z/OS

UNIX System Services Command Reference: APAR OA10314

SA22-7802-06

z/OS

UNIX System Services Command Reference: APAR OA10314

SA22-7802-06

About this document

This document supports APAR OA10314 for z/OS UNIX System Services. This document is only available on the z/OS UNIX System Services Web site at: http://www.ibm.com/servers/eserver/zseries/zos/unix/release/apar.html

Chapter 1. cp — Copy a file

Format

cp [**–cfimMUv**] [**–p|F format|B|T|X**] [**–P params**]*file1 file2*
cp [**–ACcfimMUv**] [**–p|F format|B|T|X**] [**–S suffix**] *file ... directory*
cp –R [**–cfimp**] *source... directory*
cp –r [**–cfimp**] *source... directory*

Automatic Conversion and File Tag Specific Options:

cp [**–Z**] [**–O u | c=***codeset*]

Description

cp copies files to a target named by the last argument on its command line. If the target is an existing file, **cp** overwrites it; if it does not exist, **cp** creates it. If the target file already exists and does not have write permission, **cp** denies access and continues with the next copy.

If you specify more than two pathnames, the last pathname (that is, the target) must be a directory. If the target is a directory, **cp** copies the sources into that directory with names given by the final component of the source pathname.

You can also use **cp** to copy files to and from MVS data sets. If you specify more than one file to be copied, the target (last pathname on command line) must be either a directory or a partitioned data set. If the target is an MVS partitioned data set, the source cannot be a UNIX directory.

cp does not support the copying to or from GDGs. To use those MVS data sets, user must specify the real data set name.

When copying records, the string ″ \n″ is copied the same way as the string ″\n″: both are read back as ″\n″, where ″\n″ indicates that z/OS C++ will write a record containing a single blank to the file (the default behavior of z/OS C/C++). All other blanks in your output are read back as blanks, and any empty (zero-length) records are ignored on input. However, if the environment variable **_EDC_ZERO_RECLEN** is set to Y before calling **cp**, an empty record is treated as a single newline and is not ignored. Also, if **_EDC_ZERO_RECLEN** is set to Y, a single newline is written to the file as an empty record, and a single blank will be represented by ″ \n″.

You can copy:
- One file to another file in the working directory
- One file to a new file on another directory
- A set of directories and files to another place in your file system
- A UNIX file to an MVS data set
- An MVS data set to a filesystem
- An MVS data set to an MVS data set

Options

–A Specifies that all suffixes (from the first period till the end of the target) be truncated. **–A** has precedence over **–M** and **–C** options. **–S** will be turned off if **–A** is the last option specified.

–B Specifies that the data to be copied contains binary data. When you specify

–B, **cp** operates without any consideration for <newline> characters or special characteristics of DBCS data (this type of behavior is typical when copying across a UNIX system). **–B** is mutually exclusive with **–F**, **–X**, and **–T**, i.e., you will get an error if you specify more than one of these options.

–C Specifies truncating the filename(s) to 8 characters to meet the restriction in the MVS data set member.

–c (UNIX to UNIX only)
Prompts you to change the diskette if there is not enough room to complete a copy operation. This option has no effect on systems without floppy drives.

Note: The parent directories must already exist on the new target diskette.

–F *format*
Specifies if a file is binary or text and for text files, specifies the end-of-line delimeter. Also sets the file format to *format* if the target is a UNIX file. For text files, when copying from UNIX to MVS, the end-of-line delimeter will be stripped. When copying from MVS to UNIX, the end-of-line delimeter will be added (Code page IBM-1047 will be used to check for end-of-line delimeters).

If setting *format* fails, a warning will be displayed. However, **cp** will continue to copy any remaining files specified to be copied.

–F is mutually exclusive with **–B**, **–X**, **–p**, and **–T**. If you specify one of these options with **–F**, you will get an error. If **–F** is specified more than once, the last **–F** specified will be used.

For *format* you can specify:
not not specified
bin binary data

Or the following text data delimeters:
nl newline
cr carriage return
lf line feed
crlf carriage return followed by line feed
lfcr line feed followed by carriage return
crnl carriage return followed by new line

–f (UNIX to UNIX only)
Attempts to replace files that do not have write permission.

–i When copying to a UNIX target, **–i** asks you if you want to overwrite an existing file, whether or not the file is read-only.

–M Specifies that some characters of the filename are translated when copying between a UNIX file and an MVS data set member. Characters are translated as follows:

- _ (underscore) in UNIX is translated to @ in MVS DS members and vice versa.
- . (period) in UNIX is translated to # in MVS DS members and vice versa.
- – (dash) in UNIX is translated to $ in MVS DS members and vice versa.

–m (UNIX to UNIX only)
Sets the modification and access time of each destination file to that of the corresponding source file. Normally, **cp** sets the modification time of the destination file to the present.

–P params

Specifies the parameters needed to create a sequential data set if one does not already exist. You can specify the RECFM, LRECL, BLKSIZE, and SPACE in the format the CRTL **fopen()** function uses. However, LRECL and BLKSIZE can be used for variable record format only.

`SPACE=(units,(primary,secondary)` where the following values are supported for units:

- any positive integer indicating BLKSIZE
- CYL (mixed case)
- TRK (mixed case)

For example:

```
SPACE=(500,(100,500)) units, primary, secondary
SPACE=(500,100) units and primary only
```

Note: CRTL **fopen()** arguments: LRECL specifies the length, in bytes, for fixed-length records and the maximum length for variable-length records. BLKSIZE specifies the maximum length, in bytes, of a physical block of records. RECFM refers to the record format of a data set and SPACE indicates the space attributes for MVS data sets.

–p Preserves the modification and access times (as the **–m** option does); in addition, it preserves file mode, owner, and group owner, if authorized. It also preserves extended attributes. It preserves the ACLs of files and directories, if possible. The ACLs are not preserved if a file system does not support ACLs, or if you are copying files to MVS

–p is mutually exclusive with **–F**. If you specify both, you will get an error message.

–R (UNIX to UNIX only)

"Clones" the source trees. **cp** copies all the files and subdirectories specified by *source...* into *directory*, making careful arrangements to duplicate special files (FIFO, character special). **cp** will traverse directories by following symbolic links through the file hierarchy.

–r (UNIX to UNIX only)

"Clones" the source trees, but makes no allowances for special files (FIFO, character special). Consequently, **cp** attempts to read from a device rather than duplicate the special file. This is similar to, but less useful than, the preferred **–R**.

–S *d=suffix|a=suffix*

- *d=suffix*

  Removes the specifed suffix from a file.

- *a=suffix*

  Appends the specified suffix to a file.

–S has precedence over **–M** and **–C**. It also turns off the **–A** option (if **–S** is the last specified option).

–T Specifies that the data to be copied contains text data. See "Usage Notes" on page 9 for details on how to treat text data. This option looks for IBM-1047 end-of-line delimeters, and is mutually exclusive with **–F**, **–X**, and **–B**. That is, you will get an error if you specify more than one of these options.

Note: **–T** is ignored when copying across UNIX file systems.

–U Keeps filenames in uppercase when copying from MVS data set members to UNIX files. The default is to make filenames lowercase.

–v Verbose

–X Specifies that the data to be copied is an executable. Cannot be used in conjunction with **–F**, **–X**, or **–B**.

-Z Specifies that error messages are not to be displayed when setting ACLs on the target. The return code will be zero.

Note: If you do not specify either **–FIBIT** or **X**, **cp** will first check the format of the MVS data set indicated and then try to determine the type of file.

Automatic conversion and file tag specific options

–Z Suppresses failure when default behavior is used to set the file tag. For a description of the default behavior, see "Automatic conversion and file tagging behavior for cp."

–O u | c=*codeset*

Allow automatic conversion on source and target files.

–O u If the target exists and is not empty nor already tagged, **cp** will not change the target's tag in order for the target to be a candidate for automatic conversion.

For new targets and existing, untagged, empty files, this option has no effect and **cp** behaves the same as the default. For a description of the default behavior, see "Automatic conversion and file tagging behavior for cp."

When using **cp** to copy from a UNIX file to an MVS data set, if the source is a tagged text file, then it may be a candidate for automatic conversion.

When copying executables from or to MVS, automatic conversion is disabled for both source and target.

–O c=*codeset* For a detailed description of the behavior of this option on **cp**, see "Automatic conversion and file tagging behavior for cp."

To prevent the corruption of text files, **cp** will fail if it cannot set the file tag to text or *codeset*.

Attention: If automatic conversion is not properly set or if the source is not properly tagged, the target may end up with a tag codeset that does not match the file contents.

Automatic conversion and file tagging behavior for cp

The following tables describe the behavior of file tagging and automatic conversion for various source and target scenarios depending on whether the **–O** option is specified on the **cp** command.

Table 1. Automatic conversion and file tagging behavior: ***Copying UNIX files to UNIX files***

	Default (without -O option)			
	If the target file system supports setting file tags...	**If the target file system does not support setting file tags (e.g. NFS)...**	**With -O u option**	**With -O c=*codeset* option**
File tagging	Target file is tagged the same as the source file.	• An existing target's tag is unchanged. • A new target is created with a tag according to the file system's attributes (MOUNT parm can specify TAG).	Target's tag is unchanged. **Note:** The source or target file is a candidate for automatic conversion when its txtflag is tagged TEXT.	Target's txtflag is set to TEXT and its codeset is set to *codeset*.
Automatic conversion	Disabled for source and target files	Allowed for source and target files		

Table 2. Automatic conversion and file tagging behavior: ***Copying MVS data sets to UNIX files***

	Default (without -O option)			
	If the target file system supports setting file tags...	**If the target file system does not support setting file tags (e.g. NFS)...**	**With -O u option**	**With -O c=*codeset* option**
If the SOURCE is text:				
File tagging	Target is set to UNTAG	• An existing target's tag is unchanged. • A new target is created with a tag according to the file system's attributes (MOUNT parm can specify TAG).	Target's tag is unchanged	Target's txtflag is set to TEXT and its codeset is set to *codeset*.
Automatic conversion	Disabled for target file	Allowed for target file **Note:** The target file is a candidate for automatic conversion when its txtflag is tagged TEXT.		
If the SOURCE is binary or executable:				
File tagging	Target is set to UNTAG		Target's tag is unchanged	Target's txtflag is set to BINARY and its codeset is set to *codeset*.

Table 2. Automatic conversion and file tagging behavior: ***Copying MVS data sets to UNIX files*** *(continued)*

	Default (without -O option)			
	If the target file system supports setting file tags...	**If the target file system does not support setting file tags (e.g. NFS)...**	**With -O u option**	**With -O c=*codeset* option**
Automatic conversion	Disabled for target file			

Table 3. Automatic conversion and file tagging behavior: ***Copying UNIX files to MVS data sets***

	Default (without -O option)	**With -O u option**	**With -O c=*codeset* option**
If the SOURCE is text or binary:			
File tagging	Not applicable for target data set		
Automatic conversion	Disabled for source file	Allowed for source file **Note:** The source file is a candidate for automatic conversion when its txtflag is tagged TEXT.	Disabled for source file
If the SOURCE is executable:			
File tagging	Not applicable for target data set		
Automatic conversion	Disabled for source file		

The **–p** option on **cp** does not affect file tagging.

Limits and Requirements

General Requirements

1. To specify an MVS data set name, precede the name with double slashes (//). For example, to specify the fully qualified data set names 'turbo.gammalib' and 'turbo.gammalib(pgm1)', you write:

   ```
   "//'turbo.gammalib'"
   "//'turbo.gammalib(pgm1)'"
   ```

   The same goes for data set names that are not fully qualified:

   ```
   //turbo
   ```

2. For PDS (partitioned data set) or PDSE (partitioned data set extended), to avoid parsing by the shell, the name should be quoted or minimally, the parenthesis should be escaped. For example, to specify 'turbo(pgm1)', you can use quotes:

   ```
   "//turbo(pgm1)"
   ```

   or escape the parenthesis:

   ```
   //turbo\(pgm1\)
   ```

   As indicated above, a fully qualified name must be single-quoted (as is done within TSO). To prevent the single quotes from being interpreted by the shell,

they must be escaped or the name must be placed within regular quotation marks. See the 'turbo.gammalib' examples above.

3. If you specify a UNIX file as source and the MVS data set (target) does not exist, a sequential data set will be created. If the partitioned data set exists, the UNIX file will be copied to the partitioned data set member.
4. If source is an MVS data set and target is a UNIX directory, the UNIX directory must exist.
5. You cannot have a UNIX directory, partitioned data set, or sequential data set as source if the target is a partitioned data set.
6. To copy all members from a partitioned data set, you may specify the partitioned data set as source and a UNIX directory as target.

MVS data set naming limitations

- Data set names may only contain uppercase alphabetic characters (A-Z). Lowercase characters will be converted to uppercase during any copies to MVS data sets.
- Data set names can contain numeric characters 0–9 and special characters @, #, and $.
- Data set names cannot begin with a numeric character.
- A data set member name cannot be more than 8 characters. If a filename is longer than 8 characters or uses characters that are not allowed in an MVS data set name, the file is not copied. You may use the **–C** option to truncate names to 8 characters.

Limitations: UNIX to MVS data set

1. If you specify a sequential MVS data set that is in undefined record format, the file is copied as binary.
2. If you specify a PDSE that is in undefined record format, the first file successfully copied determines in what format files will be copied. Note that PDSE does not allow mixture. So if the first successfully copied file is an executable, the PDSE will have program objects only and all other files will fail. On the other hand, if the first file is data, then all files are copied as binary.
3. If you specify a PDS that is in undefined record format, UNIX executables are saved as PDS load modules. All other files are copied as binary.
4. If you specify an MVS data set that is either in variable length or fixed record length and you have not set the file format, text files are copied as text, binaries as binary, and executables as binary. (IBM-1047 end-of-line delimeters are detected in the data)
5. If you set the file format, the set value is used to determine if data is binary or text.

Limitations: MVS data set to UNIX

1. If an HFS file does not exist, one is created using 666 mode value:

   ```
   666 mode value:  owner(rw-)  group(rw-)  other(rw-)
   ```

   whether data is binary or text. If the data to be copied is a shell script or executable, an HFS file is created using 777 mode value:

   ```
   777 mode value:  owner(rwx)  group(rwx)  other(rwx)
   ```

2. If an HFS exists and the file format is set, **cp** copies the file as that format. Otherwise,
   - load modules (PDS) are stored as UNIX executables and program objects (PDSE) are copied since they are the same as executables;

- data within data sets of undefined record format are copied as binary if the data is not a program object or load module;
- and data found within data sets of fixed length or variable record length are copied as text. (IBM-1047 end-of-line delimeters are detected in the data)

Limitations: MVS to MVS

1. Options **–A**, **–C**, **–f**, and **–S** are ignored.
2. If target and source are in undefined record format (and neither is a sequential data set), **cp** will attempt to copy the data as a load module. If that fails, then **cp** will copy the data as binary.
3. If target and source are in undefined record format and either is a sequential data set, **cp** copies the data as binary.
4. If the source has a fixed or variable record length and the target is in undefined record format, **cp** copies the data as binary.
5. If the source is in undefined record format and the target has a fixed or variable record length, **cp** copies the data as binary.
6. If both source and target are in fixed or variable record length, **cp** copies the data as text.

Limitations: Copying executables into a PDS

1. A PDS may not store load modules that incorporate program management features.
2. **c89**, by default, produces objects using the highest level of program management.
3. If you plan on copying a load module to a PDS, you may use a pre-linker which produces output compatible with linkage editor. Linkage editor generated output can always be stored in a PDS.

The following table is a quick reference for determining the correct use of options with **cp**.

Table 4. **cp** *Format: File to File and File ... (multiple files) to Directory*

Source/Target	Options Allowed	Options Ignored	Options Failed
UNIX File/UNIX File	Ffip	ABCMPSTUX	
UNIX File/Sequential Data Set	BFiPT	ACfMpSU	X
UNIX File/PDS or PDSE Member	BFiTX	ACfMPpSU	
Sequential Data Set/UNIX File	BFfiTU	ACMPpS	X
Sequential Data Set/Sequential Data Set	BFiPT	ACfMpSU	X
Sequential Data Set/PDS or PDSE Member	BFiT	ACfMPpSU	X
PDS or PDSE Member/UNIX File	BFfiTUX	ACMPpS	
PDS or PDSE Member/Sequential Data Set	BFiPT	ACfMpSU	X

Table 4. **cp** *Format: File to File and File ... (multiple files) to Directory (continued)*

Source/Target	Options Allowed	Options Ignored	Options Failed
PDS or PDSE Member/PDS or PDSE Member	BFiTX	ACfMPpSU	
UNIX File/UNIX Directory	ACFipS	BMPTUX	
PDSE or PDSE Member/UNIX Directory	BFfiMSTUX	ACMPp	
UNIX File/Partitioned Data Set	ABCFiMSTX	fPpU	
PDS or PDSE Member/PartitionedData Set	BFiTX	ACfMPpSU	
Partitioned Data Set/UNIX Directory	ABCFfiMSTUX	Pp	

The tables that follow indicate the kind of copies allowed using **cp**.

Table 5. **cp** *Format: File to File*

Source	Target	Allowed
UNIX File, Sequential Data Set, or Partitioned Data Set Member	UNIX File, Sequential Data Set, or Partitioned Data Set Member	Yes
UNIX Directory	UNIX Directory	No (unless **cp** is used with **–R** or **–r**)
Partitioned Data Set	UNIX Directory (dir) NOTE: results in each member of data set are moved to dir.	Yes
UNIX Directory	Partitioned Data Set	No
Partitioned Data Set	Partitioned Data Set	No
UNIX File or Partitioned Data Set Member	UNIX Directory (must exist) or Partitioned Data Set	Yes
Partitioned Data Set Member	Partitioned Data Set	Yes

Table 6. **cp** *Format: File... (multiple files) to Directory*

Source	Target	Allowed
Any combination of UNIX File or Partitioned Data Set Member	UNIX Directory or Partitioned Data Set	Yes
Any combination of UNIX Directory or Sequential Data Set	Partitioned Data Set or UNIX Directory	No
Partitioned Data Set	UNIX Directory	Yes
Partitioned Data Set	Partitioned Data Set	No

Usage Notes

UNIX to MVS

1. To copy from UNIX to a partitioned data set, you must allocate the data set before doing the **cp**.
2. If an MVS data set does not exist, **cp** will allocate a new sequential data set of variable record format.
3. For text files, all <newline> characters are stripped during the copy. Each line in the file ending with a <newline> character is copied into a record of the MVS data set. If text file format is specified or already exists for the source file, that file format will be used for end-of-line delimeter instead of <newline>. Note that **cp** looks for IBM-1047 end-of-line delimeters in data.

   You cannot copy a text file to an MVS data set that has an undefined record format:

   - For an MVS data set in fixed record format, any line copied longer than the record size will cause **cp** to fail with a displayed error message and error code. If the line is shorter than the record size, the record is padded with blanks.
   - For an MVS data set in variable record format: Any line copied longer than the largest record size will cause **cp** to fail with a displayed error message and error code. Record length is set to the length of the line.
4. For binary files, all copied data is preserved:
   - For an MVS data set in fixed record format, data is cut into chunks of size equal to the record length. Each chunk is put into one record. The last record is padded with blanks.
   - For an MVS data set in variable record format, data is cut into chunks of size equal to the largest record length. Each chunk is put into one record. The length of the last record is equal to length of the data left.
   - For an MVS data set in undefined record format, data is cut into chunks of size equal to the block size. Each chunk is put into one record. The length of the last record is equal to the length of the data left.
5. For load modules, the partitioned data set specified must be in undefined record format otherwise the executable will not be copied.
6. If more than one filename is the same, the file is overwritten on each subsequent copy.
7. If a UNIX filename contains characters that are not allowed in an MVS data set, it will not be copied. If the UNIX filename has more than 8 characters, it can not be copied to an MVS data set member. (See the **–ACMS** options for converting filenames)
8. You are not allowed to copy files into data sets with spanned records.
9. PDSE cannot have a mixture of program objects and data members. PDS allows mixing, but it is not recommended.
10. Special files such as character special, external links, and fifo will not be copied to an MVS data set.
11. If a file is a symbolic link, **cp** will copy the resolved file, not the link itself.
12. UNIX file attributes are lost when copying to MVS. If you wish to preserve file attributes, you should use the **pax** utility.

MVS to UNIX

1. If the specified target HFS file exists, the new data overwrites the existing data. The mode of the file is unchanged.

2. If the specified HFS file does not exist, it will be created using 666 mode value if binary or text (this is subject to **umask**). If the data to be copied is a shell script or executable, the HFS file will be created with 777 mode value (also subject to **umask**).
3. Allocating an MVS dataset to either RECFM(VB) or RECFM(U) will preserve trailing blanks when copying from MVS to HFS.
4. When you copy MVS data sets to UNIX binary files, the <newline> character is not appended to the record.
5. You cannot use **cp** to copy data sets with spanned record lengths.

Examples

1. To specify **–P params** for a non-existing sequential target:

   ```
   cp -P "RECFM=U,space=(500,100)"file "//'turbo.gammalib'"
   ```

2. To copy file **f1** to a fully qualified sequential data set `'turbo.gammalib'` and treat it as a binary:

   ```
   cp -F bin f1 "//'turbo.gammalib'"
   ```

3. To copy all members from a fully qualified PDS `'turbo.gammalib'` to an existing UNIX directory `dir`:

   ```
   cp "//turbo.gammalib'" dir
   ```

4. To drop .c suffixes before copying all files in UNIX directory `dir` to an existing PDS `'turbo.gammalib'`:

   ```
   cp -S d=.c dir/* "//'turbo.gammalib'"
   ```

Environment Variable

cp uses the following envrionment variable when copying records to or from MVS data sets:

_EDC_ZERO_RECLEN
: If set to Y before calling **cp**, an empty record (zero-length) is treated as a single newline and is not ignored. Also, a single newline is written to the file as an empty record, and a single blank will be represented by ” \n”. If you do not set this environment variable when copying records, then the string ” \n” is copied the same way as the string ”\n”: both are read and written as ”\n”, where ”\n” indicates that z/OS C/C++ will write a record containing a single blank to the file (the default behavior of z/OS C/C++). All other blanks in the output are read back as blanks, and any empty records are ignored.

Localization

cp uses the following localization environment variables:

- **LANG**
- **LC_ALL**
- **LC_COLLATE**
- **LC_CTYPE**
- **LC_MESSAGES**
- **LC_SYNTAX**
- **NLSPATH**

Exit Values

0 Successful completion

1 Failure due to any of the following:

- An argument had a trailing slash (/) but was not a directory
- Inability to find a file
- Inability to open an input file for reading
- Inability to create or open an output file
- A read error occurred on an input file
- A write error occurred on an output file
- The input and output files were the same file
- An irrecoverable error when using **–r** or **–R**. Possible irrecoverable **–r** or **–R** errors include:
  - Inability to access a file
  - Inability to change permissions on a target file
  - Inability to read a directory
  - Inability to create a directory
  - A target that is not a directory
  - Source and destination directories are the same

2 Failure due to any of the following:
- An incorrect command-line option
- Too few arguments on the command line
- A target that should be a directory but isn't
- No space left on target device
- Insufficient memory to hold the data to be copied
- Inability to create a directory to hold a target file

Messages

Possible error messages include:

cannot allocate target string
cp has no space to hold the name of the target file. Try to release some memory to give **cp** more space.

name **is a directory (not copied)**
You did not specify **–r** or **–R**, but one of the names you asked to copy was the name of a directory.

*target name***?**
You are attempting to copy a file with the **–i** option, but there is already a file with the target name. If you have specified **–f**, you can write over the existing file by typing `y` and pressing <Enter>. If you do not want to write over the existing file, type `n` and press <Enter>. If you did not specify **–f** and the file is read-only, you are not given the opportunity to overwrite it.

source *name* **and target** *name* **are identical**
The source and the target are actually the same file (for example, because of links). In this case, **cp** does nothing.

unreadable directory *name*
cp cannot read the specified directory—for example, because you do not have appropriate permission.

Portability

POSIX.2, X/Open Portability Guide, UNIX systems.

The **–f** and **–m** options are extensions of the POSIX standard.

Related Information

cat, **cpio**, **ln**, **mv**, **rm**

Chapter 2. mv — Rename or move a file or directory

Format

mv [**–fiMPUv**] [**–F format|B|T|X**] [**–P params**]*file1 file2*
mv [**–ACfiMUv**] [**–F format|B|T|X**] [**–S suffix**] *file ... directory*
mv –Rr [**–fi**] *directory1 directory2*

Automatic Conversion and File Tag Specific Options:

mv [**–Z**] [**–O u | c=***codeset*]

Description

mv renames files or moves them to a different directory. If you specify multiple files, the target (that is, the last pathname on the command line) must be a directory. **mv** moves the files into that directory and gives them names that match the final components of the source pathnames. When you specify a single source file and the target is not a directory, **mv** moves the source to the new name, by a simple rename if possible.

You can also use **mv** to move files to and from MVS data sets. If you specify more than one file to be moved, the target (last pathname on command line) must be either a directory or a partitioned data set. If the target is an MVS partitioned data set, the source cannot be a UNIX directory.

mv does not support the moving to or from GDGs. To use those MVS data sets, user must specify the real data set name.

When moving records, the string ″ \n″ is moved the same way as the string ″\n″: both are read back as ″\n″, where ″\n″ indicates that z/OS C++ will write a record containing a single blank to the file (the default behavior of z/OS C/C++). All other blanks in your output are read back as blanks, and any empty (zero-length) records are ignored on input. However, if the environment variable **_EDC_ZERO_RECLEN** is set to Y before calling **cp**, an empty record is treated as a single newline and is not ignored. Also, if **_EDC_ZERO_RECLEN** is set to Y, a single newline is written to the file as an empty record, and a single blank will be represented by ″ \n″.

A file can be moved by any user who has write permission to the directory containing the file, unless that directory has its sticky bit turned on. If the file is in a directory whose sticky bit is turned on, only the file owner or a superuser can move the file.

You can move:
- One file to another file in the working directory
- One file to a new file on another directory
- A set of directories and files to another place in your file system
- A UNIX file to an MVS data set
- An MVS data set to a file system
- An MVS data set to an MVS data set

Options

–A Specifies that all suffixes (from the first period till the end of the target) be truncated. **–A** has precedence over **–M** and **–C** options. **–S** will be turned off if **–A** is the last option specified.

–B Specifies that the data to be moved contains binary data. When you specify **–B**, **mv** operates without any consideration for <newline> characters or special characteristics of DBCS data (this type of behavior is typical when moving across a UNIX system). **–B** is mutually exclusive with **–F**, **–X**, and **–T**, i.e., you will get an error if you specify more than one of these options.

–C Specifies truncating the filename(s) to 8 characters to meet the restriction in the MVS data set member.

–F *format*
Specifies if a file is binary or text and for text files, specifies the end-of-line delimeter. Also sets the file format to *format* only if the source is an MVS data set and the target is a UNIX file. Only **cp** sets the file format for UNIX to UNIX operations. For text files, when moving from UNIX to MVS, the end-of-line delimeter will be stripped. When moving from MVS to UNIX, the end-of-line delimeter will be added (Code page IBM-1047 will be used to check for end-of-line delimeters).

–F is mutually exclusive with **–B**, **–X**, **–p**, and **–T**. If you specify one of these options with **–F**, you will get an error. If **–F** is specified more than once, the last **–F** specified will be used.

For *format* you can specify:
not not specified
bin binary data

Or the following text data delimeters:
nl newline
cr carriage return
lf line feed
crlf carriage return followed by line feed
lfcr line feed followed by carriage return
crnl carriage return followed by new line

–f (UNIX to UNIX only)
Does not ask if you want to overwrite an existing destination without write permission; it automatically behaves as if you answered `yes`. If you specify both **–f** and **–i**, **mv** uses the option that appears last on the command line.

–i When moving to a UNIX target, always prompts before overwriting an existing file, but does not overwrite the file if you do not have permission. If you specify both **–f** and **–i**, **mv** uses the option that appears last on the command line.

–M Specifies that some characters of the filename are translated when moving between a UNIX file and a data set member. Characters are translated as follows:

- _ (underscore) in UNIX is translated to @ in MVS DS members and vice versa.
- . (period) in UNIX is translated to # in MVS DS members and vice versa.
- – (dash) in UNIX is translated to $ in MVS DS members and vice versa.

–P params
Specifies the parameters needed to create a sequential data set if one does

not already exist. You can specify the RECFM, LRECL, BLKSIZE, and SPACE in the format CRTL **fopen()** function uses. However, LRECL and BLKSIZE can be used for variable record format only.

`SPACE=(units,(primary,secondary)` where the following values are supported for units:

- any positive integer indicating BLKSIZE
- CYL (mixed case)
- TRK (mixed case)

  For example:

```
SPACE=(500,(100,500)) units, primary, secondary
SPACE=(500,100) units and primary only
```

Note: The CRTL**fopen()** arguments: LRECL specifies the length, in bytes, for fixed-length records and the maximum length for variable-length records. BLKSIZE specifies the maximum length, in bytes, of a physical block of records. RECFM refers to the record format of a data set and SPACE indicates the space attributes for MVS data sets.

–R (UNIX to UNIX only)
Moves a directory and all its contents (files, subdirectories, files in subdirectories, and so on). For example:

```
mv –R dir1 dir2
```

moves the entire contents of **dir1** to **dir2/dir1**. **mv** creates any directories that it needs.

–r (UNIX to UNIX only)
Is identical to **–R**.

–S *d=suffix|a=suffix*

- *d=suffix*

  Removes the specifed suffix from a file.

- *a=suffix*

  Appends the specified suffix to a file.

–S has precedence over **–M** and **–C**. It also turns off the **–A** option (if **–S** is the last specified option).

–T Specifies that the data to be moved contains text data. See “Usage Notes” on page 24 for details on how to treat text data. This option looks for IBM-1047 end-of-line delimeters, and is mutually exclusive with **–F**, **–X**, and **–B**, i.e., you will get an error if you specify more than one of these options.

Note: **–T** is ignored when moving across UNIX file systems.

–U Keeps filenames in uppercase when moving from MVS data set members to UNIX files. The default is to make filenames lowercase.

–v Verbose

–X Specifies that the data to be moved is an executable. Cannot be used in conjunction with **–F**, **–X**, or **–B**.

-Z Specifies that error messages are not to be displayed when setting ACLs on the target. The return code will be zero. **mv** will try to preserve the ACLs, if possible. The ACLs are not preserved if a file system does not support ACLs, or if you are moving files to MVS

Note: If you do not specify **–F|B|T** or **X**, **mv** will first check the format of the MVS data set indicated and then try to determine the type of file.

Automatic conversion and file tag specific options

–Z Suppresses failure when setting the file tag by default or on empty, untagged files. For a description of the default behavior, see "Automatic conversion and file tagging behavior for mv."

–O u | c=*codeset*
Allow automatic conversion on source and target files.

–O u If the target exists and is not empty nor already tagged, **mv** will not change the target's tag in order for the target to be a candidate for automatic conversion.

For new targets and existing, untagged, empty files, this option has no effect and **mv** behaves the same as the default. For a description of the default behavior, see "Automatic conversion and file tagging behavior for mv."

When using **mv** to move from a UNIX file to an MVS data set, if the source is a tagged text file, then it may be a candidate for automatic conversion.

When moving executables from or to MVS, automatic conversion is disabled for both source and target.

–O c=*codeset* For a detailed description of the behavior of this option on **mv**, see "Automatic conversion and file tagging behavior for mv."

To prevent the corruption of text files, **mv** will fail if it cannot set the file tag to text or *codeset*.

Attention: If automatic conversion is not properly set or if the source is not properly tagged, the target may end up with a tag codeset that does not match the file contents.

Automatic conversion and file tagging behavior for mv

The following tables describe the behavior of file tagging and automatic conversion for various source and target scenarios depending on whether the **–O** option is specified on the **mv** command.

Table 7. Automatic conversion and file tagging behavior: ***Moving UNIX files to UNIX files***

	Default (without -O option)			
	If the target file system supports setting file tags...	**If the target file system does not support setting file tags (e.g. NFS)...**	**With -O u option**	**With -O c=*codeset* option**
File tagging	Target file is tagged the same as the source file.	• An existing target's tag is unchanged. • A new target is created with a tag according to the file system's attributes (MOUNT parm can specify TAG).	Target's tag is unchanged. **Note:** The source or target file is a candidate for automatic conversion when its txtflag is tagged TEXT.	Target's txtflag is set to TEXT and its codeset is set to *codeset*.
Automatic conversion	Disabled for source and target files	Allowed for source and target files		

Table 8. Automatic conversion and file tagging behavior: ***Moving MVS data sets to UNIX files***

	Default (without -O option)			
	If the target file system supports setting file tags...	**If the target file system does not support setting file tags (e.g. NFS)...**	**With -O u option**	**With -O c=*codeset* option**
If the SOURCE is text:				
File tagging	Target is set to UNTAG	• An existing target's tag is unchanged. • A new target is created with a tag according to the file system's attributes (MOUNT parm can specify TAG).	Target's tag is unchanged	Target's txtflag is set to TEXT and its codeset is set to *codeset*.
Automatic conversion	Disabled for target file	Allowed for target file **Note:** The target file is a candidate for automatic conversion when its txtflag is tagged TEXT.		
If the SOURCE is binary or executable:				
File tagging	Target is set to UNTAG		Target's tag is unchanged	Target's txtflag is set to BINARY and its codeset is set to *codeset*.

*Table 8. Automatic conversion and file tagging behavior: **Moving MVS data sets to UNIX files** (continued)*

	Default (without -O option)			
	If the target file system supports setting file tags...	**If the target file system does not support setting file tags (e.g. NFS)...**	**With -O u option**	**With -O c=*codeset* option**
Automatic conversion	Disabled for target file			

*Table 9. Automatic conversion and file tagging behavior: **Moving UNIX files to MVS data sets***

	Default (without -O option)	**With -O u option**	**With -O c=*codeset* option**
If the SOURCE is text or binary:			
File tagging	Not applicable for target data set		
Automatic conversion	Disabled for source file	Allowed for source file **Note:** The source file is a candidate for automatic conversion when its txtflag is tagged TEXT.	Disabled for source file
If the SOURCE is executable:			
File tagging	Not applicable for target data set		
Automatic conversion	Disabled for source file		

Limits & Requirements

General Requirements

1. To specify an MVS data set name, precede the name with double slashes (//). For example, to specify the fully qualified data set names 'turbo.gammalib' and 'turbo.gammalib(pgm1)', you write:

   ```
   "//'turbo.gammalib'"
   "//'turbo.gammalib(pgm1)'"
   ```

   The same goes for data set names that are not fully qualified:

   ```
   //turbo
   ```

2. For PDS (partitioned data set) or PDSE (partitioned data set extended), to avoid parsing by the shell, the name should be quoted or minimally, the parenthesis should be escaped. For example, to specify 'turbo(pgm1)', you can use quotes:

   ```
   "//turbo(pgm1)"
   ```

   or escape the parenthesis:

   ```
   //turbo\(pgm1\)
   ```

   As indicated above, a fully qualified name must be single-quoted (as is done within TSO). To prevent the single quotes from being interpreted by the shell, they must be escaped or the name must be placed within regular quotation marks. See the 'turbo.gammalib' examples above.

3. If you specify a UNIX file as source and the MVS data set (target) does not exist, a sequential data set will be created. If the partitioned data set exists, the UNIX file will be moved to the partitioned data set member.
4. If source is an MVS data set and target is a UNIX directory, the UNIX directory must exist.
5. You cannot have a UNIX directory, partitioned data set, or sequential data set as source if the target is a partitioned data set.
6. To move all members from a partitioned data set, you may specify the partitioned data set as source and a UNIX directory as target.

MVS data set naming limitations

- Data set names may only contain uppercase alphabetic characters (A-Z). Lowercase characters will be converted to uppercase during any moves to MVS data sets.
- Data set names can contain numeric characters 0–9 and special characters @, #, and $.
- Data set names cannot begin with a numeric character.
- A data set member name cannot be more than 8 characters. If a filename is longer than 8 characters or uses characters that are not allowed in an MVS data set name, the file is not moved. You may use the **–C** option to truncate names to 8 characters.

Limitations: UNIX to MVS data set

1. If you specify a sequential MVS data set that is in undefined record format, the file is moved as binary.
2. If you specify a PDSE that is in undefined record format, the first file successfully moved determines in what format files will be moved. Note that PDSE does not allow mixture. So if the first successfully moved file is an executable, the PDSE will have program objects only and all other files will fail. On the other hand, if the first file is data, then all files are moved as binary.
3. If you specify a PDS that is in undefined record format, UNIX executables are saved as PDS load modules. All other files are moved as binary.
4. If you specify an MVS data set that is either in variable length or fixed record length and you have not set the file format, text files are moved as text, binaries as binary, and executables as binary. (IBM-1047 end-of-line delimeters are detected in the data)
5. If you set the file format, the set value is used to determine if data is binary or text.

Limitations: MVS data set to UNIX

1. If an HFS file does not exist, one is created using 666 mode value:

   ```
   666 mode value:  owner(rw-)  group(rw-)  other(rw-)
   ```

   whether data is binary or text. If the data to be moved is a shell script or executable, an HFS file is created using 777 mode value:

   ```
   777 mode value:  owner(rwx)  group(rwx)  other(rwx)
   ```

2. If an HFS exists and the file format is set, **mv** moves the file as that format. Otherwise,
   - load modules (PDS) are stored as UNIX executables and program objects (PDSE) are moved since they are the same as executables;
   - data within data sets of undefined record format are moved as binary if the data is not a program object or load module;

- and data found within data sets of fixed length or variable record length are moved as text. (IBM-1047 end-of-line delimeters are detected in the data)

Limitations: MVS to MVS

1. Options **–A**, **–C**, **–f**, and **–S** are ignored.
2. If target and source are in undefined record format (and neither is a sequential data set), **mv** will attempt to move the data as a load module. If that fails, then **mv** will move the data as binary.
3. If target and source are in undefined record format and either is a sequential data set, **mv** moves the data as binary.
4. If the source has a fixed or variable record length and the target is in undefined record format, **mv** moves the data as binary.
5. If the source is in undefined record format and the target has a fixed or variable record length, **mv** moves the data as binary.
6. If both source and target are in fixed or variable record length, **mv** moves the data as text.

Limitations: Moving executables into a PDS

1. A PDS may not store load modules that incorporate program management features.
2. **c89**, by default, produces objects using the highest level of program management.
3. If you plan on moving a load module to a PDS, you may use a pre-linker which produces output compatible with linkage editor. Linkage editor generated output can always be stored in a PDS.

The following table is a quick reference for determining the correct use of options with **mv**.

*Table 10. **mv** Format: File to File and File ... (multiple files) to Directory*

Source/Target	Options Allowed	Options Ignored	Options Failed
UNIX File/UNIX File	Ffi	ABCMPSTUX	
UNIX File/Sequential Data Set	BFiPT	ACfMSU	X
UNIX File/PDS or PDSE Member	BFiTX	ACfMPSU	
Sequential Data Set/UNIX File	BFfiTU	ACMPS	X
Sequential Data Set/Sequential Data Set	BFiPT	ACfMSU	X
Sequential Data Set/PDS or PDSE Member	BFiT	ACfMPSU	X
PDS or PDSE Member/UNIX File	BFfiTUX	ACMPS	
PDS or PDSE Member/Sequential Data Set	BFiPT	ACfMSU	X
PDS or PDSE Member/PDS or PDSE Member	BFiTX	ACfMPSU	
UNIX File/UNIX Directory	Fi	ABCFMPSTUX	

*Table 10. **mv** Format: File to File and File ... (multiple files) to Directory (continued)*

Source/Target	Options Allowed	Options Ignored	Options Failed
PDSE or PDSE Member/UNIX Directory	BFfiMSTUX	ACP	
UNIX File/Partitioned Data Set	ABCFiMSTX	fPU	
PDS or PDSE Member/PartitionedData Set	BFiTX	ACfMPSU	
UNIX Directory/UNIX Directory	fi	ABCFMPSTUX	
Partitioned Data Set/UNIX Directory	ABCFfiMSTUX	P	

The tables that follow indicate the kind of moves allowed using **mv**.

*Table 11. **mv** Format: File to File*

Source	Target	Allowed
UNIX File, Sequential Data Set, or Partitioned Data Set Member	UNIX File, Sequential Data Set, or Partitioned Data Set Member	Yes
UNIX Directory (dir)	UNIX Directory (dir2 exists)	YES (NOTE: Results will be found in dir2/dir1/ ..).
UNIX Directory (dir)	UNIX Directory (dir2 does not exist)	YES (NOTE: Results will be found in dir2/...).
Partitioned Data Set	UNIX Directory (dir) NOTE: results in each member of data set are moved to dir.	Yes
UNIX Directory	Partitioned Data Set	No
Partitioned Data Set	Partitioned Data Set	No
UNIX File, UNIX Directory, or Partitioned Data Set Member	UNIX Directory	Yes
Partitioned Data Set Member	Partitioned Data Set (must exist)	Yes

*Table 12. **mv** Format: File... (multiple files) to Directory*

Source	Target	Allowed
Any combination of UNIX File and/or Partitioned Data Set Member	UNIX Directory or Partitioned Data Set	Yes
Any combination of UNIX Directory, Partitioned Data Set, Sequential Data Set	Partitioned Data Set	No
Sequential Data Set	UNIX Directory	No
Any combination of UNIX Directory, UNIX File, Partitioned Data Set, Partitioned Data Set Member	UNIX Directory	Yes

Usage Notes

UNIX to MVS

1. To move from UNIX to a partitioned data set, you must allocate the data set before doing the **mv**.
2. If an MVS data set does not exist, **mv** will allocate a new sequential data set of variable record format.
3. For text files, all <newline> characters are stripped during the move. Each line in the file ending with a <newline> character is moved into a record of the MVS data set. If text file format is specified or already exists for the source file, that file format will be used for end-of-line delimeter instead of <newline>. Note that **mv** looks for IBM-1047 end-of-line delimeters in data.

   You cannot move a text file to an MVS data set that has an undefined record format:

   - For an MVS data set in fixed record format, any line moved longer than the record size will cause **mv** to fail with a displayed error message and error code. If the line is shorter than the record size, the record is padded with blanks.
   - For an MVS data set in variable record format: Any line moved longer than the largest record size will cause **mv** to fail with a displayed error message and error code. Record length is set to the length of the line.
4. For binary files, all moved data is preserved:
   - For an MVS data set in fixed record format, data is cut into chunks of size equal to the record length. Each chunk is put into one record. The last record is padded with blanks.
   - For an MVS data set in variable record format, data is cut into chunks of size equal to the largest record length. Each chunk is put into one record. The length of the last record is equal to length of the data left.
   - For an MVS data set in undefined record format, data is cut into chunks of size equal to the block size. Each chunk is put into one record. The length of the last record is equal to the length of the data left.
5. For load modules, the partitioned data set specified must be in undefined record format otherwise the executable will not be moved.
6. If more than one filename is the same, the file is overwritten on each subsequent move.
7. If a UNIX filename contains characters that are not allowed in an MVS data set, it will not be moved. If the UNIX filename has more than 8 characters, it can not be moved to an MVS data set member. (See the **–ACMS** options for converting filenames)
8. You are not allowed to move files into data sets with spanned records.
9. PDSE cannot have a mixture of program objects and data members. PDS allows mixing, but it is not recommended.
10. Special files such as character special, external links, and fifo will not be moved to an MVS data set.
11. If a file is a symbolic link, **mv** will move the resolved file, not the link itself.
12. UNIX file attributes are lost when moving to MVS. If you wish to preserve file attributes, you should use the **pax** utility.

MVS to UNIX

1. If the specified target HFS file exists, the new data overwrites the existing data.

2. If the specified HFS file does not exist, it will be created using a 666 mode value whether the data is binary or text (this is subject to **umask**). If the data to be moved is a shell script or executable, the HFS file will be created with a 777 mode value (also subject to **umask**).
3. When you move MVS data sets to UNIX text files, a <newline> character is appended to the end of each record. If trailing blanks exist in the record, the <newline> character is appended after the trailing blanks. If the file format option is specified or the target file has the file format set, that file format is used as the end-of-line delimeter instead of <newline>.
4. When you move MVS data sets to UNIX binary files, the <newline> character is not appended to the record.
5. You cannot use **mv** to move data sets with spanned record lengths.

Examples

1. To specify **–P** *params* for a non-existing sequential target:

   ```
   mv -P "RECFM=U,space=(500,100)"file "//'turbo.gammalib'"
   ```

2. To move file **f1** to a fully qualified sequential data set `'turbo.gammalib'` and treat it as a binary:

   ```
   mv -F bin f1 "//'turbo.gammalib'"
   ```

3. To move all members from a fully qualified PDS `'turbo.gammalib'` to an existing UNIX directory `dir`:

   ```
   mv "//turbo.gammalib'" dir
   ```

4. To drop .c suffixes before moving all files in UNIX directory `dir` to an existing PDS `'turbo.gammalib'`:

   ```
   mv -S d=.c dir/* "//'turbo.gammalib'"
   ```

Environment Variable

mv uses the following environment variable when moving records to or from MVS data sets:

_EDC_ZERO_RECLEN
If set to Y before calling **mv**, an empty record is treated as a single newline and is not ignored. Also, a single newline is written to the file as an empty record, and a single blank will be represented by ” \n”. If you do not set this environment variable when moving records, then the string ” \n” is moved the same way as the string ”\n”: both are read and written as ”\n”, where ”\n” indicates that z/OS C/C++ will write a record containing a single blank to the file (the default behavior of z/OS C/C++). All other blanks in the output are read back as blanks, and any empty (zero-length) records are ignored on input.

Localization

mv uses the following localization environment variables:

- **LANG**
- **LC_ALL**
- **LC_COLLATE**
- **LC_CTYPE**
- **LC_MESSAGES**
- **LC_SYNTAX**
- **NLSPATH**

Exit Values

0 Successful completion

1 Failure due to any of the following:
- The argument had a trailing / but was not a directory
- Inability to find file
- Inability to open input file for reading
- Inability to create or open output file for output
- Read error on an input file
- Write error on an output file
- Input and output files identical
- Inability to unlink input file
- Inability to rename input file
- Irrecoverable error when using the **–r** option, such as:
  - Inability to access a file
  - Inability to read a directory
  - Inability to remove a directory
  - Inability to create a directory
  - A target that is not a directory
  - Source and destination directories identical

2 Failure due to any of the following:
- Incorrect command-line option
- Too few arguments on the command line
- A target that should be a directory but isn't
- No space left on target device
- Out of memory to hold the data to be moved
- Inability to create a directory to hold a target file

Messages

Possible error messages include:

cannot allocate target string
mv has no space to hold the name of the target file. Try to free some memory to give **mv** more space.

filename?
You are attempting to move a file, but there is already a file with the target name and the file is read-only. If you really want to write over the existing file, type `y` and press <Enter>. If you do not want to write over the existing file, type `n` and press <Enter>.

source *name* **and target** *name* **are identical**
The source and the target are actually the same file (for example, because of links). In this case, **mv** does nothing.

unreadable directory *name*
mv cannot read the specified directory—for example, because you do not have appropriate permissions.

Portability

POSIX.2, X/Open Portability Guide, UNIX systems.

The **–R** and **–r** options are extensions of the POSIX standard.

Related Information

cp, **cpio**, **rm**

Notices

This information was developed for products and services offered in the U.S.A.

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 publication documents intended Programming Interfaces that allow the customer to write programs that use z/OS UNIX System Services (z/OS UNIX).

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

IBM
ibm.com
MVS
z/OS

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.

Acknowledgments

InterOpen Shell and Utilities is a source code product providing POSIX.2 (Shell and Utilities) functions to the z/OS UNIX services offered with MVS. InterOpen/POSIX Shell and Utilities is developed and licensed by Mortice Kern Systems (MKS) Inc. of Waterloo, Ontario, Canada.

Program Number: 5694-A01, 5655-G52

Printed in USA

SA22-7802-06