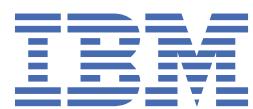


z/VM
7.4

*OpenExtensions POSIX Conformance
Document*



Note:

Before you use this information and the product it supports, read the information in “[Notices](#)” on page [73](#).

This edition applies to version 7, release 4 of IBM® z/VM® (product number 5741-A09) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2024-09-18

© Copyright International Business Machines Corporation 1993, 2024.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Tables.....	ix
About This Document.....	xi
Intended Audience.....	xi
Conventions Used in This Document.....	xi
Where to Find More Information.....	xi
Links to Other Documents and Websites.....	xi
How to provide feedback to IBM.....	xiii
Summary of Changes for z/VM: OpenExtensions POSIX Conformance Document... 	xv
GC24-6298-74, z/VM 7.4 (September 2024).....	xv
GC24-6298-73, z/VM 7.3 (December 2023).....	xv
GC24-6298-73, z/VM 7.3 (September 2022).....	xv
GC24-6298-01, z/VM 7.2 (September 2020).....	xv
Part 1. POSIX.1 Conformance Document.....	1
Section 1. General.....	3
1.3 Conformance.....	3
1.3.1 Implementation Conformance.....	3
1.3.3 Language-Dependent Services for the C Programming Language.....	3
Section 2. Terminology and General Requirements.....	5
2.2 Definitions.....	5
2.2.2 General Terms.....	5
2.3 General Concepts.....	6
2.3.1 extended security controls.....	6
2.3.2 file access permissions.....	6
2.4 Error Numbers.....	7
2.5 Primitive System Data Types.....	7
2.6 Environment Description.....	8
2.8 Numerical Limits.....	8
2.8.3 Run-Time Increasable Values.....	8
2.8.4 Run-Time Invariant Values.....	8
2.8.5 Pathname Variable Values.....	8
2.9 Symbolic Constants.....	9
2.9.3 Compile-Time Symbolic Constants for Portability Specifications.....	9
2.9.4 Execution-Time Constants for Portability Specifications.....	9
Section 3. Process Primitives.....	11
3.1 Process Creation and Execution.....	11
3.1.1 Process Creation.....	11
3.1.2 Execute a File.....	11
3.2 Process Termination.....	12
3.2.1 Wait for Process Termination.....	12
3.2.2 Terminate a Process.....	12
3.3 Signals.....	12
3.3.1 Signal Concepts.....	13
3.3.2 Send a Signal to a Process.....	13

3.3.3 Manipulate Signal Sets.....	13
3.3.4 Examine and Change Signal Action.....	13
3.3.6 Examine Pending Signals.....	14
3.4 Timer Operations.....	14
3.4.3 Delay Process Execution.....	14
 Section 4. Process Environment.....	15
4.2 User Identification.....	15
4.2.3 Get Supplementary Group IDs.....	15
4.2.4 Get User Name.....	15
4.4 System Identification.....	15
4.4.1 Get System Name.....	15
4.6 Environment Variables.....	16
4.6.1 Environment Access.....	16
4.7 Terminal Identification.....	16
4.7.1 Generate Terminal Pathname.....	17
4.7.2 Determine Terminal Device Name.....	17
4.8 Configurable System Variables.....	17
4.8.1 Get Configurable System Variables.....	17
 Section 5. Files and Directories.....	19
5.1 Directories.....	19
5.1.2 Directory Operations.....	19
5.2 Working Directory.....	19
5.2.2 Get Working Directory Pathname.....	19
5.3 General File Creation.....	20
5.3.1 Open a File.....	20
5.3.3 Set File Creation Mask.....	20
5.3.4 Link to a File.....	20
5.4 Special File Creation.....	20
5.4.1 Make a Directory.....	20
5.4.2 Make a FIFO Special File.....	21
5.5 File Removal.....	21
5.5.1 Remove Directory Entries.....	21
5.5.2 Remove a Directory.....	21
5.5.3 Rename a File.....	21
5.6 File Characteristics.....	22
5.6.2 Get File Status.....	22
5.6.3 Check File Accessibility.....	22
5.6.4 Change File Modes.....	22
5.6.5 Change Owner and Group of a File.....	22
5.7 Configurable Pathname Variables.....	22
5.7.1 Get Configurable Pathname Variables.....	22
 Section 6. Input and Output Primitives.....	25
6.3 File Descriptor Deassignment.....	25
6.3.1 Close a File.....	25
6.4 Input and Output.....	25
6.4.1 Read from a File.....	25
6.4.2 Write to a File.....	25
6.5 Control Operations on Files.....	25
6.5.2 File Control.....	26
6.5.3 Reposition Read/Write File Offset.....	26
 Section 7. Device- and Class-Specific Functions.....	27
7.1 General Terminal Interface.....	27
7.1.1 Interface Characteristics.....	27
7.1.2 Parameters That Can Be Set.....	28

7.1.3 Baud Rate Functions.....	29
7.2 General Terminal Interface Control Functions.....	29
7.2.1 Get and Set State.....	29
7.2.2 Line Control Functions.....	30
 Section 8. Language-Specific Services for the C Programming Language.....	31
8.1 Referenced C Language Routines.....	31
8.1.1 Extensions to Time Functions.....	31
8.1.2 Extensions to setlocale() Function.....	31
8.2 C Language Input/Output Functions.....	32
8.2.1 Map a Stream Pointer to a File Descriptor.....	32
8.2.2 Open a Stream on a File Descriptor.....	32
8.2.3 Interactions of Other FILE-Type C Functions.....	32
8.3 Other C Language Functions.....	32
8.3.2 Set Time Zone.....	32
 Section 9. System Databases.....	33
9.1 System Databases.....	33
9.2 Database Access.....	33
9.2.1 Group Database Access.....	33
9.2.2 User Database Access.....	34
 Section 10. Data Interchange Format.....	35
10.1 Archive/Interchange File Format.....	35
10.1.1 Extended tar Format.....	35
10.1.2 Extended cpio Format.....	35
10.1.3 Multiple Volumes.....	36
 Part 2. POSIX.2 Conformance Document.....	37
 Section 1. General.....	39
1.3 Conformance.....	39
1.3.1 Implementation Conformance.....	39
 Section 2. Terminology and General Requirements.....	41
2.2 Definitions.....	41
2.2.2 General Terms.....	41
2.4 Character Set.....	42
2.4.1 Character Set Description File.....	42
2.5 Locale.....	42
2.6 Environment Variables.....	43
2.9 Dependencies on Other Standards.....	43
2.9.1 Features Inherited from POSIX.1.....	43
2.11 Utility Description Defaults.....	43
2.11.5 External Influences.....	43
2.13 Configuration Values.....	44
2.13.1 Symbolic Limits.....	44
2.13.2 Symbolic Constants for Portability Specifications.....	45
2.14 Terminal Characteristics.....	45
 Section 3. Shell Command Language.....	47
3.5 Parameters and Variables.....	47
3.5.3 Variables.....	47
3.6 Word Expansions.....	47
3.7 Redirection.....	47
 Section 4. Execution Environment Utilities.....	49

4.1 awk – Pattern Scanning and Processing Language.....	49
4.1.7 Extended Description.....	49
4.2 basename – Return Nondirectory Portion of Pathname.....	49
4.2.2 Description.....	49
4.5 cd – Change Working Directory.....	49
4.5.2 Description.....	49
4.5.4 Operands.....	49
4.7 chmod – Change File Modes.....	49
4.7.2 Description.....	50
4.7.7 Extended Description.....	50
4.13 cp – Copy Files.....	50
4.13.2 Description.....	50
4.13.3 Options.....	51
4.18 dirname – Return Directory Portion of Pathname.....	51
4.18.2 Description.....	51
4.19 echo – Write Arguments to Standard Output.....	51
4.19.4 Operands.....	51
4.20 ed – Edit Text.....	52
4.20.7 Extended Description.....	52
4.24 find – Find Files.....	52
4.24.4 Operands.....	52
4.33 ln – Link Files.....	52
4.33.2 Description.....	52
4.33.4 Operands.....	52
4.34 locale – Get Locale-Specific Information.....	52
4.34.3 Options.....	52
4.34.4 Operands.....	53
4.35 localedef – Define Locale Environment.....	53
4.35.2 Description.....	53
4.35.3 Options.....	53
4.35.4 Operands.....	53
4.35.9 Consequences of Errors.....	53
4.36 logger – Log Messages.....	53
4.36.2 Description.....	53
4.39 ls – List Directory Contents.....	54
4.39.3 Options.....	54
4.39.5 External Influences.....	54
4.39.6 External Effects.....	54
4.40 mailx – Process Messages.....	54
4.40.4 Operands.....	54
4.40.6 External Effects.....	55
4.40.7 Extended Description.....	55
4.43 mv – Move Files.....	55
4.43.2 Description.....	55
4.45 od – Dump Files in Various Formats.....	55
4.45.7 Extended Description.....	55
4.48 pax – Portable Archive Interchange.....	56
4.48.2 Description.....	56
4.48.3 Options.....	57
4.48.5 External Influences.....	57
4.48.6 External Effects.....	57
4.55 sed – Stream Editor.....	58
4.55.7 Extended Description.....	58
4.56 sh – Shell, the Standard Command Language Interpreter.....	58
4.59 stty – Set the Options for a Terminal.....	58
4.59.2 Description.....	58
4.59.4 Operands.....	58
4.62 test – Evaluate Expression.....	59

4.62.4 Operands.....	59
4.63 touch – Change File Access and Modification Times.....	59
4.63.3 Options.....	59
4.64 tr – Translate Characters.....	59
4.64.7 Extended Description.....	59
4.68 uname – Return System Name.....	59
4.68.2 Description.....	60
4.68.6 External Effects.....	60
Section 5. User Portability Utilities Option.....	61
5.12 fc – Process Command History List.....	61
5.12.2 Description.....	61
5.12.5 External Influences.....	61
5.19 newgrp – Change to a New Group.....	61
5.23 ps – Report Process Status.....	61
5.23.2 Description.....	62
5.23.3 Options.....	62
5.23.6 External Effects.....	62
Section 6. Software Development Utilities Option.....	63
6.2 make – Maintain, Update, and Regenerate Groups of Programs.....	63
6.2.7. Extended Description.....	63
Annex A. C Language Development Utilities Option.....	65
A.1 c89 – Compile Standard C Programs.....	65
A.1.2 Description.....	65
A.1.3 Options.....	65
A.1.4 Operands.....	66
A.1.5 External Influences.....	66
A.1.6 External Effects.....	66
A.1.7 Extended Description.....	67
A.2 lex – Generate Programs for Lexical Tasks.....	68
A.2.6 External Effects.....	68
A.2.7 Extended Description.....	69
A.3 yacc – Yet Another Compiler Compiler.....	69
A.3.6 External Effects.....	69
A.3.7 Extended Description.....	70
Global Issues.....	71
Window Size.....	71
Modes and the Sticky Bit.....	71
Notices.....	73
Trademarks.....	74
Terms and Conditions for Product Documentation.....	74
IBM Online Privacy Statement.....	75
Bibliography.....	77
Where to Get z/VM Information.....	77
z/VM Base Library.....	77
z/VM Facilities and Features.....	78
Prerequisite Products.....	80
Related Products.....	80
Additional Publications.....	80

Tables

1. OpenExtensions Non-POSIX Error Codes.....	7
2. OpenExtensions Non-POSIX Primitive System Data Type.....	7
3. OpenExtensions POSIX Primitive System Data Types.....	7
4. OpenExtensions Runtime Invariant Values.....	8
5. OpenExtensions Path Name Variable Values.....	8
6. Compile-Time Symbolic Constants for Portability Specifications.....	9
7. Execution-Time Symbolic Constants for Portability Specifications.....	9
8. OpenExtensions Non-POSIX Signals.....	13
9. Formats for OpenExtensions utsname Members.....	15
10. Initial Values for Special Control Characters.....	29
11. Default Values for Required and OpenExtensions-Specific Categories.....	31
12. POSIX.2 Standard 2.13.1: Table 2-18: Symbolic Utility Limits.....	44
13. POSIX.2 Standard 2.13.2: Table 2-19: Optional Facility Configuration Values.....	45

About This Document

This document describes how IBM® z/VM® meets the criteria for a *conforming implementation* as defined in the following Institute of Electrical and Electronics Engineers (IEEE) Portable Operating System Interface for Computer Environments (POSIX) standards:

- International Organization for Standardization and International Electrotechnical Commission (ISO/IEC) 9945-1: 1990 (IEE Std. 1003.1-1990), hereafter referred to as **POSIX.1**
- International Organization for Standardization and International Electrotechnical Commission (ISO/IEC) 9945-2: 1992 (IEE Std. 1003.2-1992), hereafter referred to as **POSIX.2**

This document describes the z/VM implementation of those areas of the POSIX.1 and POSIX.2 standards that were declared to be *optional*, or *implementation-defined*.

The implementation of these POSIX standards in z/VM is known as OpenExtensions™, and is included in the Conversational Monitor System (CMS).

This document also describes the symbols and values in the files <**limits.h**> and <**unistd.h**>.

Intended Audience

This information is intended to help technical personnel evaluate how the POSIX.1 and POSIX.2 standards are implemented in z/VM.

Conventions Used in This Document

As an aid to cross-referencing with the POSIX.1 and POSIX.2 standards (and as required by those standards), the section and topic numbers used in each part of this document correspond to the same sections and topics in the standards. As a further aid to cross-referencing, the terminology used in the topic names in this document is the same as is used in the POSIX standards (for example, "pathname", "filename", and so on). However, in the text explaining the z/VM implementation (except in quotation marks from the POSIX standards), the z/VM convention is used (for example, "path name", "file name", and so on).

Where to Find More Information

More detailed information on the OpenExtensions implementation can be found in the following publications:

- [*z/VM: OpenExtensions User's Guide*](#)
- [*z/VM: OpenExtensions Callable Services Reference*](#)
- [*z/VM: OpenExtensions Commands Reference*](#)
- [*z/VM: OpenExtensions Advanced Application Programming Tools*](#)

For the complete list of books in the z/VM library, see the ["Bibliography"](#) on page 77.

Links to Other Documents and Websites

The PDF version of this document contains links to other documents and websites. A link from this document to another document works only when both documents are in the same directory or database, and a link to a website works only if you have access to the Internet. A document link is to a specific edition. If a new edition of a linked document has been published since the publication of this document, the linked document might not be the latest edition.

How to provide feedback to IBM

We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information. See [How to send feedback to IBM](#) for additional information.

Summary of Changes for z/VM: OpenExtensions POSIX Conformance Document

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line (|) to the left of the change.

GC24-6298-74, z/VM 7.4 (September 2024)

This edition supports the general availability of z/VM 7.4. Note that the publication number suffix (-74) indicates the z/VM release to which this edition applies.

GC24-6298-73, z/VM 7.3 (December 2023)

This edition includes terminology, maintenance, and editorial changes.

GC24-6298-73, z/VM 7.3 (September 2022)

This edition supports the general availability of z/VM 7.3. Note that the publication number suffix (-73) indicates the z/VM release to which this edition applies.

GC24-6298-01, z/VM 7.2 (September 2020)

This edition supports the general availability of z/VM 7.2.

Part 1. POSIX.1 Conformance Document

Section 1. General

1.3 Conformance

1.3.1 Implementation Conformance

1.3.1.1 Requirements

For information about how to start an interactive session, see “[2.2.2.46 login](#)” on page 5. From an interactive session, a user can run applications with the behavior specified by POSIX.1.

In addition to using the interactive environment, a C program behaves in a POSIX.1-conforming fashion when the following conditions are met:

- The program does not use z/VM services that are not part of a POSIX.1 or ANSI C library call. In other words, the program uses only POSIX.1-conforming functions.
- The program receives control through one of the exec family of C function calls.
- The environment variables **LOGNAME** and **HOME**, which are required by Federal Information Processing Standard (FIPS) 151-1, must be defined and passed to the exec call.
- Prior to the exec call to the program, STDIN, STDOUT, and STDERR are opened for either file **/dev/tty** or some other file system file.

The following meet the conditions required to give control to a C program to guarantee that it behave in a POSIX.1-conforming fashion:

- OPENVM SHELL command and the shell
- OPENVM RUN command

1.3.1.2 Documentation

OpenExtensions conforms to the IEEE Std 1003.1-1990 and ISO/IEC 9945-1:1990(E), hereafter referred to as POSIX.1 in this document, except that the OpenExtensions implementation of the `fork()` function does not meet all POSIX.1 specifications. Any extensions beyond POSIX.1 are described in other OpenExtensions books.

The OpenExtensions implementation supports the POSIX.1 standard and the FIPS 151-1 restrictions on that standard.

This report is published to satisfy the requirement of the POSIX.1 standard for a conformance document, as outlined in section 1.3.1.2 *Documentation* of the POSIX.1 standard.

This document has the same structure as the POSIX.1 standard. It lists all section numbers and titles as they appear in the POSIX.1 standard and in the same order. (It omits empty sections.) It also uses typographical conventions similar to those used in the POSIX.1 standard. This document supplements, but does not replace, the POSIX.1 standard.

1.3.3 Language-Dependent Services for the C Programming Language

1.3.3.2 C Standard Language-Dependent System Support

The OpenExtensions implementation conforms to ISO/IEC 9899:1990(E) C, as well as to the IEEE Std 1003.1-1990 C Language Binding (C Standard Language-Dependent System Support).

Section 2. Terminology and General Requirements

2.2 Definitions

2.2.2 General Terms

2.2.2.4 appropriate privileges

For those functions where IEEE Std 1003.1-1990 specifies "appropriate privilege" is required, *appropriate privilege* is defined as *superuser authority*. A user (that is, the process calling a callable service) has superuser authority if the effective UID of the calling process is 0.

You can assign a UID of 0 to a user by specifying 0 as the UID value on the POSIXINFO statement in the z/VM User Directory. This change to the directory can be done using the IBM Directory Maintenance z/VM product. You can also set an effective UID for a user's process to 0 by executing a setuid file that has an owner UID of 0. Only a superuser can create a setuid file with an owner UID of 0.

2.2.2.9 character special file

The character special files supported are the pseudoterminal files (pseudo-TTYs), the null file (**/dev/null**), and the controlling terminal file (**/dev/tty**).

2.2.2.27 file

The file system supports the file types listed in POSIX.1 except block special files. The file system also supports symbolic link files and external link files.

2.2.2.46 login

A user gains interactive access to the shell by first logging on to a z/VM user ID through existing documented z/VM externals. Once in CMS, a user can run the shell by entering the OPENVM SHELL command from the CMS command line.

2.2.2.55 parent process ID

When a process ends, the parent process ID of the children of the ended process is set to the process ID of the `init` process, which is 1.

2.2.2.57 pathname

The C functions `fopen()`, `freopen()`, `remove()`, and `rename()` interpret names with all of the following to mean that the rest of the name refers to a traditional CMS file on an accessed minidisk or shared file system directory.

- Exactly two leading slashes
- No leading blanks or other characters
- The third character is not a slash

2.2.2.69 read-only file system

A file system specified as read-only when it is mounted. No updates are made or allowed to a read-only file system. Writes are permitted to FIFO special files within a read-only file system, but FIFO data is kept only in memory and no updates are made to the disk.

2.2.2.83 supplementary group ID

An attribute of a process used in determining file access permissions.

A process has up to {NGROUPS_MAX} supplementary group IDs (GIDs).

When maintained by CP (that is, when no external security manager (ESM) exists, or an external security manager defers to CP), the list of supplementary groups normally includes the effective group ID. When a user logs on to OpenExtensions and there is no ESM support, or the ESM defers to CP, {NGROUPS_MAX} OpenExtensions groups listed in the user's CP directory entry, including the initial effective group ID, become associated with the virtual machine. When a new POSIX** application is started, the supplementary GIDs of its process are assigned from the groups that are associated with the virtual machine. When a process is created by `spawn()`, the supplementary GIDs of the child process are the same as the parent process.

2.3 General Concepts

2.3.1 extended security controls

OpenExtensions allows for the extension to the access security mechanisms by providing for the use of an External Security Manager (ESM). In addition, OpenExtensions provides the following access security extensions:

- A POSIXOPT QUERYDB option on the USER_DEFAULTS system configuration file statement allows an installation to define whether or not all users on the system are permitted to query other user's user and group database information. The system default can be overridden for users on an individual basis using the POSIXOPT directory control statement. This authorization applies to the `getpwuid()`, `getpwnam()`, `getgrgid()` and `getgrnam()` functions.
- For any one of the exec family of functions, if the executable file has the set-user-id mode bit set and the effective UID of the process is not the same as the file owner, or if the executable file has the set-group-id mode bit set and the effective GID is not the same as the file group, the caller must have CP authorization to execute these files. This authorization is defined on a system-wide basis in the system configuration file or on an individual user basis in the CP directory. It specifies whether or not a user with the appropriate file access permissions is allowed to execute a file that will result in a change of POSIX IDs.

In addition, the file server on which the object file resides must have CP authorization to specify that the POSIX IDs for the caller be changed. This permission is also defined in the CP directory. If either of these security checks fails, the existing process will be terminated, and the new file will not be invoked.

`setuid()`, `setgid()`, `seteuid()`, and `setegid()` allow a process with appropriate privileges to change its IDs to a value that is not currently defined in the CP directory. An ESM can be used to provide additional authorization and validity checks. If an ESM indicates that the input UID or GID value is undefined, each of these functions returns -1 and sets `errno` to [EINVAL]. These functions change the security environment that is checked by POSIX (OpenExtensions) functions. They do not affect the security environment checked by other z/VM services.

If a user has no UID defined in the CP directory entry or the ESM user database, the default value 4,294,967,295 (X'FFFFFFF') is assigned. If a user has no GID defined in the CP directory entry or the ESM user database, the default value 4,294,967,295 (X'FFFFFFF') is assigned.

`chown()` checks the input UID and GID value. `chown()` can set a file's owner UID or GID to a value that is not currently defined in the user database. A file with an undefined UID or GID can also result from deleting a user or group that still owns files.

2.3.2 file access permissions

Appropriate privilege is defined as having superuser authority (see “[2.2.2.4 appropriate privileges](#)” on page 5). No alternate access control mechanisms are provided.

2.4 Error Numbers

The OpenExtensions-defined values for the POSIX-defined error numbers are defined in `<errno.h>`. Table 1 on page 7 shows the codes that are provided in addition to those defined by POSIX:

Table 1. OpenExtensions Non-POSIX Error Codes

Error Code	Description
EBUflen	The buffer is not long enough for the path name.
ECMSBADCHAR	There is an incorrect character in the environment variable name.
ECMSERR	A CMS environment or internal error occurred.
ECMSINITIAL	A process initialization error occurred.
ECMSLOCK	A Token Manager locking error occurred.
ECMSNORTL	Access to the OpenExtensions version of the C run-time library is denied.
ECMSPARM	Incorrect parameters were passed to the service.
ECMSPFSFILE	The byte file system encountered a permanent file error.
ECMSPFSPERM	The byte file system encountered a system error.
ECMSSTORAGE	A storage management error occurred.
ECPERR	A security authorization facility error occurred.
EEXTLINK	The file being referenced is an external link.
ENODD	There is no pathdef for the ddname in effect.

In the OpenExtensions implementation, the [EFBIG] error occurs when the size of a file exceeds the maximum file size of 17,592,186,044,416 (2^{44}) bytes.

2.5 Primitive System Data Types

In addition to the primitive system data types specified by POSIX.1, the OpenExtensions implementation supports the non-POSIX data type whose name ends with `_t`, shown in Table 2 on page 7.

Table 2. OpenExtensions Non-POSIX Primitive System Data Type

Defined Type	Header	Description
<code>mtm_t</code>	<code><sys/types.h></code>	Mount mode

In addition to those primitive system data types specified by POSIX.1 to be defined in `<sys/types.h>`, the OpenExtensions implementation defines the POSIX.1 or ANSI C data types shown in Table 3 on page 7.

Table 3. OpenExtensions POSIX Primitive System Data Types

Defined Type	Header	Description
<code>cc_t</code>	<code><sys/types.h></code>	Control character
<code>clock_t</code>	<code><sys/types.h></code>	Number of clock ticks
<code>sigset_t</code>	<code><sys/types.h></code>	A set of signals
<code>speed_t</code>	<code><sys/types.h></code>	Baud rate
<code>tcflag_t</code>	<code><sys/types.h></code>	Terminal control flags

Table 3. OpenExtensions POSIX Primitive System Data Types (continued)

Defined Type	Header	Description
time_t	<sys/types.h>	Time since the Epoch

2.6 Environment Description

The OpenExtensions implementation permits all strings composed of 8-bit characters (except for the "=" character) to be used in environment variable names. If the "=" character is found in an environment variable name, the `getenv()` function returns a NULL value, and `errno` is set to [ECMSBADCHAR].

2.8 Numerical Limits

The following subsections list magnitude limitations imposed by the OpenExtensions implementation.

2.8.3 Run-Time Increasable Values

The default maximum number of simultaneous supplementary group IDs per process value is:

```
NGROUPS_MAX 32
```

An ESM is permitted to specify a number between 32 and 125 (inclusive) as the system's `NGROUPS_MAX` value. If no POSIX-capable ESM is installed, the value of `NGROUPS_MAX` is 32.

2.8.4 Run-Time Invariant Values

Table 4 on page 8 shows extended values that are available in the OpenExtensions implementation with the `sysconf()` function. They are not defined in <limits.h>.

Table 4. OpenExtensions Runtime Invariant Values

Name	Description
ARG_MAX	Maximum argument and environment length
CHILD_MAX	Maximum number of simultaneous processes per z/VM user
OPEN_MAX	Maximum number of simultaneous open files
STREAM_MAX	Maximum number of streams that a single process can have open at the same time
TZNAME_MAX	Maximum number of bytes supported for the name of a time zone

2.8.5 Pathname Variable Values

Table 5 on page 8 shows extended values that are available in the OpenExtensions implementation with the `pathconf()` function. They are not defined in <limits.h>.

Table 5. OpenExtensions Path Name Variable Values

Name	Description
LINK_MAX	Maximum value of the file link count
MAX_CANON	Maximum unprocessed input
MAX_INPUT	Maximum length of an input queue
NAME_MAX	Maximum length of a file name
PATH_MAX	Maximum length of a path name

Table 5. OpenExtensions Path Name Variable Values (continued)

Name	Description
PIPE_BUF	Maximum atomic pipe write

2.9 Symbolic Constants

The <unistd.h> header defines the symbolic constants and structures referred to in the following subsections.

2.9.3 Compile-Time Symbolic Constants for Portability Specifications

The constants for portability specifications in Table 6 on page 9 can be used by application programs at compile time to determine which optional facilities are present.

Table 6. Compile-Time Symbolic Constants for Portability Specifications

Name	Description	Value
_POSIX_JOB_CONTROL	Job control	1
_POSIX_SAVED_IDS	Saved set-user or group IDs	1
_POSIX_VERSION	Standard publish date	199009L

2.9.4 Execution-Time Constants for Portability Specifications

The constants for portability specifications shown in Table 7 on page 9 are available in the OpenExtensions implementation with the pathconf() or fpathconf() function. They are not defined in <unistd.h>.

Table 7. Execution-Time Symbolic Constants for Portability Specifications

Name	Description
_POSIX_CHOWN_RESTRICTED	chown() restricted
_POSIX_NO_TRUNC	Error if the path name length is greater than {NAME_MAX}
_POSIX_VDISABLE	The value used to disable terminal special characters

Section 3. Process Primitives

3.1 Process Creation and Execution

3.1.1 Process Creation

Function: `fork()`

3.1.1.2 Description

ATTENTION: You should use this function only during a porting exercise until you can convert to `spawn()` or threading functions.

The child process created by the OpenExtensions implementation of the `fork()` function is not an exact copy of the parent, but is another process residing within the same CMS session. The child has all the attributes associated with CMS processes. For more information about CMS processes, see [z/VM: CMS Application Multitasking](#).

The OpenExtensions implementation of the `fork()` function has the following restrictions:

- The child process is not allowed to issue an `exit()` call or to call any function that will invoke `exit()` before the child process issues the `exec()` function.
- The child process is not allowed to issue any function that will cause the child process to be blocked (for example, a pipe `read()` or a `pause()`), before the child issues the `exec()` function.
- Any local variables in the application that are changed in the child process before the `exec()` is issued will be changed in the parent process as well. This is because the child and parent processes are still using the same program storage. The `exec()` function causes the child process to begin using its own program storage.
- Any global or environment variables in the application that are changed in the child process before the `exec()` is issued will be changed in the parent process as well. This is because the child and parent processes are still using the same program storage. The `exec()` function causes the child process to begin using its own program storage.

3.1.1.4 Errors

The following additional errors may be returned by the system:

- If the CMS FORK function is not turned on, the `fork()` function returns a -1 and sets `errno` to [ENOSYS].
- If the child process issues `exit()` or calls any function that invokes `exit()` before the child issues the `exec()` function, the request fails with an abnormal end code of X'AE5'.
- If the child process calls any function that causes the child to be blocked before the child issues the `exec()` function, the request fails with an abnormal end code of X'AE6'.

3.1.1.5 Cross-References

Because the OpenExtensions implementation of `fork()` is not full-function, the only valid cross-reference is to `exec()`, 3.1.2.

3.1.2 Execute a File

Functions: `execl()`, `execv()`, `execle()`, `execve()`, `execlp()`, `execvp()`

3.1.2.2 Description

If the **PATH** environment variable is not present and the file name given to `execlp()` or `execvp()` does not have a slash, the file is accessed as given (no additional implementation-defined search lists are used) in the argument.

The number of bytes for {ARG_MAX} includes the argument strings, environment strings, and the null bytes that end these strings.

If an exec function fails but was able to locate the *process image file*, the `st_atime` field of the file is updated.

For any one of the exec family of functions, if the executable file has the set-user-id mode bit set and the effective UID of the process is not the same as the file owner or if the executable file has the set-group-id mode bit set and the effective GID is not the same as the file group, the caller must have CP authorization to execute these files. This permission is defined in the CP directory on an individual or system-wide basis, and specifies whether or not a user with the appropriate file access permissions is allowed to execute a file that will result in a change of POSIX IDs. In addition, the file server on which the object file resides must have CP authorization to specify that the POSIX IDs for the caller be changed. This permission is also defined in the CP directory. If either of these security checks fails, the existing process will be terminated, and the new file will not be invoked.

For any one of the exec family of functions, if the file in the Byte File System associated with the specified path name cannot successfully be opened for execution, the path name is interpreted as a CMS file identifier. A search is done for this file using the accessed file modes and, under certain circumstances, the nucleus extensions. If the file is found in the record file system, it is executed. Otherwise, the exec function returns with an **errno** that reflects the reason for the original open failure.

3.1.2.4 Errors

The exec functions return an **errno** of [ENOMEM] if the new process requires more memory than is permitted by the hardware or operating system. The exec functions return an **errno** of [ECMSERR] if there is insufficient storage to allow the system to perform the request.

Nonregular files cannot be executed.

3.2 Process Termination

3.2.1 Wait for Process Termination

Functions: `wait()` and `waitpid()`

3.2.1.2 Description

In addition to returning status for child processes, `wait()` or `waitpid()` may return status for processes that are being debugged.

3.2.2 Terminate a Process

Function: `_exit()`

3.2.2.2 Description

If a process ends, any children for which `wait()` has not been run are inherited by the `init` process, whose process ID is 1.

The `init` process waits for and discards the status for any terminated children that it inherits.

3.3 Signals

3.3.1 Signal Concepts

3.3.1.1 Signal Names

[Table 8 on page 13](#) shows signals that are supported in addition to those specified by POSIX.1.

Table 8. OpenExtensions Non-POSIX Signals

Symbolic Constant	Description
SIGABND	Abend signal

In addition, the symbol **SIGCLD** is provided, with the same signal value as **SIGCHLD**.

For the list of default actions and values associated with these signals, see [z/VM: OpenExtensions Callable Services Reference](#).

3.3.1.2 Signal Generation and Delivery

If the action associated with a blocked signal is to ignore the signal and if that signal is generated for the process, the OpenExtensions implementation leaves the signal pending.

Signals are not queued. If a subsequent occurrence of a pending signal is generated, the signal is delivered only once.

3.3.2 Send a Signal to a Process

Function: `kill()`

3.3.2.2 Description

A signal sent to process ID 0 is sent to all processes in the current process group, with no system-defined exclusions.

The range of a signal is limited to processes in the same virtual machine as the sending process; signals may not be sent to processes in other virtual machines.

3.3.3 Manipulate Signal Sets

Functions: `sigemptyset()`, `sigfillset()`, `sigaddset()`, `sigdelset()`, and `sigismember()`

3.3.3.4 Errors

`sigaddset()`, `sigdelset()`, and `sigismember()` generate **errno** [EINVAL] if the signal number is less than 1 or greater than 64. They do not detect unsupported signal numbers between 1 and 64.

3.3.4 Examine and Change Signal Action

Function: `sigaction()`

3.3.4.2 Description

The contents of *oact* returned by `sigaction()` for a signal whose action was last set by `signal()` rather than `sigaction()` is:

```
struct sigaction {
    void (*)() sa_handler; -- will contain the address of the user
                           signal catcher function specified in
                           signal()
    sigset_t sa_mask;    -- will be set to the empty set
    int       sa_flags;  -- the SA_OLD_STYLE flag will be set
}
```

An attempt to set the action for a signal that cannot be caught or ignored to **SIG_DFL** is returned with a return value of 0, and **errno** is not set to [EINVAL].

3.3.6 Examine Pending Signals

Function: `sigpending()`

3.3.6.4 Errors

`sigpending()` may return **errno** set to a value defined in the OpenExtensions implementation.

3.4 Timer Operations

3.4.3 Delay Process Execution

Function: `sleep()`

3.4.3.2 Description

The following list describes actions for a **SIGALRM** signal generated during the execution of `sleep()`:

- If the calling process has **SIGALRM** being blocked before calling `sleep()`, then `sleep()` does not return when this **SIGALRM** is generated and the **SIGALRM** signal is left pending when `sleep()` returns.
- If the calling process has **SIGALRM** being ignored before calling `sleep()`, then `sleep()` does not return when this **SIGALRM** is generated and the **SIGALRM** signal is ignored.
- If the calling process has **SIGALRM** being set to a signal-catching function, the **SIGALRM** signal-catching function interrupts `sleep()` and the signal-catching function receives control. The `sleep()` function returns any unslept amount of time, as it does for any other type of signal.

If a signal-catching function interrupts the `sleep()` function and either examines or changes the time a **SIGALRM** is scheduled to be generated, the action associated with the **SIGALRM** signal is the same as it is for any other function that is interrupted by a signal-catching function.

Section 4. Process Environment

4.2 User Identification

4.2.3 Get Supplementary Group IDs

Function: `getgroups()`

4.2.3.4 Errors

For the `getgroups()` function, the following error conditions are detected:

[ECMSERR]

Unforeseen CMS error

[ECPERR]

Unforeseen CP or ESM error

4.2.4 Get User Name

Function: `getlogin()`

4.2.4.3 Returns

The data returned by `getlogin()` does not point to static data that will be overwritten by each call.

4.2.4.4 Errors

`getlogin()` is never expected to fail and does not have any error conditions.

4.4 System Identification

4.4.1 Get System Name

Function: `uname()`

4.4.1.2 Description

In the OpenExtensions implementation, the structure **utsname** contains the members and formats shown in [Table 9 on page 15](#). Each member is padded with blanks to fill out the structure.

Table 9. Formats for OpenExtensions utsname Members

Member Name	Description	Format
sysname	Name of implementation	char [16]
nodename	Network node name	char [32]

Table 9. Formats for OpenExtensions utsname Members (continued)

Member Name	Description	Format
release	The level of CMS in use, expressed as the string CMS_ <i>l_s_f</i> , where: <i>l</i> is the CMS level as returned by the QUERY CMSLEVEL command. <i>s</i> is the 4-digit CMS service level as it appears in DMSVLTB. <i>f</i> is the CMS level code returned by the DMSQEFL routine in its output parameter <i>cms_level</i> . For example, the release (CMS) information for z/VM Version 6 Release 1.0 is: CMS_25_0000_80.	char [64]
version	The level of CP in use, expressed as the string CP_ <i>v.r.m_s_f</i> , where: <i>v</i> is the CP version number returned by the QUERY CPLEVEL command. <i>r</i> is the CP release number returned by QUERY CPLEVEL. <i>m</i> is the CP modification level returned by QUERY CPLEVEL. <i>s</i> is the 4-digit CP service level as it appears in the output of QUERY CPLEVEL. <i>f</i> is the CP level code returned by the DMSQEFL routine in its output parameter <i>cp_level</i> . For example, the version (CP) information for z/VM Version 7 Release 4.0 is: CP_7.4.0_0000_104.	char [64]
machine	Machine hardware name	char [16]

4.6 Environment Variables

4.6.1 Environment Access

Function: `getenv()`

4.6.1.4 Errors

The following errors may be returned by the system:

- If not enough memory exists to return the environment variable data, the `getenv()` function returns a NULL value and sets **errno** to [ENOMEM].
- If the "=" character is found in an environment variable name, the `getenv()` function returns a NULL value and sets **errno** to [ECMSBADCHAR].

4.7 Terminal Identification

4.7.1 Generate Terminal Pathname

Function: `ctermid()`

4.7.1.4 Errors

No error conditions are returned.

4.7.2 Determine Terminal Device Name

Functions: `ttyname()` and `isatty()`

4.7.2.4 Errors

No error conditions are returned.

4.8 Configurable System Variables

4.8.1 Get Configurable System Variables

Function: `sysconf()`

4.8.1.2 Description

`sysconf()` supports system variables other than those listed in the IEEE Std 1003.1-1990. See [z/VM: OpenExtensions Callable Services Reference](#) for more information.

4.8.1.5 Special Symbol {CLK_TCK}

The special symbol {CLK_TCK} is evaluated at run time.

Section 5. Files and Directories

5.1 Directories

5.1.2 Directory Operations

Functions: `opendir()`, `readdir()`, `rewinddir`, and `closedir()`

5.1.2.2 Description

Directory streams use a file descriptor internally, so the `opendir()` function is subject to the limit on open files.

The OpenExtensions implementation does not return entries for dot and dot-dot on a call to `readdir()`.

If a file is removed from or added to the directory after the most recent call to `opendir()` or `rewinddir()`, whether a subsequent call to `readdir()` returns an entry for that file depends on the circumstances. If the new entry was added to the directory at a point beyond the entries being buffered by the runtime library, the entry is returned; otherwise, it is not returned.

All directory streams are closed after any of the `exec` family of functions is run.

If both the child and parent use `readdir()` or `rewinddir()` for a directory stream after a `spawn()`, the results are indeterminate.

5.1.2.4 Errors

For the `opendir()` function, the OpenExtensions implementation detects the condition and returns the corresponding **errno** values for [EMFILE] and [ENFILE].

If the *directory-stream* argument passed to `readdir()` or `closedir()` does not refer to a currently open directory stream:

- The functions set **errno** to [EBADF]
- `readdir()` returns a value of null
- `closedir()` returns a -1

5.2 Working Directory

5.2.2 Get Working Directory Pathname

Function: `getcwd()`

5.2.2.2 Description

If *buf* is a NULL pointer, `getcwd()` returns a NULL pointer and sets **errno** to [EINVAL].

5.2.2.3 Returns

The contents of *buf* after an error is indeterminate.

5.2.2.4 Errors

When read permission is denied for a component of the path name, a call to `getcwd()` returns the current working directory.

When search permission is denied for a component of the path name, a call to `getcwd()` returns a value of NULL and `errno` of [EACCES].

5.3 General File Creation

5.3.1 Open a File

Function: `open()`

5.3.1.2 Description

If bits other than the file permission bits are set in the *mode* argument when a file is being created, these bits are ignored by the OpenExtensions implementation.

If `O_CREAT` is specified, the file's group ID is set to the group ID of the directory in which the file is being created.

`O_EXCL` is ignored if `O_CREAT` is not set.

`O_NONBLOCK` is ignored on file types other than FIFO and character special file.

`O_TRUNC` is ignored on file types other than regular files.

If `O_TRUNC` and `O_RDONLY` are set on, the request fails and `errno` is set to [EINVAL].

5.3.3 Set File Creation Mask

Function: `umask()`

5.3.3.2 Description

In the OpenExtensions implementation, only the permission bits are put in the file mode creation mask. Any other bits in *cmask* are ignored.

5.3.3.3 Returns

The file permission bits from the process's current file mode creation mask are returned. Other bits in the returned value are set to 0.

5.3.4 Link to a File

Function: `link()`

5.3.4.2 Description

The OpenExtensions implementation:

- Does not support linking of files across file systems
- Does not support using `link()` on directories
- Requires that the calling process has permission to access the existing file

5.4 Special File Creation

5.4.1 Make a Directory

Function: `mkdir()`

5.4.1.2 Description

If bits other than the file permission bits are set in the *mode* argument, these bits are ignored by the OpenExtensions implementation.

5.4.2 Make a FIFO Special File

Function: `mkfifo()`

5.4.2.2 Description

If bits other than the file permission bits are set in the *mode* argument, these bits are ignored by the OpenExtensions implementation.

5.5 File Removal

5.5.1 Remove Directory Entries

Function: `unlink()`

5.5.1.2 Description

The OpenExtensions implementation does not support using `unlink()` on directories.

5.5.2 Remove a Directory

Function: `rmdir()`

5.5.2.2 Description

If the named directory is the root directory of any file system, or the working directory of any process, `rmdir()` succeeds, provided that the named directory is not the root of a physical file system.

If the named directory is the root of a physical file system, the operation fails and sets **errno** to [EPERM].

5.5.2.4 Errors

If the named directory is not empty, the request fails with [ENOTEMPTY].

5.5.3 Rename a File

Function: `rename()`

5.5.3.2 Description

If the *old* argument points to the path name of a directory, write access permission is not required for the directory named by the *old* name nor for a directory named by the *new* name, if it exists.

If the named directory is the root of a physical file system, the operation fails and sets **errno** to [EPERM].

5.5.3.4 Errors

[EBUSY]

The directory named by *old* or *new* is being used by the system as either the system root or a mount point. Renaming such directories is not permitted.

[EXDEV]

The links named by *old* and *new* are on different file systems. Renaming across file systems is not permitted.

5.6 File Characteristics

5.6.2 Get File Status

Functions: `stat()` and `fstat()`

5.6.2.2 Description

There are no additional or alternate file access control mechanisms used by the OpenExtensions implementation.

5.6.3 Check File Accessibility

Function: `access()`

5.6.3.2 Description

Regardless of whether the process has appropriate privileges, `X_OK` does not indicate success for nondirectory files if none of the execute file permission bits are set.

For directory files, a process with appropriate privileges is given search access, even if none of the execute file permission bits are set.

5.6.3.4 Errors

If the `access_mode` parameter is incorrect, the function returns an `errno` of [EINVAL].

5.6.4 Change File Modes

Function: `chmod()`

5.6.4.2 Description

There are no implementation-defined restrictions that cause the `S_ISUID` and `S_ISGID` bits in `mode` to be ignored.

There is no effect on reading and writing of files that are open at the time of the `chmod()` function. However, there are several functions—for example, `utime()` and `stat()`—that can provide differing results when they are performed before and after a `chmod()` function.

5.6.5 Change Owner and Group of a File

Function: `chown()`

5.6.5.2 Description

The `S_ISUID` and `S_ISGID` bits of the file mode are always cleared upon successful completion of `chown()`, even if the process has appropriate privileges, and regardless of the file type.

5.6.5.4 Errors

If the owner or group ID supplied is incorrect, the function returns an `errno` of [EINVAL].

5.7 Configurable Pathname Variables

5.7.1 Get Configurable Pathname Variables

Functions: `pathconf()` and `fpathconf()`

5.7.1.2 Description

The OpenExtensions implementation does not support any configurable file name variables that do not appear in Table 5-2 of IEEE Std 1003.1-1990.

5.7.1.3 Returns

If *name* refers to MAX_CANON, MAX_INPUT, or _POSIX_VDISABLE, the following applies:

- If *path* or *fd* does not refer to a terminal file, the function returns -1 and sets **errno** to [EINVAL].

If *name* refers to PC_NAME_MAX, PC_PATH_MAX, or PC_NO_TRUNC, the following applies:

- If *path* or *fd* does not refer to a directory, the function still returns the requested information, with reference to the parent directory.

If *name* refers to PC_PIPE_BUF the following applies:

- If *path* or *fd* refers to any other type of file besides a pipe or a FIFO special file, the function returns -1 and sets **errno** to [EINVAL].

5.7.1.4 Errors

If search permission is denied for a component of the path prefix, the function returns an **errno** of [EACCES].

If the configurable file name variable is not supported for the specified file, the function returns an **errno** of [EINVAL].

If the path name is longer than 1023 characters, or some component of the path name is longer than 255 characters, the function returns -1 and sets **errno** to [ENAMETOOLONG].

If the named file does not exist, or if the path name points to an empty string, the function returns -1 and sets **errno** to [ENOENT].

If a component of the path prefix is not a directory, the function returns -1 and sets **errno** to [ENOTDIR].

If the file descriptor is incorrect, the function returns -1 and sets **errno** to [EBADF].

Section 6. Input and Output Primitives

6.3 File Descriptor Deassignment

6.3.1 Close a File

Function: `close()`

6.3.1.2 Description

If the `close()` function is interrupted by a signal that is to be caught, it returns -1 with **errno** set to [EINTR], and the *fildes* argument is closed.

6.3.1.4 Errors

[EIO] may be generated by a `close()` if an I/O operation fails within z/VM.

6.4 Input and Output

6.4.1 Read from a File

Function: `read()`

6.4.1.2 Description

If the value of *nbyte* is greater than {SSIZE_MAX}, the function returns -1 and sets **errno** to [EINVAL].

6.4.1.4 Errors

[EIO] may be generated by a `read()` if the I/O operation fails within z/VM.

6.4.2 Write to a File

Function: `write()`

6.4.2.2 Description

If a `write()` is interrupted by a signal after it successfully writes some data, it returns the number of bytes successfully written. (Partial transfers are reported.) This can happen only with a write to a nonpipe file, a FIFO special file, a socket, or a nonregular file.

If *nbyte* is 0 and the file is not a regular file, the `write()` function returns 0 and has no other results.

If the value of *nbyte* is greater than {SSIZE_MAX}, `write()` returns -1 and sets **errno** to [EINVAL].

6.4.2.4 Errors

[EIO] may be generated by a `write()` if the I/O operation fails within z/VM.

6.5 Control Operations on Files

6.5.2 File Control

Function: `fcntl()`

6.5.2.2 Description

F_SETFD

The OpenExtensions implementation also supports the setting of the FD_CLOFORK bit. After this bit has been set, it cannot be turned off. `_OPEN_SYS` is the name of the feature test macro that can be invoked to make `FD_CLOFORK` visible.

F_GETFL

The OpenExtensions implementation also returns the *oflag* values for `open()`, as described in Table 6-4 of the IEEE Std 1003.1-1990 standard. You can extract the file access modes defined in Table 6-6 of the standard, and the file status flags defined in Table 6-5 of the standard, from the return value by using the mask `O_GETFL`, which is defined in `<fcntl.h>`.

F_SETFL

If any bits in *arg* other than those mentioned here are changed, they are ignored.

F_SETLK, F_SETLKW, F_GETLK

The *l_len* value cannot be a negative value. A return value of -1 and a return code of [EINVAL] are returned if a negative *l_len* is specified.

F_CLOSPD

A process can use `fcntl()` to close a range of file descriptors by specifying `F_CLOSPD` for *cmd*. The file descriptor specified by *filedes* is the lower limit of the range of file descriptors to be closed. The third parameter then specifies a file descriptor to be the upper limit of the range. The third parameter can be set to -1 to indicate that all file descriptors greater than or equal to *filedes* are to be closed.

The `F_CLOSPD` command can be used only to close file descriptors that could be closed with a `close()` request. If not all files in the range can be closed, those that can be closed are closed, and an `errno` of [EPERM] is reported.

6.5.2.4 Errors

If a deadlock condition is detected for a `F_SETLKW` request, the function returns an `errno` of [EDEADLK].

If the action specified was `F_CLOSPD`, and the file descriptor specified as the upper limit for the range is less than the file descriptor specified as the lower limit (but is not equal to -1), an error of [EINVAL] is reported.

If the action requested was `F_CLOSPD`, and all the file descriptors in the specified range were not closed, an error of [EPERM] is reported.

6.5.3 Reposition Read/Write File Offset

Function: `lseek()`

6.5.3.2 Description

On files incapable of seeking, `lseek()` sets the file offset to the specified value. However, the offset is not honored by functions that read from or write to such files.

Section 7. Device- and Class-Specific Functions

7.1 General Terminal Interface

The OpenExtensions implementation does not support any devices that support asynchronous serial communication. The CMS command OPENVM SHELL is provided, which uses a terminal to provide the interactive environment. The full programming interface is provided.

Only canonical mode is supported.

7.1.1 Interface Characteristics

7.1.1.2 Process Groups

The condition described in POSIX in which a terminal's process group ID does not match any existing process group ID, but does match an existing process ID, cannot occur in the OpenExtensions implementation.

7.1.1.3 The Controlling Terminal

If a session leader without a controlling terminal opens a terminal device file not already associated with a session without specifying the O_NOCTTY option, then this terminal becomes the controlling terminal for the session leader. This is how a controlling terminal is acquired.

7.1.1.5 Input Processing and Reading Data

The OpenExtensions implementation imposes the limit {MAX_INPUT} on the number of bytes that may be stored in the input queue.

{MAX_INPUT} cannot be exceeded.

7.1.1.6 Canonical Mode Input Processing

{MAX_CANON} is defined for terminals.

{MAX_CANON} cannot be exceeded.

7.1.1.7 Noncanonical Mode Input Processing

The OpenExtensions implementation does not provide noncanonical mode input processing. Any attempt to put a terminal in noncanonical mode is ignored.

7.1.1.8 Writing Data and Output Processing

Data written to a terminal is buffered. Therefore, when a `write()` completes, the data has not necessarily been presented to the user.

7.1.1.9 Special Characters

No multibyte special sequences are supported.

7.1.1.10 Modem Disconnect

A terminal connection does not depend on the state of the modem status lines (the terminal is assumed to be connected to the system locally).

7.1.2 Parameters That Can Be Set

7.1.2.2 Input Modes

A break condition does not exist.

Parity errors cannot occur. Therefore, the settings of PARMRK, IGNPAR, and INPCK have no effect. Attempts to change them from their default values are ignored.

Since input is in EBCDIC and requires all 8 bits, ISTRIP is inappropriate and attempts to set it are ignored.

Since MAX_INPUT cannot be exceeded, the setting of IXOFF has no effect. Attempts to set IXOFF are ignored. STOP and START characters are never sent as a result of buffer conditions.

Flags ICRNL and IGNBRK are always set. Attempts to change them from their default values are ignored.

The initial *c_iflag* setting after `open()` is defined as:

ICRNL

Map carriage return to newline on input

IGNBRK

Ignore break condition

IXON

Enable start/stop output control

7.1.2.3 Output Modes

The initial *c_oflag* setting for the OPOST flag is On after `open()`. When the OPOST flag is set in *c_flag*, tab expansion is performed; enough blank characters are inserted to reach the next multiple of 8 bytes on a line.

7.1.2.4 Control Modes

A program can request the changing of any flag, but attempts to change them from their default values are ignored.

Flags CREAD, CSIZE, and CLOCAL are always set.

The initial *c_cflag* setting after `open()` is defined to be:

CREAD

Enable receiver

CSIZE

Set to CS8 for 8 bits per byte

CLOCAL

Ignore modem status lines

7.1.2.5 Local Modes

A program can request the changing of any flag, but changes to IEXTEN and ICANON are ignored. Attempts to change them from their default values are ignored.

The initial *c_lflag* setting after `open()` is defined to be:

ECHO

Enable echo

ICANON

Canonical input processing

ISIG

Enable signals

7.1.2.6 Special Control Characters

The number of special control characters in array c_cc (NCCS) is 11. [Table 10 on page 29](#) shows the initial values of these characters.

Table 10. Initial Values for Special Control Characters

Control Character	Control Sequence (See Note)	Hexadecimal Value	EBCDIC character
VEOF	\$D	00	None
VEOL	\$J	15	NL
VERASE	\$H	16	BS
VINTR	\$C	03	ETX
VKILL	\$U	3D	NAK
VMIN	None	00	None
VQUIT	\$V	32	SYN
VSTART	None	00	None
VSTOP	None	00	None
VSUSP	\$Z	3F	SUB
VTIME	None	00	None

Note: The control sequence is a character sequence that must be entered from the terminal to cause the special control character to be generated. The prefix character (\$) can be set by the user.

The (\$) is not passed to the user; it merely tells the terminal driver to treat the following character as a control character. Thus, for instance, typing \$D results in an EBCDIC EOT character ('X'37') to be passed to the user.

7.1.3 Baud Rate Functions

Functions: cfgetispeed(), cfgetospeed(), cfsetispeed(), and cfsetospeed()

7.1.3.2 Description

See “[7.1.3.4 Errors](#)” on page 29 for a description of processing when an unsupported baud rate is specified.

All POSIX-defined baud rates are accepted by tcsetattr(), but they have no effect on the terminal connection. A subsequent tcgetattr() returns the baud rates set by an earlier tcsetattr().

7.1.3.4 Errors

If an unsupported baud rate is specified for the cfsetispeed() or cfsetospeed() functions, a return of -1 is generated and **errno** is set to [EINVAL].

7.2 General Terminal Interface Control Functions

7.2.1 Get and Set State

Functions: tcgetattr(), tcsetattr()

7.2.1.2 Description

The `tcsetattr()` function does not support the TCDRAIN action. If it is specified, the `tcsetattr()` function returns -1 with `errno` set to [EINVAL].

7.2.2 Line Control Functions

Functions: `tcsendbreak()`, `tcdrain()`, `tcflush()`, and `tcflow()`

7.2.2.2 Description

The `tcsendbreak()` function does not generate a break condition. Unless issued under circumstances requiring a **SIGTTOU** signal, the function is successful without taking any action.

The `tcdrain()` function is not supported. It returns -1 with `errno` set to [EINVAL].

Section 8. Language-Specific Services for the C Programming Language

8.1 Referenced C Language Routines

8.1.1 Extensions to Time Functions

In the OpenExtensions implementation, **TZ** environment variables of the form

:characters

are *not* supported.

TZ environment variables with the expanded format described in section 8.1.1 are supported. These **TZ** environment variables have the form

stdoffset[dst[offset][,start[/time], end[/time]]]

If parsing of the **TZ** environment variable fails, time zone values specified by the C proprietary LC_TOD locale category are used to establish default values. See [z/OS: XL C/C++ Programming Guide](https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/cbcpx01_v2r5.pdf) (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/cbcpx01_v2r5.pdf) for a description of the LC_TOD category.

8.1.2 Extensions to setlocale() Function

Function: `setlocale()`

8.1.2.2 Description

For the `setlocale()` function, the default values for the required categories and those categories specific to the OpenExtensions implementation are defined in [Table 11 on page 31](#).

Table 11. Default Values for Required and OpenExtensions-Specific Categories

Category	Default Value
LC_CTYPE	"C"
LC_COLLATE	"C"
LC_TIME	"C"
LC_NUMERIC	"C"
LC_MONETARY	"C"
LC_MESSAGES	"C"
LC_TOD	"C"

See [z/OS: XL C/C++ Programming Guide](https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/cbcpx01_v2r5.pdf) (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/cbcpx01_v2r5.pdf) for:

- A description of LC_TOD category
- Information on the contents of the string that is returned when the locale name is an explicit string
- Information on the contents of the string that is returned when the pointer to the locale name is null

The following locale values are recognized by `setlocale()`: "C", "POSIX", "FRAN", "GERM", "ITAL", "SPAI", "S370", "UK", and "USA". These locales are described in [z/OS: XL C/C++ Programming Guide](#) (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/cbcpx01_v2r5.pdf).

If:

1. The **LC_ALL** environment variable is not specified or is set to the empty string,
2. The environment variable corresponding to the category named on the `setlocale()` call is not specified or is set to the empty string, and
3. The **LANG** environment variable is not set or is set to the empty string,
then `setlocale()` defaults to the "C" locale.

8.2 C Language Input/Output Functions

8.2.1 Map a Stream Pointer to a File Descriptor

Function: `fileno()`

8.2.1.4 Errors

If the *stream-pointer* argument is not valid or refers to a CMS native record file, the `fileno()` function returns -1, and sets **errno** to [EBADF].

8.2.2 Open a Stream on a File Descriptor

Function: `fdopen()`

8.2.2.2 Description

The *type* argument can have a b as the second or third character to indicate binary. This b is ignored.

8.2.2.4 Errors

If the first character of the *type* argument is not r, w, or a, the `fdopen()` function returns a NULL stream pointer and sets **errno** to [EINVAL].

If the *file descriptor* argument is not a valid open file descriptor, the `fdopen()` function returns a NULL stream pointer and sets **errno** to [EBADF].

8.2.3 Interactions of Other FILE-Type C Functions

When applications obey all the rules specified in POSIX.1 section 8.2.3, input is always seen exactly once.

8.2.3.10 `fseek()`

If the stream is opened in append mode, the result of `fseek()` on that stream is the current file position.

8.3 Other C Language Functions

8.3.2 Set Time Zone

Function: `tzset()`

8.3.2.2 Description

If **TZ** is absent from the environment or cannot be parsed, the time zone values specified by the C proprietary locale category, **LC_TOD**, are used to establish default values.

Section 9. System Databases

9.1 System Databases

The system default for the numerical user ID field is 4,294,967,295 (X'FFFFFF').

The user name field is the lowercased z/VM user ID.

The system default for the numerical group ID field is 4,294,967,295 (X'FFFFFF').

The system default for the initial user program field is the program **/bin/sh**. This is the default shell.

If the initial working directory field is null, the initial working directory is the root directory: **/**.

No other implementation-defined fields in the user or group databases are supported.

The system databases can be kept in the CP directory or an External Security Manager (ESM).

9.2 Database Access

9.2.1 Group Database Access

Functions: `getgrgid()` and `getgrnam()`

9.2.1.2 Description

To be authorized to obtain a group database entry either:

- An ESM must grant the requestor authority to read the entry, or
- An ESM must not be installed or must defer authorization to CP, and
 - The caller's effective UID must be 0, or
 - The caller's real or effective GID must match the GID of the designated group, or
 - The caller must be a member of the specified group, or
 - The caller must have the attribute POSIXOPT QUERYDB ALLOW, either through a statement in its CP directory entry or through a setting, specified or defaulted, in the system configuration file, which is not overridden in the directory entry.

9.2.1.3 Returns

In the OpenExtensions implementation, the return values for the `getgrgid()` and `getgrnam()` functions point to data that may be overwritten by each call.

9.2.1.4 Errors

For the `getgrgid()` and `getgrnam()` functions, the following error conditions are detected:

[EINVAL]

If the group name specified was less than 1 or greater than 8 characters long

[ECMSERR]

Unforeseen CMS error, such as insufficient storage

[ECPERR]

CP or ESM error

9.2.2 User Database Access

Functions: `getpwuid()` and `getpwnam()`

9.2.2.2 Description

To be authorized to obtain a user database entry either:

- An ESM must grant the requestor authority to read the entry, or
- An ESM must not be installed or must defer authorization to CP, and
 - The caller's effective UID must be 0, or
 - The caller's real or effective UID must match the UID in the entry, or
 - The caller must have the attribute `POSIXOPT QUERYDB ALLOW`, either through a statement in its CP directory entry or through a setting, specified or defaulted, in the system configuration file, which is not overridden in the directory entry.

The `getpwnam()` service is not sensitive to the case of the user name specified on input. This means that a user name of DANIEL is considered the same as a user name of daniel. However the user name returned in the database entry is always in lower case.

The `getgrname()` service is not sensitive to the case of the group name.

9.2.2.3 Returns

For the `getpwuid()` and `getpwnam()` functions, the return values point to data that may be overwritten on each call.

9.2.2.4 Errors

In the OpenExtensions implementation, the `getpwuid()` and `getpwnam()` functions detect the following error conditions:

[EINVAL]

If the user name specified was less than 1 or greater than 8 characters long

[ECMSERR]

Unforeseen CMS error, such as insufficient storage

[ECPERR]

CP or ESM error

Section 10. Data Interchange Format

10.1 Archive/Interchange File Format

An archive being introduced into an OpenExtensions implementation from an external medium is first copied intact into a file in the OpenExtensions file system using the CMS command OPENVM PARCHIVE. It is then read using the format-reading pax utility of OpenExtensions.

An archive being exported from an OpenExtensions implementation to an external medium is first created in the byte file system (BFS) with the format-creating pax utility of OpenExtensions. It is then copied to the external medium with the OPENVM PARCHIVE command.

See [z/VM: OpenExtensions Commands Reference](#) for a description of the pax utility and the OPENVM PARCHIVE command and the interfaces to them.

10.1.1 Extended tar Format

The OpenExtensions implementation supports the use of characters outside the portable file name character set in names for files, users, or groups. For interchange purposes, such characters are mapped to ISO 8859-1. Any characters in a name to be archived that are not in the ISO 8859-1 character set are converted to underscore when stored in an extended `tar` archive.

If a file name found in an archive contains characters outside the ISO 8859-1 character set, such characters are converted to underscore before being put into the file system. No names can result from this conversion that are incorrect in the hierarchical file system.

If a file to be archived has the `filemode` bit `S_ISVTX` set, the `TSVTX` bit is set in the archive and vice versa.

10.1.2 Extended cpio Format

10.1.2.1 cpio Header

For character special files, `c_rdev` contains a leading-zero-filled octal representation of the 18-bit binary number formed by concatenating the low-order 9 bits of the `devmajor` field (in the high-order 9 bits of `c_rdev`) and the low-order 9 bits of the `devminor` field (in the low-order 9 bits of `c_rdev`). This can result in ambiguity if character devices are archived whose `devmajor` or `devminor` numbers contain more than 9 bits of significance. The OpenExtensions implementation supports 16-bit values in these fields.

The OpenExtensions implementation does not support block special files.

10.1.2.2 cpio Filename

In the OpenExtensions implementations, if a file name found in an archive contains characters outside the ISO 8859-1 character set, such characters are converted to underscore before being put into the file system. No names can result from this conversion that are incorrect in the byte file system.

10.1.2.5 cpio Values

In the OpenExtensions implementation, other than those file types defined in Table 10-3 of the IEEE Std 1003.1-1990 standard, the following file type is supported in cpio archives: symbolic links. The `typeflag` for a symbolic link in cpio archives is `C_ISLNK`.

If a file to be archived has the `filemode` bit `S_ISVTX` set, the `C_ISVTX` bit is set in the archive and vice versa.

10.1.3 Multiple Volumes

In the OpenExtensions implementation, the pax utility of OpenExtensions determines which file to read or write for the next volume of a multivolume archive by prompting to **stdout** and reading the reply from **stdin**.

Part 2. POSIX.2 Conformance Document

Section 1. General

1.3 Conformance

1.3.1 Implementation Conformance

The following sections describe the behavior of OpenExtensions in situations that the POSIX.2 standard defines as implementation-defined.

1.3.1.1 Requirements

The POSIX.2 standard states:

(3)

The system may provide additional or enhanced utilities, functions, or facilities not required by this standard. Nonstandard extensions should be identified as such in the system documentation. Nonstandard extensions, when used, may change the behavior of utilities, functions, or facilities defined by this standard. In such cases, the implementation's conformance document (see 2.2.1.3) shall define an execution environment (i.e., shall provide general operating instructions) in which an application can be run with the behavior specified by this standard. In no case shall such an environment require modification of a Strictly Conforming POSIX.2 Application.

The `xargs` utility has a known problem, and currently, there is no execution environment that can circumvent it. The `xargs` utility of OpenExtensions conforms to XPG4, and as such, supports the "`-e`" option, which means that `xargs` supports the default logical EOF string of "`_`" which is an extension of POSIX.2 and may change the behavior of conforming applications that use `xargs`.

1.3.1.2 Documentation

OpenExtensions conforms to the IEEE Std 1003.2-1992 and ISO/IEC DIS 9945-2:1992, hereafter referred to as POSIX.2 in this document. This report is published to satisfy the requirement of the POSIX.2 standard for a conformance document, as outlined in section *1.3.1.2 Documentation* of the POSIX.2 standard.

This document has the same structure as the POSIX.2 standard. It lists all section numbers and titles as they appear in the POSIX.2 standard and in the same order. (It omits empty sections.) It also uses typographical conventions similar to those used in the POSIX.2 standard. This document supplements, but does not replace, the POSIX.2 standard.

OpenExtensions POSIX.2 support resides on a system that conforms to IEEE Std 1003.1-1990 and ISO/IEC 9945-1:1990(E), referred to as POSIX.1 in this document, except that the OpenExtensions implementation of the `fork()` function does not meet all POSIX.1 specifications.

Section 2. Terminology and General Requirements

2.2 Definitions

2.2.2 General Terms

2.2.2.8 appropriate privileges

The POSIX.2 standard states that "the means for associating privileges to a process is implementation defined."

The POSIX.2 standard refers several times to "appropriate privileges," a concept taken from POSIX.1 (see POSIX.1 section 2.3, General Terms, which defines this concept).

- **3.5.3 Variables:** See section 3.5.3 of this document.
- **4.7.2 Description {of chmod}:** See POSIX.1 section 5.6.4.2 Description {of chmod ()}.
- **4.30.2 Description {of id}:** See POSIX.1 section 2.2.2.4.

2.2.2.27 byte

The POSIX.2 standard states that "a byte is composed of a contiguous sequence of bits, the number of which is implementation defined."

OpenExtensions uses bytes composed of 8 bits.

2.2.2.61 extended security controls

The POSIX.2 standard states, "the access control ... and privilege ... mechanisms have been defined to allow implementation-defined extended security controls."

OpenExtensions allows an external security manager to define alternate or additional file access controls.

2.2.2.65 file

The POSIX.2 standard states that "other types of files may be defined by the implementation."

The OpenExtensions implementation also defines the symbolic link file type, the external link file type, and the socket file type.

2.2.2.68 file group class

The POSIX.2 standard states that "other members of the class may be implementation defined."

The OpenExtensions implementation defines no other members of the class.

2.2.2.93 job control

The POSIX.2 standard states, "POSIX.1-conforming implementations may optionally support job control facilities."

The OpenExtensions implementation supports job control.

2.2.2.120 parent process ID

The POSIX.2 standard states that "after the creator's lifetime has ended, the parent process ID is the process ID of an implementation-defined system process."

The OpenExtensions implementation makes the parent process the Init process, which is process ID 1.

2.2.2.121 pathname

The POSIX.2 standard states that "a pathname that begins with two successive slashes may be interpreted in an implementation-defined manner, although more than two leading slashes shall be treated as a single slash."

See POSIX.1 section 2.2.2.57.

2.2.2.141 read-only file system

The POSIX.2 standard states that a read-only file system is "a file system that has implementation-defined characteristics restricting modifications."

The OpenExtensions implementation supports read-only file systems, and read-only file systems cannot be modified.

2.2.2.189 variable assignment [assignment]

The sh utility of OpenExtensions supports subscripted variable assignment using the syntax name[expr]=value. For more information, see the entry for sh in [z/VM: OpenExtensions User's Guide](#) and [z/VM: OpenExtensions Commands Reference](#).

2.4 Character Set

The POSIX.2 standard states, "use of a locking-shift encoding with any of the standard utilities or the optional C-language functions (Annex B) that describe character (versus byte) or text-file manipulation is implementation-defined."

In the OpenExtensions implementation, only single-byte characters are supported. Multibyte characters, and thus locking shift encodings, are not currently supported.

2.4.1 Character Set Description File

The POSIX.2 standard states that "it is implementation defined whether or not users or applications can provide additional character set description files."

Because OpenExtensions does not define the symbolic constant {POSIX2_LOCALEDEF}, this section is not applicable.

The POSIX.2 standard states that "implementations supporting other byte sizes (other than 8-bit) may allow constants to represent values larger than those that can be represented in 8-bit bytes, and to allow additional digits in constants. ... The manner in which constants are represented in the character is implementation defined."

Current implementations of OpenExtensions support only 8-bit bytes.

2.5 Locale

The OpenExtensions implementation supports any locale that uses the IBM-1047, IBM-1027, or IBM-939 character set.

2.6 Environment Variables

The POSIX.2 standard states that, for each of the following environment variables, "additional semantics of this variable, if any, are implementation defined."

LANG
LC_COLLATE
LC_CTYPE
LC_MESSAGES
LC_MONETARY
LC_NUMERIC
LC_TIME

In the OpenExtensions implementation, there are no additional semantics for locale environment variables.

The POSIX.2 standard states that "if **PATH** is unset or is set to null, the path search is implementation defined."

OpenExtensions searches the **/bin** directory.

The POSIX.2 standard states that "if the **LANG** variable is not set or is set to the empty string, the implementation-defined default locale shall be used."

In the OpenExtensions implementation, the POSIX Locale with the name "C" is the default locale.

The POSIX.2 standard states that "if **LANG** (or any of the **LC_*** environment variables) contains one of a set of implementation-defined values, the standard utilities shall behave in accordance with the rules in a corresponding implementation-defined locale description for the associated category."

No OpenExtensions implementation-defined values are supported.

2.9 Dependencies on Other Standards

2.9.1 Features Inherited from POSIX.1

2.9.1.4 File Read, Write, and Creation

The POSIX.2 standard states that "for other file types, the effect is implementation defined."

In the OpenExtensions implementation, symbolic links can be created only with the symlink callable service (BPX1SYM) and be read by the readlink callable service (BPX1RDL). They are deleted or renamed the same as files are.

2.9.1.5 File Removal

The POSIX.2 standard states that "when a directory that is the root directory or current working directory of any process is removed, the effect is implementation defined."

See POSIX.1 section 5.5.2.2, Description (in "Remove a Directory").

2.11 Utility Description Defaults

2.11.5 External Influences

2.11.5.2 Input Files

The POSIX.2 standard states that "implementations shall define ... those utilities that are limited by constraints other than file system space, available memory, and other limits specifically cited by this

standard, and identify what the constraint is, and indicate a way of estimating when the constraint would be reached."

Utilities that use regular expressions and the `regcomp()` and `regexec()` functions (for example, `ed`, `sed`, and `awk`) use a preallocated *backtrack stack* to improve performance. This stack is sufficiently large that any pattern matched against a string of `{LINE_MAX}` characters or less should not cause it to overflow. For larger strings, it is possible to construct a regular expression that would cause possible backtracking decisions to overflow this stack.

The `join` utility supports up to 256 input fields, and a maximum of 512 output fields can be specified in the "`-o`" argument list.

The `ed` utility can only edit a file that has less than 500⁺000 lines, and has a limit of `{LINE_MAX}` characters in the global command string and in the remembered regular expression string.

The `awk` utility limits each input record to a length of 20⁺000 characters, limits the number of fields in a record to 4000, only allows up to 32 occurrences of the "`-f`" option, and limits the recursion level to 3000.

The `sed` utility provides a pattern space buffer of size $(5 \times \{LINE_MAX\})$ with a limit of 8192 characters.

2.13 Configuration Values

2.13.1 Symbolic Limits

The POSIX.2 standard describes, in Table 2-17, the minimum values for utility limits (such as `POSIX2_BC_DIM_MAX`, the "maximum number of elements permitted in any array by the `bc` utility") and states that "implementations may provide more liberal, or less restrictive, values than shown in Table 2-17. These possibly more liberal values are accessible using the symbols in Table 2-18."

Table 12. POSIX.2 Standard 2.13.1: Table 2-18: Symbolic Utility Limits

Symbolic Limit	Description	Min. Value	OpenExtensions Value
<code>{BC_BASE_MAX}</code>	The largest <i>obase</i> value allowed by the <code>bc</code> utility.	99	<code>{SHRT_MAX}</code> 32 ⁺ 767
<code>{BC_DIM_MAX}</code>	The maximum number of elements permitted in an array by the <code>bc</code> utility.	2048	<code>{SHRT_MAX}</code> 32 ⁺ 767
<code>{BC_SCALE_MAX}</code>	The largest <i>scale</i> value allowed by the <code>bc</code> utility.	99	<code>{SHRT_MAX}</code> 32 ⁺ 767
<code>{BC_STRING_MAX}</code>	The maximum length of a string constant accepted by the <code>bc</code> utility.	1000	2048
<code>{COLL_WEIGHTS_MAX}</code>	The maximum number of weights that can be assigned to an entry of the <code>LC_COLLATE</code> <code>order</code> keyword in the locale definition file.	2	2
<code>{EXPR_NEST_MAX}</code>	The largest number of expressions that can be nested within parentheses by the <code>expr</code> utility. Because OpenExtensions implements <code>expr</code> using the <code>yacc</code> utility, this value is the depth of the <code>yacc</code> stack.	32	32
<code>{LINE_MAX}</code>	The maximum length, in bytes, of a utility's input line for processing text files.	2048	2048

Table 12. POSIX.2 Standard 2.13.1: Table 2-18: Symbolic Utility Limits (continued)

Symbolic Limit	Description	Min. Value	OpenExtensions Value
{RE_DUP_MAX}	The largest number of repeated occurrences of a regular expression permitted when using the interval notation $\{m,n\}$. The OpenExtensions implementation is unlimited (within underlying system memory resources).	255	255

Table 12 on page 44, based on Table 2-18 in the POSIX.2 standard, describes the corresponding symbolic limits, the minimum values dictated by the POSIX.2 standard, and the values provided by OpenExtensions. These values are retrieved by means of the `getconf` utility, described in section 4.26 of the POSIX.2 standard.

2.13.2 Symbolic Constants for Portability Specifications

The POSIX.2 standard states that "Table 2-19 lists symbols that can be used by the application to determine which optional facilities are present on the implementation. ... Each shall be defined on the system with a value of 1 if the corresponding option is supported; otherwise, the symbol shall be undefined."

Table 13. POSIX.2 Standard 2.13.2: Table 2-19: Optional Facility Configuration Values

Symbolic Limit	Description	OpenExtensions Value
{POSIX2_C_BIND}	The C Language development facilities in Annex A support the C language Bindings Option (see POSIX.2 Annex B).	1
{POSIX2_C_DEV}	The system supports the C Language Development Utilities Option (see POSIX.2 Annex A).	1
{POSIX2_FORT_DEV}	The system supports the FORTRAN Development Utilities Option (see POSIX.2 Annex C).	Undefined
{POSIX2_FORT_RUN}	The system supports the FORTRAN Runtime Utilities Option (see POSIX.2 Annex C).	Undefined
{POSIX2_LOCALEDEF}	The system supports the creation of locales, as described in POSIX.2 section 4.35.	Undefined
{POSIX2_SW_DEV}	The system supports the Software Development Utilities Option (see POSIX.2 section 6).	1
{POSIX2_UPE}	The system supports the User Portability Utilities Option (see POSIX.2 section 5).	1

2.14 Terminal Characteristics

The POSIX.2 standard states that the implementation shall document which terminal type it supports and which of these features and utilities are not supported by each terminal. This implementation-defined list of terminals:

- Shall include at least one terminal type that is capable of supporting all of the standard utilities and all of their features, if the **{POSIX2_CHAR_TERM}** option is provided.
- May group terminals in terms of families or equivalences to other documented terminal types.
- Need not consist of an exhaustive list of terminal modes when the implementer considers that some terminal types are used too infrequently to be listed.

The OpenExtensions implementation supports the TTY terminal type running in canonical mode.

Section 3. Shell Command Language

3.5 Parameters and Variables

3.5.3 Variables

The POSIX.2 standard states under the variable **PS1**, "for users who have specific additional implementation-defined privileges ... the default may be another, implementation-defined, value."

If a user's effective user ID has the value of 0, the default value of **PS1** changes from "\$ " to "# "

3.6 Word Expansions

The POSIX.2 standard states that "if an unquoted \$ is followed by a character that is either not numeric, the name of one of the special parameters (see 3.5.2), a valid first character of a variable name, a left curly brace {}, or a left parenthesis, the result is unspecified."

The sh utility of OpenExtensions allows arithmetic substitution with the syntax "\$[arithmetic expression]." This sequence is replaced with the value of *arithmetic expression*. For more information, see the entry for sh in [z/VM: OpenExtensions Commands Reference](#) and the discussion in [z/VM: OpenExtensions User's Guide](#).

3.7 Redirection

The POSIX.2 standard states that for file descriptors (the decimal numbers [starting with zero] that represent open files), "it is implementation defined what the largest value can be."

In the OpenExtensions implementation, the largest file descriptor value is 1023.

Section 4. Execution Environment Utilities

4.1 awk – Pattern Scanning and Processing Language

4.1.7 Extended Description

4.1.7.6 Actions

4.1.7.6.2 Functions

4.1.7.6.2.3 Input/Output and General Functions

The POSIX.2 standard states that "the limit on the number of open *expression* arguments [to the *close* function] is implementation defined."

In the OpenExtensions implementation, the default value is 64.

4.1.7.8 awk Lexical Conventions

The POSIX.2 standard states (in the \ddd entry of Table 4-2): "If the size of a byte on the system is greater than nine bits, the valid escape sequence used to represent a byte is implementation defined."

The awk utility of OpenExtensions does not currently support byte sizes that are different from 8 bits.

4.2 basename – Return Nondirectory Portion of Pathname

4.2.2 Description

The POSIX.2 standard states that "if *string* is //, it is implementation defined whether steps (2) through (5) are skipped or processed."

The basename utility of OpenExtensions takes no special action when *string* is //.

4.5 cd – Change Working Directory

4.5.2 Description

The POSIX.2 standard states that "if **HOME** is empty or is undefined, the default behavior is implementation defined."

The cd utility of OpenExtensions issues an error message stating that **HOME** is unset.

4.5.4 Operands

The POSIX.2 standard states that "if *directory* is –, the results are implementation defined."

The cd utility of OpenExtensions changes *directory* to the contents of **\$OLDPWD**.

4.7 chmod – Change File Modes

4.7.2 Description

The POSIX.2 standard states that "it is implementation defined whether and how the chmod utility affects any alternate or additional file access control mechanism ... being used for the specified file."

OpenExtensions defines no alternate or additional file access control mechanisms.

4.7.7 Extended Description

The POSIX.2 standard states that "when using the symbolic mode form on a regular file, it is implementation defined whether or not:

(1)

Requests to set the set-user-ID-on-execution or set-group-ID-on-execution bit when all execute bits are currently clear and none are being set are ignored,

(2)

Requests to clear all execute bits also clear the set-user-ID-on-execution and set-group-ID-on-execution bits, or

(3)

Requests to clear the set-user-ID-on-execution or set-group-ID-on-execution bits when all execute bits are currently clear are ignored."

OpenExtensions does not ignore requests [as in (1) and (3)] or clear bits [as in (2)].

The POSIX.2 standard states that "when using the symbolic mode form on other file types [other than a regular file], it is implementation defined whether or not requests to set or clear the set-user-ID-on-execution or set-group-ID-on-execution bits are honored."

OpenExtensions does not ignore requests [as in (1) and (3)] or clear bits [as in (2)].

The POSIX.2 standard states that "for each bit set in the octal number, the corresponding file permission bit ... shall be set; all other file permission bits shall be cleared ... For other file types [other than regular], it is implementation defined whether or not requests to set or clear the set-user-ID-on-execution or set-group-ID-on-execution bits are honored."

OpenExtensions does not ignore the settings of the S_ISUID or S_ISGID bits.

4.13 cp – Copy Files

4.13.2 Description

The POSIX.2 standard states in (2)(c) that for each *source_file* of type directory, "if *dest_file* exists and it is a file type not specified by POSIX.1 {8}, the behavior is implementation defined."

If the destination file type is a symbolic link, and the link points to a directory, then the contents of the source directory will be copied into the directory pointed to by the symbolic link.

The POSIX.2 standard states in step (4)(a) that "if the -r option was specified, the behavior is implementation defined."

The cp utility of OpenExtensions defines the behavior of the -r option as being similar to the behavior of the -R option except when copying special files. The -r option actually tries to read special files, whereas -R re-creates them. For example, if the -R option is specified, and the source file is of type FIFO, the destination is another file of type FIFO. If, instead, the -r option is specified, the destination file will be a regular file, consisting of the contents of the FIFO.

The POSIX.2 standard states in step (4)(b)[2] that with the -R option, if the *source_file* is not of type FIFO, "... the permissions, owner ID, and group ID of *dest_file* are implementation defined."

The cp utility of OpenExtensions sets the *dest_file* privileges to those of the *source_file* and the owner ID and group ID to the current effective user and group IDs.

The POSIX.2 standard states that "if the implementation provides additional or alternate access control mechanisms ... their effect on copies of files is implementation defined."

The cp utility of OpenExtensions defines no alternate or additional file access control mechanisms.

4.13.3 Options

The POSIX.2 standard states that under the **-p** option "other, implementation-defined bits may be duplicated as well."

The cp utility of OpenExtensions duplicates all of the bits of *st_mode* from the stat() function (as described in POSIX.1 section 5.6.2, "Get File Status"). cp duplicates the "sticky bit". See "[Modes and the Sticky Bit](#)" on page 71.

The POSIX.2 standard states that with the **-r** option, "the treatment of special files is implementation defined."

With **-r**, cp attempts to open the special file, and copy its contents. For example, if the **-R** option is specified, and the source file is of type FIFO, the destination is another file of type FIFO. If, instead, the **-r** option is specified, the destination file will be a regular file, consisting of the contents of the FIFO.

4.18 dirname – Return Directory Portion of Pathname

4.18.2 Description

The POSIX.2 standard states that "if the remaining *string* is //, it is implementation defined whether steps (7) and (8) are skipped or processed."

The dirname utility of OpenExtensions skips steps (7) and (8) when converting a *string* of // to a file name; that is, dirname // converts to //.

4.19 echo – Write Arguments to Standard Output

4.19.4 Operands

The POSIX.2 standard states that "if the first operand is **-n** or if any of the operands contain a backslash (\) character, the results are implementation defined."

- The echo utility of OpenExtensions takes no special action for **-n**: the text is echoed directly.
- OpenExtensions supports the historical SVID functionality as an extension to the standard, which includes the following escape sequences in the echo operands:

- **\a**
Write an <alert> character.
- **\b**
Write a <backspace> character.
- **\c**
Suppress the <newline> character that otherwise follows the final argument in the output, with everything after \c in input being ignored.
- **\f**
Write a <form-feed> character.
- **\n**
Write a <newline> character.
- **\r**
Write a <carriage-return> character.
- **\t**
Write a <tab> character.

\v	Write a <vertical-tab> character.
\\"	Write a backslash character.
\Onum	Write an 8-bit value that is the 1-, 2-, or 3-digit octal number <i>num</i> .
\X	When <i>X</i> is not one of the preceding characters, the echo utility of OpenExtensions simply echoes it.

4.20 ed – Edit Text

4.20.7 Extended Description

4.20.7.3 ed Commands

4.20.7.3.13 List Command

The POSIX.2 standard states that "if the size of a byte on the system is greater than 9 bits, the format used for nonprintable characters is implementation defined."

The ed utility of OpenExtensions does not currently support byte sizes that are different from 8 bits.

4.24 find – Find Files

4.24.4 Operands

The POSIX.2 standard states that for the -exec operand "if a *utility_name* or *argument* string contains the two characters { }, but not just the two characters { }, it is implementation defined whether find replaces those two characters with the current path name or uses the string without change."

The find utility of OpenExtensions uses the string without change.

4.33 ln – Link Files

4.33.2 Description

The POSIX.2 standard states that "if the last operand specifies an existing file of a type not specified by POSIX.1 {8}, the behavior is implementation defined."

If the file type is not specified by POSIX.1 {8}, it is treated as a nondirectory file.

4.33.4 Operands

The POSIX.2 standard states that "whether a directory can be linked is implementation defined."

In the OpenExtensions implementation, directories cannot be linked.

4.34 locale – Get Locale-Specific Information

4.34.3 Options

The POSIX.2 standard states that under the -a option, "the manner in which the implementation determines what other locales are available is implementation defined."

- `locale -a` searches the directory **/usr/lib/nls/locale** for any locale file names to list.

Note: The compiled locales reside in the SCEERUN LOADLIB.

4.34.4 Operands

The POSIX.2 standard states that "it is implementation defined whether any keyword values are written for the categories **LC_CTYPE** and **LC_COLLATE**."

For **LC_CTYPE**, the `locale` utility of OpenExtensions displays the full **CTYPE** character classes and mapping values. For **LC_COLLATE**, it does not write any values.

4.35 `localedef` – Define Locale Environment

4.35.2 Description

The POSIX.2 standard states "it is implementation defined whether users shall have the capability to create new locales, in addition to those supplied by the implementation. If the symbolic constant {POSIX2_LOCALEDEF} is defined, the system supports the creation of new locales."

Because OpenExtensions does not define the symbolic constant {POSIX2_LOCALEDEF}, this section is not applicable.

The POSIX.2 standard states that "in addition [to the categories specified by the POSIX.2 standard] the input may contain source for implementation-defined categories."

Because OpenExtensions does not define the symbolic constant {POSIX2_LOCALEDEF}, this section is not applicable.

4.35.3 Options

The POSIX.2 standard states that "if the `-f` option is not present, an implementation-defined default *charmap* file shall be used."

In the OpenExtensions implementation, if `-f` is not specified, the IBM-1047 *charmap* is used.

4.35.4 Operands

The POSIX.2 standard states that "if *name* does not contain any slash characters, the interpretation of the name is implementation defined and the locale shall be public. This capability may be restricted to users with appropriate privileges."

Because OpenExtensions does not define the symbolic constant {POSIX2_LOCALEDEF}, this section is not applicable.

4.35.9 Consequences of Errors

The POSIX.2 standard states that "other implementation-defined conditions can also cause warnings."

Because OpenExtensions does not define the symbolic constant {POSIX2_LOCALEDEF}, this section is not applicable.

4.36 `logger` – Log Messages

4.36.2 Description

The POSIX.2 standard states, "it is implementation defined whether messages written in locales other than the POSIX Locale are effective."

Messages in `logger` are just treated as a sequence of bytes. If the output destination is `stderr` (for example, if the `-s` option is specified by the user), no attempt is made to do any type of code conversion, and the sequence of bytes are written unmodified.

Timestamps are always in the POSIX locale.

4.39 ls – List Directory Contents

4.39.3 Options

The POSIX.2 standard states that "entries beginning with a period (.) shall not be listed unless explicitly referenced, the `-a` option is supplied, or an implementation-defined condition causes them to be listed."

The `ls` utility of OpenExtensions includes an `-f` option, which also lists entries beginning with a period (.).

4.39.5 External Influences

4.39.5.3 Environment Variables

The POSIX.2 standard states that "if **COLUMNS** is not set or invalid, an implementation-defined number of column positions shall be assumed, based on the implementation's knowledge of the output device."

The `ls` utility of OpenExtensions obtains the numbers of columns as described in "Window Size" on [page 71](#).

4.39.6 External Effects

4.39.6.1 Standard Output

The POSIX.2 standard states that "if the output is to a terminal, the format is implementation defined."

The `ls` utility of OpenExtensions uses a multicolumn format, as if the user specified `-C`.

The POSIX.2 standard states that "if the file is a character special or block special file, the size of the file may be replaced with implementation-defined information associated with the device in question."

The `ls` utility of OpenExtensions replaces the file size with major and minor device numbers of the file and displays them with the format "%u, %u".

The POSIX.2 standard states that "implementations may add other characters to this list [of *entry type* characters] to represent other, implementation-defined, file types."

The `ls` utility of OpenExtensions also recognizes the *entry types* of x for "none of the above," l for "symbolic links," E for "external links", and s for "socket files".

The POSIX.2 standard states that "implementations may add other characters to this list [of *owner, group, and other permissions*] for the third character position."

The `ls` utility of OpenExtensions uses T and t to designate the "sticky bit" if it is set in the mode returned by `stat()`. See "Modes and the Sticky Bit" on [page 71](#).

4.40 mailx – Process Messages

4.40.4 Operands

The POSIX.2 standard states that "an implementation-defined way for a user with a login-name address to retrieve the message shall be provided by the implementation."

The mail retrieval capabilities in the mailx utility of OpenExtensions are as described in that standard, section 4.41, "mailx Interactive Message Processing System." For details, see the entry on mailx in z/VM: *OpenExtensions Commands Reference*.

4.40.6 External Effects

4.40.6.3 Output Files

The POSIX.2 standard states that "when a message from the system mailbox or entered by the user is not a text file, it is implementation defined how such a message is stored in files written by mailx."

The mailx utility of OpenExtensions does not directly support binary file transfer.

4.40.7 Extended Description

4.40.7.1 Internal Variables

The POSIX.2 standard states that if the *crt* variable "is set to null, the value used is implementation defined."

If the *crt* variable is set to null, the mailx utility of OpenExtensions treats it the same as a setting of 0: It pipes all messages through **PAGER**.

4.40.7.3 Command Escapes

The POSIX.2 standard states that, on terminals, the ~h command escape prompts for a Subject line and the To, Cc, and Bcc lists and that "other implementation-defined headers may be presented for editing."

The mailx utility of OpenExtensions presents no additional headers for editing.

4.43 mv – Move Files

4.43.2 Description

The POSIX.2 standard states that "if any operand specifies an existing file of a type not specified by POSIX.1 {8}, the behavior is implementation defined."

If the source file is of type directory, and the destination file type is not specified by POSIX.1, an error is given.

The OpenExtensions implementation supports the symbolic-link file type. The behavior of mv is to refer to the symbolic link file itself when validating the existence of the *source* file arguments and to refer to the file to which the symbolic link points when validating and referring to the *target* argument.

4.45 od – Dump Files in Various Formats

4.45.7 Extended Description

The POSIX.2 standard states that "the default number of bytes transformed by output type specifiers d, f, o, u, and x shall correspond to the various C-language types as follows. If the c89 compiler is present on the system, these specifiers shall correspond to the sizes used by default in that compiler. Otherwise, these sizes are implementation defined." The POSIX.2 standard expands on this with the explanation that "for the type specifier characters d, o, u, and x, the default number of bytes shall correspond to the size of the underlying implementation's basic integral data type," and "for the type specifier character f, the default number of bytes shall correspond to the number of bytes in the underlying implementation's basic double precision floating point data type."

- As specified by the POSIX.2 standard, the od utility of OpenExtensions bases the default number of bytes transformed by the specifiers d, o, u, and x on the C-language type *int*. It also supports *char*, *short*, and *long* types.

For the "c89 compiler," the number of bytes corresponding to the these C-language types are as follows:

- char*: 1 byte
- short*: 2 bytes
- int*: 4 bytes
- long*: 4 bytes

The od utility of OpenExtensions bases the number of bytes that the f specifier transforms on the C-language type *double*. It also supports *float* and *long double* types for the number of bytes in this identifier.

For the "c89 compiler," the number of bytes corresponding to the various C-language types are as follows:

- float*: 4 bytes
- double*: 8 bytes
- long double*: 16 bytes

The POSIX.2 standard states that "for these specifier characters [d, o, u, and x], the implementation shall support values of the optional number of bytes to be converted corresponding to the number of bytes in the C-language types *char*, *short*, *int*, and *long*. The byte order used when interpreting numeric values is implementation defined, but shall correspond to the order in which a constant of the corresponding type is stored in memory on the system."

OpenExtensions treats the first byte as the most significant byte, the second byte as the next most significant, and so on.

The POSIX.2 standard states that "if the size of a byte on the system is greater than 9 bits, the format used for nonprintable characters is implementation defined."

The od utility of OpenExtensions uses a byte size of 8 bits, so this is not relevant.

The POSIX.2 standard states that "when either the -j *skip* or -N *count* option is specified along with the c type specifier, and this results in an attempt to start or finish in the middle of a multibyte character, the result is implementation defined."

If the -j option is used and depending if the starting byte is not the first byte of a character, the od utility of OpenExtensions will result in a misinterpretation of that and subsequent characters. This misinterpretation will continue until od encounters a <newline>, at which point it is once again synchronized with the first byte of a multibyte character.

If od -N is being used to process a mulitbyte character when it encounters the last byte, which is not the last byte of a character, od displays ??? rather than this character.

4.48 pax – Portable Archive Interchange

4.48.2 Description

The POSIX.2 standard states that "if the destination directory is a file of a type not defined by POSIX.1 {8}, the results are implementation defined."

In this situation, the pax utility of OpenExtensions issues a diagnostic message.

The POSIX.2 standard states that "the default output archive format shall be implementation defined."

The pax utility of OpenExtensions uses the extended tar format, as described in POSIX.1 section 10.1.1, "Extended tar Format."

The POSIX.2 standard states that "the pax utility shall determine, in an implementation-defined manner, what file to read or write as the next file."

- Upon encountering one of the following:
 - End-of-file condition
 - Error on reading file
 - Error on writing file
 - Partial write on writing file

the pax utility of OpenExtensions either uses the file pattern given as an argument to the -V option and waits for the user to press the <Enter> key after exchanging media; or it prompts the user for a new device name.

4.48.3 Options

The POSIX.2 standard states that with the -a option, "it is implementation defined which devices on the system support appending."

The pax utility of OpenExtensions supports append on all regular files.

The POSIX.2 standard states that for the -p option, "the string shall consist of the specification characters a, e, m, o, and p and/or other, implementation-defined characters."

The pax utility of OpenExtensions supports no additional characters.

The POSIX.2 standard states that under the -p option, the specification character e shall "preserve the user ID, group ID, file mode bits ... access time, modification time, and any other, implementation-defined, file characteristics."

The pax utility of OpenExtensions supports no additional file characteristics.

The POSIX.2 standard states that under the -p option, the specification character p shall "preserve the file mode bits. Other, implementation-defined file-mode attributes may be preserved."

The pax utility of OpenExtensions also preserves the "sticky bit". See "[Modes and the Sticky Bit](#)" on [page 71](#).

The POSIX.2 standard states that under the -x option, "implementation-defined formats shall specify a default block size as well as any other block sizes supported for character special archive files."

The pax utility of OpenExtensions supports two implementation-defined formats: tar format and cpio format. The default block sizes for the tar and cpio formats are the same as for ustar and cpio formats, respectively.

4.48.5 External Influences

4.48.5.2 Input Files

The POSIX.2 standard states that "the input file ... shall be a file formatted according to one of the specifications in POSIX.1 {8} 10.1, or some other, implementation-defined, format."

The pax utility of OpenExtensions supports both of the file formats specified in POSIX.1 (that is, extended tar format and extended cpio format). In addition, it supports a binary format cpio (select -x cpio option), a tar format (select -x tar option), and compressed versions of both formats (select -z option in addition to -x). Compressed format files are identical to files upon which the compress utility has been used.

4.48.6 External Effects

4.48.6.1 Standard Output

The POSIX.2 standard states that "if the **-w** option is specified and neither the **-f** nor the **-x** options are specified, the standard output shall be the archive formatted according to one of the specifications in POSIX.1 {8} 10.1, or some other implementation-defined format."

The pax utility of OpenExtensions uses the extended **tar** format in this case.

4.48.6.3 Output Files

The POSIX.2 standard states that "if the **-w** option is specified, and neither the **-f** nor **-x** are specified, the standard output shall be the archive formatted according to one of the specifications in POSIX.1 {8} 10.1, or some other implementation-defined format."

The pax utility of OpenExtensions uses the extended **tar** format in this case.

4.55 sed – Stream Editor

4.55.7 Extended Description

4.55.7.3 Editing Commands

The POSIX.2 standard states that under the **l** command, "if the size of a byte on the system is greater than 9 bits, the format used for nonprintable characters is implementation defined."

The sed utility of OpenExtensions uses a byte size of 8 bits, so this is not relevant.

4.56 sh – Shell, the Standard Command Language Interpreter

The POSIX.2 standard states that under the variable **PS1**, "for users who have specific additional implementation-defined privileges (see 2.2.2.8), the default may be another implementation-defined, value."

If a user's effective user ID has the value zero (0), then the default value of **PS1** changes from "\$ " to "# ".

4.59 stty – Set the Options for a Terminal

4.59.2 Description

The POSIX.2 standard states that "without options or operands specified, it shall report the settings of certain characteristics, usually those that differ from implementation-defined defaults."

Refer to ["7.1.2 Parameters That Can Be Set" on page 28](#) in the POSIX.1 part of this document for the modes that are on by default.

4.59.4 Operands

4.59.4.6 Combination Modes

The POSIX.2 standard states under the definition of the *sane* mode: "Reset all modes to some reasonable, unspecified, values."

The following modes are those that are set on reset: **opost**, **isig**, **echo**, **echok** and **echoe**.

For special control characters that can be set or reset, see [Table 10 on page 29](#).

4.62 test – Evaluate Expression

4.62.4 Operands

- The test utility of OpenExtensions provides the following additional primaries:

-a

True if both *expression1* and *expression2* are true.

-o

True if either *expression1* and *expression2* is true.

()

Parentheses allow primaries to be grouped as single expressions, for use with the -a and -o primaries.

-k

True if the "sticky bit" is on (see ["Modes and the Sticky Bit" on page 71](#)).

-nt

True if **file1** is newer than **file2**.

-ot

True if **file1** is older than **file2**.

-ef

True if **file1** has the same device and inode as **file2**; that is, they are the same file.

-L

True if **file** is a symbolic link.

-h

True if **file** is a hard link.

See [z/VM: OpenExtensions Commands Reference](#) for details.

OpenExtensions test provides no additional operators.

4.63 touch – Change File Access and Modification Times

4.63.3 Options

The POSIX.2 standard states that "the range of valid times past the Epoch is implementation defined."

In the OpenExtensions implementation, this item depends on the size of a **time_t** structure, which is 4 bytes.

4.64 tr – Translate Characters

4.64.7 Extended Description

The POSIX.2 standard states that when using the *octal* convention to specify characters or collating elements, "if the size of a byte on a system is greater than 9 bits, the valid escape sequence used to represent a byte is implementation defined."

The **tr** utility of OpenExtensions uses a byte size of 8 bits, so this is not relevant.

4.68 uname – Return System Name

4.68.2 Description

The POSIX.2 standard states that "when options are specified, symbols representing one or more system characteristics shall be written to the standard output. The format and contents of the symbols are implementation defined."

The uname utility of OpenExtensions supports the five fields described in the POSIX.1 standard, section 4.4.1.2, "Description" [of `uname()`], and displays the requested fields in the order in which the POSIX.1 standard describes them. For more information, refer to "[4.4.1 Get System Name](#)" on page 15 in the POSIX.1 part of this document.

4.68.6 External Effects

4.68.6.1 Standard Output

The POSIX.2 standard states that "additional implementation-defined symbols may be written."

The uname utility of OpenExtensions writes no additional symbols.

Section 5. User Portability Utilities Option

5.12 fc – Process Command History List

5.12.2 Description

The POSIX.2 standard states that "when the number reaches an implementation-defined upper limit, which shall be no smaller than the value in **HISTSIZE** or 32⁵767 (whichever is greater), the shell may wrap the numbers starting the next command with a lower number (usually 1)."

The upper limit is the maximum positive value of an integer, which is 2³¹147⁴83⁵647.

5.12.5 External Influences

5.12.5.3 Environment Variables

The POSIX.2 standard states that, regarding the **HISTFILE** variable, "an implementation may choose to access this variable only when initializing the history file; this initialization shall occur when fc or sh first attempts to retrieve entries from, or add entries to, the file, as the result of commands issued by the user, the file named by the ENV variable, or implementation-defined system startup files. Therefore, it is implementation defined whether changes made to **HISTFILE** after the history file has been initialized are effective."

The **HISTFILE** environment variable is examined only when the history file is opened for the first time.

The POSIX.2 standard states that "implementations may choose to disable the history list mechanism for users with appropriate privileges who do not set **HISTFILE**; the specific circumstances under which this will occur are implementation defined."

There are no circumstances under which OpenExtensions disables the history list mechanisms.

The POSIX.2 standard states that, regarding the **HISTSIZE** variable, "an implementation may choose to access this variable only when initializing the history file, as described under **HISTFILE**. Therefore, it is implementation defined whether changes made to **HISTSIZE** after the history file has been initialized are effective."

The **HISTSIZE** environment variable is examined only when the history file is opened for the first time.

5.19 newgrp – Change to a New Group

The POSIX.2 standard states that, On systems where the supplementary group list also contains the new effective group ID, or where the previous effective group ID was actually in the supplementary group list:

- If the supplementary group list also contains the new effective group ID, newgrp changes the effective group ID.
- If the supplementary group list does not contain the new effective group ID, newgrp adds it to the list (if there is room).

The newgrp utility of OpenExtensions implements this behavior.

5.23 ps – Report Process Status

5.23.2 Description

The POSIX.2 standard states that "when the -o option is not specified, information about processes selected shall be written in an implementation-defined manner."

- In the OpenExtensions implementation, the default formats are:

```
For ps:   PID   TTY   TIME   COMMAND
For ps -f:  UID   PID   PPID   STIME   TTY   TIME   COMMAND
For ps -j:  PID   SID   PGRP   TTY   TIME   COMMAND
For ps -l: STATE UID   PID   PPID   NI SZ TTY TIME COMMAND
```

5.23.3 Options

The POSIX.2 standard states that with the -t option, "terminal identifiers shall be given in an implementation-defined format."

The format implemented by OpenExtensions uses the terminal (tty) device name—for example **/dev/tty**. This is the same format returned by the **tty** command. Using the **tty** name without **/dev/** is also acceptable.

5.23.6 External Effects

5.23.6.1 Standard Output

The POSIX.2 standard states that "when the -o option is not specified, the standard output format is implementation defined."

See section "[5.23.2 Description](#)" on page 62.

The POSIX.2 standard states that the **args** format specifier may have its value truncated "to the field width; it is implementation defined whether any further truncation occurs."

The **ps** utility of OpenExtensions truncates the value of the **args** format specifier at 40 bytes.

The POSIX.2 standard states that "any implementation-defined variables shall specify in the conformance document if the field may contain <blank>s, as well as for the default header."

In the **ps** utility of OpenExtensions, no implementation-defined variable allows its field or its default header to contain blanks.

Section 6. Software Development Utilities Option

6.2 make – Maintain, Update, and Regenerate Groups of Programs

6.2.7. Extended Description

6.2.7.1 Makefile Syntax

The POSIX.2 standard states that "if neither **./makefile** nor **./Makefile** are found, other implementation-defined path names may also be tried."

The make utility of OpenExtensions permits you to specify alternative path names with the **.MAKEFILES** target.

6.2.7.2 Makefile Execution

The POSIX.2 standard states that "the macros from the command line to make shall be added to make's environment. Other implementation-defined variables may also be added to the environment."

- The make utility of OpenExtensions adds the following variables to the environment:

.EPILOG	MAKEFLAGS
.IGNORE	MAKESTARTUP
.PRECIOUS	MFLAGS
.PROLOG	NULL
.SETDIR	OS
.SILENT	OSRELEASE
DIRSEPSTR	OSVERSION
GROUPFLAGS	PWD
GROUPSHELL	SHELL
GROUPSUFFIX	SHELLMETAS
INCDEPTH	SHELLFLAGS
MAKECMD	SWITCHAR
MAKEDIR	

The POSIX.2 standard states that "if the **MAKEFLAGS** variable is not set ... it shall be created by make, and shall contain all options specified on the command line except for the -f and -p options. It may also contain implementation-defined options."

In the make utility of OpenExtensions, **MAKEFLAGS** can also contain the -E, -e, -V, -v, and -x options.

6.2.7.3 Target Rules

The POSIX.2 standard states that "the interpretation of targets containing the characters "%" and " is implementation defined."

The make utility of OpenExtensions treats targets containing "%" as metarules (rules for defining rules) unless the user specifies the **.POSIX** special target, in which case, it ignores metarules.

OpenExtensions make uses the " character in pairs for quoting, that is, it treats special characters contained within a " pair as though they had no special meaning.

6.2.7.4 Macros

The POSIX.2 standard states that "other effects of defining **SHELL** in the makefile or on the command line are implementation defined."

If the user specifies the **.POSIX** special target, **SHELL** has no special effect. If the user does not specify **.POSIX** and defines **SHELL** as a macro in the makefile, make uses the shell specified by **SHELL** but does not change the value of the **SHELL** environment variable in the environment passed to child processes unless the user specified the -x option. If the user does not specify **.POSIX** and includes **SHELL=shell_path** on the command line, make uses *shell_path* as its shell and also assigns it as the value of the **SHELL** environment variable in the environment passed to child processes.

Annex A. C Language Development Utilities Option

A.1 c89 – Compile Standard C Programs

A.1.2 Description

The POSIX.2 standard states that "it is unspecified whether the linking occurs entirely within the operation of c89; some systems may produce objects that are not fully resolved until the file is executed."

Objects are fully resolved by c89. However, instructions providing dynamic linkage to library functions are bound to the objects, rather than to the actual library functions.

The POSIX.2 standard states that "if the -c option is not specified, it is unspecified whether such .o files are created or deleted for the *file.o* operands."

file.o files are always created. They are deleted only if the corresponding compilation fails (even if only one *file.c* operand is specified).

A.1.3 Options

The POSIX.2 standard states that for the -g option "the nature of this information is unspecified, and may be modified by implementation-defined interactions with other options."

Symbolic information produced by the -g option is equivalent to the information produced by the C compiler options TEST(ALL) and GONUMBER.

The symbolic information produced by the -g option is not affected by any other option. However, if the -E option is specified, -g is ignored. Also, when both the -g and -s options are specified, the one specified last is honored.

The POSIX.2 standard states that for the -s option, "when both the -g and -s options are present, the action taken is unspecified."

When both the -g and -s options are specified, the one specified last is honored.

The POSIX.2 standard states that for the -o option, "if the -o option is present with -c or -E, the result is unspecified."

If the -c or -E option is also specified, validation of the form of the -o option-argument is still performed, but the output file is not otherwise used.

The POSIX.2 standard states that for the -D option, "additional implementation-defined *names* may be provided by the compiler."

- The following macros are automatically specified, but may be overridden by -D or -U options specifying the same *names*.

```
errno=(*__errno())
__OPEN_DEFAULT
__OPEN_VM
```

The POSIX.2 standard states that for the -L option, "if a directory specified by a -L option contains files named **libc.a**, **libm.a**, **libl.a**, or **liby.a**, the results are unspecified."

The operand values **-l c**, **-l m**, **-l l**, and **-l y** will be recognized and used when searching -L option directories. However, the usual places (which are represented as CMS native record files rather than regular files) will still be used for any symbols left unresolved. Usurped library functions will never affect the behavior of other library functions (you cannot expect that one library function will use another library function).

The POSIX.2 standard states that for the -O option, "the nature of optimization is unspecified."

The c89 defaults for optimization are the C compiler options OPTIMIZE(0) and NOINLINE(AUTO,REPORT,250,1000).

The c89 -O option results in C compiler options OPTIMIZE(1) and INLINE(NOAUTO,NOREPORT,250,1000).

The C OPTIMIZE option is always set according to the corresponding c89 option. The C INLINE option may be overridden using the c89 -W option.

The optimization techniques used are fully described in the chapter on optimization in *z/OS: XL C/C++ Programming Guide* (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/cbcpx01_v2r5.pdf).

A.1.4 Operands

The POSIX.2 standard states that for the *file.a* operand, "implementations may recognize implementation-defined suffixes other than .a as denoting object file libraries."

By default, no file suffixes other than .a are recognized as object file libraries; however, you can override the default by exporting an environment variable recognized by c89.

The POSIX.2 standard states that for the *file.o* operand, "implementations may recognize implementation-defined suffixes other than .o as denoting object files."

By default, no file suffixes other than .o are recognized as object files, but CMS native record files can be specified.

The POSIX.2 standard states that for path name operands, "the processing of other files is implementation defined."

By default, no file operand suffixes other than those stated in the POSIX.2 standard are recognized. As previously stated, CMS native record files are also recognized; they are specified with a leading double slash (//).

The POSIX.2 standard states that for the -l *library* operand (the letter ell), "implementations may recognize implementation-defined suffixes other than .a as denoting libraries."

By default, no file suffixes other than .a are recognized as libraries; however, CMS record files can be specified and must be C Object Libraries that are CMS TXTLIBs.

A.1.5 External Influences

A.1.5.2 Input Files

The POSIX.2 standard states that for input files, "implementations may supply additional utilities that produce files in these formats. Additional input file formats are implementation defined."

The user creates C source (text) files (using an editor).

The c89 command produces object files.

The ar command produces archive files (also called archive libraries).

The C370LIB utility produces C Object Libraries.

The CMS TXTLIB command creates TXTLIBs.

No additional utilities or input file formats are defined.

A.1.6 External Effects

A.1.6.1 Standard Output

The POSIX.2 standard states that for standard output, if more than one file operand ending in .c (or possibly other unspecified suffixes) is given, for each such file:

```
"%s:\n", <file>
```

may be written.

By default, no file suffixes other than .c are recognized; however, CMS record files can be specified.

The POSIX.2 standard states that for standard output messages, "these messages, if written, shall precede the processing of each input file; they shall not be written to the standard output if they are written to the standard error, as described in A.1.6.2."

c89 writes a message to **stderr** preceding the compilation of each file, when more than one file is being compiled.

A.1.6.2 Standard Error

The POSIX.2 standard states that for standard error, if more than one file operand ending in .c (or possibly other unspecified suffixes) is given, for each such file:

```
"%s:\n", <file>
```

may be written to allow identification of the diagnostic and warning messages with the appropriate input file.

By default, no file suffixes other than .c are recognized; however, CMS record files can be specified.

The POSIX.2 standard states that for standard error messages, "these messages, if written, shall precede the processing of each input file; they shall not be written to the standard error if they are written to the standard output, as described in A.1.6.1."

c89 writes a message to **stderr** preceding the compilation of each file, when more than one file is being compiled.

The POSIX.2 standard states that for standard error, "this utility may produce warning messages about certain conditions that do not warrant returning an error (nonzero) exit value."

c89 allows for a return code of greater than zero (0) from the compiler and prelinker. The maximum result allowed is four (4), which corresponds to a warning message. A result of zero (0) may also produce informational messages.

A.1.6.3 Output Files

The POSIX.2 standard states that for output files, "object files or executable files or both are produced in unspecified formats."

Object files are produced according to the rules of the C compiler. For more information, see [z/OS: XL C/C++ Programming Guide](#) (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/cbcpx01_v2r5.pdf).

Executable files are produced according to the rules of the CMS module generation process. For more information, see [z/VM: CMS Commands and Utilities Reference](#).

A.1.7 Extended Description

A.1.7.1 Standard Libraries

The POSIX.2 standard states that for the C functions standard library operand, -l c "if the status {of getconf} is nonzero, it is unspecified whether these functions are available."

`getconf _POSIX_VERSION` returns a zero status: All POSIX.1 functions are available.

`getconf _POSIX2_C_BIND` returns a nonzero status: Not all POSIX.2 C Language Bindings (functions) are available.

The POSIX.2 standard states that for the math functions standard library operand -l m, "an implementation may search this library in the absence of this operand."

The library containing the math functions is always searched, even if not specified.

The POSIX.2 standard states that for standard libraries, "it is unspecified whether the libraries **libc.a**, **libm.a**, **lbl.a**, and **liby.a** exist as regular files."

All the standard libraries specified exist as regular files. However, **libc.a** and **libm.a** are empty and CMS record files are used to resolve those library functions.

The POSIX.2 standard states that for the library operand, "the implementation may accept as -l operands names of objects that do not exist as regular files."

c89 accepts as -l operands only the names of libraries that exist as a regular files.

A.1.7.2 External Symbols

The POSIX.2 standard states that for external symbol support, "the C compiler and link editor shall support the significance of external symbols up to a length of at least 31 bytes; the action taken upon encountering symbols exceeding the implementation-defined maximum symbol length is unspecified."

The C compiler determines how any external symbols exceeding the implementation maximum are handled. The maximum supported value is 255 characters. A duplicate symbol error results if there is a collision beyond the implementation maximum, and a nonzero exit value results.

The POSIX.2 standard states that for external symbol support, "the compiler and link editor shall support a minimum of 511 external symbols per source or object file, and a minimum of 4095 external symbols total. A diagnostic message shall be written to the standard output if the implementation-defined limit is exceeded; other actions are unspecified."

The c89 command supports a maximum of at least 65,535 symbols, both for each source or object file, and also for the total number of symbols of an executable file. The actual limit is dependent on the amount of storage available to the compiler and linkage editor (prelinker and CMS module build process), and so may be less. If the limit is exceeded, an appropriate error message is written to the standard error, and a nonzero exit value results.

A.2 lex – Generate Programs for Lexical Tasks

A.2.6 External Effects

A.2.6.1 Standard Output

The POSIX.2 standard states that if the -t option is not specified:

- "Implementation-defined information, error, and warning messages concerning the contents of lex source code input shall be written to either the standard output or standard error."

The lex utility of OpenExtensions writes all information, error, and warning messages to the standard error.

- "If the -v option is specified and the -n option is not specified, lex statistics shall also be written to either the standard output or standard error, in an implementation-defined format."

OpenExtensions writes lex statistics to the standard error.

A.2.6.2 Standard Error

The POSIX.2 standard states that "if the -t option is specified, implementation-defined informational, error, and warning messages concerning the contents of lex source code input shall be written to the standard error."

The lex utility of OpenExtensions writes all information, error, and warning messages to the standard error.

The POSIX.2 standard states that if the -t option is not specified:

- "Implementation-defined information, error, and warning messages concerning the contents of lex source code input shall be written to either the standard output or standard error."

The lex utility of OpenExtensions writes all information, error, and warning messages to the standard error.

- "If the -v option is specified and the -n option is not specified, lex statistics shall also be written to either the standard output or standard error, in an implementation-defined format."

OpenExtensions writes the lex statistics to the standard error.

A.2.7 Extended Description

The POSIX.2 standard states that "as explained in A.2.7.1, the type can be explicitly selected using the %array or %pointer declarations, but the default is implementation defined."

By default, the lex utility of OpenExtensions behaves as if the user had explicitly selected the %array declaration.

A.2.7.1 Definitions

The POSIX.2 standard states that "the default type of yytext is implementation defined."

In the lex utility of OpenExtensions the default type of yytext is char[].

The POSIX.2 standard states that (for Table A-1 – lex Table Size Declarations in the standard) "the exact meaning of these table size numbers is implementation defined. The implementation shall document how these numbers affect the lex utility and how they are related to any output that may be generated by the implementation should space limitations be encountered during the execution of lex."

- Only three of the table size numbers represent actual fixed limits (%e, %n, %p). Only the amount of system memory limits the rest (%a, %k, %o). Depending on the system configuration and the available resources, limits may affect what input lex can successfully compile.

If OpenExtensions encounters space limitations, it can use the lex statistics (which show the number of elements used and the maximum size of the table for each of %e, %n and %p) to identify the table that has reached its maximum size (for example, the number of elements equals the maximum size of the table in the statistics). For more details, see the entry for lex in [z/VM: OpenExtensions Commands Reference](#).

A.2.7.4 Regular Expressions

The POSIX.2 standard states that for the escape sequence digits, "if the size of a byte on the system is greater than 9 bits, the valid escape sequence used to represent a byte is implementation-defined."

The lex utility of OpenExtensions uses a byte size of 8 bits, so this is not relevant.

A.3 yacc – Yet Another Compiler Compiler

A.3.6 External Effects

A.3.6.3 Output Files

A.3.6.3.3 Description File

The POSIX.2 standard states that "limits for internal tables (see A.3.7.9) also shall be reported, in an implementation-defined manner."

The only limitation on table sizes in the yacc utility of OpenExtensions is system memory (see section A.3.7.9 in this document). The -v option reports memory usage in the statistics file **y.output**.

A.3.7 Extended Description

A.3.7.9 Limits

The POSIX.2 standard states that (for Table A-3 – yacc Internal Limits) "the exact meaning of these [minimum maximum] values is implementation defined. The implementation shall define the relationship between these values and between them and any error messages that the implementation may generate should it run out of space for any internal structure."

In OpenExtensions, all the internal tables for yacc are dynamically allocated. The only limitation is the system memory; if that is exceeded, the message "Out of memory at <size> bytes" is produced. For more information, see [z/VM: OpenExtensions Advanced Application Programming Tools](#).

Global Issues

The following topics discuss certain global issues, to which entries on specific implementation-defined features can point.

Window Size

To accommodate a wide range of architectures and implementations, OpenExtensions uses an extension to the POSIX.1 interface to get the system's notion of the number of lines and columns in the current "window."

OpenExtensions obtains the number of lines and columns from the environment variables **INES** and **COLUMNS**. If these variables are not set, OpenExtensions uses a "window" size of 24 lines by 80 columns.

Modes and the Sticky Bit

Historically, UNIX® systems have supported a variable S_ISVTX, which designated the presence or absence of the so-called "sticky bit". When the "sticky bit" is turned on, the system keeps a process in swap space. However, OpenExtensions does not support this function of the "sticky bit".

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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 grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan*

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 jurisdictions 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide 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 Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US*

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 document 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.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., in the United States and/or other countries. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on [IBM Copyright and trademark information](https://www.ibm.com/legal/copytrade) (<https://www.ibm.com/legal/copytrade>).

Adobe is either a registered trademark or a trademark of Adobe Systems Incorporated in the United States, and/or other countries.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Terms and Conditions for Product Documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal Use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see:

- The section entitled **IBM Websites** at [IBM Privacy Statement](https://www.ibm.com/privacy) (<https://www.ibm.com/privacy>)
- [Cookies and Similar Technologies](https://www.ibm.com/privacy#Cookies_and_Similar_Technologies) (https://www.ibm.com/privacy#Cookies_and_Similar_Technologies)

Bibliography

This topic lists the publications in the z/VM library. For abstracts of the z/VM publications, see [z/VM: General Information](#).

Where to Get z/VM Information

The current z/VM product documentation is available in [IBM Documentation - z/VM \(https://www.ibm.com/docs/en/zvm\)](#).

z/VM Base Library

Overview

- [z/VM: License Information](#), GI13-4377
- [z/VM: General Information](#), GC24-6286

Installation, Migration, and Service

- [z/VM: Installation Guide](#), GC24-6292
- [z/VM: Migration Guide](#), GC24-6294
- [z/VM: Service Guide](#), GC24-6325
- [z/VM: VMSES/E Introduction and Reference](#), GC24-6336

Planning and Administration

- [z/VM: CMS File Pool Planning, Administration, and Operation](#), SC24-6261
- [z/VM: CMS Planning and Administration](#), SC24-6264
- [z/VM: Connectivity](#), SC24-6267
- [z/VM: CP Planning and Administration](#), SC24-6271
- [z/VM: Getting Started with Linux on IBM Z](#), SC24-6287
- [z/VM: Group Control System](#), SC24-6289
- [z/VM: I/O Configuration](#), SC24-6291
- [z/VM: Running Guest Operating Systems](#), SC24-6321
- [z/VM: Saved Segments Planning and Administration](#), SC24-6322
- [z/VM: Secure Configuration Guide](#), SC24-6323

Customization and Tuning

- [z/VM: CP Exit Customization](#), SC24-6269
- [z/VM: Performance](#), SC24-6301

Operation and Use

- [z/VM: CMS Commands and Utilities Reference](#), SC24-6260
- [z/VM: CMS Primer](#), SC24-6265
- [z/VM: CMS User's Guide](#), SC24-6266
- [z/VM: CP Commands and Utilities Reference](#), SC24-6268

- [z/VM: System Operation](#), SC24-6326
- [z/VM: Virtual Machine Operation](#), SC24-6334
- [z/VM: XEDIT Commands and Macros Reference](#), SC24-6337
- [z/VM: XEDIT User's Guide](#), SC24-6338

Application Programming

- [z/VM: CMS Application Development Guide](#), SC24-6256
- [z/VM: CMS Application Development Guide for Assembler](#), SC24-6257
- [z/VM: CMS Application Multitasking](#), SC24-6258
- [z/VM: CMS Callable Services Reference](#), SC24-6259
- [z/VM: CMS Macros and Functions Reference](#), SC24-6262
- [z/VM: CMS Pipelines User's Guide and Reference](#), SC24-6252
- [z/VM: CP Programming Services](#), SC24-6272
- [z/VM: CPI Communications User's Guide](#), SC24-6273
- [z/VM: ESA/XC Principles of Operation](#), SC24-6285
- [z/VM: Language Environment User's Guide](#), SC24-6293
- [z/VM: OpenExtensions Advanced Application Programming Tools](#), SC24-6295
- [z/VM: OpenExtensions Callable Services Reference](#), SC24-6296
- [z/VM: OpenExtensions Commands Reference](#), SC24-6297
- [z/VM: OpenExtensions POSIX Conformance Document](#), GC24-6298
- [z/VM: OpenExtensions User's Guide](#), SC24-6299
- [z/VM: Program Management Binder for CMS](#), SC24-6304
- [z/VM: Reusable Server Kernel Programmer's Guide and Reference](#), SC24-6313
- [z/VM: REXX/VM Reference](#), SC24-6314
- [z/VM: REXX/VM User's Guide](#), SC24-6315
- [z/VM: Systems Management Application Programming](#), SC24-6327
- [z/VM: z/Architecture Extended Configuration \(z/XC\) Principles of Operation](#), SC27-4940

Diagnosis

- [z/VM: CMS and REXX/VM Messages and Codes](#), GC24-6255
- [z/VM: CP Messages and Codes](#), GC24-6270
- [z/VM: Diagnosis Guide](#), GC24-6280
- [z/VM: Dump Viewing Facility](#), GC24-6284
- [z/VM: Other Components Messages and Codes](#), GC24-6300
- [z/VM: VM Dump Tool](#), GC24-6335

z/VM Facilities and Features

Data Facility Storage Management Subsystem for z/VM

- [z/VM: DFSMS/VM Customization](#), SC24-6274
- [z/VM: DFSMS/VM Diagnosis Guide](#), GC24-6275
- [z/VM: DFSMS/VM Messages and Codes](#), GC24-6276
- [z/VM: DFSMS/VM Planning Guide](#), SC24-6277

- [z/VM: DFSMS/VM Removable Media Services](#), SC24-6278
- [z/VM: DFSMS/VM Storage Administration](#), SC24-6279

Directory Maintenance Facility for z/VM

- [z/VM: Directory Maintenance Facility Commands Reference](#), SC24-6281
- [z/VM: Directory Maintenance Facility Messages](#), GC24-6282
- [z/VM: Directory Maintenance Facility Tailoring and Administration Guide](#), SC24-6283

Open Systems Adapter

- [Open Systems Adapter/Support Facility on the Hardware Management Console](#) (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC14-7580-02.pdf), SC14-7580
- [Open Systems Adapter-Express ICC 3215 Support](#) (<https://www.ibm.com/docs/en/zos/2.3.0?topic=osa-icc-3215-support>), SA23-2247
- [Open Systems Adapter Integrated Console Controller User's Guide](#) (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC27-9003-02.pdf), SC27-9003
- [Open Systems Adapter-Express Customer's Guide and Reference](#) (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/ioa2z1f0.pdf), SA22-7935

Performance Toolkit for z/VM

- [z/VM: Performance Toolkit Guide](#), SC24-6302
- [z/VM: Performance Toolkit Reference](#), SC24-6303

The following publications contain sections that provide information about z/VM Performance Data Pump, which is licensed with Performance Toolkit for z/VM.

- [z/VM: Performance](#), SC24-6301. See [z/VM Performance Data Pump](#).
- [z/VM: Other Components Messages and Codes](#), GC24-6300. See [Data Pump Messages](#).

RACF® Security Server for z/VM

- [z/VM: RACF Security Server Auditor's Guide](#), SC24-6305
- [z/VM: RACF Security Server Command Language Reference](#), SC24-6306
- [z/VM: RACF Security Server Diagnosis Guide](#), GC24-6307
- [z/VM: RACF Security Server General User's Guide](#), SC24-6308
- [z/VM: RACF Security Server Macros and Interfaces](#), SC24-6309
- [z/VM: RACF Security Server Messages and Codes](#), GC24-6310
- [z/VM: RACF Security Server Security Administrator's Guide](#), SC24-6311
- [z/VM: RACF Security Server System Programmer's Guide](#), SC24-6312
- [z/VM: Security Server RACROUTE Macro Reference](#), SC24-6324

Remote Spooling Communications Subsystem Networking for z/VM

- [z/VM: RSCS Networking Diagnosis](#), GC24-6316
- [z/VM: RSCS Networking Exit Customization](#), SC24-6317
- [z/VM: RSCS Networking Messages and Codes](#), GC24-6318
- [z/VM: RSCS Networking Operation and Use](#), SC24-6319
- [z/VM: RSCS Networking Planning and Configuration](#), SC24-6320

TCP/IP for z/VM

- [z/VM: TCP/IP Diagnosis Guide](#), GC24-6328
- [z/VM: TCP/IP LDAP Administration Guide](#), SC24-6329
- [z/VM: TCP/IP Messages and Codes](#), GC24-6330
- [z/VM: TCP/IP Planning and Customization](#), SC24-6331
- [z/VM: TCP/IP Programmer's Reference](#), SC24-6332
- [z/VM: TCP/IP User's Guide](#), SC24-6333

Prerequisite Products

Device Support Facilities

- Device Support Facilities (ICKDSF): User's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/ickug00_v2r5.pdf), GC35-0033

Related Products

XL C++ for z/VM

- [XL C/C++ for z/VM: Runtime Library Reference](#), SC09-7624
- [XL C/C++ for z/VM: User's Guide](#), SC09-7625

z/OS

IBM Documentation - z/OS (<https://www.ibm.com/docs/en/zos>)

Additional Publications

[XL C/C++ for z/VM: User's Guide](#), SC09-7625

[z/OS: XL C/C++ Programming Guide](https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/cbcpx01_v2r5.pdf) (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/cbcpx01_v2r5.pdf), SC09-4765



Product Number: 5741-A09

Printed in USA

GC24-6298-74

