OS/390

GC28-1895-00

OpenEdition MVS POSIX.1 Conformance Document

OS/390

GC28-1895-00

OpenEdition MVS POSIX.1 Conformance Document

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page xi.

First Edition (March 1996)

This edition applies to Release 1 of OS/390 (5645-001) and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

IBM welcomes your comments. A form for your comments appears at the back of this publication. If the form has been removed, address your comments to:

International Business Machines Corporation
Department 55JA, Mail Station P384
522 South Road
Poughkeepsie, N. Y. 12601-5400
United States of America

FAX: (International Access Code)+1+914+432-9405

IBMLink (United States customers only): KGNVMC(MHVRCFS)
IBM Mail Exchange: USIB6TC9 at IBMMAIL
Internet: mhvrcfs@vnet.ibm.com

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Tables

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user′s responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, New York 10594
USA

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
522 South Road
Poughkeepsie, NY 12601-5400
USA
Attention: Information Requests

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks

The following terms, **denoted by an asterisk (*),** used in this publication, are trademarks of the IBM Corporation in the United States or other countries:

IBM
MVS/DFP
MVS/ESA
OpenEdition
RACF

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company, Limited.

Other company, product, or service names, **which may be denoted by a double asterisk (**),** used in this book, may be trademarks or service marks of others.

ANSI	American National Standards Institute
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
POSIX	Institute of Electrical and Electronics Engineers
UNIX	UNIX System Laboratories, Inc.

About This Book

This book describes how the OpenEdition* implementation meets the criteria for a *conforming implementation* as defined in the International Organization for Standardization and International Electrotechnical Commission (ISO/IEC) 9945-1: 1990 (Institute of Electrical and Electronics Engineers [IEEE**] Std 1003.1-1990) POSIX.1 standard. POSIX** stands for Information Technology—Portable Operating System Interface.

This book is intended to help technical personnel evaluate how IBM is implementing the IEEE Std 1003.1-1990 (POSIX.1) standard with its MVS/ESA* system. There are two OpenEdition features:

- OpenEdition Shell and Utilities
- OpenEdition Debugger

This book describes the OpenEdition implementation of those areas of the standard that were declared to be *optional*, or *implementation-defined*. The topics included correspond to topics in the POSIX.1 standard.

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

Note: The chapter numbers employed in this document refer to the actual chapter numbers in the IEEE Std 1003.1-1990 standard and are a requirement of POSIX.1 section 1.3.1.2. They also make the task of cross-referencing that document easier.

Related Books

Where necessary, this book references information in other books, using shortened versions of the book title. For complete titles and order numbers of the books for all products that are part of OS/390, see *OS/390 Information Roadmap*.

* OpenEdition is a trademark of IBM Corporation.

** IEEE and POSIX are trademarks of Institute of Electrical and Electronics Engineers.

* MVS/ESA is a trademark of IBM Corporation.

A Task-Oriented Guide to OS/390 OpenEdition MVS Information

Books that apply to more than one task occur more than once in this guide. Most of the book titles listed in this guide are included in the CD-ROM collection kit *OS/390 Collection*. You can order this using SK2T-6700.

BookManager READ/MVS provides access to online information on a CD-ROM. Also available are BookManager READ/DOS and BookManager READ/2, which allow you to download online books to your workstation and read these books on DOS or OS/2, respectively.

With BookManager READ installed on your system, you can enter the command BOOKMGR to start BookManager and display a list of books available to you. If you know the name of the book that you want to view, you can use the OPEN command to open the book directly. You can read, search, make notes, and select sections of text to print.

The OS/390 OpenEdition MVS OHELP TSO/E command lets you go directly to specific pages of information from your TSO/E session. OHELP requires the BookManager READ product.

And See Our Home Page on the Web!

To keep current with new OS/390 OpenEdition MVS announcements, events, and other information, see the OS/390 OpenEdition MVS home page on the World Wide Web at:

`http://www.s390.ibm.com/products/oe/index.html`

Getting a Basic Understanding

Book Title	Order Number
OS/390 OpenEdition MVS Introduction	GC28-1889
Introducing OS/390	GC28-1725
OS/390 Documentation Roadmap	GC28-1727
OS/390 OpenEdition DCE Introduction	GC28-1581
OS/390 Language Environment Concepts Guide	GC28-1945
C/C++ General Information	GC09-2060
DFSMS/MVS General Information	GC26-4900

Planning, Installing, and Customizing

Book Title	Order Number
OS/390 OpenEdition MVS Planning	SC28-1890
OS/390 Language Environment Installation and Customization on MVS	SC28-1941
OS/390 OpenEdition DCE Planning	SC28-1582
OS/390 OpenEdition DCE Configuring and Getting Started	SC28-1583

Administration

Book Title	Order Number
OS/390 OpenEdition MVS Planning	SC28-1890
OS/390 OpenEdition DCE Administration Guide	SC28-1584

Using the Shell and Utilities or the Hierarchical File System

Book Title	Order Number
OS/390 OpenEdition MVS User's Guide	SC28-1891
OS/390 OpenEdition MVS Command Reference	SC28-1892
OS/390 OpenEdition MVS Messages and Codes	SC28-1908
OS/390 OpenEdition MVS Programming Tools	SC28-1904
DFSMS/MVS: Network File System User's Guide	SC26-7028

Application Programming: Standards

Book Title	Order Number
POSIX.1 standard: *Information technology—Portable Operating System Interface (POSIX)—Part 1: System Application Program Interface (API)*[*C Language*], International Standard ISO/IEC 9945-1: 1990 (IEEE Std 1003.1-1990).	N/A
POSIX.2 standard: *Information technology—Portable Operating System Interface (POSIX)—Part 2: Shell and Utilities,* International Standard ISO/IEC 9945-2: 1992 (IEEE Std 1003.2-1992).	N/A
FIPS-2 standard: Federal Information Processing Standards Publication (FIPS PUB) 151-2.	N/A
OS/390 OpenEdition MVS POSIX.1 Conformance Document	GC28-1895
OS/390 OpenEdition MVS POSIX.2 Conformance Document	GC28-1896
OS/390 OpenEdition MVS XPG4 Conformance Document	GC28-1897

Designing and Coding Programs

Book Title	Order Number
OS/390 Language Environment Programming Guide	SC28-1939
OS/390 Language Environment Programming Reference	SC28-1940
C/C++ for MVS/ESA Library Reference	SC23-3881
C/C++ for MVS/ESA Language Reference	SC09-2150
C/C++ for MVS/ESA User's Guide	SC09-2205
C/C++ for MVS/ESA Programming Guide	SC09-2164
C/MVS Library Reference: OpenEdition MVS Curses	SC23-3876
OS/390 OpenEdition MVS Programming: Assembler Callable Services Reference	SC28-1899
OS/390 OpenEdition MVS Programming Tools	SC28-1904
OS/390 OpenEdition MVS Communications Server Guide	SC28-1906
OS/390 OpenEdition MVS Using REXX and OpenEdition MVS	SC28-1905
OS/390 OpenEdition MVS File System Interface Reference	SC28-1909
OS/390 OpenEdition DCE Application Development Reference	SC28-1590

Compiling and Running Programs

Book Title	Order Number
C/C++ for MVS/ESA User's Guide	SC09-2205
OS/390 OpenEdition MVS Programming Tools	SC28-1904
OS/390 OpenEdition DCE User's Guide	SC28-1586

Debugging Programs

Book Title	Order Number
OS/390 OpenEdition MVS Programming Tools	SC28-1904
OS/390 OpenEdition MVS Command Reference	SC28-1892
OS/390 OpenEdition MVS Messages and Codes	SC28-1908
OS/390 Language Environment Debugging Guide and Run-Time Messages	SC28-1942
OS/390 OpenEdition DCE Messages and Codes	SC28-1591

Diagnosing Problems

Book Title	Order Number
OS/390 OpenEdition MVS Messages and Codes	SC28-1908
OS/390 OpenEdition MVS Programming: Assembler Callable Services Reference	SC28-1899
OS/390 OpenEdition DCE Messages and Codes	SC28-1591

Elements in OS/390

You can use the following table to see the relationship of a product you are familiar with and how it is referred to in OS/390. OS/390 is made up of elements that contain function at or beyond the release level of the products listed in the following table. The table gives the name and level of each product on which an OS/390 element is based, identifies the OS/390 name of the element, and indicates whether the OS/390 element is part of the base or optional. For more compatibility information about OS/390 elements see *OS/390 Up and Running!*, GC28-1726.

Product Name and Level	**Name in OS/390**	**Base or Optional**
BookManager BUILD/MVS V1R3	BookManager BUILD	optional
BookManager READ/MVS V1R3	BookManager READ	base
MVS/Bulk Data Transfer V2	Bulk Data Transfer (BDT)	base
MVS/Bulk Data Transfer File-to-File V2	Bulk Data Transfer (BDT) File-to-File	optional
MVS/Bulk Data Transfer SNA NJE V2	Bulk Data Transfer (BDT) SNA NJE	optional
IBM C/C++ for MVS/ESA V3R2	C/C++	optional
DFSMSdfp V1R3	DFSMSdfp	base
DFSMSdss	DFSMSdss	optional
DFSMShsm	DFSMShsm	optional
DFSMSrmm	DFSMSrmm	optional
DFSMS/MVS Network File System V1R2	DFSMS/MVS Network File System	base
EREP MVS V3R5	EREP	base
GDDM/MVS V3R1.1 • GDDM-OS/2 LINK • GDDM-PCLK	GDDM	base
GDDM-REXX/MVS V3R1.1	GDDM-REXX	optional
IBM High Level Assembler for MVS V1R2	High Level Assembler	base
IBM High Level Assembler Toolkit	High Level Assembler Toolkit	optional
ICKDSF R16	ICKDSF	base
ISPF V4R2	ISPF	base
Language Environment for MVS V1R5	Language Environment	base
Language Environment V1R5 Data Decryption	Language Environment Data Decryption	optional
MVS/ESA SP V5R2.2		
• BCP	• BCP or MVS	base
• ESCON Director Support	• ESCON Director Support	base
• Hardware Configuration Definition (HCD)	• Hardware Configuration Definition (HCD)	base
• JES2 V5R2.0	• JES2	base
• JES3 V5R2.1	• JES3	optional
• LANRES/MVS V1R3.1	• LANRES	base
• IBM LAN Server for MVS V1R1	• LAN Server	base
• MICR/OCR Support	• MICR/OCR Support	base
• OpenEdition MVS Services	• OpenEdition MVS Services	base
• OpenEdition MVS Debugger	• OpenEdition MVS Debugger	base
• OpenEdition MVS Shell & Utilities	• OpenEdition MVS Shell & Utilities	base
• OpenEdition DCE Base Services (OSF DCE level 1.1)	• OpenEdition DCE Base Services	base
• OpenEdition DCE Distributed File Service (DFS) (OSF DCE level 1.0.3a)	• OpenEdition DCE Distributed File Service (DFS)	base
• OpenEdition DCE User Data Privacy	• OpenEdition DCE User Data Privacy	optional

Product Name and Level	Name in OS/390	Base or Optional
• SOMobjects for MVS Application Development Environment (ADE) V1R1 • SOMobjects Runtime Library (RTL) • SOMobjects service classes	• SOMobjects Application Development Environment (ADE) • SOMobjects for MVS Runtime Library (RTL) • SOMobjects service classes	optional base base
Open Systems Adapter Support Facility (OSA/SF) R1	Open Systems Adapter Support Facility (OSA/SF)	base
MVS/ESA RMF V5R2	RMF	optional
RACF V2R2	Security Server • RACF • OpenEdition DCE Security Server	optional
SMP/E V1R8.1	SMP/E	base
SystemView for MVS base	SystemView for MVS base	base
IBM TCP/IP V3R1 • TCP/IP CICS Sockets • TCP/IP IMS Sockets • TCP/IP Kerberos • TCP/IP Network Print Facility (NPF) • TCP/IP OpenEdition Applications • TCP/IP OS/2 Offload	TCP/IP • TCP/IP CICS Sockets • TCP/IP IMS Sockets • TCP/IP Kerberos • TCP/IP Network Print Facility (NPF) • TCP/IP OpenEdition Applications • TCP/IP OS/2 Offload	base optional optional optional optional optional optional
TIOC R1	TIOC	base
Time Sharing Option Extensions (TSO/E) V2R5	TSO/E	base
VisualLift for MVS V1R1.1	• VisualLift Run-Time Environment (RTE) • VisualLift Application Development Environment (ADE)	base optional
VTAM V4R3 with the AnyNet feature	VTAM	base

Summary of Changes

Summary of Changes
for GC28-1895-00
OS/390 Release 1

This book contains information previously presented in *OpenEdition MVS POSIX.1 Conformance Document*, GC23-3011, which supports OpenEdition MVS.

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 4. 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 MVS services that are not part of a POSIX.1 or ANSI** C library call. In other words, the program uses only POSIX.1 functions.
- The program receives control through one of the *exec* family of C function calls.
- Prior to the *exec* call to the program, STDIN, STDOUT, and STDERR are opened for either file **/dev/null** or some other file system file.

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

- TSO/E OMVS command and the shell
- BPXBATCH
- rlogin and the shell

1.3.1.2 Documentation

The OpenEdition implementation conforms to the International Organization for Standardization and International Electrotechnical Commission (ISO/IEC) 9945-1: 1990 (Institute of Electrical and Electronics Engineers [IEEE] Std. 1003.1-1990) POSIX.1 standard, approved on September 28, 1990. Any extensions beyond the IEEE Std 1003.1-1990 are described in other OpenEdition books.

Other de facto, national, and international software standards that are available with the OpenEdition implementation are described in related system documentation.

** ANSI is a trademark of American National Standards Institute.

1.3.3 Language-Dependent Services for the C Programming Language

1.3.3.2 C Standard Language-Dependent System Support

The OpenEdition 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).

** ISO/IEC 9899:1990(E) C is technically identical to ANS X3.159-1989, Programming Language C (ANSI C).

Section 2. Terminology and General Requirements

2.2 Definitions

2.2.2 General Terms

2.2.2.4 appropriate privileges: The following terms are sometimes used in this document to refer to the creation and deletion of an OpenEdition process:

- **Dub.** To make an MVS address space known to OpenEdition MVS. Once dubbed, an address space is considered to be an OpenEdition "process." Address spaces created by fork() are automatically dubbed when they are created; other address spaces become dubbed if they invoke an OpenEdition service. Dubbing also applies to MVS tasks. A dubbed task is considered an OpenEdition "thread." Tasks are dubbed if they invoke an OpenEdition service.
- **Undub.** The inverse of *dub*. Normally, a task (dubbed a thread) is undubbed when it ends. An address space (dubbed a process) is undubbed when the last thread ends.

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 or if the user has Resource Access Control Facility (RACF*) trusted or privileged attributes.

For the setuid and seteuid functions, the MVS identity of the address space is changed in addition to the POSIX identity. This capability can be further restricted by requiring the caller of these functions to be defined to a special list called the BPX.DAEMON FACILITY class profile within RACF. If the BPX.DAEMON profile is defined, all modules that are present in the process must be controlled. "Controlled" refers to the module being defined to RACF with an RDEFINE command, with the WHEN(PROGRAM) operand specified.

A user not currently running as a superuser can switch to superuser (effective UID=0) if the user is permitted to the BPX.SUPERUSER FACILITY class profile within RACF. The OpenEdition MVS ISPF Shell and the su shell command provide the external interface for a user to switch to superuser.

You can assign a UID of 0 to a user in the RACF user profile using the RACF ADDUSER or ALTUSER commands. When an address space (that is, TSO/E session, started task, or initiated batch job) started with the user ID for that profile is dubbed an OpenEdition process, the UID from the profile becomes the effective UID for the process. 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.

The trusted or privileged attribute is an attribute associated with a started procedure address space. If the address space is dubbed a process, the attribute also

* RACF is a registered trademark of IBM Corporation.

applies to the process. An installation defines which started procedures have the trusted or privileged attributes by coding a started procedure table module and replacing the module (which is empty) shipped by RACF.

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

2.2.2.27 file: The file system also supports symbolic link files and external links, and does not support block special files.

2.2.2.46 login: A user gains interactive access to the shell by first logging on to a TSO/E user ID through existing documented MVS externals. Once logged on to TSO/E, a user can run the shell by entering the OMVS command from the TSO/E READY prompt.

Users can also login from remote terminals with the **rlogin** command.

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 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 MVS data set:

- 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 on the MOUNT command or parmlib statement that mounted the file system. No updates are made or allowed to a read-only file system.

2.2.2.83 supplementary group ID: There is no support to ensure that the effective GID is always included in or always excluded from the list of supplemental GIDs. The effective GID is in the list if the user is a member of the effective GID of the process. This is true when a process is first dubbed. The effective GID of a process is not in the list if, for example, the process runs a setgid file that sets the effective GID to a group of which the user is not a member.

2.3 General Concepts

2.3.1 extended security controls: The inclusion of the RACF trusted and privileged attributes as part of the definition of superuser is the only extension to the access security mechanisms (see “2.2.2.4 appropriate privileges” on page 3).

2.3.2 file access permissions: *Appropriate privilege* is defined as having superuser authority (see “2.2.2.4 appropriate privileges” on page 3). No alternate access control mechanisms are provided.

2.4 Error Numbers

The OpenEdition-defined values for the POSIX-defined error numbers are defined in <**errno.h**>. Table 1 shows the codes that are provided in addition to those defined by POSIX:

Table 1. OpenEdition Non-POSIX Error Codes

Error Code	Description
EILSEQ	Incorrect byte sequence.
ELOOP	A loop exists in symbolic links encountered during pathname resolution.
EMVSBADCHAR	There is an incorrect character in the environment variable name.
EMVSCATLG	Catalog obtain error.
EMVSCVAF	Catalog Volume Access Facility error.
EMVSDYNALC	Dynamic allocation error.
EMVSERR	An MVS environment or internal error occurred.
EMVSINITIAL	A process initialization error occurred.
EMVSNORTL	Access to the OpenEdition version of the C runtime library is denied.
EMVSNOTUP	OpenEdition services are not active.
EMVSPARM	Incorrect parameters were passed to the service.
EMVSPATHOPTS	An access mode argument conflicts with the PATHOPTS parameter.
EMVSPFSFILE	The HFS data set encountered a permanent file error.
EMVSPFSPERM	The HFS data set encountered a system error.
EMVSSAF2ERR	A Security Authorization Facility/Resource Access Control Facility (SAF/RACF) error occurred.
EMVSSAFEXTRERR	A Security Authorization Facility/Resource Access Control Facility (SAF/RACF) extract error occurred.
EMVSTODNOTSET	The system time-of-day (TOD) clock is not set.

2.5 Primitive System Data Types

In addition to the primitive system data types specified by POSIX.1, the OpenEdition implementation supports the non-POSIX data type whose name ends with _***t***, shown in Table 2.

Table 2. OpenEdition Non-POSIX Primitive System Data Type

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

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

Table 3. OpenEdition POSIX Primitive System Data Types

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

2.6 Environment Description

The OpenEdition implementation permits all strings composed of 8-bit characters (except for the “=” character) to be used in environment variable names.

2.7 C Language Definitions

2.7.2 POSIX.1 Symbols

In addtion to _POSIX_SOURCE, OpenEdition MVS supports the following feature test macros:

_POSIX1_SOURCE
_POSIX_C_SOURCE
_OPEN_SYS
_OPEN_THREADS
_BSD

2.8 Numerical Limits

The following subsections list magnitude limitations imposed by the OpenEdition implementation:

2.8.3 Run-Time Increasable Values

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

NGROUPS_MAX 300

2.8.4 Run-Time Invariant Values

Table 4 shows extended values that are available in the OpenEdition implementation with the sysconf() function. They are not defined in <**limits.h**>.

Table 4. OpenEdition Run-Time Invariant Values

Name	Description
ARG_MAX	Maximum argument and environment length
CHILD_MAX	Maximum number of simultaneous processes per real user ID
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 shows extended values that are available in the OpenEdition implementation with the pathconf() function. They are not defined in <**limits.h**>.

Table 5. OpenEdition Pathname 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 filename
PATH_MAX	Maximum length of a pathname
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 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

2.9.4 Execution-Time Constants for Portability Specifications

The constants for portability specifications shown in Table 7 are available in the OpenEdition 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 pathname length is greater than {NAME_MAX}
_POSIX_VDISABLE	The value used to disable terminal special characters

The values of _POSIX_CHOWN_RESTRICTED and _POSIX_NO_TRUNC apply to all files. The value of _POSIX_VDISABLE applies to all terminal files.

Section 3. Process Primitives

3.1 Process Creation and Execution

3.1.1 Process Creation

Function: fork()

3.1.1.2 Description

Each open directory stream in the child can share directory stream positioning with the corresponding directory stream of the parent.

3.1.1.4: Errors

fork() returns an **errno** of [ENOMEM] if the process requires more space than is available.

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 filename 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.

A process keeps the MVS JES address space characteristics after an exec.

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

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 [EFAULT] if a bad address was received as an argument on the call or the user exit program checked.

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 with ptrace().

3.2.1.4 Errors

If one of the parameters specified contained an address of storage that is not accessible to the caller, wait() or waitpid() sets **errno** to [EFAULT].

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 shows signals that are supported in addition to those specified by POSIX.1.

Table 8. OpenEdition Non-POSIX Signals

Symbolic Constant	Description
SIGABND	Abend signal
SIGDCE	Exclusive use by Distributed Computing Environment (DCE)
SIGIO	Completion of input or output
SIGIOER	Error on input/output
SIGTRAP	Trace trap

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

For the list of default actions associated with these signals, see *AD/Cycle C/370 Library Reference, SC09-1761*.

For the list of values associated with these signals, see *OS/390 OpenEdition MVS Programming: Assembler 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 OpenEdition 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.

Unexpected MVS abends result in either a **SIGKILL** or **SIGABND** signal being generated.

A **SIGIO** signal is generated for non-file-system I/O errors when a catcher has been defined for **SIGIO**.

A **SIGTRAP** signal is generated to process ptrace callable services.

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.

3.3.3 Manipulate Signal Sets

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

3.3.3.4 Errors

sigaddset(), sigdelset(), and sigismember() generate [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].

3.3.4.4 Errors

When the specified address for New_sa_handler_address or Old_sa_handler_address is incorrect, sigaction() sets **errno** to [EFAULT].

When an MVS environmental or internal error has occurred, sigaction() sets **errno** to [EMVSERR].

3.3.5 Examine and Change Blocked Signals

Function: sigprocmask()

3.3.5.4 Errors

Access violations when the signal mask is filled in for sigprocmask() cause an error return with **errno** set to [EFAULT].

When an MVS environmental or internal error has occurred, sigprocmask() sets **errno** to [EMVSERR].

3.3.6 Examine Pending Signals

Function: sigpending()

3.3.6.4 Errors

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

When an MVS environmental or internal error has occurred, sigpending() sets **errno** to [EMVSERR].

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()

4.2.3.2 Description

If the return value from getgroups() is less than the array size argument gidsetsize, the array elements greater than the return value remain unchanged.

4.2.4 Get User Name

Function: getlogin()

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 OpenEdition implementation, the structure **utsname** contains the members and formats shown in Table 9. Each member is padded with blanks to fill out the structure.

Table 9. Formats for OpenEdition utsname Members

Member Name	Description	Format
sysname	Name of implementation	char [16]
nodename	Network node name	char [32]
release	Release level	char [8]
version	Version of release	char [8]
machine	Machine hardware name	char [16]

4.4.1.4 Errors

Access violations when the **utsname** structure is filled in cause an error return with **errno** set to [EFAULT].

4.5 Time

4.5.1 Get System Time

Function: time()

4.5.1.4 Errors

If the system time-of-day (TOD) clock is not set, time() returns ((**time_t**)−1) after setting **errno** [EMVSTODNOTSET].

4.5.2 Get Process Times

Function: times()

4.5.2.4 Errors

An overflow when a time value is calculated causes an error return with **errno** set to [ERANGE].

Access violations when the **struct_tms** buffer is filled in causes an error return with **errno** set to [EFAULT].

4.6 Environment Variables

4.6.1 Environment Access

Function: getenv()

4.6.1.4 Errors

The following error 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].

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. For more information, see *C/C++ for MVS Library Reference,* SC23-3881.

Section 5. Files and Directories

5.1 Directories

5.1.1 Format of Directory Entries

The link count of a directory is incremented when a subdirectory is created in that directory.

5.1.2 Directory Operations

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

5.1.2.2 Description

The OpenEdition implementation returns 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.

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

5.1.2.4 Errors

For the opendir() function, the OpenEdition 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].

5.2.2.4 Errors

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

When search permission is denied for a component of the pathname, 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 OpenEdition 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. If O_CREAT is used and the file exists, the group is not changed.

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 OpenEdition 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 OpenEdition 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 OpenEdition implementation.

The group ownership of a newly created directory is the same as that of the parent directory.

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 OpenEdition implementation.

The group ownership of a newly created FIFO is the same as that of the parent directory.

5.5 File Removal

5.5.1 Remove Directory Entries

Function: unlink()

5.5.1.2 Description

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

5.5.1.4 Errors

[EBUSY] The file cannot be unlinked, because it is being used by the system.

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, rmdir() fails and sets **errno** to [EBUSY].

If the named directory is the working directory of a process, rmdir() succeeds.

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 pathname 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.

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 OpenEdition 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 fstat()—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 OpenEdition implementation does not support any configurable filename variables that do not appear in Table 5-2 of IEEE Std 1003.1-1990.

If name is _PC_MAX_CANON, _PC_MAX_INPUT, or _PC_VDISABLE, the following applies:

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

If name is _PC_NAME_MAX, _PC_PATH_MAX, or _PC_NO_TRUNC, the following applies:

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

If name is _PC_PIPE_BUF the following applies:

- If path or fildes 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 filename variable is not supported for the specified file, the function returns an **errno** of [EINVAL].

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

If the named file does not exist, or if the pathname 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 MVS.

6.4 Input and Output

6.4.1 Read from a File

Function: read()

6.4.1.2 Description

When read() is interrupted after some of the bytes have been transferred, the number of bytes transferred is returned.

All reads occurring at the end-of-file for a master pseudoterminal shall return a value of 0. (The end-of-file condition for the master pseudoterminal exists when there is no data on the output queue and the last file descriptor for the associated slave pseudoterminal has been closed while the **termios** control mode HUPCL flag was set. See “7.1.2.4 Control Modes” on page 29.)

A read that occurs at the end-of-file for a slave pseudoterminal shall return a value of 0. (The end-of-file condition exists for the slave pseudoterminal when either of the following is true:

- The associated master pseudoterminal has closed and there is no more data in the input buffer

or:

- An EOF character was written, by itself, to the master pseudoterminal and there was no data in the input data buffer.)

In the first case, the end-of-file condition continues to exist; in the second case, the slave read clears the end-of-file condition.

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 MVS.

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 does not happen with a write to a pipe file (or FIFO special file) or to a regular 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 MVS.

6.5 Control Operations on Files

6.5.2 File Control

Function: fcntl()

6.5.2.2 Description

F_SETFD The OpenEdition 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_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. Advisory record locking is supported only for regular files.

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_CLOSFD, 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_CLOSFD, 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 OpenEdition implementation does not provide any devices that support asynchronous serial communication. It provides support for some features of the general terminal interface by way of pseudoterminals, which act like and appear as asynchronous terminals to the OpenEdition shell and application programs. The MVS TSO/E command OMVS is provided, which uses a pseudoterminal 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 OpenEdition 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 OpenEdition implementation imposes the limit {MAX_INPUT} on the number of bytes that may be stored in the input queue.

If {MAX_INPUT} is exceeded when incoming data is being processed, either the process writing to the master terminal is blocked, or the number of characters accepted is returned, depending on whether the master terminal was opened in blocking or nonblocking mode. If it was opened in nonblocking mode, and no characters can be accepted, the return value is set to −1 and the **errno** to [EAGAIN].

There are several exceptions to this rule:

- If the character that would cause {MAX_INPUT} to be exceeded is a SUSP, START, or STOP character, the requested function is performed.
- If, while in canonical mode, the character that would cause {MAX_INPUT} to be exceeded is the ERASE or KILL character, then normal ERASE or KILL processing is performed.
- In addition, if the character is EOF, the canonical line is completed.

  Note: It is not possible to exceed {MAX_INPUT} while in canonical mode without at least one canonical line, because that would also mean exceeding {MAX_CANON}. Therefore, it is not necessary to discard characters when {MAX_INPUT} is exceeded in canonical mode.

7.1.1.6 Canonical Mode Input Processing

{MAX_CANON} is defined for pseudoterminals.

If {MAX_CANON} is exceeded while incoming data is being processed, all characters except for line-editing characters are ignored.

7.1.1.7 Noncanonical Mode Input Processing

The OpenEdition 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 slave terminal is buffered, waiting for a subsequent read() by the master.

7.1.1.9 Special Characters

The START and STOP characters can be changed.

No multibyte special sequences are supported.

7.1.1.10 Modem Disconnect

A close() of the master pseudoterminal is treated as a modem disconnect situation.

When a modem is disconnected, the EOF condition is returned to subsequent reads by a background process.

7.1.2 Parameters That Can Be Set

7.1.2.2 Input Modes

For pseudoterminals, 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.

Metering of input data is done by blocking or returning an **errno** of [EAGAIN] to master writes, as appropriate. Therefore, 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.

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

BRKINT Signal interrupt on break
ICRNL Map carriage return to newline on input
IGNBRK Ignore break condition
IXON Enable start/stop output control

7.1.2.3 Output Modes

No flags are set in the initial c_oflag setting after open(). When OPOST is set in c_oflag, tab expansion is performed; enough blank characters are inserted to reach the next multiple of 8 bytes on a line. The OMVS TSO/E command also performs tab expansion, so the OPOST flag is off by default.

7.1.2.4 Control Modes

A program can request the changing of any flag, but all except for HUPCL and CLOCAL are ignored. Attempts to change them from their default values are ignored. HUPCL indicates whether to indicate the end of the file to a read of the master terminal after the last close of the slave. If set, the end of the file is indicated; if clear, the master read is given a return value of −1 and errno of [EAGAIN].

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

CREAD Enable receiver
CSIZE Set to CS8 for 8 bits per byte
HUPCL Hang up on last close

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.

If the KILL character is written to a master pseudoterminal, and ECHOK and ICANON are set, all characters waiting to be echoed, up to any preceding "\n," are deleted. If there are no characters to erase, the KILL character is ignored.

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

ECHO Enable echo
ECHOE Echo ERASE as an error-correcting backspace
ECHOK Echo kill
ECHONL Echo \n
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 30 shows the initial values of these characters.

Table 10. Initial Values for Special Control Characters

Control Character	Hexadecimal Value	EBCDIC character
VEOF	37	EOT
VEOL	15	NL
VERASE	16	BS
VINTR	03	ETX
VKILL	3D	NAK
VMIN	00	None
VQUIT	32	SYN
VSTART	11	DC1
VSTOP	13	DC3
VSUSP	3F	SUB
VTIME	00	None

7.1.3 Baud Rate Functions

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

7.1.3.2 Description

See “7.1.3.4 Errors” for a description of processing when an unsupported baud rate is specified.

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

Different input and output baud rates are supported.

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 against pseudoterminals. Unless issued under circumstances requiring a **SIGTTOU** signal, the function is successful without taking any action.

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 OpenEdition implementation, **TZ** environment variables of the form:

:characters

are *not* supported.

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 OpenEdition implementation are defined in Table 11.

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

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

See *C/MVS Programming Guide*, SC09-2062, for:

- A description of the LC_SYNTAX and LC_TOD categories
- 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
- A list of supported locales and where to find the online information about the locales.

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, 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 an MVS data set, 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.

8.3 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/370 proprietary locale category, LC_TOD, are used to establish default values.

Section 9. System Databases

9.1 System Databases

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.

9.2 Database Access

9.2.1 Group Database Access

Functions: getgrgid() and getgrnam()

9.2.1.3 Returns

In the OpenEdition implementation, the return values for the getgrgid() and getgrnam() functions *do not point* to static data that is 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

[**EMVSSAF2ERR**] Unforeseen Security Authorization Facility/Resource Access Control Facility (SAF/RACF) error

9.2.2 User Database Access

Functions: getpwuid() and getpwnam()

9.2.2.3 Returns

For the getpwuid() and getpwnam() functions, the return values *do not point* to static data that is overwritten on each call.

9.2.2.4 Errors

In the OpenEdition 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

[**EMVSSAFEXTRERR**]
The group ID was set up incorrectly in Security Authorization Facility/Resource Access Control Facility (SAF/RACF)

[**EMVSSAF2ERR**]
Unforeseen Security Authorization Facility/Resource Access Control Facility (SAF/RACF) error

Section 10. Data Interchange Format

10.1 Archive/Interchange File Format

An archive being introduced into an OpenEdition implementation from an external medium is first copied intact into a file in the OpenEdition file system using a standard MVS utility named OCOPY. It is then read using the standard OpenEdition format-reading utility named pax.

An archive being exported from an OpenEdition implementation to an external medium is first created in the hierarchical file system (HFS) with the standard format-creating utility named pax. It is then copied to the external medium with the standard MVS utility named OCOPY.

See *OS/390 OpenEdition MVS Command Reference* for a description of the pax and the OCOPY utilities and the interfaces to them.

10.1.1 Extended tar Format

The OpenEdition implementation supports the use of characters outside the portable filename 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 filename 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 major device number (in the high-order 9 bits of c_rdev) and the low-order 9 bits of the minor device number (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 OpenEdition implementation supports 16-bit values in these fields.

c_dev and c_ino are taken from the stat structure, st_dev and st_ino. They are truncated to fit within the allotted space.

The OpenEdition implementation does not support block special files.

10.1.2.2 cpio Filename

In the OpenEdition implementations, if a filename 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.

10.1.2.5 cpio Values

In the OpenEdition implementation, other than those file types defined in Table 10-3 of the IEEE Std 1003.1-1990 standard, the following file types are 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 OpenEdition implementation, the `pax` utility 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**.

Communicating Your Comments to IBM

OS/390
OpenEdition MVS POSIX.1 Conformance Document

Publication No. GC28-1895-00

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a readers′ comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:

  FAX: (International Access Code)+1+914+432-9405
- If you prefer to send comments electronically, use this network ID:
  - IBMLink (United States customers only): KGNVMC(MHVRCFS)
  - IBM Mail Exchange: USIB6TC9 at IBMMAIL
  - Internet e-mail: mhvrcfs@vnet.ibm.com
  - World Wide Web: http://www.s390.ibm.com/os390

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies.

Readers′ Comments — We′d Like to Hear from You

OS/390
OpenEdition MVS POSIX.1 Conformance Document
Publication No. GC28-1895-00

You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author′s department for whatever review and action, if any, are deemed appropriate.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Today′s date: _____

What is your occupation?

Newsletter number of latest Technical Newsletter (if any) concerning this publication:

How did you use this publication?

[ ]	As an introduction	[ ]	As a text (student)
[ ]	As a reference manual	[ ]	As a text (instructor)
[ ]	For another purpose (explain)		

Is there anything you especially like or dislike about the organization, presentation, or writing in this manual? Helpful comments include general usefulness of the book; possible additions, deletions, and clarifications; specific errors and omissions.

Page Number: Comment:

Name	Address
Company or Organization	
Phone No.	

Readers′ Comments — We′d Like to Hear from You
GC28-1895-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
522 South Road
Poughkeepsie NY 12601-5400

Fold and Tape **Please do not staple** Fold and Tape

GC28-1895-00

Cut or Fold
Along Line

Program Number: 5645-001

Printed in U.S.A.

GC28-1895-00

DSMPSP544E PAGE SEGMENT FRONT811 PSEG3820 NOT FOUND.
DSMMOM395I ′.EDFAWRK′ LINE 920: .si front811 inline
DSMMOM397I ′.EDFAWRK′ WAS IMBEDDED AT LINE 2330 OF ′.EDF#CV′
DSMMOM397I ′.EDF#CV′ WAS IMBEDDED AT LINE 190 OF ′.EDF#FCV7′
DSMMOM397I ′.EDF#FCV7′ WAS IMBEDDED AT LINE 330 OF ′.EDFCOVER′
DSMMOM397I ′.EDFCOVER′ WAS IMBEDDED AT LINE 43 OF ′BPXA2MST′
DSMPSP544E PAGE SEGMENT BACK811 PSEG3820 NOT FOUND.
DSMMOM395I ′.EDFAWRK′ LINE 920: .si back811 inline
DSMMOM397I ′.EDFAWRK′ WAS IMBEDDED AT LINE 2 OF ′XAGDB′
DSMMOM397I ′XAGDB′ WAS IMBEDDED AT LINE 187 OF ′EDFPRF40′
DSMBEG323I STARTING PASS 2 OF 3.
DSMPSP544E PAGE SEGMENT FRONT811 PSEG3820 NOT FOUND.
DSMMOM395I ′.EDFAWRK′ LINE 920: .si front811 inline
DSMMOM397I ′.EDFAWRK′ WAS IMBEDDED AT LINE 2330 OF ′.EDF#CV′
DSMMOM397I ′.EDF#CV′ WAS IMBEDDED AT LINE 190 OF ′.EDF#FCV7′
DSMMOM397I ′.EDF#FCV7′ WAS IMBEDDED AT LINE 330 OF ′.EDFCOVER′
DSMMOM397I ′.EDFCOVER′ WAS IMBEDDED AT LINE 43 OF ′BPXA2MST′
DSMPSP544E PAGE SEGMENT BACK811 PSEG3820 NOT FOUND.
DSMMOM395I ′.EDFAWRK′ LINE 920: .si back811 inline
DSMMOM397I ′.EDFAWRK′ WAS IMBEDDED AT LINE 2 OF ′XAGDB′
DSMMOM397I ′XAGDB′ WAS IMBEDDED AT LINE 187 OF ′EDFPRF40′
DSMBEG323I STARTING PASS 3 OF 3.
DSMPSP544E PAGE SEGMENT FRONT811 PSEG3820 NOT FOUND.
DSMMOM395I ′.EDFAWRK′ LINE 920: .si front811 inline
DSMMOM397I ′.EDFAWRK′ WAS IMBEDDED AT LINE 2330 OF ′.EDF#CV′
DSMMOM397I ′.EDF#CV′ WAS IMBEDDED AT LINE 190 OF ′.EDF#FCV7′
DSMMOM397I ′.EDF#FCV7′ WAS IMBEDDED AT LINE 330 OF ′.EDFCOVER′
DSMMOM397I ′.EDFCOVER′ WAS IMBEDDED AT LINE 43 OF ′BPXA2MST′
DSMPSP544E PAGE SEGMENT BACK811 PSEG3820 NOT FOUND.
DSMMOM395I ′.EDFAWRK′ LINE 920: .si back811 inline
DSMMOM397I ′.EDFAWRK′ WAS IMBEDDED AT LINE 2 OF ′XAGDB′
DSMMOM397I ′XAGDB′ WAS IMBEDDED AT LINE 187 OF ′EDFPRF40′