

## 8.2 The BPXBATCH Utility

---

. Copyright IBM Corp. 1996, 1998

BPXBATCH is an MVS utility that you can use to run shell commands or shell scripts and to run executable files through the MVS batch environment. You can invoke BPXBATCH:

- \* In JCL
- \* From the TSO/E READY prompt
- \* From TSO CLISTS and REXX execs
- \* From a program

BPXBATCH has logic in it to detect when it is run from JCL. If the BPXBATCH program is running as the only program on the job step task level, it sets up the stdin, stdout, and stderr and execs the requested program. If BPXBATCH is not running as the only program at the job step task level, the requested program will run as the second step of a JES batch address space from JCL in batch. If run from any other environment, the requested program will run in a WLM initiator in the OMVS subsys category.

This book provides examples of how you can use BPXBATCH. For more detailed information about BPXBATCH, see OS/390 UNIX System Services Command Reference.

### 8.2.1 Defining Standard Input, Output, and Error for BPXBATCH

---

. Copyright IBM Corp. 1996, 1998

OS/390 c/c++ programs require that stdin, stdout, and stderr be defined as either a file or a terminal. Many C functions use stdin, stdout, and stderr. For example, getchar() obtains a character from stdin, printf() directs the output to stdout, and perror() directs the output to stderr. See "Understanding Standard Input, Standard Output, and Standard Error" in topic 5.3 for more information.

For BPXBATCH, the default for stdin and stdout is /dev/null. The BPXBATCH default for stderr is the same file defined for stdout. For example, if you define stdout to be /tmp/output1 and do not define stderr, then both printf() and perror() output is directed to /tmp/output1.

For BPXBATCH, you can define stdin, stdout, and stderr using one of these:

- \* The TSO/E ALLOCATE command, using the ddnames STDIN, STDOUT, and STDERR.
- \* A JCL DD statement with the PATH operand, using the ddnames STDIN, STDOUT, and STDERR.
- \* Redirection. For example, even if stdout defaults to /dev/null, the command:

```
BPXBATCH SH ps -el >>/tmp/ps.out
```

entered in TSO/E redirects the output of the ps command to be appended to the file /tmp/ps.out.

### 8.2.2 Defining an Environment Variable File for BPXBATCH

---

. Copyright IBM Corp. 1996, 1998

When you are using BPXBATCH to run a program, you typically pass the program a file that sets the environment variables. If you do not pass an environment variable file when running a program with BPXBATCH or if the

HOME and LOGNAME variables are not set in the environment variable file, those two variables are set from your logon RACF profile. LOGNAME is set to the user name and HOME is set to the initial working directory from the RACF profile.

To pass environment variables to BPXBATCH, you define a file containing the variable definitions; it can be one of these:

- \* An HFS file identified with a STDENV statement
- \* An MVS data set identified with a STDENV statement

The default is /dev/null.

If you define an HFS file:

- \* m It must be a text file defined with read access only.
- \* Specify one variable per line, in the format variable=value. Environment variable names must begin in column 1.
- \* An environment variable file cannot have sequence numbers in it. If you use the ISPF editor to create the file, set the sequence numbers off by typing number off on the command line before you begin typing the data. If sequence numbers already exist, type UNNUM to remove them and set number mode off.

If you define an MVS data set:

- \* It must be a sequential data set, a partitioned data set (PDS) member, or a SYSIN data set. Record format can be fixed or variable (nonspanned).
- \* Specify one environment variable per record, in the format variable=value. Environment variable names must begin in column 1. Do not use terminating nulls.
- \* An environment variable file cannot have sequence numbers in it. If you use the ISPF editor to create the file, set the sequence numbers off by typing number off on the command line before you begin typing the data.

You can specify the environment variables as part of an in-stream JCL SYSIN data set--for example:

```
//STDENV DD *
variable1=aaaaaaa
variable2=bbbbbbbb
.
.
variable5=fffffff
/*
```

Note: Trailing blanks are truncated for SYSIN data sets, but not for other data sets.

For BPXBATCH, you can specify the environment variable file by using one of these:

- \* The TSO/E ALLOCATE command--for example:

```
ALLOCATE DDN(STDENV) DSN('TURBO.ENV.FILE') SHR
```

- \* A JCL STDENV DD statement. To identify an HFS file, use the PATH operand and the PATHOPTS ORDONLY. For example:

```
//STDENV DD PATH='u/turbo/env.file',PATHOPTS=ORDONLY
```

- \* JCL with a SYSIN data set for the environment variable definitions.
- \* SVC 99 dynamic allocation, if you are running BPXBATCH from a program.

### 8.2.2.1 Example: Setting Up Code Page Support in a STDENV file

-----  
 Copyright IBM Corp. 1996, 1998

To enable national language support for BPXBATCH, set the locale variables in the STDENV file. For example, you could put these lines in the file:

```
LANG=Da_DK.IBM-277
LC_ALL=Da_DK.IBM-277
export LANG LC_ALL
```

After you allocate this file to STDENV, you can test it by typing:

```
OSHELL echo $HOME
```

The pathname of your home directory should be displayed, instead of just \$HOME.

### 8.2.2.2 ! \_BPX\_BATCH\_SPAWN and \_BPX\_BATCH\_UMASK Environment Variables

-----  
 Copyright IBM Corp. 1996, 1998

! BPXBATCH uses two environment variables for execution that are specified by STDENV:

```
! *   _BPX_BATCH_UMASK=0755
! *   _BPX_BATCH_SPAWN=YES!NO
```

! \_BPX\_BATCH\_UMASK allows the user the flexibility of modifying the permission bits on newly created files instead of using the default mask (when PGM is specified).

! Note: This variable will be overridden by umask (usually set from within /etc/profile) if BPXBATCH is invoked with the 'SH' option (SH is the default). SH causes BPXBATCH to execute a login shell which runs the /etc/profile script (and runs the user's .profile) and which may set the umask before execution of the intended program.

! \_BPX\_BATCH\_SPAWN causes BPXBATCH to use spawn instead of fork/exec and allows data definitions to be carried over into the spawned process. When \_BPX\_BATCH\_SPAWN is set to YES, spawn will be used. If it is set to NO, which is equivalent to the default behavior, fork/exec will be used to execute the program.

! If \_BPX\_BATCH\_SPAWN is set to YES, then you must consider two other environment variables that affect spawn (BPX1SPN):

```
! *   _BPX_SHAREAS=YES!NO!REUSE
```

! When YES or REUSE, the child process created by spawn will run in the same address space as the parent's under these conditions:

- ! - The child process is not setuid or setgid to a value different from the parent
- ! - The spawned filename is not an external link or a sticky bit file
- ! - The parent has enough resources to allow the child process to reside in the same address space

```
! - The NOSHAREAS extended attribute is not set

! When NO, the child and parent run in separate address spaces.

! * _BPX_SPAWN_SCRIPT=YES

! When YES, the spawn will treat the specified file as a shell script
! and will invoke the shell to run the shell script.

! For more information about spawn, see BPX1SPN in OS/390 UNIX System
! Services Programming: Assembler Callable Services Reference .
```

### 8.2.3 Invoking BPXBATCH in JCL

---

. Copyright IBM Corp. 1996, 1998

You can create a job that uses BPXBATCH to run a shell command, a shell script, or an executable file.

BPXBATCH is invoked in JCL in this way:

```
//stepname EXEC PGM=BPXBATCH,PARM='SH!PGM program_name'
```

where:

- \* When SH is specified, program\_name is the name of a shell command or a file containing a shell script. SH is the default; therefore, you can omit PARM= and supply the name of a shell script for STDIN, and it will be invoked.
- \* When PGM is specified, program\_name is the name of an executable file or a REXX exec that is stored in an HFS file.

If you specify data sets in a STEPLIB DD statement, all the data sets must be cataloged. UNIT= and VOL=SER= parameters are not propagated to the process that is being executed by BPXBATCH.

If the job needs to run with a group other than your default group, on the job card you need to code GROUP=grpname to specify the group your job needs to run under. For BPXBATCH, the group needs to have an OMVS segment and a GID defined for it.

If your job requires a REGION size greater than the default on your system, you may receive this abend code:

```
ABEND 4093 reason code 0000001c
```

To fix this, use a larger REGION size. For example, you could specify:

```
//SHELLCMD EXEC PGM=BPXBATCH,REGION=8M,PARM='SH shell_cmd'
```

#### 8.2.3.1 Running a Shell Script in Batch

---

. Copyright IBM Corp. 1996, 1998

You can use BPXBATCH to run a shell script through MVS batch and redirect the output error messages to an HFS file. Because the default is PARM='SH', the PARM is not specified in the following example. The shell script associated with STDIN is invoked. You can allocate STDIN, STDOUT, and STDERR as files, using the PATH operand. In this example, for user TURBO:

- \* The pathname of the STDIN file (the shell script) is

```
/u/turbo/bin/myscript.in
* The pathname of the STDOUT file is /u/turbo/bin/mystd.out
* The pathname of the STDERR file is /u/turbo/bin/mystd.err
```

```
//jobname JOB ...
//stepname EXEC PGM=BPXBATCH,REGION=8M
//STDIN DD PATH='/u/turbo/bin/myscript.in',PATHOPTS=(ORDONLY)
//STDOUT DD PATH='/u/turbo/bin/mystd.out',PATHOPTS=(OWRONLY,OCREAT),
// PATHMODE=SIRWXU
//STDERR DD PATH='/u/turbo/bin/mystd.err',PATHOPTS=(OWRONLY,OCREAT),
// PATHMODE=SIRWXU
```

#### 8.2.3.2 Running a Shell Command in Batch

-----  
Copyright IBM Corp. 1996, 1998

In the following example, BPXBATCH runs the shell command `ls` to print the contents of the directory `/usr/lib` to the file `/u/turbo/bin/mystd.out`. To start the next JCL job step before the `ls` command completes, in this example:

- \* `nohup` is specified with the command. This prevents the process running `ls` from being ended when the job step ends.
- \* `ls` is specified with `&`, so it runs in the background.

A job can be run without the `&` and `nohup` shown in the example; in that case, the job stays active until the shell command ends. However, if you do use BPXBATCH to run a job in background, you must use both the `&` (ampersand) symbol and `nohup` shell command together. If the `&` is used without `nohup`, the results are unpredictable.

In this example, `STDOUT` and `STDERR` are HFS files allocated using the `PATH` operand:

- \* The pathname of the standard output (`STDOUT`) file is `/u/turbo/bin/mystd.out`.
- \* The pathname of the standard error (`STDERR`) file is `/u/turbo/bin/mystd.err`.

```
//jobname JOB ...
//stepname EXEC PGM=BPXBATCH,REGION=8M,PARM='SH nohup ls /usr/lib&'
//STEPLIB DD DSN=CEE.SCEERUN,DISP=SHR
//STDOUT DD PATH='/u/turbo/bin/mystd.out',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),PATHMODE=SIRWXU
//STDERR DD PATH='/u/turbo/bin/mystd.err',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),PATHMODE=SIRWXU
```

The `STEPLIB` is propagated for the execution of the shell and for any processes created by the shell.

#### 8.2.4 Invoking BPXBATCH from TSO/E READY

-----  
Copyright IBM Corp. 1996, 1998

You can use BPXBATCH to run a shell command or a shell script from TSO/E, but it is even easier to use the `OSHELL exec`, which invokes BPXBATCH.

The syntax for using BPXBATCH to run a shell command from TSO/E is:

```
BPXBATCH SH shellcmd
```





```

!   IF (RC ^= 254) & (RC ^= 255) THEN
!
!       DO
!           "ALLOCATE FILE(out1) DA(*) LRECL(255) RECFM(F) REUSE"
!           "OCOPY indd(STDOUT) outdd(out1) TEXT PATHOPTS(OVERRIDE)"
!           "FREE DDNAME(out1)"
!
!       END
!
!   "FREE DDNAME(STDOUT)"
!
!
!
!
!-----!

```

Figure 14. OSHELL REXX exec

### 8.3 Using TSO/E REXX for OS/390 UNIX System Services Processing

Copyright IBM Corp. 1996, 1998

! You can use a set of OS/390 UNIX extensions to TSO/E REXX--host commands  
! and functions--to access kernel callable services. The OS/390 UNIX  
! extensions, called syscall commands, have names that correspond to the  
! names of the callable services that they invoke--for example, access,  
! chmod, and chown.

You can run a REXX program with OS/390 UNIX extensions from MVS, TSO/E,  
the shell, or a C program. The exec is not portable to an operating system  
that does not have OS/390 UNIX installed.

For more information about the REXX extensions that call OpenMVS services,  
see OS/390 Using REXX and OS/390 UNIX System Services.

### 8.4 Using the ISPF Shell

Copyright IBM Corp. 1996, 1998

With the ISPF shell, a user or system programmer can use ISPF dialogs  
instead of shell commands to perform many tasks, especially those related  
to file systems and files. An ordinary user can use the ISPF shell to work  
with:

- \* Directories
- \* Regular files
- \* FIFO special files
- \* Symbolic links, including external links

You can also run shell commands, REXX programs, and C programs from the  
ISPF shell. The ISPF shell can direct stdout and stderr only to an HFS  
file, not to your terminal. If it has any contents, the file is displayed  
when the command or program completes.

#### 8.4.1 Invoking the ISPF Shell

Copyright IBM Corp. 1996, 1998

You can invoke the shell by:

- \* Typing the TSO/E ISHELL command. See "Entering a TSO/E Command" in

topic 11.6.1 for information on entering TSO/E commands in TSO/E, the shell, and ISPF.

- \* Selecting the ISPF shell from the ISPF menu, if a menu option is installed.

#### 8.4.2 Working in the ISPF Shell

-----  
Copyright IBM Corp. 1996, 1998

Figure 15 is the main panel, which you see when you invoke the ISPF shell. At the top of the panel is the action bar, with seven choices:

```
File
Directory
Special file
Tools
File systems
Options
Setup
Help
```

When you select one of these choices, a pulldown panel with a list of actions is displayed.

```
!-----!
! File Directory Special_file Tools File_systems Options Setup Help !
!-----!
!                               ISPF Shell                               !
!
! Enter a pathname and do one of these:                               !
!
!   - Press Enter.                                                    !
!   - Select an action bar choice.                                     !
!   - Specify an action code or command on the command line.        !
!
! Return to this panel to work with a different pathname.           !
!
! /                                                                    More:  + !
! _____                                                            !
! _____                                                            !
! _____                                                            !
!
! (C) Copyright IBM Corp., 1993. All rights reserved.                !
! Command ==> _____                                                !
! F1=Help   F3=Exit   F5=Retrieve F6=Keyshelp F7=Backward F8=Forward  !
! F10=Actions F11=Command F12=Cancel                                     !
!-----!
```

Figure 15. ISPF Shell: The Main Panel

In the center of the panel, you see three lines. Here you can type the pathname of a file (a directory is a type of file) that you want to work with. It can be the name of an existing file or a new file that you are creating.

In the lower part of the panel, you see a command line. Here you can type an action code, a one-character code that specifies an action that you want to perform on the pathname you are working with. For example, D is the action code for "delete." (To familiarize yourself with the action codes, press <F1> on the main panel. On the help panel that is displayed,

position your cursor under the highlighted words action code and press <F1>.)

Work in the ISPF shell is a two-step sequence:

1. Select an object--the pathname of a new or existing file.
2. Select an action for that object.

#### 8.4.3 Selecting an Object

---

. Copyright IBM Corp. 1996, 1998

When you select an object, this pathname becomes the object of most subsequent actions until you change it or delete it. You can select an object in either of these ways:

- \* Specify the pathname on the main panel
- \* Select a file on a directory list panel

##### 8.4.3.1 On the Main Panel

---

. Copyright IBM Corp. 1996, 1998

On the main panel, you can supply a pathname for a directory, a regular file, a FIFO special file, symbolic link, external link, or, if you are a superuser, a character special file.

If you type a pathname for a file that does not exist and press <Enter>, you see a popup panel from which you can:

- \* Create a directory
- \* Create a regular file using ISPF File Edit
- \* Create a regular file by copying another file into it
- \* Create a regular file by copying a data set into it
- \* Create a FIFO special file
- \* Create a symbolic link or an external link
- \* Create a hard link to a file

An external link is a type of symbolic link and is handled as such.

If you type the name of an existing file and press <Enter>, the system performs the default action for that file type. The default actions, which you can change using the Options pulldown, depend on the file type:

- \* Directory. List the files in the directory
- \* Regular file. Browse the file
- \* Character special file. Display its attributes
- \* FIFO special file. Display its attributes
- \* Symbolic link. Display its attributes

##### 8.4.3.2 From a Directory List

---

. Copyright IBM Corp. 1996, 1998

From the Directory pulldown, you can select a directory list. All the files in a directory are displayed in a list; you can then select a file by typing a slash (/) or an s or an action code next to its name.

#### 8.4.4 Selecting an Action

---

. Copyright IBM Corp. 1996, 1998

To select an action, you can do one of these:

- \* Specify an action code on the command line of the main panel.
- \* Select an action from an action bar pulldown panel.
- \* Specify an action on a directory list panel.

#### 8.4.4.1 On the Command Line

-----  
 . Copyright IBM Corp. 1996, 1998

If you type a pathname on the main panel and then type an action code on the command line and press <Enter>, you can perform the same file system tasks that are available from the pulldown panels.

To familiarize yourself with the action codes, press <F1> on the main panel. On the help panel that is displayed, position your cursor under the highlighted words action code and press <F1>.

#### 8.4.4.2 Using the Action Bar

-----  
 . Copyright IBM Corp. 1996, 1998

Supply a pathname on the main panel. To use the action bar, position your cursor under one of the choices and press <Enter>. You then see a pulldown panel with a list of actions; Figure 16 shows what you would see if you selected the Directory pulldown.

```

!-----!
!
!   File  Directory  Special file  File systems  Options  Setup  Help
! -----!-----!-----!-----!-----!-----!
! BPXWP ! 1. List directory(L)... ! Shell
!      ! 2. New(N)... !
! Enter ! 3. Attributes(A)... ! t an action bar choice. You can
! also ! 4. Delete(D)... ! d line.
!      ! 5. Rename(R)... !
! A bla ! 6. Copy to PDS(C)... ! of your working directory.
!      ! 7. Copy from PDS(I)... ! More:
! /t    ! 8. Print(P) !
!      ! 9. Compare(M)... !
!      ! 10. Find strings(F)... !
!      ! 11. Set working directory(W) !
!-----!
!
! Command ==>
! F1= Help   F3=Exit   F5=Retrieve  F6=Keyshelp  F7=Backward  F8=Forward!
! F10=Actions F11=Command F12=Cancel
!
!-----!

```

Figure 16. A Pulldown Panel Selected from the Action Bar

To select an action from the pulldown you can do either of these:

- \* Specify the number of the action you want to select and press <Enter>.
- \* Position the cursor next to the action and press <Enter>.

The letter in parentheses following each action is the action code for that action.

#### 8.4.4.3 On a Directory List

-----  
 . Copyright IBM Corp. 1996, 1998

From the Directory pulldown, you can select a directory list. All the

files in a directory are displayed in a list; you can then select a file and specify an action by typing the action code next to the filename.

#### 8.4.5 Using the Online Help Facility

-----  
Copyright IBM Corp. 1996, 1998

In the ISPF shell, you can get help information for:

- \* Panels
- \* Fields on panels
- \* Highlighted words on panels

Position your cursor on one of those locations and press <F1>.

For more information on the online help facility when you begin working in the ISPF shell, select the Help choice on the action bar and read the information there.

#### 8.4.6 Working with the File Pulldown

-----  
Copyright IBM Corp. 1996, 1998

Using the File pulldown, you can perform the following tasks with a regular file.

!-----!	
! Table 3. Using the ISPF Shell to Work with a Regular File !	
!-----!	
! Task	! Pulldown Choice
!-----!	
! Create a file.	! 1. New (N)
!-----!	
! Display and alter file attributes (such as permissions, owning user, owning group, audit settings).	! 2. Attributes (A)
!-----!	
! Delete a file.	! 3. Delete (D)
!-----!	
! Rename a file.	! 4. Rename (R)
!-----!	
! Edit a file.	! 5. Edit (E)
!-----!	
! Browse a file with fixed-length records.	! 6. Browse text (B)
!-----!	
! Browse a text file that has delimited lines.	! 7. Browse records (V)
!-----!	
! Copy a file to another file.	! 8. Copy to (C)
!-----!	
! Copy a file to a sequential data set, specifying one or both of these options: binary copy or code page conversion.	! 8. Copy to (C)
!-----!	
! Replace the contents of a file with another file.	! 9. Replace from (I)
!-----!	
! Replace the contents of a file with a sequential data set, specifying one or both of these options: binary copy and code page conversion.	! 9. Replace from (I)
!-----!	
! Print a file to the ISPF list data set.	! 10. Print (P)
!-----!	
! Compare a file with another file and put the results in an output file.	! 11. Compare (M)
!-----!	
! Search a file for a specified text string.	! 12. Find Strings (F)
!-----!	

Run executable files.	13. Run (X)
Create a hard link to a file.	14. Link
Display the attributes for the file system the file is in	15. File System (U)

#### 8.4.7 Working with the Directory Pulldown

Copyright IBM Corp. 1996, 1998

Using the Directory pulldown, you can perform the following tasks with a directory. To customize the way a directory list is sorted or displayed, use the Options pulldown.

Table 4. Using the ISPF Shell to Work with a Directory	
Task	Pulldown Choice
List all files and subdirectories in a directory.	1. List directory (L)
Create new subdirectories within a directory.	2. New (N)
Display and alter directory attributes (such as permissions).	3. Attributes (A)
Delete a directory.	4. Delete (D)
Rename a directory.	5. Rename (R)
Copy all files in a directory to a partitioned data set or a PDS/E. If desired, specify one or more of these options: binary copy, inclusion of files with lowercase names, code page conversion, or stripping the suffix from the filenames.	6. Copy to PDS (C)
Copy all or selected members from a partitioned data set or PDS/E into a directory. If desired, specify one or more of these options: binary copy, conversion of data set names to lowercase, code page conversion, or adding a suffix to the filenames.	7. Copy from PDS (I)
Print a directory list to the ISPF list data set.	8. Print (P)
Compare two directories and put the results in an output file.	9. Compare (M)
Search all the files in a directory for a specified text string and put the results in an output file.	10. Find strings (F)
Change the working directory.	11. Set working directory (W)
Display the attributes for the file system the directory or a selected file is in	12. File System (U)

#### 8.4.8 Working with the Special File Pulldown

Copyright IBM Corp. 1996, 1998

Using the Special File pulldown, you can perform the following tasks with a FIFO special file or a symbolic link.

Table 5. Using the ISPF Shell to Work with a FIFO Special File or a Symbolic Link	
Task	Pulldown Choice
Create a FIFO special file.	1. New FIFO
Create a symbolic link or an external link.	2. New symbolic link(N)
Display or modify permission for a FIFO special file.	3. Attributes(A)
Show attributes for a FIFO special file (such as date created, link count, and size).	3. Attributes(A)
Display the attributes and contents of a symbolic link.	3. Attributes(A)
Delete a FIFO special file or symbolic link.	4. Delete(D)
Rename a FIFO special file or symbolic link.	5. Rename(R)
Create a hard link to a FIFO special file.	6. Link

#### 8.4.9 Working with the Tools Pulldown

Copyright IBM Corp. 1996, 1998

Using the Tools pulldown, you can perform the following tasks.

Table 6. Using the ISPF Shell's Tools Pulldown	
Task	Pulldown Choice
Display process attributes or send a signal to a process	1. Work with processes(PS)
Run a shell command, using the login shell /bin/sh -Lc	2. Run shell command(SH)
Run a program or shell command, using the spawn service instead of a shell	3. Run program(EX)

#### 8.4.10 Working with the File Systems Pulldown

Copyright IBM Corp. 1996, 1998

Using the File Systems pulldown, you can perform the following tasks with a file system.

Table 7. Using the ISPF Shell to Work with a File System	
Task	Pulldown Choice
Display all mounted file systems.	1. Mount table

! Display the attributes of a mounted file system (such as total blocks, blocks in use, and ddname).	! 1. Mount table	!
! Allocate an HFS data set.	! 2. New	!
! Mount a File System (superuser only)	! 3. Mount (O)	!

#### 8.4.11 Working with the Options Pulldown

. Copyright IBM Corp. 1996, 1998

Using the Options pulldown, you can select the following customization options.

! Table 8. Tailoring the ISPF Shell for Your Use		
! Task	!	! Pulldown Choice
! Select sorting options for a directory list (for example, case-insensitive sort by filename).	!	! 1. Directory list
! Select display options for a directory list (for example, show file type or permissions).	!	! 1. Directory list
! Select the default action to be taken for each file type when you type a pathname on the main panel and press <Enter>.	!	! 2. Default actions
! Select Edit and Browse options.	!	! 3. Edit or browse
! Enter Edit profile name.	!	! 3. Edit or browse
! Enter Edit initial macro and select option to bypass the initial macro panel when you select Edit.	!	! 3. Edit or browse
! Position the command line on the bottom of the screen during an Edit or Browse session.	!	! 3. Edit or browse
! Change the position of the command line.	!	! 4. Command line on top
! Activate or deactivate the prompt that asks you to confirm a deletion (for example, deleting a file).	!	! 5. Bypass delete confirmation
! Activate or deactivate the prompt that asks you to confirm that you want to exit the shell.	!	! 6. Bypass exit confirmation

While using the ISPF shell, you can enter ISPF system commands to perform customization tasks, such as:

- \* KEYLIST, to customize the function keys
- \* PFSHOW, to control the display of function keys

Enter the command from the command line in the ISPF shell. (You can also enter the LIST command to process the ISPF list data set.) For more details on these ISPF system commands, see ISPF Dialog Management Guide and Reference

#### 8.4.12 System Programmer Tasks

. Copyright IBM Corp. 1996, 1998

A system programmer can use the ISPF shell to perform the following tasks, which are further discussed in OS/390 UNIX System Services Planning. These tasks require superuser authority.

Table 9. Using the ISPF Shell for System Programmer Tasks		
Task	Action Bar Choice	Pulldown Choice
Mount a file system.	File systems	3. Mount(O)
Unmount a file system.	File systems	1. Mount table
Reset a pending unmount.	File systems	1. Mount table
Reset a quiesce status.	File systems	1. Mount table
Change attributes for an OS/390 UNIX user.	Setup	1. User
Authorize an OS/390 UNIX user.	Setup	1. User
Display a list of users and sort by name, UID, or GID.	Setup	2. User list
Print a list of users.	Setup	2. User list
Set up all OS/390 UNIX users.	Setup	3. All users
Set up kernel groups.	Setup	4. All groups
Permit users to alter their own home directory and initial program.	Setup	5. Permit field access
Create character special files.	Setup	6. Character special
Set up the root file system.	Setup	7. Root
Switch to or from superuser status	Setup	8. Enable superuser mode (SU)

## 9.0 Chapter 9. Performance: Running Executable Files

Copyright IBM Corp. 1996, 1998

A process is a collection of threads that execute within an address space, along with the required system resources. A user's login shell is one example of a process.

- \* The OMVS command creates two processes per login: a process to control the terminal and then a process for the login shell.
- \* rlogin and telnet logins each create two processes: one to control the socket connection to the user, another for the login shell.

- \* Communications Server logins require only one process per login. Consequently, there is no method for requesting a shared address space for the Communications Server login shell.

Most utilities invoked from the shell command line run in new processes that the shell creates.

There is a systemwide limit on:

- \* The number of OS/390 UNIX processes across the system
- \* The number of OS/390 UNIX processes per user

For a discussion of these limits, see OS/390 UNIX System Services Planning.

The shell and other OS/390 UNIX commands and daemons can assign multiple processes to the same MVS address space; this is called a shared address space. Using a shared address space offers these advantages:

- \* A new process in the same address space can be started faster than a new process in another address space.
- \* A new process in the same address space requires fewer system resources (storage, for example) than a new process in another address space.

For rlogin, the system administrator must update /usr/sbin/inetd.conf by adding -m to the rlogind entry to enable shared address space. When -m is added, the socket connection process and the login shell process share the same address space.

For the OMVS command, use the SHAREAS keyword to enable shared address space. When the SHAREAS keyword is used, the login shell process is nested in the user's TSO address space. Any other login shells started with the OMVS OPEN subcommand are also nested in the user's TSO address space. (With NOSHAREAS, other login shells started with the OMVS OPEN subcommand will each consume another address space.)

To enable shared address space for the shell, issue the command

```
export _BPX_SHAREAS=YES
```

interactively or place it in your \$HOME/.profile. Then all simple commands (any command run in the foreground and that is not in a pipeline) will run in processes nested in the shell's address space. If this variable is not set or is not set to the value YES, the shell creates all processes in separate address spaces. No matter how the shell is started (with or without shared address space enabled), you must set \_BPX\_SHAREAS=YES if processes started by the shell itself are to run in processes nested in the shell's address space.

User applications can use shared address spaces as well. See the description of the spawn() function and the BPX1SPN and BPX1ATX callable services for details.

Some processes cannot execute correctly in a shared address space. For example, if a process needs to reserve MVS system resources that are common to all processes in an MVS address space, it must run by itself. If two processes using the same MVS resource attempted to execute concurrently in the same address space, they would compete for these resources thus causing at least one of them to fail. When a potential storage shortage is detected, the new processes are created in their own address spaces, even if \_BPX\_SHAREAS=YES is present in the invoker's environment. For more details about these restrictions, see the descriptions of the spawn() function and BPX1SPN callable service.

Improving Shell Script Performance: You can improve shell script

performance by setting the `_BPX SPAWN SCRIPT` environment variable to a value of YES. However, when `_BPX SPAWN SCRIPT=YES`, the behavior will not conform completely to the XPG4 Commands & Utilities specification. See "Improving the Performance of Shell Scripts" in topic 4.6 for more information.

## 10.0 Chapter 10. Communicating with Other Users

---

Copyright IBM Corp. 1996, 1998

You can communicate only with users in the same environment you are working in. For example, if you are working in the TSO/E environment, you cannot use MVS facilities to send a message to a user working in the shell.

OS/390 UNIX shell users who want to exchange messages with other shell users at the same system can use shell commands. Other users may prefer to use TSO/E facilities in order to be able to exchange messages with all TSO/E users, not just those using the shell.

Within the shell, you can send and receive messages using these shell commands:

- \* mailx
- \* mail
- \* write
- \* talk
- \* wall

Alternatively, you can switch into your TSO/E session and send messages to any TSO/E user by using TSO/E facilities, through either the OFFICE option of the Information Center Facility (ICF), if it is installed on your system, or through TSO/E commands. You can also receive messages using TSO/E.

If your system has Transmission Control Protocol/Internet Protocol (TCP/IP) or other network management facilities installed, you can log in to the TCP/IP network and send messages to users at other systems.

If your system has UUCP (Unix-to-Unix Copy Program) installed and set up, you can use this facility to send files to, or run commands or custom applications at, other sites in the UUCP network.