 4 pseudo-TTY pairs for each user (MAXUIDS * 4). The MAXPTYS value influences the number of pseudo-TTY pairs that can be defined in the file system.

MAXRTYS = maxrtys
Specifies the maximum number of remote-terminal sessions that can be active concurrently. The range is from 1 to 10 000; the default and the value in BPXPRMXX is 256.

MAXRTYS lets you manage the number of interactive shell sessions that are accessed by OCS terminal support. When you specify this value, each interactive session requires one remote TTY. You should avoid specifying an arbitrarily high value for MAXRTYs. However, because each interactive user may have more than one session, we recommend that you allow 4 remote TTY files for each user (MAXUIDS * 4).

The value of this parameter has no effect unless OCS support is active.

The MAXRTYS value influences the configuration of OCS nodes and associated terminal files.

MAXSHAREPAGES = maxsharepages
Specifies the maximum number of shared storage pages that can be

concurrently in use by OS/390 UNIX functions. This can be used to control the amount of ESQA consumed, since the shared storage pages cause the consumption of ESQA storage.

The range is from 0 to 32 768 000. The default is 131 072 pages.

MAXTHREADS = maxthreads
Specifies the maximum number of pthread_created threads, including those running, queued, and exited but not detached, that a single process can have currently active. Specifying a value of 0 prevents applications from using pthread_create. The range is 0 to 100 000; the default and the value in BPXPRMXX is 200.

MAXTHREADTASKS = maxthreadtasks
Specifies the maximum number of MVS tasks created with pthread_create (BPX1PTC) that a single user may have concurrently active in a process. The range is 1 to 32 768; the default and the value in BPXPRMXX is 50.

MAXTHREADTASKS lets you limit the amount of system resources available to a single user process.

- The minimum value of 1 prevents a process from performing any pthread_creates.
- A high MAXTHREADTASKS value may affect storage and performance. Each task requires additional storage for:
  - The control blocks built by the OpenMVS kernel
  - The control blocks and data areas required by the runtime library
  - System control blocks such as the TCB and RB

Individual processes can alter these limits dynamically.

MAXUIDS = maxuids
Specifies the maximum number of unique OMVS user IDs (UIDs) that can use OS/390 UNIX at the same time. The UIDs are for interactive users or for programs that requested OS/390 UNIX. The range is 1 to 32 767; the default and the value in BPXPRMXX is 200.

MAXUIDS lets you limit the number of active UIDs. Select a MAXUIDS by considering:

- Each OS/390 UNIX user is likely to run with 3 or more concurrent processes. Therefore, OS/390 UNIX users require more system resources than typical TSO/E users.
- If the MAXUIDS value is too high relative to the MAXPROCSYS value, too many users can invoke the shell. All users may be affected, because forks may begin to fail.

  For example, if your installation can support 400 concurrent processes — MAXPROCSYS(400) — and each UID needs an average of 4 processes, then the system can support 100 users. For this operating system, specify MAXUIDS(100).

In assigning a value to MAXUIDS, consider if the security administrator assigned the same OMVS UID to more than one TSO/E user ID.

PID=pid,processlimitname=value
Dynamically changes a process-level limit for the process represented by *pid*.

PRIORITYGOAL = (n) | NONE
Specify from 1 to 40 service classes. These classes can be from 1 to 8

characters. If you do not specify this statement, or if you specify NONE, no array is created for it. All service classes specified on the PRIORITYGOAL option must also be specified in your workload manager service policy.

Generally, we do not recommend that you set PRIORITYGOAL.

PRIORITYPG = (n) | NONE

Specify from 1 to 40 performance group numbers, in a range from 1 to 999. If you do not specify this statement, or if you specify NONE, no array is created for it. If you specify fewer than 40 values, the last value specified is propagated to the end of the array.

All performance groups specified on the PRIORITYPG statement must also be specified in the IEAIPSxx parmlib member.

Generally, we do not recommend that you set PRIORITYPG.

RESET = (xx)

Specifies the parmlib file containing parameters that are to be applied immediately to the running OS/390 UNIX System Services environment. The variable specifies the character suffix of the BPXPRMxx member that is to be used to change the environment. Any properly constructed BPXPRMxx member can be used. This parameter accepts only the single keyword and parmfile specification. Additional keywords separated by commas are not accepted.

The SETOMVS RESET command is similar to the SET OMVS command. The following table shows the acceptable parameters for each.

Note: SETOMVS RESET accepts only a single parameter; SET OMVS accepts more than one parameter.

Table 4. Acceptable Parameter Statements and Their Applicability

Parameter Statement	SET OMVS= (xx, yy, ...)	SETOMVS RESET= (xx)
MAXASSIZE	Yes	Yes
MAXCPUTIME	Yes	Yes
MAXCORESIZE	Yes	Yes
MAXFILESIZE	Yes	Yes
MAXFILEPROC	Yes	Yes
MAXMMAPAREA	Yes	Yes
MAXPROCSYS	Yes	Yes
MAXPROCUSER	Yes	Yes
MAXPTYS	Yes	Yes
MAXSHAREPAGES	Yes	Yes
MAXTHREADTASKS	Yes	Yes
MAXTHREADS	Yes	Yes
MAXRTYS	Yes	Yes
MAXUIDS	Yes	Yes
IPCMSGNIDS	Yes	Yes
IPCMSGQBYTES	Yes	Yes
IPCMSGQMNUM	Yes	Yes
IPCSEMNIDS	Yes	Yes

Table 4. Acceptable Parameter Statements and Their Applicability (continued)

Parameter Statement	SET OMVS= (xx, yy, ...)	SETOMVS RESET= (xx)
IPCSEMNOPS	Yes	Yes
IPCSEMNSEMS	Yes	Yes
IPCSHMMPAGES	Yes	Yes
IPCSHMNIDS	Yes	Yes
IPCSHMNSEGS	Yes	Yes
IPCSHMSPAGES	Yes	Yes
FORKCOPY	Yes	Yes
STEPLIBLIST	Yes	Yes
USERIDALIASTABLE	Yes	Yes
PRIORITYPG	Yes	Yes
PRIORITYGOAL	Yes	Yes
TTYGROUP	Yes	Yes
SUPERUSER	Yes	Yes
CTRACE	No	No
SYSCALL_COUNTS	Yes	Yes
FILESYSTYPE	No	Yes
SUBFILESYSTYPE	No	Yes
NETWORK	No	Yes
MOUNT	No	No
ROOT	No	No
RUNOPTS	No	No
STARTUP_PROC	No	No
STARTUP_EXEC	No	No
SYSPLEX	No	No
AUTOMOVE	No	No
FILESYS	No	No
FILESYSTEM	No	No
FROMSYS	No	No
MOUNTPOINT	No	No
SYSNAME	No	No
VERSION	Yes	Yes

STEPLIBLIST = 'stepliblist'

Specifies the path name of a hierarchical file system (HFS) file. This file is intended to contain a list of data sets that are sanctioned by the installation for use as step libraries during the running of set-user-ID and set-group-ID executable programs.

SUPERUSER = superuser

This statement specifies a superuser name. You can specify a 1-to-8-character name that conforms to restrictions for an OS/390 user ID. The user ID specified on SUPERUSER must be defined to the security product and should have a

UID of 0 assigned to it. The user ID specified with **setuid()** is used when a daemon switches to an unknown identity with a UID of 0.

The default is SUPERUSER(BPXROOT).

SYNTAXCHECK=(xx)
Specifies that the operator wishes to check the syntax of the designated parmlib member. For example, to check the syntax of BPXPRMZ1 the operator enters:

```
SETOMVS SYNTAXCHECK=(Z1)
```

The system returns a message indicating either that the syntax is correct or that syntax errors were found and written into the hard copy log. This command parses the parmlib member in the same manner, and with the same messages as during IPL.

Note: **SYNTAXCHECK** checks only syntax and does not verify that HFS data sets or mount points are valid.

SYSCALL_COUNTS = (YES|NO)
Specifies whether to accumulate syscall counts in internal kernel data areas so that the RMF data gatherer can record this information.

The default is NO.

If you specify YES, the path length for the most frequently used kernel system calls is increased by more than 150 instructions.

**SYSNAME=sysname|*
sysname is the 1-8 alphanumeric name of a system participating in shared HFS. This system must be IPLed with SYSPLEX(YES). **sysname** specifies the particular system on which a mount should be performed. This system will then become the owner of the file system mounted. If ***(asterisk) is specified, it represents any other randomly selected system taking part in shared HFS. The asterisk specification is not available with the **FROMSYS** parameter.

For examples of the use of this parameter when making move or change requests, see ″Shared HFS in a Sysplex″ in *OS/390 UNIX System Services Planning*.

TTYGROUP = ttygroup
This specifies a 1-to-8-character name that must conform to the restrictions for an OS/390 group name. Slave pseudoterminals (ptys) and OCS rtys are given this group name when they are first opened. This group name should be defined to the security product and have a unique GID. No users should be connected to this group.

The name is used by certain **setgid()** programs, such as **talk** and **write**, when attempting to write to another user's pty or rty.

The default is TTYGROUP(TTY).

USERIDALIASTABLE = 'useridaliastable'
Enables installations to associate alias names with MVS user IDs and group names. If specified, the alias names are used in OS/390 UNIX System Services processing for the user IDs and group names listed in the table.

Specifying USERIDALIASTABLE causes performance to degrade slightly. The more names that you define, the greater the performance degradation. Installations are encouraged to continue using uppercase-only user IDs and group names.

The USERIDALIASTABLE statement specifies the pathname of a hierarchical file system (HFS) file. This file is intended to contain a list of MVS user IDs and group names with their associated alias names.

VERSION = 'nnnn'

The VERSION statement applies only to systems that are exploiting shared HFS. VERSION allows multiple releases and service levels of the binaries to coexist and participate in HFS sharing. A directory with the value *nnnn* specified on VERSION is dynamically created at system initialization under the sysplex root and is used as a mount point for the version HFS. This directory, however, is only dynamically created if the sysplex root HFS is mounted read/write.

Note: *nnnn* is a case-sensitive character string no greater than 8 characters in length. It indicates a specific instance of the version HFS. The most appropriate values for *nnnn* are the name of the target zone, &SYSR1, or another qualifier meaningful to the system programmer. For example, if the system is at V2R9, you can specify REL9 for VERSION.

When SYSPLEX(YES) is specified, you must also specify the VERSION parameter.

The VERSION value is substituted in the content of symbolic links that contain $VERSION. For scenarios describing the use of the version HFS, see ″Shared HFS in a Sysplex″ in *OS/390 UNIX System Services Planning*.

When testing or changing to a new Maintenance Level (PTF), you can change the VERSION value dynamically by using the SETOMVS command:

```
SETOMVS VERSION='string'
```

You can also change the settings of this parameter via SET OMVS=(xx) and SETOMVS RESET=(xx) parmlib specifications.

Note: We do not recommend changing version dynamically if you have any users logged on or running applications; replacing the system files for these users may be disruptive.

Chapter 3. APAR OW43776: OS/390 MVS System Commands Summary

Display *or* D OMVS

Example: The following DISPLAY command shows information about OS/390 UNIX System Services (OS/390 UNIX):

```
D OMVS[{,SUMMARY|S}            ]
      |,{ASID|A}=ALL
      |,{ASID|A}=asid
      |,U=userid
      |,{PID}=processid[,BRL]
      |,{FILE|F[,CAPS|C]}
      |,{VSERVER|V}
      |,{PFS|P}
      |,{CINET|CI}=All|TPname
      |,{OPTIONS|O}
      |,{LIMITS|L[,PID=ProcessId][,RESET]}

[,L={a|cc|cca|name|name-a}]
```

SETOMVS Command

Purpose: Use the SETOMVS command to change the options dynamically that OS/390 UNIX System Services uses. These options are originally set in the BPXPRMxx member of SYS1.PARMLIB at the time of initially program loading (IPL'ing) the system.

The complete syntax for the SETOMVS command is:

SETOMVS Command

SETOMVS	SETOMVS EXTENSIONS (sysplex exclusive)
SETOMVS [FORKCOPY=(COPY|COW)] [,IPCSEMNIDS=ipcsemnids] [,IPCSEMNOPS=ipcsemnops] [,IPCSEMNSEMS=ipcsemnsems] [,IPCMSGQBYTES=ipcmsgqbytes] [,IPCMSGNIDS=ipcmsgnids] [,IPCSHMMPAGES=ipcshmmpages] [,IPCSHMNIDS=ipcshmnids] [,IPCSHMNSEGS=ipcshmnsegs] [,IPCSHMSPAGES=ipcshmspages] [,IPCMSGQMNUM=ipcmsgqmnum] [,MAXASSIZE=maxassize] [,MAXCORESIZE=maxcoresize] [,MAXCPUTIME=maxcputime] [,MAXFILEPROC=maxfileproc] [,MAXFILESIZE=(maxfilesize|NOLIMIT)] [,MAXMMAPAREA=maxmmaparea] [,MAXPROCSYS=maxprocsys] [,MAXPROCUSER=maxprocuser] [,MAXPTYS=maxptys] [,MAXRTYS=maxrtys] [,MAXSHAREPAGES=maxsharepages] [,MAXTHREADS=maxthreads] [,MAXTHREADTASKS=maxthreadtasks] [,MAXUIDS=maxuids] [,PID=pid,processlimitname=newvalue] [,PRIORITYGOAL=(n) | NONE] [,PRIORITYPG=(n) | NONE] ; [,STEPLIBLIST='stepliblist'] [,SUPERUSER=superuser] [,SYNTAXCHECK='parmlibmember'] [,TTYGROUP=ttygroup] [,USERIDALIASTABLE=useridaliastable] [,VERSION='string'] [,LIMMSG=[NONE|SYSTEM|ALL]]	SETOMVS FILESYS ,FILESYSTEM=filesystem ,AUTOMOVE=YES|NO ,SYSNAME=sysname|* or SETOMVS FILESYS ,FILESYSTEM=filesystem ,AUTOMOVE=YES|NO or SETOMVS FILESYS ,FILESYSTEM=filesystem ,SYSNAME=sysname|* or SETOMVS FILESYS ,MOUNTPOINT=mountpoint ,AUTOMOVE=YES|NO ,SYSNAME=sysname|* or SETOMVS FILESYS ,MOUNTPOINT=mountpoint ,AUTOMOVE=YES|NO or SETOMVS FILESYS ,MOUNTPOINT=mountpoint ,SYSNAME=sysname|* or SETOMVS FILESYS ,FROMSYS=sysname ,SYSNAME=sysname|* **Note: FILESYSTEM**, **MOUNTPOINT**, and **FROMSYS** are mutually exclusive parameters. When you specify **FILESYS**, you must supply one of these three parameters.

Chapter 4. APAR OW43776: OS/390 MVS Initialization and Tuning Reference

BPXPRMxx (OS/390 UNIX System Services Parameters)

BPXPRMxx contains the parameters that control the OS/390 UNIX System Services (OS/390 UNIX) environment and the file systems. IBM recommends that you have two BPXPRMxx parmlib members, one defining the values to be used for system setup and the other defining the file systems. This makes it easier to migrate from one release to another, especially when using the ServerPac method of installation.

After installation is complete, the operator needs to specify OMVS=xx in the IEADYDxx parmlib member. To specify which BPXPRMxx parmlib member to start with, the operator can include OMVS=xx in the reply to the IPL message or OMVS=xx in the IEASYSxx parmlib member. The two alphanumeric characters, represented by xx, are appended to BPXPRM to form the name of the BPXPRMxx parmlib member.

If OMVS=xx is not specified in the reply to the IPL message or is not in the IEASYSxx member, or if OMVS=DEFAULT is specified, defaults are used for each parameter and OMVS is started in minimum mode. For more information about running in minimum mode and full function mode, see *OS/390 UNIX System Services Planning*. If the operator specifies OMVS=xx in the IPL reply to the message, it overrides the OMVS=xx specified in IEASYSxx.

Note: The START OMVS,OMVS=xx command is not valid when issued from the command console. OMVS=xx is not valid in parmlib COMMNDxx.

You can use multiple parmlib members to start OMVS. This is shown by the following reply to the IPL message:

```
R 0,CLPA,SYSP=R3,LNK=(R3,R2,L),OMVS=(AA,BB,CC)
```

The parmlib member BPXPRMCC would be processed first, followed by and overridden by BPXPRMBB, followed by and overridden by BPXPRMAA. This means that any parameter in BPXPRMAA has precedence over the same parameter in BPXPMRBB and BPXPRMCC.

For example, if you specify MAXFILESIZE in all three parmlib members, the value MAXFILESIZE in BPXPRMAA will be the value used to start OMVS.

You can also specify multiple OMVS parmlib members in IEASYSxx. For example:

```
OMVS=(AA,BB,CC)
```

If MOUNT statements are specified in each parmlib member, the files are mounted in the following order: BPXPRMAA, BPXPRMBB, and BPXPRMCC.

To modify BPXPRMxx parmlib settings without re-IPLing, you can use the SETOMVS operator command, or you can dynamically change the BPXPRMxx parmlib members that are in effect by using the SET OMVS operator command. See “Dynamically Changing the BPXPRMxx Values” in *OS/390 UNIX System Services Planning* for more information. See *OS/390 MVS System Commands* for more information about the SETOMVS and SET OMVS commands.

Syntax Rules for BPXPRMxx

When customizing BPXPRMxx, the following rules apply:

- If the member contains duplicates of these statements, the last occurrence is used. If a statement that has a default is omitted, the default is used.
- Use columns 1 through 71 for data; columns 72 through 80 are ignored.
- Enter one or more statements on a line, or use several lines for one statement.
- Use blanks as delimiters. Multiple blanks are interpreted as a single blank. Blanks are allowed between parameters and values; for example, MAXPROCSYS(500) and MAXPROCSYS (500) are allowed and have the same meaning.
- Comments may appear in columns 1-71 and must begin with "/*" and end with "*/".
- Enter values in uppercase, lowercase, or mixed case. The system converts the input to uppercase, except for values enclosed in single quotes, which are processed without changing the case.
- Values that require single quotes and that are the only ones allowed to be in single quotes are:
  - STEPLIBLIST
  - USERIDALIASTABLE
  - FILESYSTEM in the ROOT and MOUNT statements
  - MOUNTPOINT in the MOUNT statement
  - PARM in the FILESYSTYPE, ROOT, MOUNT, and SUBFILESYSTYPE statements
  - RUNOPTS
  - VERSION
- Enclose values in single quotes, using the following rules:
  - Two single quotes next to each other on the same line are considered as a single quote. For example, `John''s file` is considered to be `John's file`. One quote in column 71 and another in column 1 of the next line are *not* considered as a single quote. This input is treated as two strings or an error.
  - Because some values can be up to 1023 characters, a value can require multiple lines. Place one quote at the beginning of the value, stop the value in column 72 of each line, continue the value in column 1 of the next line, and complete the value with one quote. For example:

```
column                                                          column
1                                                                  71
|                                                                   |
   MOUNT FILESYSTEM('HFS.WORKDS') MOUNTPOINT('/u/john/namedir1/namedir2
/namedir3/namedir4') TYPE(HFS) MODE(RDWR)
```

Syntax of BPXPRMxx

```
{MAXPROCSYS(nnnnn)}

{MAXPROCUSER(nnnnn)}

{MAXUIDS(nnnnn)}

{MAXFILEPROC(nnnnn)}

{MAXTHREADTASKS(nnnnn)}

{MAXTHREADS(nnnnnn)}

{MAXPTYS(nnnnn)}
```

```
{MAXRTYS(nnnnn)}

{MAXFILESIZE(nnnnn|NOLIMIT)}

{MAXCORESIZE(nnnnn)}

{MAXASSIZE(nnnnn)}

{MAXCPUTIME(nnnnn)}

{MAXMMAPAREA(nnnnn)}

{MAXSHAREPAGES(nnnnn)}

{SHRLIBRGNSIZE(nnnnn)}

{SHRLIBMAXPAGES(nnnnn)}

{PRIORITYPG(n1,n2,...n40|NONE)}

{PRIORITYGOAL(service_class_name1,service_class_name2,...service_class_name40|NONE)}

{IPCMSGNIDS(nnnnn)}

{IPCMSGQBYTES(nnnnn)}

{IPCMSGQMNUM(nnnnn)}

{IPCSEMNIDS(nnnnn)}

{IPCSEMNOPS(nnnnn)}

{IPCSEMNSEMS(nnnnn)}

{IPCSHMMPAGES(nnnnn)}

{IPCSHMNIDS(nnnnn)}

{IPCSHMNSEGS(nnnnn)}

{IPCSHMSPAGES(nnnnn)}

{FORKCOPY(COW|COPY)}

{SUPERUSER(user_name)}

{TTYGROUP(group_name)}

{CTRACE(parmlib_member_name)}

{STEPLIBLIST('/etc/steplib')}

{USERIDALIASTABLE('/etc/tablename')}




{FILESYSTYPE TYPE(type_name)
       ENTRYPOINT(entry_name)
       PARM('parm')}

{SYSPLEX(YES|NO)}

{VERSION('nnnn')}
```

```
 {ROOT  FILESYSTEM('fsname') or DDNAME(ddname)
       TYPE(type_name)
       MODE(access)
       PARM('parameter')
       SETUID|NOSETUID
       SYSNAME(sysname)
       AUTOMOVE|NOAUTOMOVE}

{MOUNT FILESYSTEM('fsname') or DDNAME(ddname)
       TYPE(type_name)
       MOUNTPOINT('pathname')
       MODE(access)
       PARM('parameter')
       SETUID|NOSETUID
       WAIT|NOWAIT
       SECURITY|NOSECURITY
       SYSNAME(sysname)
       AUTOMOVE|NOAUTOMOVE}

{NETWORK DOMAINNAME(sockets_domain_name)
       DOMAINNUMBER(sockets_domain_number)
       MAXSOCKETS(nnnnn)
       TYPE(type_name)
       INADDRANYPORT(starting_port_number)
       INADDRANYCOUNT(number_of_ports_to_reserve)}

{SUBFILESYSTYPE NAME(transport_name)
       TYPE(type_name)
       ENTRYPOINT(entry_name)
       PARM('parameter')
       DEFAULT}

{STARTUP_PROC(procname)}

{STARTUP_EXEC('dsname(membername)',class)}

{RUNOPTS('string')}

{SYSCALL_COUNTS(YES/NO)}

{MAXQUEUEDSIGS(nnnnnn)}

{LIMMSG(NONE|SYSTEM|ALL)}
```

Syntax Example of BPXPRMxx

```
MAXPROCSYS(400)
MAXPROCUSER(16)
MAXUIDS(200)
MAXFILEPROC(20)
MAXTHREADTASKS(100)
MAXTHREADS(500)
MAXPTYS(100)
MAXRTYS(100)
MAXFILESIZE(1000)
MAXCORESIZE(4194304)
MAXASSIZE(41943040)
MAXCPUTIME(1000)
MAXMMAPAREA(4096)
MAXSHAREPAGES(32768)
PRIORITYPG(7,7,7,7,7,6,5,999,3,2,1)
PRIORITYGOAL(CICS4,CICS4,CICS4,CICS3,CICS2,CICS1,TSO2,TSO1,BAT3,BAT2)
IPCMSGNIDS(500)
IPCMSGQBYTES(262144)
IPCMSGQMNUM(100000)
IPCSEMNIDS(500)
IPCSEMNOPS(25)
IPCSEMNSEMS(25)
IPCSHMMPAGES(256)
IPCSHMNIDS(500)
IPCSHMNSEGS(10)
IPCSHMSPAGES(262144)
FORKCOPY(COW)
SUPERUSER(BPXROOT)
TTYGROUP(TTY)
CTRACE(CTCBPX23)
STEBLIBLIST('/etc/steplib')
USERIDALIASTABLE('/etc/tablename')
SYSPLEX(YES)
 VERSION('REL9')
 FILESYSTYPE TYPE(HFS)
      ENTRYPOINT(GFUAINIT)
      PARM('SYNCDEFAULT(0) FIXED(2) VIRTUAL(128)')
ROOT  FILESYSTEM('OMVS.ROOT')
      TYPE(HFS)
      MODE(RDWR)
      SYSNAME(SY1)
      AUTOMOVE
MOUNT FILESYSTEM('OMVS.USER.JONES')
      TYPE(HFS)
      MOUNTPOINT('/u/jones')
      MODE(RDWR)
      SYSNAME(SY1)
      NOAUTOMOVE
FILESYSTYPE TYPE(INET)
      ENTRYPOINT(EZBPFINI)
NETWORK DOMAINNAME(AF_INET)
      DOMAINNUMBER(2)
      MAXSOCKETS(2000)
      TYPE(INET)
STARTUP_PROC(OMVS)
STARTUP_EXEC('OMVS.ROOT(REXX01)',A)
RUNOPTS('RTLS(ON) LIBRARY(SYSCEE) VERSION(OS24)')
SYSCALL_COUNTS(YES)
MAXQUEUEDSIGS(1000)
```

IBM-Supplied Default for BPXPRMxx

There is no default BPXPRMxx parmlib member. A sample parmlib member BPXPRMXX is provided in SYS1.SAMPLIB.

Statements and Parameters for BPXPRMxx

For guidance information about selecting values for the statements, see **Customizing the OS/390 UNIX Environment** in *OS/390 UNIX System Services Planning*.

MAXPROCSYS(nnnnn)

Specifies the maximum number of OS/390 UNIX processes that the system allows.

Value Range: *nnnnn* is a decimal value from 5 to 32767.

Default: 200

You can use the SETOMVS or SET OMVS command to dynamically increase or decrease the value of MAXPROCSYS. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

If you are using SETOMVS or SET OMVS to change the value, the new value must be within a certain range, or you will get an error message. The range that you can use has a minimum value of 5; the maximum value is based on the following calculation:

```
MIN(32767,MAX(4096,3*initial value))
```

The initial value is the MAXPROCSYS value that was specified during BPXPRMxx initialization. You cannot use a value less than 5. If you want to use a value greater than the current maximum (as calculated by the formula) but lower than the initial maximum (32767), you will have to change the value in BPXPRMxx and re-IPL. For an example of how to calculate the maximum value in the range, see “Dynamically Changing Certain BPXPRMxx Parameter Values“ in *OS/390 UNIX System Services Planning*.

For additional information, see **MAXPROCSYS** in *OS/390 UNIX System Services Planning*.

MAXPROCUSER(nnnnn)

Specifies the maximum number of processes that a single OS/390 UNIX user ID can have concurrently active, regardless of how the processes were created. MAXPROCUSER is the same as the CHILD_MAX variable in the POSIX standard.

A value of 25 is required for FIPS 151-2 compliance and a value of 16 is required for POSIX.1 (ISO/IEC 9945-1:1990[E] IEEE Std 1003.1-1990) standard compliance.

The number of processes is tracked by user ID (UID). When a user attempts to create a new process, the limit value for the user (defined by either the user profile or the default OPTN value) is compared to the value maintained for the user’s UID. If the user maximum is larger than the current process count for the UID, the user can create another process. If not, the user is not allowed to create a new process. For example, if user “A“, with a user-defined limit of 10, tries to create a process and the UID limit is already 12, user “A“ is not allowed to create the new process. Since only 12 processes are currently created, user “B“, with a user-defined limit of 20, is allowed to create a new process.

Use the SETOMVS or SET OMVS command to dynamically increase or decrease the MAXPROCUSER values. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

For additional information, see **MAXPROCUSER** in *OS/390 UNIX System Services Planning*.

Value Range: *nnnnn* is a decimal value from 3 to 32767.

Default: 25

MAXUIDS(nnnnn)
Specifies the maximum number of OS/390 UNIX user IDs (UIDs) that can operate concurrently.

Value Range: *nnnnn* is a decimal value from 1 to 32767.

Default: 200

Use the SETOMVS or SET OMVS command to dynamically increase or decrease the value of MAXUIDS. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

For additional information, see **MAXUIDS** in *OS/390 UNIX System Services Planning*.

MAXFILEPROC(nnnnn)
Specifies the maximum number of files that a single process can have concurrently active or allocated. MAXFILEPROC is the same as the OPEN_MAX variable in the POSIX standard.

Value Range: *nnnnn* is a decimal value from 3 to 65535.

Default: 64

Use the SETOMVS or SET OMVS command to dynamically increase or decrease the value of MAXFILEPROC. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

For additional information, see **MAXFILEPROC** in *OS/390 UNIX System Services Planning*.

MAXTHREADTASKS(nnnnn)
Specifies the maximum number of MVS tasks that a single process can have concurrently active for pthread_created threads.

Value Range: *nnnnn* is a decimal value from 0 to 32768.

Default: 50

You can change the value of MAXTHREADTASKS dynamically using the SETOMVS or SET OMVS command. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

For additional information, see **MAXTHREADTASKS** in *OS/390 UNIX System Services Planning*.

MAXTHREADS(nnnnnn)
Specifies the maximum number of pthread_created threads, including running, queued, and exited but undetached, that a single process can have concurrently active. Specifying a value of 0 prevents applications from using pthread_create.

Value Range: *nnnnnn* is a decimal value from 0 to 100000.

Default: 200

You can change the value of MAXTHREADS dynamically using the SETOMVS or SET OMVS command. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

For additional information, see **MAXTHREADS** in *OS/390 UNIX System Services Planning*.

MAXPTYS(nnnnn)
Specifies the maximum number of pseudoterminals (pseudo-TTYs or PTYs) for the system.

Value Range: *nnnnn* is a decimal value from 1 to 10000.

Default: 256

You can use the SETOMVS or SET OMVS command to dynamically increase the value of MAXPTYS. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

If you are using SETOMVS or SET OMVS to change the value, the new value must be within a certain range. If it is outside the range, you will get an error message. To use a value that is outside this range, you must change the MAXPTYS specification in BPXPRMxx and re-IPL. The range's minimum value is 1 and the maximum is based on the following calculation:

```
MIN(10000,MAX(256,2*initial value)
```

The initial value is the MAXPTYS value that was specified during BPXPRMxx initialization. For an example of how to calculate the maximum value in the range, see "Dynamically Changing Certain BPXPRMxx Parameter Values" in *OS/390 UNIX System Services Planning*.

For additional information, see **MAXPTYS** in *OS/390 UNIX System Services Planning*.

MAXRTYS(nnnnn)
Specifies the maximum number of remote terminal sessions that can be active at the same time.

Value Range: *nnnnn* is a decimal value from 1 to 10000.

Default: 256

You can use the SETOMVS or SET OMVS command to dynamically increase the value of MAXRTYS. To decrease the value of MAXRTYS, you will have to change BPXPRMxx and re-IPL. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

If you are using SETOMVS or SET OMVS to change the value, the new value must be within a certain range, or you will get an error message. To use a value that is outside this range, you must change the MAXRTYS specification in BPXPRMxx and re-IPL. The range's minimum value is the current setting of MAXRTYS, and the maximum is based on the following calculation:

```
MIN(10000,MAX(256,2*initial value)
```

The initial value is the MAXRTYS value that was specified during BPXPRMxx initialization. If you want to use a value greater than the current maximum (as calculated by the formula) but lower than the initial maximum (10000), you will have to change the value in BPXPRMxx and re-IPL. For an example of how to calculate the maximum value in the range, see "Dynamically Changing Certain BPXPRMxx Parameter Values" in *OS/390 UNIX System Services Planning*.

For additional information, see **MAXRTYS** in *OS/390 UNIX System Services Planning*.

MAXFILESIZE(nnnnn|NOLIMIT)

Specifies the RLIMIT_FSIZE soft and hard resource values that a process receives when it is identified as a process. RLIMIT_FSIZE indicates the maximum file size (in 4KB increments) that a process can create. It also specifies the limit when they are initiated by a daemon process using an **exec()** after a **setuid()**. For more information about RLIMIT_FSIZE, see the description of **setrlimit()** in *OS/390 UNIX System Services Programming: Assembler Callable Services Reference*.

Value Range: *nnnnn* is a decimal value from a minimum of 0 to a maximum of greater than 2147483647 (2 gigabytes) in 4 kilobyte increments. If MAXFILESIZE is not specified or MAXFILESIZE(NOLIMIT) is specified, there will be no limit to the size of files created, except for the architectural limit of the system.

If you specify 0, the process does not create any files. Omitting this statement indicates an unlimited file size.

Default: 1000

Use the SETOMVS or SET OMVS command to dynamically increase or decrease the value of MAXFILESIZE. To make a permanent change, edit the BPXPRMxx member that will be used in IPLs.

MAXCORESIZE(nnnnn)

Specifies the RLIMIT_CORE soft and hard resource values that a process receives when it is identified as a process. RLIMIT_CORE indicates the maximum core dump file size (in bytes) that a process can create. It also specifies the limit when they are initiated by a daemon process using an **exec()** after a **setuid()**. For more information about RLIMIT_CORE, see the description of **setrlimit()** in *OS/390 UNIX System Services Programming: Assembler Callable Services Reference*.

Value Range: *nnnnn* is a decimal value from 0 to 2147483647 (2 gigabytes).

Default: 4194304 (4 megabytes) Specifying a value of 2147483647 (2 gigabytes) indicates an unlimited core file size.

Use the SETOMVS or SET OMVS command to dynamically increase or decrease the value of MAXCORESIZE. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

MAXASSIZE(nnnnn)
Specifies the RLIMIT_AS resource values that a process receives when it is identified as a process. RLIMIT_AS indicates the address space region size. For more information about RLIMIT_AS, refer to the description of **setrlimit** in *OS/390 UNIX System Services Programming: Assembler Callable Services Reference*.

The soft limit is obtained from MVS; if it is greater than the MAXASSIZE value, the hard limit is set to the soft limit. This value is also used when processes are initiated by a daemon process using an **exec()** after **setuid()**. In this case, both the RLIMIT_ AS hard and soft limit values are set to the MAXASSIZE specified value.

When processes are initiated by a daemon process using an **exec()** after **setuid()**, this value is used. Therefore, MAXASSIZE will be the region size for all processes created via rlogin or telnet. In this case, both the RLIMIT_AS hard and soft limit values are set to the MAXASSIZE value.

A superuser can override this value by specifying a new region size in the spawn inheritance structure on __**spawn()**. Or you can change the value of MAXASSIZE dynamically by using the SETOMVS or SET OMVS command.This change only affects the new processes created after the change was made.

Note: The IEFUSI user exit can modify the region size of an address space. Users are strongly discouraged from altering the region size of address spaces in the OMVS subsystem category.

Value Range: *nnnnn* is a decimal value from 10485760 (10 megabytes) to 2147483647 (2 gigabytes).

Default: 41943040 (40 megabytes)

Use the SETOMVS or SET OMVS command to dynamically increase or decrease the value of MAXASSIZE. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

For additional information, see **MAXASSIZE** in *OS/390 UNIX System Services Planning*.

MAXCPUTIME(nnnnn)
Specifies the RLIMIT_CPU resource values that a process receives when it is identified as a process. RLIMIT_CPU indicates the CPU time, in seconds, that a process can use. For more information about RLIMIT_CPU, refer to the description of **setrlimit()** in *OS/390 UNIX System Services Programming: Assembler Callable Services Reference*.

If the soft limit value from MVS is greater than the MAXCPUTIME value, the hard limit is set to the soft limit. This value is also used when processes are initiated by a daemon process using an **exec()** after **setuid()**. In this case, both the RLIMIT_CPU hard and soft limit values are set to the MAXCPUTIME value.

A superuser can override this value by specifying a new time limit in the spawn inheritance structure on __**spawn()**.

For processes running in or forked from TSO or BATCH, the MAXCPUTIME value has no effect. The TIME limit is inherited from the parent. If a TIME parameter is specified on the JCL for the started task, then that value is used. If not, then the TIME value is taken from the JES default TIME value.

For processes created by the **rlogin** command or other daemons, MAXCPUTIME is the time limit for the address space.

Value Range: *nnnnn* is a decimal value from 7 to 2147483647 (2 gigabytes).

Default: 1000

Use the SETOMVS or SET OMVS command to dynamically increase the value of MAXCPUTIME. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

For additional information, see **MAXCPUTIME** in *OS/390 UNIX System Services Planning*.

MAXMMAPAREA(nnnnn)

Specifies the maximum amount of data space storage space (in pages) that can be allocated for memory mappings of HFS files. Storage is not allocated until the memory mapping is active.

Using memory map services causes a large amount of system memory to be consumed. For each page (4KB) that is memory-mapped, 96 bytes of ESQA are consumed when a file is not shared with any other users. When a file is shared by multiple users, each user after the first causes 32 bytes of ESQA to be consumed for each shared page. Assuming that the default of 4096 pages is taken, and assuming that no sharing is done by **mmap()** users, a maximum of 384KB of ESQA could be consumed. The ESQA storage is consumed when the **mmap()** function is invoked rather than when the page is accessed by the memory mapping application program.

If you have applications using the __MAP_MEGA option, you can map very large files without the system overhead in ESQA. For more information, see “Extended System Queue Area (ESQA“ in *OS/390 UNIX System Services Planning*.

Value Range: *nnnnn* is a decimal value from 1 to 16777216.

Default: 4096

You can change the value of MAXMMAPAREA dynamically using the SETOMVS or SET OMVS command. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

For additional information, see **MAXMMAPAREA** in *OS/390 UNIX System Services Planning*.

MAXSHAREPAGES(nnnnn)

Specifies the maximum amount of shared system storage pages that OS/390 UNIX functions can use. This limit applies to the **mmap**, **shmat**, **ptrace**, and **fork** functions.

The fork service uses shared storage only when FORKCOPY(COW) is specified. Because the **fork** and **ptrace** functions use the shared storage pages

as a boost to performance, the usage is not critical to the completion of these functions. For this reason, when the amount of shared storage pages being used reaches approximately 60% of the specified limit, these functions no longer use the shared storage to complete their function. Because the **shmat()** function is considered the most critical of the functions, it continues to use the shared storage pages until the total consumption reaches the specified limit. The **mmap** function continues to use the shared storage pages until total shared storage consumption reaches approximately 80% of the limit.

Because each page of shared storage requires the associated consumption of extended system queue area (ESQA) storage, limiting the shared storage usage provides a way to limit the ESQA usage by OS/390 UNIX users. If you use the __IPC_MEGA or __MAP_MEGA options, then the shared pages limits are not affected because MEGA does not affect the system ESQA overhead.

Value Range: *nnnnn* is a decimal value from 0 to 32768000.

Default: 131072

Use the SETOMVS or SET OMVS command to dynamically increase or decrease the MAXSHAREPAGES value. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

SHRLIBRGNSIZE(nnnnn)
Specifies the size of the shared library region for address spaces that load system shared library modules. For these address spaces, the size specified is allocated from high-end private storage and is used for the loading of system shared library modules. This storage is not allocated in an address space until it loads a system shared library module.

Value Range: *nnnnn* is a decimal value between 16777215 (16 megabytes) and 1610612735 (1.5 gigabytes).

Default: 67108863

For information on setting the value, see **SHRLIBRGNSIZE** in *OS/390 UNIX System Services Planning*.

SHRLIBMAXPAGES(nnnnn)
Specifies the number of data space storage pages that can be allocated for non-system shared library modules. The data space storage is not allocated until a non-system shared library is loaded.

Value Range: *nnnnn* is a decimal value between 1 and 16777215.

Default: 4096

For information on setting the value, see **SHRLIBMAXPAGES** in *OS/390 UNIX System Services Planning*.

PRIORITYPG(n1,n2,...n40)
Specifies a list of 1 to 40 performance group numbers separated by commas, which are used in association with the **setpriority**, **nice** and **chpriority** callable services when the system is running in compatibility mode. These functions allow a program to alter the priority of one or more processes.

Generally, it is recommended that you not set PRIORITYPG unless the nice(), setpriority() or chpriority() values must be enabled.

If the list has less than 40 entries, the system propagates the last performance group specified into the remaining unspecified entries in the table. For example:

```
PRIORITYPG(7,7,7,7,7,6,5,999,3,2,1)
```

The performance groups specified on the PRIORITYPG statement must also be specified in the IEAIPSxx parmlib member.

PRIORITYPG(NONE) means that there are no values. If you do not specify PRIORITYPG, that means that there are no values.

Only superusers can increase their values. Regular users can only decrease their priority values; they cannot increase their priority values. If you do not want to allow your users to increase the priority but still want to enable the **nice()** and **setpriority()** functions, define a range of performance groups or service classes with priority increments on a base that is normal for the users. Using these functions lets the user order the priority of processes, but will not let a user improve performance over that of other users.

Value Range: *n* is a decimal value from 1 to 999.

Default: None

You can use the SETOMVS or SET OMVS command to specify a new range of priority settings. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

For additional information, see **PRIORITYPG** in *OS/390 UNIX System Services Planning*.

PRIORITYGOAL(service_class_name1,service_class_name2,...service_class_name40)

Specifies a list of 1 to 40 service class names of 8 characters or less separated by commas, which are used in association with the **setpriority**, **nice** and **chpriority** callable services when the system is running in goal mode. These functions allow a program to alter the priority of one or more processes.

Generally, it is recommended that you not set PRIORITYGOAL unless the nice(), setpriority() or chpriority() values must be enabled.

If the list has less than 40 entries, the system propagates the last service class specified into the remaining unspecified entries in the table. For example:

```
PRIORITYGOAL(CICS4,CICS4,CICS4,CICS3,CICS2,CICS1,TSO2,TSO1,BAT3,BAT2)
```

If you do not specify this statement, arrays are not created for it. All service classes specified on the PRIORITYGOAL statement must also be specified in your workload manager service policy.

PRIORITYGOAL(NONE) means that there are no values. If you do not specify PRIORITYGOAL, that means that there are no values.

If you do not want to allow users to increase the priority but still want to enable the **nice()** and **setpriority()** functions, define a range of performance groups or service classes with priority increments on a base that is normal for the users.

Using these functions lets the user order the priority of processes, but will not let a user improve performance over that of other users.

Value Range: *service_class_name* is a 1 to 8 character value.

Default: None

You can dynamically change the values of PRIORITYGOAL by using the SETOMVS or SET OMVS command. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

For additional information, see **PRIORITYGOAL** in *OS/390 UNIX System Services Planning*.

IPCMSGNIDS(nnnnn)
Specifies the maximum number of unique system-wide message queues.

Value Range: *nnnnn* is a decimal value from 1 to 20000.

Default: 500

You can change the value of IPCMSGNIDS dynamically using the SETOMVS or SET OMVS command. The new minimum is the current value. The new maximum is calculated as follows:

```
MIN(initial maximum,MAX(4096,3*initial value))
```

You can increase but not decrease the value, as described in *OS/390 UNIX System Services Planning*.

IPCMSGQBYTES(nnnnn)
Specifies the maximum number of bytes in a single message queue.

Value Range: *nnnnn* is a decimal value from 0 to 2147483647.

Note: The high end of this range is not obtainable due to storage constraints. The actual maximum range varies due to storage allocation and system usage.

Default: 262144

You can change the value of IPCMSGQBYTES dynamically using the SETOMVS or SET OMVS command.

IPCMSGQMNUM(nnnnn)
Specifies the maximum number of system-wide messages for each queue.

Value Range: *nnnnn* is a decimal value from 0 to 2147483647.

Note: The high end of this range is not obtainable due to storage constraints. The actual maximum range varies due to storage allocation and system usage.

Default: 10000

You can change the value of IPCMSGQMNUM dynamically using the SETOMVS or SET OMVS command.

IPCSEMNIDS(nnnnn)
Specifies the maximum number of unique system-wide semaphore sets.

Value Range: *nnnnn* is a decimal value from 1 to 20000.

Default: 500

You can change the value of IPCSEMNIDS dynamically using the SETOMVS or SET OMVS command, as described in *OS/390 UNIX System Services Planning*.

IPCSEMNOPS(nnnnn)
Specifies the maximum number of operations for each **semop** call.

Value Range: *nnnnn* is a decimal value from 0 to 32767.

Default: 25

You can change the value of IPCSEMNOPS dynamically using the SETOMVS or SET OMVS command.

IPCSEMNSEMS(nnnnn)
Specifies the maximum number of semaphores for each semaphore set.

Value Range: *nnnnn* is a decimal value from 0 to 32767.

Default: 25

You can change the value of IPCSEMNSEMS dynamically using the SETOMVS or SET OMVS command.

IPCSHMMPAGES(nnnnn)
Specifies the maximum number of pages for shared memory segments.

Value Range: *nnnnn* is a decimal value from 1 to 524287.

Note: The high end of this range is not obtainable due to storage constraints. The actual maximum range varies due to storage allocation and system usage.

Default: 256

You can change the value of IPCSHMMPAGES dynamically using the SETOMVS or SET OMVS command.

IPCSHMNIDS(nnnnn)
Specifies the maximum number of unique system-wide shared memory segments.

Value Range: *nnnnn* is a decimal value from 1 to 20000.

Default: 500

You can change the value of IPCSHMNIDS dynamically using the SETOMVS or SET OMVS command. The new minimum is the same as the current value. The new maximum is calculated as follows:

```
MIN(initial maximum,MAX(4096,3*initial value))
```

You can increase but not decrease the value, as described in *OS/390 UNIX System Services Planning*.

IPCSHMNSEGS(nnnnn)
Specifies the maximum number of attached shared memory segments for each address space.

Value Range: *nnnnn* is a decimal value from 0 to 1000.

Default: 10

You can change the value of IPCSHMNSEGS dynamically using the SETOMVS or SET OMVS command.

IPCSHMSPAGES(nnnnn)
Specifies the maximum number of system-wide shared pages created by calls to the **fork** and **shmat** functions.

Value Range: *nnnnn* is a decimal value from 0 to 2621440.

Default: 262144

You can change the value of IPCSHMSPAGES dynamically using the SETOMVS or SET OMVS command. The new minimum is the same as the current value. The new maximum is calculated as follows:

```
MIN(initial maximum,MAX(4096,3*initial value))
```

You can increase but not decrease the value, as described in *OS/390 UNIX System Services Planning*.

FORKCOPY(COW|COPY)
Specifies how user storage is to be copied from the parent process to the child process during a **fork()** system call.

FORKCOPY(COW) specifies that all **fork()** calls are processed with the copy-on-write mode if the suppression-on-protection (SOP) hardware feature is available. Before the storage is modified, both the parent and child process refer to the same view of the data. The parent storage is copied to the child only if either the parent or the child modifies the storage. FORKCOPY(COW) causes the system to use the ESQA to manage page sharing.

FORKCOPY(COPY) specifies that **fork()** immediately copies the parent storage to the child, whether the SOP is available or not. Use this option to avoid any additional ESQA use in support of fork.

Follow these guidelines:
- If the run-time library is in the link pack area, specify FORKCOPY(COPY).
- If the run-time library is not in the link pack area, specify FORKCOPY(COW).

Default: COW

You can change the value of FORKCOPY dynamically using the SETOMVS or SET OMVS command. To make a permanent change, edit the BPXPRMxx member used for IPLs.

SUPERUSER(user_name)
Superuser name, which must conform to the restrictions for an OS/390 user ID. The user name must also be defined to RACF (or another security product) and

must have an OS/390 UNIX user ID (UID) of 0. For example, in RACF, specify OMVS(UID(0)) on the ADDUSER command.

When a daemon issues a **setuid()** to set a UID to 0 and the user ID is not known, **setuid()** uses the user ID from the SUPERUSER statement.

Never permit the userid BPXROOT to the BPX.DAEMON profile (described in "Setting Up the BPX.* FACILITY Class Profiles" in *OS/390 UNIX System Services Planning*). This warning applies even if you use a name other than BPXROOT.

Value Range: *user_name* is a 1 to 8 character value.

Default: BPXROOT

Use the SETOMVS or SET OMVS command to dynamically change the value of SUPERUSER. To make a permanent change, edit the BPXPRMxx member that is used for IPLs.

TTYGROUP(group_name)
Specifies the OS/390 group name given to slave pseudoterminals (PTYs) and OCS remote terminals (RTYs). This group name should be defined to the security product and must have a unique group ID (GID). No users should be connected to this group.

The group_name is used by certain **setgid()** programs, such as talk and write, when writing to another user's PTY or RTY.

Value Range: *group_name* is a 1 to 8 character value.

Default: TTY

You can change the value of TTYGROUP dynamically using the SETOMVS or SET OMVS command. To make a permanent change, edit the BPXPRMxx member that will be used for future IPLs.

CTRACE(parmlib_member_name)
Specifies the parmlib member that contains the initial tracing options to be used for the OS/390 UNIX component. Use this statement to provide tracing while the kernel is starting and to avoid having to issue a TRACE operator command to set tracing options.

Default: CTIBPX00

STEPLIBLIST('/etc/steplib')
Specifies the pathname of a hierarchical file system (HFS) file. This file is intended to contain a list of MVS datasets that are sanctioned by the installation for use as step libraries for programs that have the set-user-ID and set-group-ID bit set.

Use the SETOMVS or SET OMVS command to dynamically change the value of STEPLIBLIST. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

For additional information, see **STEPLIBLIST** in *OS/390 UNIX System Services Planning*.

USERIDALIASTABLE('/etc/tablename')
Specifies the pathname of a hierarchical file system (HFS) file. This file is intended to contain a list of MVS user IDs and group names with their corresponding alias names. The alias names can contain any characters in the portable filename character set.

You can change USERIDALIASTABLE dynamically using the SETOMVS or SET OMVS command. To make a permanent change, edit the BPXPRMxx member that will be used for IPLs.

Once a user is logged into the system, changing the user ID or group name alias table does not change the alias name immediately. If a change needs to be activated sooner, you can use the SETOMVS or SET OMVS command to change the table more quickly.

For additional information, see **USERIDALIASTABLE** in *OS/390 UNIX System Services Planning*.

FILESYSTYPE TYPE(type_name) ENTRYPOINT(entry_name) PARM('parm')

ASNAME(proc_name)
Specifies the type of file system that is to be started. BPXPRMxx can contain more than one FILESYSTYPE statement.

When SYSPLEX(YES) is specified, each FILESYSTYPE in use within the participating shared HFS group must be defined for all systems participating in shared HFS. The easiest way to accomplish this is by having a single BPXPRMxx member that contains file system information for each system participating in shared HFS. If you decide to define a BPXPRMxx for each system, the FILESYSTYPE statements must be identical on each system. For more information on shared HFS, see “Shared HFS in a Sysplex“ in *OS/390 UNIX System Services Planning*.

Note that any facilities required for a particular FILESYSTYPE must be initiated on all systems participating in shared HFS. For example, NFS requires TCP/IP, so if you specify an NFS FILESYSTYPE, you must also initialize TCP/IP on NFS initialization.

The SETOMVS RESET command can be used to dynamically specify new FILESYSTYPE statements. To make a permanent change, edit the BPXPRMxx member used for IPLs. For more information, see “Dynamically Adding FILESYSTYPE Statements in BPXPRMxx“ in *OS/390 UNIX System Services Planning*.

The parameters are:

TYPE(type_name)
Specifies the name of the file system type that is to control the file system.

In the FILESYSTYPE statement, specify a name for the TYPE file system. For example, you could use the following, or assign your own names:
- HFS for a hierarchical file system (HFS)
- TFS for a temporary file system (TFS)
- UDS for UNIX domain (AF_UNIX) sockets
- INET for network (AF_INET) sockets
- LINET for local (AF_INET) sockets
- CINET for common INET sockets

- AUTOMNT for an automounted file system
- DFSC for accessing global namespace.
- NFS for accessing remote files.

For additional information, see **FILESYSTYPE** in *OS/390 UNIX System Services Planning*.

TYPE is a required parameter. The name is 1 to 8 characters; the system converts the name to uppercase.

ENTRYPOINT(entry_name)
Specifies the name of the load module containing the entry point into the file system type.

ENTRYPOINT is a required parameter. The name is 1 to 8 characters; the system converts the name to uppercase. Refer to the documentation for the specific physical file system for valid entry point names.

PARM('parm')
Provides a parameter to be passed directly to the file system type. The parameter format and content are specified by the file system type.

PARM is an optional parameter. The parameter is up to 1024 characters long; the characters can be in uppercase, lowercase, or both. The parameter must be enclosed in single quotes.

If the physical file system specified does not expect a PARM operand, it ignores all PARM operands.

SYNCDEFAULT(t), VIRTUAL(max), FIXED(min) and **NOWRITEPROTECT** are valid only when **ENTRYPOINT** is GFUAINIT.

Note: If a syntax error is found in any of these four parameters (**SYNCDEFAULT, VIRTUAL, FIXED,** or **NOWRITEPROTECT**), an error message is issued and **all four parameters** are set to the default values.

SYNCDEFAULT(t)

t specifies the number of seconds used as a default for the sync daemon interval. When the sync daemon is active, the meta data for a file system is hardened. Setting *t* to 0 indicates that the file system should harden meta data synchronously with syscall requests.

Sync interval values are rounded up to the next 30-second value. For example, specifying 31 seconds results in a sync interval of 60 seconds.

The maximum value that can be specified for *t* is 65535. Values between 65535 and 99999 are rejected.

A value of 99999 specifies that no sync daemon intervals are specified, and thus, the meta data is not hardened.

Default: 60 seconds

VIRTUAL(max)

max specifies the maximum amount of virtual storage (in megabytes) that HFS data and meta data buffers should use. The minimum value that can be specified is 32M. If less than 32M is specified, an informational message is issued and *max* is set to 32M. The maximum limit can be changed dynamically by invoking the

configfhs shell command. See *OS/390 UNIX System Services Command Reference* for more information about the **configfhs** shell command.

Note: HFS may temporarily exceed the limit set in *max* to avoid failure of a file read or write request, but the amount of buffers used is reduced to the *max* specification or less as soon as possible.

If you do not specifically set a value for VIRTUAL(max), the system assigns to *max* a default value which is equal to half the amount of real storage available to the system at HFS initialization. (**Note:** The sample BPXPRMxx parmlib member provided in SYS1.SAMPLIB uses this default.) It is recommended that you consider how this storage change will affect your current system storage usage.

Also, starting in R7, OS/390 uses more buffers. IBM recommends that you monitor the paging of your system. If paging is increasing, you might need to set a lower value on the VIRTUAL parameter to relieve the situation.

Default: 50% of real storage available to the system at HFS initialization time.

FIXED(min)

min specifies the amount of virtual storage (in megabytes) that is fixed at HFS initialization time and remains fixed even if HFS activity drops to zero. *min* must be less than or equal to **VIRTUAL(max)**.

min cannot exceed 50% of real storage available to the system. If the allowed amount of storage is exceeded, an informational message is issued and *min* is set to 50% of real storage. The minimum limit can be changed dynamically by invoking the **configfhs** shell command. See *OS/390 UNIX System Services Command Reference* for more information about the **configfhs** shell command.

Default: 0

NOWRITEPROTECT

– This keyword overrides the WRITEPROTECT function introduced with OS/390 R7 (DFSMS 1.5). When **NOWRITEPROTECT** is specified, the file system is not protected from being read/write mounted by multiple systems simultaneously. Read/write mounting by multiple systems corrupts the file system.

  Extreme care should be taken when specifying this keyword. It should only be used when there is no possibility of the file system being mounted by multiple systems.

  Use of the **NOWRITEPROTECT** keyword avoids an additional file system read operation that is required at Sync time to support the WRITEPROTECT function.

– **Default:** WRITEPROTECT

ASNAME(proc_name)

Specifies the name of a procedure in SYS1.PROCLIB that is to be used to start the address space that is initialized by the physical file system (PFS). Specify ASNAME for any PFS that does not run in the kernel address space. The name you specify is also used for the name of the address space.

ASNAME is an optional parameter. The name is 1 to 8 characters; the system converts the name to uppercase. If you do not specify ASNAME, or specify in ASNAME the name of the kernel address space, the PFS is initialized in the kernel address space.

If the physical file system specified does not expect an ASNAME operand, it ignores all ASNAME operands. Refer to the documentation for the specific physical file system for valid ASNAME operands.

SYSPLEX(YES|NO)

For OS/390 UNIX System Services, the SYSPLEX statement specifies whether a system should join the SYSBPX XCF group to share HFS resources across the sysplex. If SYSPLEX(YES) is specified, the system participates in shared HFS. If SYSPLEX(NO) is specified, the system *does not* participate in shared HFS. If the SYSPLEX statement is not provided, the default is SYSPLEX(NO). Also, to participate in shared HFS, the systems must be at R9 level or later. For more information on shared HFS, see “Shared HFS in a Sysplex“ in *OS/390 UNIX System Services Planning*.

Note: You cannot adjust the SYSPLEX field dynamically. There is no SETOMVS, SET OMVS, or SETOMVS RESET=(xx) capability. To change the value of SYSPLEX, you must re-IPL the system.

Default: NO

VERSION('nnnn')

The VERSION statement applies only to systems that are exploiting shared HFS. VERSION allows multiple releases and service levels of the binaries to coexist and participate in shared HFS. A directory with the value *nnnn* specified on VERSION is dynamically created at system initialization under the sysplex root and is used as a mount point for the version HFS. This directory, however, is only dynamically created if the sysplex root HFS is mounted read/write.

Note: *nnnn* is a case-sensitive character string no greater than 8 characters in length. It indicates a specific instance of the version HFS. The most appropriate values for *nnnn* are the name of the target zone, &SYSR1, or another qualifier meaningful to the system programmer. For example, if the system is at V2R9, you can specify REL9 for VERSION.

When SYSPLEX(YES) is specified, you must also specify the VERSION parameter.

The VERSION value is substituted in the content of symbolic links that contain $VERSION. For scenarios describing the use of the version HFS, see “Shared HFS in a Sysplex“ in *OS/390 UNIX System Services Planning*.

When testing or changing to a new Maintenance Level (PTF), the VERSION value can be changed dynamically by using the SETOMVS command:

```
SETOMVS VERSION='string'
```

You can also change the settings of this parameter via SET OMVS=(xx) and SETOMVS RESET=(xx) parmlib specifications.

Note: We do not recommend changing VERSION dynamically if you have any users logged on or running applications; replacing the system files for these users may be disruptive.

**ROOT FILESYSTEM('fsname') DDNAME(ddname) TYPE(type_name)
MODE(access) PARM('parameter') SETUID|NOSETUID**

AUTOMOVE|NOAUTOMOVESYSNAME(sysname)

Specifies a file system that OS/390 UNIX is to logically mount as the root file system.

Note: The ROOT statement is optional. If not specified, a TFS file system is mounted as the root.

To change the value of the ROOT statement without having to re-IPL, you can use the TSO/E MOUNT and UNMOUNT commands.

The root file system can be unmounted using the TSO/E UNMOUNT command or ISHELL. Ensure that you specify the IMMEDIATE option.

The parameters are:

FILESYSTEM('fsname')
The name of the root file system. The name must be unique in the system.

Either FILESYSTEM or DDNAME is required; do not specify both. The name is 1 to 44 characters; the characters can be in uppercase, lowercase, or both. The name must be enclosed in single quotes. An HFS dataset name must conform to the rules of MVS dataset names.

DDNAME(ddname)
The ddname on the JCL DD statement that defines the root file system. To use the DDNAME parameter, a DD statement for the HFS dataset containing the root file system should be placed in the OS/390 UNIX cataloged procedure.

Either FILESYSTEM or DDNAME is required; do not specify both. The ddname is 1 to 8 characters; the system converts the ddname to uppercase.

TYPE(type_name)
Specifies the name of a file system type identified in a FILESYSTYPE statement. The TYPE(type_name) parameter must be the same as the TYPE(type_name) parameter on a FILESYSTYPE statement.

TYPE is a required parameter. The name is 1 to 8 characters; the system converts the name to uppercase.

MODE(access)
Specifies access to the root file system by all users:
- READ: Users can only read the root file system.
- RDWR: Users can read and write in the root file system.

Default: RDWR

PARM('parameter')
Provides a parameter to be passed directly to the file system type. The parameter format and content are specified by the file system type.

PARM is an optional parameter. The parameter is up to 1024 characters long; the characters can be in uppercase, lowercase, or both. The parameter must be enclosed in single quotes.

If the physical file system specified does not expect a PARM operand, it ignores all PARM operands. Refer to the documentation for the specific physical file system for valid entry point names.

SYNC(t), VIRTUAL(max), FIXED(min) and **NOWRITEPROTECT** are valid only when **ENTRYPOINT** is GFUAINIT.

Note: If a syntax error is found in any of these parameters (**SYNC, VIRTUAL, FIXED,** or **NOWRITEPROTECT**), an error message is issued and all four parameters are set to the default values.

SYNC(t)

t specifies the number of seconds used as a default for the sync daemon interval. When the sync daemon is active, the meta data for a file system is hardened. Setting *t* to 0 indicates that the file system should harden meta data synchronously with syscall requests.

Sync interval values are rounded up to the next 30-second value. For example, specifying 31 seconds results in a sync interval of 60 seconds.

The maximum value that can be specified for *t* is 65535. Values between 65535 and 99999 are rejected.

A value of 99999 specifies that no sync daemon intervals are specified, and thus, the meta data is not hardened.

Default: 60 seconds

VIRTUAL(max)

max specifies the maximum amount of virtual storage (in megabytes) that HFS data and meta data buffers should use. The minimum value that can be specified is 32M. If less than 32M is specified, an informational message is issued and *max* is set to 32M. The maximum limit can be changed dynamically by invoking the **confighfs** shell command. See *OS/390 UNIX System Services Command Reference* for more information about the **confighfs** shell command.

Note: HFS may temporarily exceed the limit set in *max* to avoid failure of a file read or write request, but the amount of buffers used is reduced to the *max* specification or less as soon as possible.

If you do not specifically set a value for VIRTUAL(max), the system assigns to *max* a default value which is equal to half the amount of real storage available to the system at HFS initialization. (**Note:** The sample BPXPRMxx parmlib member provided in SYS1.SAMPLIB uses this default.) It is recommended that you consider how this storage change will affect your current system storage usage.

Also, starting in R7, OS/390 uses more buffers. IBM recommends that you monitor the paging of your system. If paging is increasing, you might need to set a lower value on the VIRTUAL parameter to relieve the situation.

Default: 50% of real storage available to the system at HFS initialization time.

FIXED(min)

min specifies the amount of virtual storage (in megabytes) that is fixed at HFS initialization time and remains fixed even if HFS activity drops to zero. *min* must be less than or equal to **VIRTUAL(max)**.

min cannot exceed 50% of real storage available to the system. If the allowed amount of storage is exceeded, an informational message is issued and *min* is set to 50% of real storage. The minimum limit can be changed dynamically by invoking the **confighfs** shell command. See *OS/390 UNIX System Services Command Reference* for more information about the **confighfs** shell command.

Default: 0

NOWRITEPROTECT

– This keyword overrides the WRITEPROTECT function. When **NOWRITEPROTECT** is specified, the file system is not protected from being read/write mounted by multiple systems simultaneously. Read/write mounting by multiple systems corrupts the file system.

Extreme care should be taken when specifying this keyword. It should only be used when there is no possibility of the file system being mounted by multiple systems.

Use of the **NOWRITEPROTECT** keyword avoids an additional file system read operation that is required at Sync time to support the WRITEPROTECT function.

– **Default:** WRITEPROTECT

SETUID|NOSETUID

SETUID specifies that the **setuid()** and **setgid()** mode bit on an executable file will be supported.

NOSETUID specifies that the **setuid()** and **setgid()** mode bit on an executable file will **not** be supported. The UID or GID will **not** be changed when the program is executed and the APF and Program Control extended attributes are **not** honored. The entire HFS is uncontrolled.

Default: SETUID

AUTOMOVE|NOAUTOMOVE

For a description, see AUTOMOVE|NOAUTOMOVE on the MOUNT statement. In OS/390 R9 and later, to ensure that the root is always available, use the default.

Default: AUTOMOVE

SYSNAME(sysname)

For a description, see SYSNAME on the MOUNT statement. In OS/390 R9 and later, to ensure that the root is always available, use the default.

Default: The name of the system the command is processed on.

MOUNT FILESYSTEM('fsname') DDNAME(ddname) TYPE(type_name) MOUNTPOINT('pathname') MODE(access) PARM('parameter')

SETUID|NOSETUID WAIT|NOWAIT SECURITY|NOSECURITY

AUTOMOVE|NOAUTOMOVESYSNAME(sysname)

Specifies a file system that OS/390 UNIX is to logically mount onto the root file system or another file system.

Mount statements are processed in the sequence in which they appear. If they are cascading, the system will mount the first file system first. Make sure that a

mount point exists before the file system is mounted. If you mount a file system over an existing directory containing files, you will cover up the existing files.

If a MOUNT statement uses a DDNAME parameter to identify the HFS data set, allocate that HFS data set in the OMVS cataloged procedure. See “Customizing the OMVS Cataloged Procedure to Run the Kernel Initialization Program“ in *OS/390 UNIX System Services Planning*.

The MOUNT statement is optional; the BPXPRMxx member can contain one or more MOUNT statements.

The MOUNT parameters are:

FILESYSTEM('fsname')
The name of the file system. The name must be unique in the system.

Either FILESYSTEM or DDNAME is required; do not specify both. The name is 1 to 44 characters; the characters can be in uppercase, lowercase, or both. The name must be enclosed in single quotes. An HFS dataset name must conform to the rules of MVS dataset names.

DDNAME(ddname)
The ddname on the JCL DD statement that defines the file system. To use the DDNAME parameter, a DD statement for the HFS dataset containing the mountable file system should be placed in the OMVS cataloged procedure.

Either FILESYSTEM or DDNAME is required; do not specify both. The name is 1 to 8 characters; the system converts the ddname to uppercase.

TYPE(type_name)
Specifies the name of a file system type identified in a FILESYSTYPE statement. The TYPE(type_name) parameter must be the same as the TYPE(type_name) parameter on a FILESYSTYPE statement.

TYPE is a required parameter. The name is 1 to 8 characters; the system converts the name to uppercase.

MOUNTPOINT('pathname')
Specifies the pathname of the directory onto which the file system is to be mounted.

Mount point restrictions are:

- The mount point must be a directory.
- Any files in the directory are not accessible while the file system is mounted.
- Only one mount can be active at any time for a mount point.
- A file system can be mounted at only one directory at any time.

MOUNTPOINT is required. The pathname is up to 1023 characters long; the characters can be in uppercase, lowercase, or both. The pathname must be enclosed in single quotes.

MODE(access)
Specifies access to the mounted file system by all users:
- READ: Users can only read the file system being mounted.
- RDWR: Users can read and write in the file system being mounted.

Default: RDWR

PARM('parameter')

Provides a parameter to be passed directly to the file system type. The parameter format and content are specified by the file system type.

PARM is an optional parameter. The parameter is up to 1024 characters long; the characters can be in uppercase, lowercase, or both. The parameter must be enclosed in single quotes.

If the physical file system specified does not expect a PARM operand, it ignores all PARM operands. Refer to the documentation for the specific physical file system for valid entry point names.

SYNC(t), VIRTUAL(max), FIXED(min) and **NOWRITEPROTECT** are valid only when **ENTRYPOINT** is GFUAINIT.

Note: If a syntax error is found in any of these parameters (**SYNC, VIRTUAL, FIXED,** or **NOWRITEPROTECT**), an error message is issued and all four parameters are set to the default values.

SYNC(t)

t specifies the number of seconds used as a default for the sync daemon interval. When the sync daemon is active, the meta data for a file system is hardened. Setting *t* to 0 indicates that the file system should harden meta data synchronously with syscall requests.

Sync interval values are rounded up to the next 30-second value. For example, specifying 31 seconds results in a sync interval of 60 seconds.

The maximum value that can be specified for *t* is 65535. Values between 65535 and 99999 are rejected.

A value of 99999 specifies that no sync daemon intervals are specified, and thus, the meta data is not hardened.

Default: 60 seconds

VIRTUAL(max)

max specifies the maximum amount of virtual storage (in megabytes) that HFS data and meta data buffers should use. The minimum value that can be specified is 32M. If less than 32M is specified, an informational message is issued and *max* is set to 32M. The maximum limit can be changed dynamically by invoking the **confighfs** shell command. See *OS/390 UNIX System Services Command Reference* for more information about the **confighfs** shell command.

Note: HFS may temporarily exceed the limit set in *max* to avoid failure of a file read or write request, but the amount of buffers used is reduced to the *max* specification or less as soon as possible.

If you do not specifically set a value for VIRTUAL(max), the system assigns to *max* a default value which is equal to half the amount of real storage available to the system at HFS initialization. (**Note:** The sample BPXPRMxx parmlib member provided in SYS1.SAMPLIB uses this default.) It is recommended that you consider how this storage change will affect your current system storage usage.

Also, starting in R7, OS/390 uses more buffers. IBM recommends that you monitor the paging of your system. If paging is increasing, you might need to set a lower value on the VIRTUAL parameter to relieve the situation.

Default: 50% of real storage available to the system at HFS initialization time.

FIXED(min)

min specifies the amount of virtual storage (in megabytes) that is fixed at HFS initialization time and remains fixed even if HFS activity drops to zero. *min* must be less than or equal to **VIRTUAL(max)**.

min cannot exceed 50% of real storage available to the system. If the allowed amount of storage is exceeded, an informational message is issued and *min* is set to 50% of real storage. The minimum limit can be changed dynamically by invoking the **confighfs** shell command. See *OS/390 UNIX System Services Command Reference* for more information about the **confighfs** shell command.

Default: 0

NOWRITEPROTECT

– This keyword overrides the WRITEPROTECT function. When **NOWRITEPROTECT** is specified, the file system is not protected from being read/write mounted by multiple systems simultaneously. Read/write mounting by multiple systems corrupts the file system.

  Extreme care should be taken when specifying this keyword. It should only be used when there is no possibility of the file system being mounted by multiple systems.

  Use of the **NOWRITEPROTECT** keyword avoids an additional file system read operation that is required at Sync time to support the WRITEPROTECT function.

– **Default:** WRITEPROTECT

SETUID|NOSETUID

SETUID specifies that the **setuid()** and **setgid()** mode bit on an executable file will be supported.

NOSETUID specifies that the **setuid()** and **setgid()** mode bit on an executable file will **not** be supported. The UID or GID will **not** be changed when the program is executed and the APF and Program Control extended attributes are **not** honored. The entire HFS is uncontrolled.

Default: SETUID

WAIT|NOWAIT

WAIT specifies that processing should not continue during an asynchronous mount.

NOWAIT specifies that processing should continue during an asynchronous mount.

Default: WAIT

SECURITY|NOSECURITY

SECURITY specifies that security checks should be performed.

NOSECURITY specifies that security checks should not be performed.

Default: SECURITY

AUTOMOVE|NOAUTOMOVE

The AUTOMOVE|NOAUTOMOVE parameters apply only in a sysplex where systems are participating in shared HFS. The AUTOMOVE and NOAUTOMOVE parameters indicate what happens if the system that owns a file system goes down. AUTOMOVE indicates that ownership of the file system automatically changes to another system participating in shared HFS. NOAUTOMOVE indicates that ownership of the file system is not moved if the owning system goes down; as a result, the file system becomes inaccessible.

For file systems that are mostly used by DFS clients, consider specifying NOAUTOMOVE on the MOUNT statement. By doing so, the file systems will not change ownership if the system is suddenly recycled, and they will be available for automatic re-export by DFS. This is recommended because a file system can only be exported by the DFS server at the system that owns the file system. Once a file system has been exported by DFS, it cannot be moved until it has been unexported from DFS. When recovering from system outages, you need to weigh sysplex availability against availability to the DFS clients. When an owning system recycles and a DFS-exported file system has been taken over by one of the other systems, DFS cannot automatically re-export that file system. The file system will have to be moved from its current owner back to the original DFS system— the one that has just been recycled— and then exported again.

Default: AUTOMOVE

SYSNAME(sysname)

For systems participating in shared HFS, SYSNAME specifies the particular system on which a mount should be performed. This system will then become the owner of the file system mounted. This system must be IPLed with SYSPLEX(YES).

Default: The name of the system, if IPLed with SYSPLEX(YES), that the mount is processed on.

Note: In OS/390 R9 and later, to ensure that the root is always available, use the defaults for SYSNAME and AUTOMOVE.

For additional information, see **MOUNT** in *OS/390 UNIX System Services Planning*.

NETWORK DOMAINNAME(sockets_domain_name) DOMAINNUMBER(sockets_domain_number) MAXSOCKETS(number) TYPE(type_name) INADDRANYPORT(starting_port_number)

INADDRANYCOUNT(number_of_ports_to_reserve)

Specifies that a socket physical file system domain should be readied for use. The UDS matches the TYPE on the previous FILESYSTYPE statement.

Use the SETOMVS RESET command to dynamically change the NETWORK values. To make a permanent change, edit the BPXPRMxx member used for IPLs. For more information, see “Dynamically Adding FILESYSTYPE Statements in BPXPRMxx“ in *OS/390 UNIX System Services Planning*.

Provide a NETWORK statement for each socket file system domain to be initialized.

- For AF_UNIX file systems, always include a FILESYSTYPE statement specifying ENTRYPOINT(BPXTUINT) and a NETWORK statement with a matching TYPE, usually TYPE(UDS), on both.
- For TCP/IP sockets, always include a FILESYSTYPE statement specifying ENTRYPOINT(EZBPFINI) and a NETWORK statement with a matching TYPE, usually TYPE(INET), on both.
- For CINET sockets, include a FILESYSTYPE statement with ENTRYPOINT (BPXCTCINT) and a NETWORK statement with a matching TYPE, usually TYPE(CNET), that specifies INADDRANYPORT and INADDRANYCOUNT. See “Specifying INADDRANYPORT and INADDRANYCOUNT“ in *OS/390 UNIX System Services Planning* for more information.

DOMAINNAME(sockets_domain_name)
The 1 to 16 character name by which this socket file system domain is to be known.

DOMAINNUMBER(sockets_domain_number)
A number that matches the value defined for this domain name. The currently supported values for this field are:
1 AF_UNIX
2 AF_INET

The following table shows some supported domain names, domain numbers, and their associated entry point names. See the documentation for the physical file system you are using to get the correct entry point name.

Table 5. Supported Domains

Domain name	Domain number	Entry point
AF_UNIX	1	BPXTUINT
AF_INET	2	EZBPFINI, BPXTCINT, BPXTLINT

MAXSOCKETS(nnnnn)
Specifies the maximum number of sockets supported by this file system. You can specify a value from 0 to 64498. This is an optional parameter. The maximum value that this field can have is defined by each domain. If a value larger than the maximum is specified, an informational message is issued and the value used is the maximum. If this parameter is omitted, a default value of 100 is used.

Note: Ensure that this number is large enough for socket connections for all applications using your OS/390 UNIX environment. This upper limit is set when the NETWORK statement is processed during IPL. It can only be changed if the NETWORK statement is changed using the SETOMVS RESET command, and the physical filesystem named in the FILESYSTYPE statement associated with that NETWORK statement (such as TCP/IP) can be stopped and restarted.

TYPE(type_name)
Specifies the name of a file system type identified in a FILESYSTYPE statement. The TYPE(type_name) must be the same as the TYPE(type_name) parameter on a FILESYSTYPE statement.

TYPE is a required parameter. The name is 1 to 8 characters; the system converts the name to uppercase.

INADDRANYPORT(starting_port_number)
Specifies the starting port number for the range of port numbers that the system reserves for use with PORT 0, INADDR_ANY binds. This value is only needed for CINET.

Value Range: *starting_port_number* is a decimal value from 1024 to 65535. Ports 1 — 1023 are well-known ports that cannot be reserved for use with PORT 0, INADDR_ANY binds.

Default: If neither INADDRANYPORT or INADDRANYCOUNT is specified, the default for INADDRANYPORT is 63000. Otherwise, no ports are reserved (0).

Note: If you do not want to support INADDRANY with CINET, you should specify INADDRANYPORT(*xx*), where *xx* is a valid value, without specifying INADDRANYCOUNT.

INADDRANYCOUNT(number_of_ports_to_reserve)
Specifies the number of ports that the system reserves, starting with the port number specified in the INADDRANYPORT parameter. This value is only needed for CINET.

Value Range: *number_of_ports_to_reserve* is a decimal value from 1 to 4000.

Default: If neither INADDRANYPORT or INADDRANYCOUNT is specified, the default for INADDRANYCOUNT is 1000. Otherwise, no ports are reserved (0).

SUBFILESYSTYPE NAME(transport_name) TYPE(type_name)

ENTRYPOINT(entry_name) PARM('parameter') DEFAULT
Specifies an AF_INET physical file system that is to run underneath the INET socket file system. TCPIP and TCPIP2 are the names that TCP/IP uses to identify itself during its initialization, and CINET matches the TYPE operand on the previous FILESYSTYPE and NETWORK statements. In the case of TCP/IP, the NAME() value is the procname. The system attaches the EZBPFINI load module during initialization, and this file system should be used as the default INET physical file system.

The SUBFILESYSTYPE statement is associated with its corresponding FILESYSTYPE and NETWORK statements by matching the value specified in the TYPE operand.

The value specified on all of the TYPE operands must match, but can be any 1- to 8-character value. The value specified on the NAME parameter on the SUBFILESYSTYPE statement is the name that will be used by the physical file system when it is initialized.

For SecureWay Communications Server, the SUBFILESYSTYPE statement must match the TCPIPJOBNAME of that stack. See “Customizing the File System Statements on the BPXPRMxx Member“ in *OS/390 UNIX System Services Planning* for more details.

New SUBFILESYSTYPE statements can be added dynamically. However, you cannot dynamically change (or delete) a value. For more information, see "Dynamically Adding FILESYSTYPE Statements in BPXPRMxx" in *OS/390 UNIX System Services Planning*.

The parameters are:

NAME(transport_name)
Specifies the name that identifies this file system to the CINET physical file system.

NAME is a required parameter. The name is 1 to 8 characters; the system converts the name to uppercase. The value specified by the NAME parameter on the SUBFILESYSTYPE statement is the name that the physical file system uses to identify itself when it is initialized. For example, for TCP/IP, this is the starting procedure name.

TYPE(type_name)
Specifies the name of the CINET file system type identified in a FILESYSTYPE statement. The TYPE(type_name) parameter must be the same name that was used for the TYPE(type_name) parameter on the FILESYSTYPE statement for the CINET physical file system.

TYPE is a required parameter. The name is 1 to 8 characters; the system converts the name to uppercase.

ENTRYPOINT(entry_name)
Specifies the name of the load module containing the entry point into the file system type.

ENTRYPOINT is a required parameter. The name is 1 to 8 characters; the system converts the name to uppercase.

PARM('parameter')
Provides a parameter to be passed to the transport driver. The parameter format and content are specified by the file system receiving the data.

PARM is an optional parameter. The parameter is up to 1024 characters long; the characters can be in uppercase, lowercase, or both. If the characters are not all in uppercase, the parameter must be enclosed in single quotes.

If the physical file system specified does not expect a PARM operand, it ignores all PARM operands. Refer to the documentation for the specific physical file system for valid entry point names.

DEFAULT
Identifies this file system as the default CINET file system.

DEFAULT is an optional parameter. If it is not specified, the file system specified in the first SUBFILESYSTYPE statement found in the parmlib member is designated as the default. See "Setting Up for CINET AF_INET Sockets" in *OS/390 UNIX System Services Planning* for more information about the use of the DEFAULT parameter.

For additional information, see **SUBFILESYSTYPE** in *OS/390 UNIX System Services Planning*.

STARTUP_PROC

This statement specifies a 1-to-8-character name of a started JCL procedure that initializes the kernel. The name specified in this statement must exist on the system before IPL or errors will occur.

Using a started procedure other than OMVS is **strongly discouraged**. If you want to change the value of STARTUP_PROC, you will have to edit the BPXPRMxx member and then re-IPL. You cannot use the SET OMVS or SETOMVS command to change the value.

If you decide to use a started procedure other than OMVS:

- The replacement started procedure must also be a single jobstep procedure that invokes the BPXINIT program (EXEC PGM=BPXINIT). If it invokes any other program, the OMVS initialization will fail.
- Change the procedure name in the RACF started procedures table or the definitions in the STARTED Class. See “Preparing the RACF Security Program“ in *OS/390 UNIX System Services Planning*.

Note: Renaming OMVS to some other value may affect the setup of other products such as TCP/IP.

Default: STARTUP_PROC(OMVS).

STARTUP_EXEC

STARTUP_EXEC names a REXX exec that does application environment initialization for OS/390 UNIX. This statement is optional; if it is specified, the BPXOINIT process will not run **/etc/init**. The startup exec is typically used by an installation that does not have an HFS, but is using a TFS for a file system. It can be used to populate the TFS with any directories and files that are needed. It is specified as:

```
STARTUP_EXEC('Dsname(Memname)',SysoutClass)
```

where:

- Dsname is a 1-to-44-character valid dataset name.
- Memname is a 1-to-8-character valid REXX exec member.
- SysoutClass is 1 character and is alphanumeric and specifies the sysout class that the REXX exec will run under. Specifying SysoutClass is optional.

If you want to change the value of STARTUP_EXEC, you will have to edit the BPXPRMxx member and then reIPL. You cannot use the SET OMVS or SETOMVS command to change the value.

Default: There is no default value for STARTUP_EXEC.

RUNOPTS('string')

Specifies the _CEE_RUNOPTS environment variable used when OS/390 UNIX initialization invokes **/etc/init** or **/usr/sbin/init**. This string provides runtime options to Language Environment programs in environments where these options are not available from other sources. OS/390 UNIX passes the _CEE_RUNOPTS value and all programs invoked from **/etc/rc** to the shell.

If you want to change the value of RUNOPTS, you will have to edit the BPXPRMxx member and then re-IPL. You cannot use the SET OMVS or SETOMVS command to change the value. After the value is specified in BPXPRMxx, you can use one of the following methods to change this string:

- The system is re-IPLed with a new BPXPRMxx RUNOPTS string.

- The user or installation sets _CEE_RUNOPTS in **/etc/rc** or **/etc/init.config**.
- A program or shell script sets _CEE_RUNOPTS.

If you do not specify a value for RUNOPTS, the RUNOPTS string or _CEE_RUNOPTS environment variable is not provided.

The TSO/E OMVS command uses the specified options as the Language Environment run-time options, by default.

The setting of RUNOPTS has no effect on BPXBATCH jobs.

If RTLS will be used to access the Language Environment run-time library, RUNOPTS should specify the RTLS(ON), LIBRARY, and, optionally, VERSION run-time options. Use the RUNOPTS parameter only when using RTLS. Before using RTLS, you must set up FACILITY profiles as documented in the **CSVRTLxx** description.

Specifying the RUNOPTS parameter causes the kernel to set the **_CEE_RUNOPTS** environment variable when starting **/etc/init**, or when the TSO/E OMVS command is entered. This environment variable is normally propagated to subsequent processes (such as **/etc/init** to **/bin/sh** to **/etc/rc** to **/bin/inetd** to **/bin/rlogind** to **/bin/sh** for shell users).

To do this, you must make sure that any other steps in the flow (such as export statements in **/etc/rc**) do not overwrite the value of **_CEE_RUNOPTS**. If additional run-time options are needed, they should be concatenated to the old value of **_CEE_RUNOPTS**.

Value Range: From 1 to 250 characters.

Default: No RUNOPTS string or _CEE_RUNOPTS environment variable is provided.

Restrictions:

- The string must be enclosed in parentheses and quotes (").
- An empty string (' ') is not valid.
- Although all characters are allowed, nulls, slashes (/), unbalanced SO/SI, and unbalanced parentheses and quotes cause unpredictable problems in areas such as the TSO/E OMVS command.

For more information on specifying RUNOPTS strings, see “Customizing the BPXPRMxx Parmlib Member“ in *OS/390 UNIX System Services Planning*.

SYSCALL_COUNTS(YES/NO)

Specifies that syscall counts are to be accumulated in internal kernel data areas so that the RMF data gatherer can record the information.

If you specify YES, the path length for the most frequently used OS/390 UNIX system calls is increased by more than 150 instructions.

Default: NO

Use the SETOMVS or SET OMVS command to dynamically change the value of SYSCALL_COUNT. To make a permenent change, edit the BPXPRMxx member used for IPLs.

MAXQUEUEDSIGS(nnnnnn)

Specifies the maximum number of signals that OS/390 UNIX allows to be concurrently queued within a single process.

Value Range: *nnnnnn* is a decimal value from 1 to 100000.

Default: 1000

You can change the value of MAXQUEUEDSIGS dynamically using the SETOMVS or SET OMVS command. To make a permanent change, edit the BPXPRMxx member that will be used for future IPLs.

LIMMSG(NONE|SYSTEM|ALL)

Specifies how console messages that indicate when parmlib limits are reaching critical levels are to be displayed:

NONE No console messages are to be displayed when any of the parmlib limits have been reached.

SYSTEM

Console messages are to be displayed for all processes that reach system limits. In addition, messages are to be displayed for each process limit of a process if:

- The process limit or limits are defined in the OMVS segment of the owning User ID
- The process limit or limits have been changed with a SETOMVS PID=*pid*,*process_limit*

ALL Console messages are to be displayed for the system limits and for the process limits, regardless of which process reaches a process limit.

Default: NONE

Chapter 5. APAR OW43776: OS/390 MVS System Messages

BPX Messages

BPXI038I TASK*procname* HAS ABNORMALLY ENDED. *text*

Explanation: The OS/390 UNIX task abnormally ended and cannot be recovered. The end of task exit routine (ETXR) failed to reattach it after a preset number of attempts.

text is one of the following:

MEMORY MAP PROCESSING IS SUSPENDED UNTIL

THE NEXT IPL
OS/390 UNIX memory map processing is being suspended until the next IPL.

MODIFY BPXOINIT PROCESSING IS SUSPENDED
OS/390 UNIX MODIFY BPXOINIT console commands are being suspended until the next IPL.

NETWORK DISPATCHER WORKLOAD BALANCING

IS SUSPENDED
The OS/390 UNIX Network Dispatcher workload balancing function is being suspended until the next IPL.

In the message text:

procname
The name of the OS/390 UNIX task that abnormally ended

Source: OS/390 UNIX System Services kernel (BPX)

Module: BPXQETXR

System Action: The system continues.

Operator Response: None.

System Programmer Response: The identified OS/390 UNIX task has ended. The function is unavailable until the next IPL. The system should have presented other information that identifies the cause of the task failure.

BPXI039E SYSTEM LIMIT *limitname* HAS REACHED *nn*% OF ITS CURRENT CAPACITY OF *currentlimit*

Explanation: The specified OS/390 UNIX system limit has reached a critical level.

In the message text:

limitname
The name of the OS/390 UNIX system limit

nn The percentage of the system limit that has been reached

currentlimit
The current setting for the named system limit

Source: OS/390 UNIX System Services kernel (BPX)

Module: BPXMSLIM

System Action: No action is taken.

Operator Response: If the condition persists, contact the system programmer.

System Programmer Response: If it is determined that the limit is too restrictive, raise the specified limit using the **SETOMVS** command.

BPXI040I PROCESS LIMIT *limitname* HAS REACHED *nn*% OF ITS CURRENT CAPACITY OF *currentlimit* FOR PID=*pid* IN JOB *jobname* RUNNING IN ADDRESS SPACE *asid*

Explanation: The specified OS/390 UNIX process limit has reached a critical level.

In the message text:

limitname
The name of the OS/390 UNIX process limit

nn The percentage of the process limit that has been reached

currentlimit
The current setting for the named process limit

pid The process ID of the process that encountered the limit

jobname
The Jobname of the address space that encountered the limit

asid
The address space ID of the address space that encountered the limit

Source: OS/390 UNIX System Services kernel (BPX)

Module: BPXMSLIM

System Action: No action is taken.

Operator Response: If the condition persists, contact the system programmer.

System Programmer Response: If it is determined that the specified limit is too restrictive, raise it using the **SETOMVS** command.

BPXI041I RESOURCE SHORTAGE FOR *limitname*HAS BEEN RELIEVED

Explanation: The resource shortage for limit *limitname* has been relieved.

In the message text:

limitname
The name of the OS/390 UNIX system limit

Source: OS/390 UNIX System Services kernel (BPX)

Module: BPXSLIM

System Action: No action is taken.

BPXI042I RESOURCE SHORTAGE FOR *limitname* FOR PID=*pid* HAS BEEN RELIEVED

Explanation: The resource shortage for limit *limitname* for process *pid* has been relieved.

In the message text:

limitname
The name of the OS/390 UNIX process limit

pid The process ID of the process that encountered the limit

Source: OS/390 UNIX System Services kernel (BPX)

Module: BPXMSLIM

System Action: No action is taken.

BPXO017I SETOMVS ERROR. LOWERING *limitname* IS CURRENTLY NOT ALLOWED. A WARNING MESSAGE FOR THIS LIMIT IS OUTSTANDING.

Explanation: The system does not allow you to lower a limit, *limitname*, for which there is an outstanding warning message. For a description of the limit, refer to the BPXPRMXX sample parmlib member.

limitname is one of the following

MAXPROCSYS
MAXUIDS
MAXPTYS
MAXMMAPAREA
MAXSHAREPAGES
IPCSMSGNIDS
IPCSEMNIDS
IPCSHMNIDS
IPCSHMSPAGES
SHRLIBRGNSIZE
SHRLIBMAXPAGES
IPCMSGQBYTES
IPCMSGQMNUM
IPCSHMMPAGES
INET MAXSOCKETS
UNIX MAXSOCKETS
MAXFILEPROC
MAXPROCUSER
MAXQUEUEDSIG
MAXTHREADS
MAXTHREADTASKS
IPCSHMNSEGS

:

Source: OS/390 UNIX System Services kernel (BPX)

Module: BPXOTASK, BPXMIMST

System Action: The system does not change the limit value.

Operator Response: None.

System Programmer Response: To solve the displayed problem, increase the limit value for the specified resource.

BPXO029I LIMMSG CHANGED FROM *oldvalue* TO *newvalue*

Explanation: The system-wide value for LIMMSG has been changed. Warning messages will now be issued using the new value.

In the message text:

oldvalue
The old value for LIMMSG

newvalue
The new value for LIMMSG

Source: OS/390 UNIX System Services kernel (BPX)

Module: BPXMU1

System Action: The LIMMSG value has been changed successfully.

Operator Response: None.

System Programmer Response: None.

Chapter 6. APAR OW43776: OS/390 MVS Routing and Descriptor Codes

BPX Messages

Message Identifier	Routing Code	Descriptor Code
BPXB001E	1,10	3
BPXB002E	1	11
BPXB003I	2	4
BPXB004E	1	11
BPXB005I	2	4
BPXC001I	2	4
BPXF001I	2	4
BPXF002I	2	4
BPXF003I	2	4
BPXF004I	2,10	4
BPXF005I	2,10	4
BPXF006I	2	4
BPXF007I	2,10	4
BPXF008I	2,10	4
BPXF009I	2,10	4
BPXF010I	2,10	4
BPXF011I	2,10	4
BPXF012I	2,10	4
BPXF013I	2	4
BPXF014D	2	2
BPXF015I	*	5
BPXF016I	2	4
BPXF017I	2	4
BPXF018I	2	4
BPXF019I	2	4
BPXF020I	2	11
BPXF021I	2	4
BPXF022I	2	4
BPXF023I	2,10	4
BPXF024I	2	4
BPXF025I	2	4
BPXF026I	2	4
BPXF027I	2	4
BPXF028I	2	4
BPXF029E	2	11
BPXF030I	2,10	4
BPXF031I	2,10	4
BPXF032D	2	2
BPXF033I	2,10	4
BPXF101E	-	5
BPXF102E	2	5
BPXF103E	2	5
BPXF104E	2	5
BPXF105E	2	2
BPXF106E	2	2
BPXF107E	2	5
BPXF108E	2	5
BPXF110E	2	2

Message Identifier	Routing Code	Descriptor Code
BPXF111E	2	2
BPXF112W	2	2
BPXF113W	2	2
BPXF114E	-	5
BPXF115E	2	5
BPXF116E	-	5
BPXF117E	2	2
BPXF118W	2	2
BPXF119W	2	2
BPXF120E	2	5
BPXF121E	2	5
BPXF123E	2	2
BPXF124E	2	2
BPXF125E	2	2
BPXF126E	2	5
BPXF127E	2	5
BPXF128E	2	5
BPXF129E	2	5
BPXF130E	2	5
BPXF131E	2	5
BPXF132E	2	5
BPXF134E	2	2
BPXF135E	2	2
BPXF136E	2	5
BPXF137E	2	2
BPXF138E	2	2
BPXF139E	2	2
BPXF140E	2	2
BPXF141E	2	2
BPXF142E	2	2
BPXF143E	2	2
BPXF144I	-	-
BPXF145E	2	2
BPXF146E	2	2
BPXF147E	2	2
BPXF148E	2	2
BPXF150I	2	5
BPXF151I	2	5
BPXF152W	2	2
BPXF153W	2	2
BPXF154E	2	2
BPXF155E	2	2
BPXF156E	2	2
BPXF157E	2	2
BPXF158E	2	2
BPXF159E	2	5
BPXF160E	2	2
BPXF161I	2	2
BPXF162E	2	2
BPXF163E	2	2
BPXF164E	2	2
BPXF165E	2	2
BPXF166E	2	2
BPXF167E	2	2

Message Identifier	Routing Code	Descriptor Code
BPXF168E	2	2
BPXF169E	2	2
BPXF170E	2	2
BPXF171E	2	2
BPXF172E	2	2
BPXF173E	2	2
BPXF174E	2	2
BPXF175E	2	2
BPXF176E	2	2
BPXF201I	2,10	4
BPXF202I	2	4
BPXF203I	2	4
BPXF204I	2	4
BPXF205I	2	4
BPXF206I	2	4
BPXF207I	2	4
BPXF208I	2	4
BPXF209I	2	4
BPXF210I	2	4
BPXF211I	2,10	4
BPXF212I	2,10	4
BPXF213E	1, 2	3
BPXF214E	2	11
BPXF215E	2	11
BPXF216E	1, 2	3
BPXF217E	1, 2	3
BPXF218I	2	4
BPXI002I	2	4
BPXI003I	2	4
BPXI004I	2	4
BPXI005I	2	4
BPXI006I	-	4
BPXI007I	-	4
BPXI008I	-	4
BPXI009I	-	4
BPXI010I	-	4
BPXI011I	-	4
BPXI012I	2,10	4
BPXI013I	2,10	4
BPXI014I	2,10	4
BPXI015I	2	4
BPXI016I	2	4
BPXI017I	2	4
BPXI018I	2	4
BPXI019I	2	4
BPXI020I	2	4
BPXI021I	2	4
BPXI022I	-	4
BPXI023I	-	4
BPXI024I	-	4
BPXI025I	-	4
BPXI026I	2	4
BPXI027I	2	4
BPXI028E	1	11

Message Identifier	Routing Code	Descriptor Code
BPXI029I	1,2,10	12
BPXI030I	1,2,10	12
BPXI031E	1	1
BPXI032E	1,10	11
BPXI033E	1,10	11
BPXI034I	2	4
BPXI035E	1	11
BPXI038I	2	4
BPXI039E	1	11
BPXI040I	1	11
BPXI041I	2,10	4
BPXI042I	2,10	4
BPXM001I	11	6
BPXM002I	11	6
BPXM004I	11	6
BPXM006I	11	6
BPXM007I	11	6
BPXM008I	11	6
BPXM009I	11	6
BPXM010I	11	6
BPXM011I	11	6
BPXM012I	11	6
BPXM013I	11	6
BPXM014I	11	6
BPXM015I	11	6
BPXM016I	11	6
BPXM017I	11	6
BPXM018I	11	6
BPXM019I	11	6
BPXM020I	11	6
BPXM021E	2	5
BPXM022E	2	5
BPXM023I	2	4
BPXM024I	2	4
BPXM025I	2	4
BPXM026I	2	4
BPXM027I	2	4
BPXM028I	2	4
BPXM029I	2	4
BPXM030I	2	12
BPXM031I	2	12
BPXM032E	1,10	11
BPXM033I	2	12
BPXM036I	2	4
BPXM037I	2	4
BPXM038I	2	4
BPXM039I	2	4
BPXM040I	2	4
BPXM041I	2	4
BPXM042I	2	4
BPXM043I	2	4
BPXM047I	11	6
BPXN001I	2	4
BPXN002I	2	4

Message Identifier	Routing Code	Descriptor Code
BPXO001I	#	5,8,9
BPXO002I	#	5,8,9
BPXO003I	#	5,8,9
BPXO006I	2	5
BPXO007I	2	5
BPXO008I	2	5
BPXO009I	2	5
BPXO012I	2	5
BPXO015I	2	5
BPXO016I	2	5
BPXO017I	2,10	4
BPXO024I	2	5
BPXO025I	2	5
BPXO026I	2	5
BPXO027I	2	5
BPXO028I	2	5
BPXO029I	2	4
BPXO030I	2	5
BPXO031I	2,10	4
BPXO032I	2	5
BPXO033I	2,10	4
BPXO034I	2	5
BPXO035I	2,10	4
BPXO036I	2	5
BPXO037E	2	5
BPXO038I	2	5
BPXO039I	2, 10	4
BPXO040I	-	5,8,9
BPXO041I	-	5,8,9
BPXO042I	-	5,8,9
BPXO043I	-	5,8,9
BPXO044I	-	5,8,9
BPXO045I	-	5,8,9
BPXO046I	-	5,8,9
BPXO047I	-	5,8,9
BPXO048I	2	5
BPXP001I	2	4
BPXP003E	1,10	11
BPXP004E	1,10	11
BPXP005I	-	4
BPXP006E	1,10	11
BPXP007E	1,10	11
BPXP008E	1,10	11
BPXT001I	2,10	4
BPXU001I	2	4
BPXU002I	2	4
BPXU003I	2	4
BPXU004I	2	4
BPXU005I	2,10	4
BPXW0000I	2	2
BPXW0001I	2	2
BPXW0002I	2	2
BPXW0003I	2	2
BPXW0004I	2	2

Message Identifier	Routing Code	Descriptor Code

Part 3. Appendixes

Notices

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming Interface Information

This book is intended to help the customer plan for, customize, operate, manage, and maintain an OS/390 system with OS/390 UNIX System Services (OS/390 UNIX).

This book primarily documents intended Programming Interfaces that allow the customer to write programs that use OS/390 UNIX.

This book also documents information that is NOT intended to be used as Programming Interfaces of OS/390 UNIX. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

NOT Programming Interface information

End of NOT Programming Interface information

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AnyNet
CICS
CICS/ESA
DFSMS/MVS
DFSMSdfp
DFSMShsm
IBM
IMS
Language Environment
OS/390
RACF
RMF

VTAM

Lotus, Domino, and Lotus Go Webserver are trademarks of the Lotus Development Corporation.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others:

DFS

Transarc Corporation

Index

Special Characters

A

B

C

D

E

F

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Program Number: 5647-A01

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.