

Microsoft_ Windows NT_

POSIX Conformance Document

Information in this document is subject to change and does not represent a commitment on the part of Microsoft Corporation. The software and/or databases described in this document are furnished under a license agreement or nondisclosure agreement. The software and/or databases may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the software in any medium except as specifically allowed in the license or nondisclosure agreement. The licensee may make one copy of the software for backup purposes. No part of this manual and/or databases may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the licensee's personal use, without the express written permission of Microsoft Corporation

© 1993 Microsoft Corporation. All rights reserved.

Microsoft, MS, MS-DOS are registered trademarks and Windows, Windows NT, NT and the Windows logo are trademarks of Microsoft Corporation.

July 3, 1993

Preface

Purpose and Audience

The purpose of this document is to meet the requirements outlined in 1.3.1.2 of POSIX.1 Standard:

A conformance document with the following information shall be available for an implementation claiming conformance to this part of ISO/IEC 9945. The conformance document shall have the same structure as this part of ISO/IEC 9945, with the information presented in the appropriately numbered sections, clauses, and subclauses. The conformance document shall not contain information about extended facilities or capabilities outside the scope of this part of ISO/IEC 9945.

The conformance document shall contain a statement that indicates the full name, number, and date of the standard that applies. The conformance document may also list international software standards that are available for use by a Conforming POSIX.1 Application. Applicable characteristics where documentation is required by one of these standards, or by standards of government bodies, may also be included.

The conformance document shall describe the limit values found in `<limits.h>` and `<unistd.h>` headers, stating values, the conditions under which those values may change, and the limits of such variations, if any.

The conformance document shall describe the behavior of the implementation for all implementation-defined features defined in this part of ISO/IEC 9945. This requirement shall be met by listing these features and providing either a specific reference to the system documentation or providing full syntax and semantics of these features. The conformance document may specify the behavior of the implementation for those features where this part of ISO/IEC 9945 states that implementations may vary or where features are identified as undefined or unspecified.

This manual is intended for C programmers who are writing POSIX-conformant programs and who need to know the behavior of the implementation-defined features discussed in the POSIX.1 Standard.

Manual Organization

This manual closely follows the POSIX.1 Standard in format and structure. Chapter 1 contains the POSIX conformance statement, the C Standard conformance statement and the contents of `<limits.h>` and `<unistd.h>`. Chapters 2 through 10 correspond to the same chapters in the POSIX.1 Standard.

Notations and Conventions

Throughout this manual, the following typographic notations are used:

ℵ The *fixed-width* font (Courier) is used for references to C language header files and utility names, and to provide examples of system input/output.

ℵ The *italic* typeface is used to reference symbolic function parameters, C language function names, C language data types, and global external variables. Examples: *printf()*, *argc*, *uid_t*, and *errno*.

- ⌘ The **bold** typeface is used to reference environment variables and with the term **NULL** pointer.
- ⌘ Symbolic constants and limits defined by various header files are enclosed in braces. Examples: {ARG_MAX}, {_CLK_TCK}.
- ⌘ Symbolic error codes which are set by many of the functions are enclosed in brackets. Examples: [ENOENT], [ENOMEM].

Section 1: General

1.1 Scope

This document defines how Microsoft Windows NT, Version 3.1, implements the implementation-defined and unspecified clauses in the ISO/IEC 9945-1:1990 POSIX Standard, henceforth known as the POSIX.1 Standard.

1.3 Conformance

1.3.1 Implementation Conformance

1.3.1.1 Requirements

For an application to run with the behavior specified by the standard, Microsoft Windows NT systems must be configured to deny the "Bypass Traverse Check" privilege to those users who run POSIX.1 applications. Denying the "Bypass Traverse Check" privilege means that when access to a file is requested, the user must have execute permission on all of the directory components of the path. If the "Bypass Traverse Check" privilege is granted, access is checked only on the last component of the path.

To deny the "Bypass Traverse Check" privilege:

1. Log in as Administrator.
2. Start the User Manager.
3. From the Policies menu, choose User Rights.
4. Choose the Show Advanced User Rights button.
5. In the Bypass Traverse Checking area, replace "Everyone" with a list of the users who don't require POSIX.1 conformance.

1.3.1.2 Documentation

Microsoft Windows NT, version 3.1, conforms to IEEE Standard 1003.1-1990, Information Technology Portable Operating System Interface (POSIX) Part 1: System Application Program Interface (API) [C Language].

1.3.3 Language-Dependent Services for the C Programming Language

1.3.3.2 C Standard Language-Dependent System Support

Microsoft Windows NT, version 3.1, meets the requirements of POSIX.1, Section 8 by reference to ISO/IEC 9989: 1990, Information Technology Programming Languages C/ANSI X3.189-1989, Programming Language C.

Section 2: Terminology and General Requirements

2.2 Definitions

2.2.2 General Terms

2.2.2.4 Appropriate Privileges

A process cannot acquire appropriate privileges.

2.2.2.55 Parent Process ID

When the lifetime of the parent process of a child process ends, the parent process ID of the child process is assigned to a special system process ID of "1".

2.2.2.57 Pathname

Pathnames which begin with two consecutive slashes are interpreted as identifying a drive specifier. The format is `//x` where `x` is a disk drive available under Windows NT. For example, `//A/tmp` would identify the `/tmp` directory on drive A.

2.2.2.69 Read-Only File System

Read-only file systems are not supported. Windows NT does not provide a mechanism to deny all write access to a file system.

2.2.2.83 Supplementary Group ID

A process's effective group ID is included in its list of supplementary group IDs.

2.3 General Concepts

2.3.1 Extended Security Controls

Windows NT does not implement any extended security controls.

2.3.2 File Access Permissions

Windows NT provides an alternate file access control mechanism that allows specification of access for users and groups other than the file owner, file group, and file other. The mechanism is implemented in the File Manager (from the Security menu, choose Permissions).

2.4 Error Numbers

Only those error codes discussed in the POSIX.1 Standard are supported.

The error [EFAULT] is reliably detected.

The maximum size of a file is 232 bytes in length (4,294,967,296 bytes).

2.5 Primitive System Data Types

Information in the following table is from the `<sys/types.h>` header file.

Type	Type Symbol
long	time_t

The following data types are defined in various header files:

Type	Type Symbol
unsigned short	wchar_t
unsigned short	wctype_t

2.7 C Language Definitions

2.7.2 POSIX.1 Symbols

Only the `{_POSIX_SOURCE}` feature test macro is supported.

2.8 Numerical Limits

2.8.2 Minimum Values

Symbolic Constant	Value
_POSIX_ARG_MAX	4096
_POSIX_CHILD_MAX	6
_POSIX_LINK_MAX	8
_POSIX_MAX_CANON	255
_POSIX_MAX_INPUT	255
_POSIX_NAME_MAX	14
_POSIX_NGROUPS_MAX	0
_POSIX_OPEN_MAX	16
_POSIX_PATH_MAX	255
_POSIX_PIPE_BUF	512
_POSIX_SSIZE_MAX	32767
_POSIX_STREAM_MAX	8
_POSIX_TZNAME_MAX	3

2.8.3 Run-Time Increaseable Values

Symbolic Constant	Value
NGROUPS_MAX	16

2.8.4 Run-Time Invariant Values

Symbolic Constant	Value
ARG_MAX	14500
CHILD_MAX	Indeterminate and undefined
OPEN_MAX	32
STREAM_MAX	20
TZNAME_MAX	10

2.8.5 Pathname Variable Values

Symbolic Constant	Value
LINK_MAX	Indeterminate
MAX_CANON	255
MAX_INPUT	255
NAME_MAX	255
PATH_MAX	512
PIPE_BUF	512

2.8.6 Invariant Values

Symbolic Constant	Value
SSIZE_MAX	32767

2.9 Symbolic Constants

Information in the following table is from the `<unistd.h>` header file.

Symbolic Variable	Value	Comments
STDIN_FILENO	0	
STDOUT_FILENO	1	
STDERR_FILENO	2	

The constants `{_POSIX_JOB_CONTROL}` and `{_POSIX_SAVED_IDS}` are defined in `<unistd.h>` and do not change under any conditions.

The constant `{_POSIX_CHOWN_RESTRICTED}` is defined in `<unistd.h>` as 1 and does not

change under any conditions.

The constant `{_POSIX_NO_TRUNC}` is defined in `<unistd.h>` as 1 and does not change under any conditions.

The constant `{_POSIX_VDISABLE}` is defined in `<unistd.h>` as -1 and does not change under any conditions.

2.9.1 Symbolic Constants for the `access()` function

Information in the following table is from the `<unistd.h>` header file.

Symbolic Variable	Value	Comments
F_OK	00	test for existence of file
X_OK	01	test for execute permission
W_OK	02	test for write permission
R_OK	04	test for read permission

2.9.2 Execution-Time Symbolic Constants for Portability Specifications

Information in the following table is from the `<unistd.h>` header file.

Symbolic Variable	Value	Comments
SEEK_SET	0	set file position relative to BOF
SEEK_CUR	1	set file position relative to current position
SEEK_END	2	set file position relative to EOF

2.9.3 Compile-Time Symbolic Constants for Portability Specifications

Information in the following table is from the `<unistd.h>` header file.

Symbolic Variable	Value	Comments
_POSIX_JOB_CONTROL		is defined
_POSIX_VERSION	199009L	version of POSIX.1 Standard supported
_POSIX_SAVED_IDS		is defined

2.9.4 Execution-Time Symbolic Constants for Portability Specifications

Information in the following table is from the `<unistd.h>` header file.

Symbolic Variable	Value	Comments
_POSIX_CHOWN_RESTRICTED	1	
_POSIX_NO_TRUNC	1	
_POSIX_VDISABLE	(-1)	

The constant `{_POSIX_CHOWN_RESTRICTED}` is defined in `<unistd.h>` with a value of 1 and the option is provided on all applicable files as described in the POSIX.1 Standard.

The constant `{_POSIX_NO_TRUNC}` is defined in `<unistd.h>` with a value of 1 and the option is provided on all applicable files as described in the POSIX.1 Standard.

The constant `{_POSIX_VDISABLE}` is defined in `<unistd.h>` with a value of -1 and therefore option is not provided on any file.

Section 3: Process Primitives

3.1 Process Creation and Execution

3.1.1 Process Creation

3.1.1.2 Description

The calling process (parent) and the new process (child) after a call to `fork()` can not share open directory streams.

3.1.1.4 Errors

The *fork()* function does not support the detection of [ENOMEM].

3.1.2 Execute a File

3.1.2.2 Description

When the environment variable **PATH** is not defined, the search for the executable file when calling *execlp()* and *execvp()* is limited to the current directory.

When an *exec* function fails to execute, but is able to locate a process image file, the *st_atime* field for the process image file is not updated.

3.1.2.4 Errors

The execution of files other than regular files is not supported.

The *exec* functions do support the detection of [ENOMEM].

3.2 Process Termination

3.2.2 Terminate a Process

3.2.2.2 Description

The child processes of a terminated process are assigned a new parent process ID of "1".

3.3 Signals

3.3.1 Signal Concepts

3.3.1.2 Signal Generation and Delivery

A signal is discarded upon generation when the signal is being blocked and the action associated with the signal is SIG_IGN.

When a signal which is already pending is generated again, the signal is only delivered once.

3.3.1.3 Signal Actions

When a process sets the actions for the SIGCHLD signal to SIG_IGN, the signal will be ignored.

A SIGCHLD signal will not be generated when a process establishes a signal-catching function for the SIGCHLD signal while it has a terminated child process for which it has not waited.

3.3.3 Manipulate Signal Sets

3.3.3.4 Errors

If the value of the *signo* argument is an unsupported signal number, then the error condition is detected by the *sigaddset()*, *sigdelset()*, and *sigismember()* functions.

3.3.6 Examine Pending Signals

3.3.6.4 Errors

The *sigpending()* function detects [EFAULT] when the value of the *set* argument is an invalid pointer.

Section 4: Process Environment

4.2 User Identification

4.2.2 Set User and Group IDs

4.2.2.2 Description

There are no methods for obtaining appropriate privileges to set the real user ID, effective user ID, saved set-user-ID, real group ID, effective group ID, or saved set-group-ID.

4.2.3 Get Supplementary Group IDs

4.2.3.2 Description

The value of the array entries in *grouplist[]* with indices larger than or equal to the returned value from *getgroups()* is unchanged by a call to *getgroups()*.

4.2.4 Get User Name

4.2.4.3 Returns

The login name pointed to by the pointer returned from a call to *getlogin()* is overwritten on each call to *getlogin()*.

4.2.4.4 Errors

The *getlogin()* function does not support the detection of errors.

4.4 System Identification

4.4.1 Get System Name

4.4.1.2 Description

The following values are supported for each member of the *utsname* structure:

Member Name	Values	Comments
sysname	Windows NT	Operating system name
nodename		Name of node within network
release	3	Current release of product
version	1	Current version level of release
machine	i386 i486 i860 R2000 R3000 R4000 Alpha 21064 Pentium	Hardware platform

4.4.1.4 Errors

The *uname()* function returns [EFAULT] when the *name* argument is an invalid address.

4.5 Time

4.5.1 Get System Time

4.5.1.4 Errors

The *time()* function sets *errno* to [EFAULT] when the *tloc* argument is an invalid address. In addition, the *time()* function returns -1 when the date is not within the range 1970 to 2105.

4.5.2 Get Process Times

4.5.1.3 Returns

The return value from a call to the *times()* function can overflow the range of type *clock_t*.

4.5.1.4 Errors

The *times()* function returns -1 and sets *errno* to [EFAULT] when the *buffer* argument is an invalid address.

4.6 Environment Variables

4.6.1 Environment Access

4.6.1.3 Returns

The pointer returned from the *getenv()* function does not address static data and is therefore not overwritten on each call to *getenv()*.

4.6.1.4 Errors

The *getenv()* function does not support the detection of errors.

4.7 Terminal Identification

4.7.1 General Terminal Pathname

4.7.1.3 Returns

The pathname addressed by the pointer returned from a call to *ctermid()* is overwritten on each call to *ctermid()*.

4.7.1.4 Errors

The *ctermid()* function does not support the detection of errors.

4.7.2 Determine Terminal Device Name

4.7.2.2 Description

The pathname addressed by the pointer returned from a call to *ttyname()* is overwritten on each call to *ttyname()*.

4.7.2.4 Errors

The *ttyname()* and *isatty()* functions do not support the detection of errors.

4.8 Configurable System Variables

4.8.1 Get Configurable System Variables

4.8.1.2 Description

Information in the following table is from the `<unistd.h>` header file.

Symbolic Variable	Value	Comments
<code>_SC_ARG_MAX</code>	1	
<code>_SC_CHILD_MAX</code>	2	
<code>_SC_CLK_TCK</code>	3	
<code>_SC_NGROUPS_MAX</code>	4	
<code>_SC_OPEN_MAX</code>	5	
<code>_SC_JOB_CONTROL</code>	6	
<code>_SC_SAVED_IDS</code>	7	
<code>_SC_VERSION</code>	8	

Only those variables specified in table listed above are supported by the *sysconf()* function.

Section 5: Files and Directories

5.1 Directories

5.1.2 Directory Operations

5.1.2.2 Description

The *readdir()* function does not return entries for dot or dot-dot.

When a file is added to or removed from a directory after a call to *opendir()* or *rewinddir()*, the change is reflected in the set of files returned via the *readdir()* function call.

The data returned by calls to the *readdir()* function is overwritten between subsequent calls to the function.

A directory pointer *dirp* is unusable after a call to *closedir()*.

5.1.2.4 Errors

The detection of [EMFILE] and [ENFILE] is supported when calling *opendir()* under the conditions described in the POSIX.1 Standard.

The detection of [EBADF] is supported for calls to *readdir()*, *closedir()*, and *rewinddir()* when *dirp* does not refer to an open directory stream.

5.2 Working Directory

5.2.2 Get Working Directory Pathname

5.2.2.2 Description

When a call is made to *getcwd()* and the first argument *buf* is **NULL** the system returns **NULL** and sets *errno* to [EFAULT].

5.2.2.3 Returns

The contents of the buffer passed to *getcwd()* after an error are unchanged.

5.2.2.4 Errors

The *getcwd()* function does not support the detection of [EACCES].

5.3 General File Creation

5.3.1 Open a File

5.3.1.2 Description

A new file's group is set to the group ID of its parent directory when the *open()* function is called with O_CREAT set.

When a call to the function *open()* is made with O_CREAT set, the bits in the third argument other than the mode bits are ignored.

When a call to the *open()* function is made with O_EXCL set and O_CREAT not set, the O_EXCL is ignored.

When a call to the *open()* function is made for file types other than FIFO, block special and character special, the status of the O_NONBLOCK flag is ignored.

When a call to the *open()* function is made for file types other than FIFO special files, regular files, and terminal device files, the value of the O_TRUNC flag is ignored.

When a call to the *open()* function is made with both O_TRUNC and O_RDONLY set, the O_TRUNC flag is ignored.

5.3.3 Set File Creation Mask

5.3.3.2 Description

Only the permission bits are supported and therefore the meaning of the bits, other than the permission bits in the argument to the *umask()* function are ignored.

5.3.4 Link to a File

5.3.4.2 Description

Linking across file systems is not supported.

Linking to a directory is not supported and therefore obtaining appropriate privileges to link to a directory is also unsupported.

The calling process needs permission to access the existing file when making a call to *link()*.

5.4 Special File Creation

5.4.1 Make a Directory

5.4.1.2 Description

When calling the *mkdir()* function the bits other than the permission bits in *mode* have no effect.

A new directory's group is set to the group ID of its parent directory when the *mkdir()* function is called.

5.4.2 Make a FIFO Special File

5.4.2.2 Description

When calling the *mkfifo()* function the bits other than the permission bits in *mode* have no effect.

A new FIFO's group is set to the group ID of its parent directory when the *mkfifo()* function is called.

5.5 File Removal

5.5.1 Remove Directory Entries

5.5.1.2 Description

The use the *unlink()* function on directories is not supported and therefore obtaining the appropriate privileges to use *unlink()* on a directory is not supported.

5.5.1.4 Errors

Because calling the *unlink()* function with a directory is unsupported, the *unlink()* function does not set *errno* to [EBUSY] and return -1 when the directory named by the *path* argument cannot be unlinked because it is being used by the system or another process.

5.5.2 Remove a Directory

5.5.2.2 Description

Calling the *rmdir()* function when the named directory is the root directory fails.

Calling the *rmdir()* function when the named directory is the root directory or the current working directory of any process does not succeed and sets *errno* to [EACCES].

5.5.2.4 Errors

A call to the *rmdir()* function does not fail when the named directory is being used by another process.

5.5.3 Rename a File

5.5.3.2 Description

Write access is required for the *old* directory and for the *new* directory (if it exists) in order for the call to *rename(old,new)* to be successful.

5.5.3.4 Errors

A call to *rename(old,new)* does not fail when the named directory is being used by another process.

5.6 File Characteristics

5.6.1 File Characteristics: Header and Data Structure

The *st_size* field in the *stat* structure for FIFO files contains its PIPE_BUF value.

5.6.2 Get File Status

5.6.2.2 Description

The *stat()* function may fail if an alternate access control mechanism denies permission to get the information for the file. This alternate access control mechanism is described in section 2.3.2.

5.6.3 Check File Accessibility

5.6.3.4 Errors

The *access()* function supports the detection of the [EINVAL] error code under the conditions described in the POSIX.1 Standard.

5.6.4 Change File Modes

5.6.4.2 Description

Calling the *chmod()* function on a file with open file descriptors has no effect on the open file descriptors.

Windows NT restricts the *chmod()* function such that no user has permission to set the {S_ISUID} or {S_ISGID} bits on any file; these bits are always ignored.

5.6.5 Change Owner and Group of a File

5.6.5.4 Errors

The *chown()* function supports the detection of the [EINVAL] error code under the conditions described in the POSIX.1 Standard.

5.6.6 Set File Access and Modification Times

5.6.6.2 Description

Windows NT does not extend the *utimebuf* structure.

5.7 Configurable Pathname Variables

5.7.1 Get Configurable Pathname Variables

5.7.1.2 Description

Information in the following table is from the `<unistd.h>` header file.

Symbolic Variable	Value	Comments
_PC_LINK_MAX	1	
_PC_MAX_CANON	2	
_PC_MAX_INPUT	3	
_PC_NAME_MAX	4	
_PC_PATH_MAX	5	
_PC_PIPE_BUF	6	
_PC_CHOWN_RESTRICTED	7	
_PC_NO_TRUNC	8	
_PC_VDISABLE	9	

Only those configurable pathname variables listed above are supported.

The association of the variable name {PIPE_BUF} is supported for all file types for calls to *pathconf()* or *fpathconf()*.

5.7.1.4 Errors

The *pathconf()* function supports the detection of [EACCES], [ENAMETOOLONG], [ENOENT], and [ENOTDIR] (but not [EINVAL]) under the conditions described in the POSIX.1 Standard.

The *fpathconf()* function supports the detection of [EINVAL] and [EBADF] under the conditions described in the POSIX.1 Standard.

Section 6: Input and Output Primitives

6.4 Input and Output

6.4.1 Read from a File

6.4.1.2 Description

When *read()* is interrupted by a signal after it has successfully read some data it returns the number of bytes read.

Because device-special files are not supported, calling the *read()* function, when the starting position is at or after the end-of-file for device-special files, is unsupported.

The return value from the *read()* function is truncated to type `ssize_t`, if necessary, when the *nbyte* parameter exceeds `SSIZE_MAX`.

6.4.1.4 Errors

When the device reports a hardware error, the *read()* function returns `-1` with *errno* set to `[EIO]`.

6.4.2 Write to a File

6.4.2.2 Description

When calling the *write()* function on a file that is not a regular file with *nbyte* set to zero, the function returns `0`.

When *write()* is interrupted by a signal after it successfully writes some data it returns the number of bytes written.

The return value from the *write()* function is truncated to type `ssize_t`, if necessary, when the *nbyte* parameter exceeds `SSIZE_MAX`.

6.4.2.2 Errors

When the device reports a hardware error, the *write()* function returns `-1` with *errno* set to `[EIO]`.

6.5 Control Operations on Files

6.5.2 File Control

6.5.2.2 Description

Advisory record locking is supported only for regular files.

If *l_len* is negative then the lock is placed on the file starting with offset *l_start* - *l_len* through *l_start* - 1. If the starting point is before the beginning of the file then *fcntl()* returns `-1` and sets *errno* to `[EINVAL]`.

6.5.2.2 Errors

The *fcntl()* function does not support the detection of the `[EDEADLK]` error code.

6.5.3 Reposition Read/Write File Offset

6.5.3.2 Description

On PIPE files, which are incapable of seeking, a call to the *lseek()* function returns `-1` and sets *errno* to `[ESPIPE]`.

Section 7: Device- And Class-Specific Functions

7.1 General Terminal Interface

Windows NT does not provide asynchronous communication ports. The interface does not support synchronous ports nor network connections.

7.1.1 Interface Characteristics

7.1.1.3 The Controlling Terminal

If a session leader has no controlling terminal and opens a terminal device file that is not already associated with a session without using the `O_NOCTTY` option, that terminal does not become the controlling terminal, because Windows NT does not provide terminal device files.

7.1.1.9 Special Characters

Changing the START and STOP characters is not supported.

7.1.2 Parameters That Can Be Set

7.1.2.2 Input Modes

The break condition is not defined for contexts other than asynchronous serial data transmission.

The initial input control value after *open()* is zero.

7.1.2.3 **Output Modes**

If OPOST is set, output data is transmitted without change.

The initial output control value after *open()* is zero.

7.1.2.4 **Control Modes**

The initial hardware control value after *open()* is zero.

7.1.2.5 **Local Modes**

Setting IEXTEN has no effect on the interpretation of the ICANON, ISIG, IXON, and IXOFF flags.

The initial local control values after *open()* are zero.

7.1.2.6 **Special Control Characters**

The initial value of all control characters is zero.

7.1.3 **Baud Rate Functions**

7.1.3.2 **Description**

The *cfsetispeed()* and *cfsetospeed()* functions do not support the detection of an error when an unsupported baud rate is set.

7.1.3.4 **Errors**

The functions *cfgetispeed()*, *cfgetospeed()*, *cfsetispeed()*, and *cfsetospeed()* do not support the detection of errors.

7.2 **General Terminal Interface Control Functions**

7.2.1 **Get and Set State**

7.2.1.2 **Description**

The system does not provide support for any devices that support the general terminal interface. As such, the *tcsetattr()* function does not support differing input and output baud rates.

7.2.2 **Line Control Functions**

7.2.2.2 **Description**

The system does not provide support for any devices that support the general terminal interface. As such, the *tcsendbreak()* function does not send breaks.

Section 8: Language Specific Services for C

8.1 **Referenced C Language Routines**

8.1.1 **Extensions to Time Functions**

If TZ is of the " : characters " format, the characters following the colon are ignored.

8.1.2 **Extensions to *setlocale()* Function**

8.1.2.2 **Description**

The *category* argument defines the functions affected by the *setlocale()* function. The *locale* argument is a pointer to a string that specified the name of the locale.

If *locale* points to an empty string, environment variables can provide a value (\$LANG, \$LC_ALL, or specific variables for the category being set). If none of these environment variables is present or if they have null values, the *locale* argument defaults to the value C (the minimal

ANSI conforming environment for C translation).

The *locale* argument can be composed of any combination of three separate string options: language, country, and code page. The *locale* argument takes the following form:

```
locale  "lang[_country[.code_page]]"  
        | ".code_page"  
        | ""  
        | NULL
```

C and POSIX are supported values for *locale*. Other possible values for *locale* are described in the following tables of language and country values:

Sublanguage	Primary Language	Language String
Albanian	Albanian	"albanian"
Albanian	Albanian	"sqi"
Arabic	Arabic	"ara"
Arabic	Arabic	"arabic"
Bahasa	Bahasa	"bah"
Bahasa	Bahasa	"bahasa"
Bulgarian	Bulgarian	"bgr"
Bulgarian	Bulgarian	"bulgarian"
Catalan	Catalan	"cat"
Catalan	Catalan	"catalan"
Chinese	Chinese	"chinese"
Chinese (simplified)	Chinese	"chinese-simplified"
Chinese (simplified)	Chinese	"chs"
Chinese (traditional)	Chinese	"chinese-traditional"
Chinese (traditional)	Chinese	"cht"
Czech	Czech	"csy"
Czech	Czech	"czech"
Danish	Danish	"dan"
Danish	Danish	"danish"
Dutch (Belgian)	Dutch	"belgian"
Dutch (Belgian)	Dutch	"dutch-belgian"
Dutch (Belgian)	Dutch	"nlb"
Dutch (default)	Dutch	"dutch"
Dutch (default)	Dutch	"nld"
English (Australian)	English	"australian"
English (Australian)	English	"ena"
English (Australian)	English	"english-aus"
English (Canadian)	English	"canadian"
English (Canadian)	English	"enc"
English (Canadian)	English	"english-can"
English (default)	English	"english"

English (New Zealand)	English	"english-nz"
English (New Zealand)	English	"enz"
English (UK)	English	"eng"
English (UK)	English	"english-uk"
English (UK)	English	"uk"
English (USA)	English	"american"
English (USA)	English	"american english"
English (USA)	English	"american-english"
English (USA)	English	"english-american"
English (USA)	English	"english-us"
English (USA)	English	"english-usa"
English (USA)	English	"enu"
English (USA)	English	"us"
English (USA)	English	"usa"
Finnish	Finnish	"fin"
Finnish	Finnish	"finnish"
French (Belgian)	French	"frb"
French (Belgian)	French	"french-belgian"
French (Canadian)	French	"frc"
French (Canadian)	French	"french-canadian"
French (default)	French	"fra"
French (default)	French	"french"
French (Swiss)	French	"french-swiss"
French (Swiss)	French	"frs"
German (Austrian)	German	"dea"
German (Austrian)	German	"german-austrian"
German (default)	German	"deu"
German (default)	German	"german"
German (Swiss)	German	"des"
German (Swiss)	German	"german-swiss"
German (Swiss)	German	"swiss"
Greek	Greek	"ell"
Greek	Greek	"greek"
Hebrew	Hebrew	"heb"
Hebrew	Hebrew	"hebrew"
Hungarian	Hungarian	"hun"
Hungarian	Hungarian	"hungarian"
Icelandic	Icelandic	"icelandic"
Icelandic	Icelandic	"isl"
Indonesian	Indonesian	"indonesian"
Italian (default)	Italian	"ita"
Italian (default)	Italian	"italian"
Italian (Swiss)	Italian	"italian-swiss"
Italian (Swiss)	Italian	"its"
Japanese	Japanese	"japanese"

Japanese	Japanese	"jpn"
Korean	Korean	"kor"
Korean	Korean	"korean"
Norwegian (Bokmal)	Norwegian	"nor"
Norwegian (Bokmal)	Norwegian	"norwegian-bokmal"
Norwegian (default)	Norwegian	"norwegian"
Norwegian (Nynorsk)	Norwegian	"non"
Norwegian (Nynorsk)	Norwegian	"norwegian-nynorsk"
Polish	Polish	"plk"
Polish	Polish	"polish"
Portuguese (Brazilian)	Portuguese	"portuguese-brazilian"
Portuguese (Brazilian)	Portuguese	"ptb"
Portuguese (default)	Portuguese	"portuguese"
Portuguese (default)	Portuguese	"ptg"
Rhaeto-Roman	Rhaeto-Roman	"rhaeto-roman"
Rhaeto-Roman	Rhaeto-Roman	"rms"
Romanian	Romanian	"rom"
Romanian	Romanian	"romanian"
Russian (default)	Russian	"rus"
Russian (default)	Russian	"russian"
Serbo-Croatian (Cyrillic)	Serbo-Croatian	"serbian"
Serbo-Croatian (Cyrillic)	Serbo-Croatian	"serbo-croatian-cyrillic"
Serbo-Croatian (Cyrillic)	Serbo-Croatian	"shc"
Serbo-Croatian (default)	Serbo-Croatian	"serbo-croatian"
Serbo-Croatian (default)	Serbo-Croatian	"shl"
Serbo-Croatian (Latin)	Serbo-Croatian	"croatian"
Serbo-Croatian (Latin)	Serbo-Croatian	"serbo-croatian-latin"
Slovak	Slovak	"sky"
Slovak	Slovak	"slovak"
Spanish (default)	Spanish	"esp"
Spanish (default)	Spanish	"spanish"
Spanish (Mexican)	Spanish	"esm"
Spanish (Mexican)	Spanish	"spanish-mexican"
Spanish (Modern)	Spanish	"esn"
Spanish (Modern)	Spanish	"spanish-modern"
Swedish	Swedish	"sve"
Swedish	Swedish	"swedish"
Thai	Thai	"tha"
Thai	Thai	"thai"
Turkish	Turkish	"trk"
Turkish	Turkish	"turkish"
Urdu	Urdu	"urd"
Urdu	Urdu	"urdu"

Country

Country String

Australia	"aus"
Australia	"australia"
Austria	"austria"
Austria	"aut"
Belgium	"bel"
Belgium	"belgium"
Brazil	"bra"
Brazil	"brazil"
Canada	"can"
Canada	"canada"
Denmark	"denmark"
Denmark	"dnk"
Finland	"fin"
Finland	"finland"
France	"fra"
France	"france"
Germany	"deu"
Germany	"germany"
Iceland	"iceland"
Iceland	"ireland"
Iceland	"isl"
Ireland	"irl"
Italy	"ita"
Italy	"italy"
Japan	"japan"
Japan	"jpn"
Mexico	"mex"
Mexico	"mexico"
Netherlands	"holland"
Netherlands	"netherlands"
Netherlands	"nld"
New Zealand	"new zealand"
New Zealand	"new-zealand"
New Zealand	"nz"
New Zealand	"nzl"
Norway	"nor"
Norway	"norway"
People's Republic of China	"china"
People's Republic of China	"chn"
People's Republic of China	"pr china"
People's Republic of China	"pr-china"
Portugal	"portugal"
Portugal	"prt"
South Korea	"kor"
South Korea	"korea"

South Korea	"south korea"
South Korea	"south-korea"
Spain	"esp"
Spain	"spain"
Sweden	"swe"
Sweden	"sweden"
Switzerland	"che"
Switzerland	"switzerland"
Taiwan	"taiwan"
Taiwan	"twm"
United Kingdom	"britain"
United Kingdom	"england"
United Kingdom	"gbr"
United Kingdom	"great britain"
United Kingdom	"uk"
United Kingdom	"united kingdom"
United Kingdom	"united-kingdom"
United States of America	"america"
United States of America	"united states"
United States of America	"united-states"
United States of America	"us"
United States of America	"usa"

8.2 C Language Input/Output Functions

8.2.1 Map a Stream Pointer to a File Descriptor

8.2.1.4 Errors

The *fileno()* function not support the detection of errors.

8.2.2 Open a Stream on a File Descriptor

8.2.2.2 Description

There are no additional values supported for the *type* argument beyond those specified in the POSIX.1 Standard.

8.2.2.4 Errors

The *fdopen()* function does support the detection of errors. If the *type* argument is not one of those specified in the POSIX.1 Standard, a call to *fdopen()* returns -1 and sets *errno* to [EINVAL].

8.2.3 Interactions of Other *FILE*-Type C Functions

Output may be seen twice if a process forks while data is buffered for a stream. Similarly, input may be seen twice under the same circumstances.

8.3 Other C Language Functions

8.3.2 Set Time Zone

8.3.2.2 Description

When **TZ** variable is absent from the environment the default value of "PST8PDT" is used for the time-zone.

Section 9: System Databases

9.1 System Databases

The initial working directory may be set from the Windows NT User Manager. If the initial working directory is **NULL** then a value of `"/ /C/"` is used. The initial working program is always `"noshell"`.

Only the fields specified in the POSIX.1 Standard for the *group* and *user* databases are available to a POSIX-compliant program.

9.2 Database Access

9.2.1 Group Database Access

9.2.1.3 Returns

The data returned from calls to the functions *getgrgid()* and *getgrnam()* overwrites the data from previous calls to either function. For example, a call to *getgrgid()* overwrites data from a previous call to *getgrnam()*.

9.2.1.4 Errors

The *getgrgid()* and *getgrnam()* functions do not support the detection of errors.

9.2.2 User Database Access

9.2.2.3 Returns

The data returned from calls to the functions *getpwuid()* and *getpwnam()* overwrites the data from previous calls to either function. For example, a call to *getpwuid()* overwrites data from a previous call to *getpwnam()*.

9.2.2.4 Errors

The *getpwuid()* and *getpwnam()* functions do not support the detection of errors.

Section 10: Data Interchange Format

10.1 Archive/Interchange File Format

The name of the *format-creating utility* and *format-reading utility* is `pax`. The `pax` utility is fully described in the IEEE POSIX.2-1992 Standard.

10.1.1 Extended `tar` Format

When a file name is found on the medium that would create an invalid file name, the `pax` utility does not store the file into the archive.

10.1.2 Extended `cpio` Format

10.1.2.2 `cpio` File Name

When a file name is found on the medium that would create an invalid pathname, the file is not stored in the archive file.

10.1.3 Multiple Volumes

The `pax` format-creating utility for the `tar` and `cpio` formats determine what file to read or write for the next volume of a multi-volume archive by prompting the user.

