NAME

unistd.h − standard symbolic constants and types

SYNOPSIS

#include <unistd.h>

DESCRIPTION

The <*unistd.h*> header defines miscellaneous symbolic constants and types, and declares miscellaneous functions. The actual values of the constants are unspecified except as shown. The contents of this header are shown below.

Version Test Macros

The following symbolic constants shall be defined:

_POSIX_VERSION

Integer value indicating version of IEEE Std 1003.1 (C-language binding) to which the implementation conforms. For implementations conforming to IEEE Std 1003.1-2001, the value shall be 200112L.

_POSIX2_VERSION

Integer value indicating version of the Shell and Utilities volume of IEEE Std 1003.1 to which the implementation conforms. For implementations conforming to IEEE Std 1003.1-2001, the value shall be 200112L.

The following symbolic constant shall be defined only if the implementation supports the XSI option; see *XSI Conformance* .

_XOPEN_VERSION

Integer value indicating version of the X/Open Portability Guide to which the implementation conforms. The value shall be 600.

Constants for Options and Option Groups

The following symbolic constants, if defined in <*unistd.h*>, shall have a value of -1, 0, or greater, unless otherwise specified below. If these are undefined, the *fpathconf*(), *pathconf*(), or *sysconf*() functions can be used to determine whether the option is provided for a particular invocation of the application.

If a symbolic constant is defined with the value -1, the option is not supported. Headers, data types, and function interfaces required only for the option need not be supplied. An application that attempts to use anything associated only with the option is considered to be requiring an extension.

If a symbolic constant is defined with a value greater than zero, the option shall always be supported when the application is executed. All headers, data types, and functions shall be present and shall operate as specified.

If a symbolic constant is defined with the value zero, all headers, data types, and functions shall be present. The application can check at runtime to see whether the option is supported by calling *fpathconf*(), *pathconf*(), or *sysconf*() with the indicated *name* parameter.

Unless explicitly specified otherwise, the behavior of functions associated with an unsupported option is unspecified, and an application that uses such functions without first checking *fpathconf*(), *pathconf*(), or *sysconf*() is considered to be requiring an extension.

For conformance requirements, refer to *Conformance* .

_POSIX_ADVISORY_INFO

The implementation supports the Advisory Information option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_ASYNCHRONOUS_IO

The implementation supports the Asynchronous Input and Output option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_BARRIERS

The implementation supports the Barriers option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_CHOWN_RESTRICTED

The use of *chown*() and *fchown*() is restricted to a process with appropriate privileges, and to changing the group ID of a file only to the effective group ID of the process or to one of its supplementary group IDs. This symbol shall always be set to a value other than -1.

_POSIX_CLOCK_SELECTION

The implementation supports the Clock Selection option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_CPUTIME

The implementation supports the Process CPU-Time Clocks option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_FSYNC

The implementation supports the File Synchronization option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_IPV6

The implementation supports the IPv6 option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_JOB_CONTROL

The implementation supports job control. This symbol shall always be set to a value greater than zero.

_POSIX_MAPPED_FILES

The implementation supports the Memory Mapped Files option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_MEMLOCK

The implementation supports the Process Memory Locking option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_MEMLOCK_RANGE

The implementation supports the Range Memory Locking option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_MEMORY_PROTECTION

The implementation supports the Memory Protection option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_MESSAGE_PASSING

The implementation supports the Message Passing option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_MONOTONIC_CLOCK

The implementation supports the Monotonic Clock option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_NO_TRUNC

Pathname components longer than {NAME_MAX} generate an error. This symbol shall always be set to a value other than -1.

_POSIX_PRIORITIZED_IO

The implementation supports the Prioritized Input and Output option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_PRIORITY_SCHEDULING

The implementation supports the Process Scheduling option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_RAW_SOCKETS

The implementation supports the Raw Sockets option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_READER_WRITER_LOCKS

The implementation supports the Read-Write Locks option. This is always set to a value greater than zero if the Threads option is supported. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_REALTIME_SIGNALS

The implementation supports the Realtime Signals Extension option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_REGEXP

The implementation supports the Regular Expression Handling option. This symbol shall always be set to a value greater than zero.

_POSIX_SAVED_IDS

Each process has a saved set-user-ID and a saved set-group-ID. This symbol shall always be set to a value greater than zero.

_POSIX_SEMAPHORES

The implementation supports the Semaphores option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_SHARED_MEMORY_OBJECTS

The implementation supports the Shared Memory Objects option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_SHELL

The implementation supports the POSIX shell. This symbol shall always be set to a value greater than zero.

_POSIX_SPAWN

The implementation supports the Spawn option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_SPIN_LOCKS

The implementation supports the Spin Locks option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_SPORADIC_SERVER

The implementation supports the Process Sporadic Server option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_SYNCHRONIZED_IO

The implementation supports the Synchronized Input and Output option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_THREAD_ATTR_STACKADDR

The implementation supports the Thread Stack Address Attribute option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_THREAD_ATTR_STACKSIZE

The implementation supports the Thread Stack Size Attribute option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_THREAD_CPUTIME

The implementation supports the Thread CPU-Time Clocks option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_THREAD_PRIO_INHERIT

The implementation supports the Thread Priority Inheritance option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_THREAD_PRIO_PROTECT

The implementation supports the Thread Priority Protection option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_THREAD_PRIORITY_SCHEDULING

The implementation supports the Thread Execution Scheduling option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_THREAD_PROCESS_SHARED

The implementation supports the Thread Process-Shared Synchronization option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_THREAD_SAFE_FUNCTIONS

The implementation supports the Thread-Safe Functions option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_THREAD_SPORADIC_SERVER

The implementation supports the Thread Sporadic Server option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_THREADS

The implementation supports the Threads option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_TIMEOUTS

The implementation supports the Timeouts option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_TIMERS

The implementation supports the Timers option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_TRACE

The implementation supports the Trace option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_TRACE_EVENT_FILTER

The implementation supports the Trace Event Filter option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_TRACE_INHERIT

The implementation supports the Trace Inherit option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_TRACE_LOG

The implementation supports the Trace Log option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_TYPED_MEMORY_OBJECTS

The implementation supports the Typed Memory Objects option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX_VDISABLE

This symbol shall be defined to be the value of a character that shall disable terminal special character handling as described in *<termios.h>* . This symbol shall always be set to a value other than -1.

_POSIX2_C_BIND

The implementation supports the C-Language Binding option. This symbol shall always have the value 200112L.

_POSIX2_C_DEV

The implementation supports the C-Language Development Utilities option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX2_CHAR_TERM

The implementation supports at least one terminal type.

_POSIX2_FORT_DEV

The implementation supports the FORTRAN Development Utilities option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX2_FORT_RUN

The implementation supports the FORTRAN Runtime Utilities option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX2_LOCALEDEF

The implementation supports the creation of locales by the *localedef* utility. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX2_PBS

The implementation supports the Batch Environment Services and Utilities option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX2_PBS_ACCOUNTING

The implementation supports the Batch Accounting option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX2_PBS_CHECKPOINT

The implementation supports the Batch Checkpoint/Restart option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX2_PBS_LOCATE

The implementation supports the Locate Batch Job Request option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX2_PBS_MESSAGE

The implementation supports the Batch Job Message Request option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX2_PBS_TRACK

The implementation supports the Track Batch Job Request option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX2_SW_DEV

The implementation supports the Software Development Utilities option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_POSIX2_UPE

The implementation supports the User Portability Utilities option. If this symbol has a value other than -1 or 0, it shall have the value 200112L.

_V6_ILP32_OFF32

The implementation provides a C-language compilation environment with 32-bit **int**, **long**, **pointer**, and **off_t** types.

_V6_ILP32_OFFBIG

The implementation provides a C-language compilation environment with 32-bit **int**, **long**, and **pointer** types and an **off_t** type using at least 64 bits.

_V6_LP64_OFF64

The implementation provides a C-language compilation environment with 32-bit **int** and 64-bit **long**, **pointer**, and **off_t** types.

_V6_LPBIG_OFFBIG
The implementation provides a C-language compilation environment with an **int** type using at least 32 bits and **long**, **pointer**, and **off_t** types using at least 64 bits.

_XBS5_ILP32_OFF32 (**LEGACY**)

The implementation provides a C-language compilation environment with 32-bit **int**, **long**, **pointer**, and **off_t** types.

_XBS5_ILP32_OFFBIG (**LEGACY**)

The implementation provides a C-language compilation environment with 32-bit **int**, **long**, and **pointer** types and an **off_t** type using at least 64 bits.

_XBS5_LP64_OFF64 (**LEGACY**)

The implementation provides a C-language compilation environment with 32-bit **int** and 64-bit **long**, **pointer**, and **off_t** types.

_XBS5_LPBIG_OFFBIG (**LEGACY**)

The implementation provides a C-language compilation environment with an **int** type using at least 32 bits and **long**, **pointer**, and **off_t** types using at least 64 bits.

_XOPEN_CRYPT

The implementation supports the X/Open Encryption Option Group.

_XOPEN_ENH_I18N
The implementation supports the Issue 4, Version 2 Enhanced Internationalization Option Group. This symbol shall always be set to a value other than -1.

_XOPEN_LEGACY
The implementation supports the Legacy Option Group.

_XOPEN_REALTIME
The implementation supports the X/Open Realtime Option Group.

_XOPEN_REALTIME_THREADS
The implementation supports the X/Open Realtime Threads Option Group.

_XOPEN_SHM
The implementation supports the Issue 4, Version 2 Shared Memory Option Group. This symbol shall always be set to a value other than -1.

_XOPEN_STREAMS
The implementation supports the XSI STREAMS Option Group.

_XOPEN_UNIX

The implementation supports the XSI extension.

Execution-Time Symbolic Constants

If any of the following constants are not defined in the <*unistd.h*> header, the value shall vary depending on the file to which it is applied.

If any of the following constants are defined to have value -1 in the <*unistd.h*> header, the implementation shall not provide the option on any file; if any are defined to have a value other than -1 in the <*unistd.h*> header, the implementation shall provide the option on all applicable files.

All of the following constants, whether defined in <*unistd.h*> or not, may be queried with respect to a specific file using the *pathconf*() or *fpathconf*() functions:

_POSIX_ASYNC_IO
Asynchronous input or output operations may be performed for the associated file.

_POSIX_PRIO_IO
Prioritized input or output operations may be performed for the associated file.

_POSIX_SYNC_IO
Synchronized input or output operations may be performed for the associated file.

Constants for Functions

The following symbolic constant shall be defined:

NULL Null pointer

The following symbolic constants shall be defined for the *access*() function:

F_OK Test for existence of file.

R_OK Test for read permission.

W_OK Test for write permission.

X_OK Test for execute (search) permission.

The constants F_OK, R_OK, W_OK, and X_OK and the expressions *R_OK*|*W_OK*, *R_OK*|*X_OK*, and *R_OK*|*W_OK*|*X_OK* shall all have distinct values.

The following symbolic constants shall be defined for the *confstr*() function:

_CS_PATH
This is the value for the *PATH* environment variable that finds all standard utilities.

_CS_POSIX_V6_ILP32_OFF32_CFLAGS
If *sysconf*(_SC_V6_ILP32_OFF32) returns -1, the meaning of this value is unspecified. Otherwise, this value is the set of initial options to be given to the *c99* utility to build an application using a programming model with 32-bit **int**, **long**, **pointer**, and **off_t** types.

_CS_POSIX_V6_ILP32_OFF32_LDFLAGS
If *sysconf*(_SC_V6_ILP32_OFF32) returns -1, the meaning of this value is unspecified. Otherwise, this value is the set of final options to be given to the *c99* utility to build an application using a programming model with 32-bit **int**, **long**, **pointer**, and **off_t** types.

_CS_POSIX_V6_ILP32_OFF32_LIBS
If *sysconf*(_SC_V6_ILP32_OFF32) returns -1, the meaning of this value is unspecified. Otherwise, this value is the set of libraries to be given to the *c99* utility to build an application using a programming model with 32-bit **int**, **long**, **pointer**, and **off_t** types.

_CS_POSIX_V6_ILP32_OFFBIG_CFLAGS
If *sysconf*(_SC_V6_ILP32_OFFBIG) returns -1, the meaning of this value is unspecified. Otherwise, this value is the set of initial options to be given to the *c99* utility to build an application using a programming model with 32-bit **int**, **long**, and **pointer** types, and an **off_t** type using at least 64 bits.

_CS_POSIX_V6_ILP32_OFFBIG_LDFLAGS
If *sysconf*(_SC_V6_ILP32_OFFBIG) returns -1, the meaning of this value is unspecified. Otherwise, this value is the set of final options to be given to the *c99* utility to build an application using a programming model with 32-bit **int**, **long**, and **pointer** types, and an **off_t** type using at least 64 bits.

_CS_POSIX_V6_ILP32_OFFBIG_LIBS
If *sysconf*(_SC_V6_ILP32_OFFBIG) returns -1, the meaning of this value is unspecified. Otherwise, this value is the set of libraries to be given to the *c99* utility to build an application using a

programming model with 32-bit **int**, **long**, and **pointer** types, and an **off_t** type using at least 64 bits.

_CS_POSIX_V6_LP64_OFF64_CFLAGS
: If *sysconf*(_SC_V6_LP64_OFF64) returns -1, the meaning of this value is unspecified. Otherwise, this value is the set of initial options to be given to the *c99* utility to build an application using a programming model with 32-bit **int** and 64-bit **long**, **pointer**, and **off_t** types.

_CS_POSIX_V6_LP64_OFF64_LDFLAGS
: If *sysconf*(_SC_V6_LP64_OFF64) returns -1, the meaning of this value is unspecified. Otherwise, this value is the set of final options to be given to the *c99* utility to build an application using a programming model with 32-bit **int** and 64-bit **long**, **pointer**, and **off_t** types.

_CS_POSIX_V6_LP64_OFF64_LIBS
: If *sysconf*(_SC_V6_LP64_OFF64) returns -1, the meaning of this value is unspecified. Otherwise, this value is the set of libraries to be given to the *c99* utility to build an application using a programming model with 32-bit **int** and 64-bit **long**, **pointer**, and **off_t** types.

_CS_POSIX_V6_LPBIG_OFFBIG_CFLAGS
: If *sysconf*(_SC_V6_LPBIG_OFFBIG) returns -1, the meaning of this value is unspecified. Otherwise, this value is the set of initial options to be given to the *c99* utility to build an application using a programming model with an **int** type using at least 32 bits and **long**, **pointer**, and **off_t** types using at least 64 bits.

_CS_POSIX_V6_LPBIG_OFFBIG_LDFLAGS
: If *sysconf*(_SC_V6_LPBIG_OFFBIG) returns -1, the meaning of this value is unspecified. Otherwise, this value is the set of final options to be given to the *c99* utility to build an application using a programming model with an **int** type using at least 32 bits and **long**, **pointer**, and **off_t** types using at least 64 bits.

_CS_POSIX_V6_LPBIG_OFFBIG_LIBS
: If *sysconf*(_SC_V6_LPBIG_OFFBIG) returns -1, the meaning of this value is unspecified. Otherwise, this value is the set of libraries to be given to the *c99* utility to build an application using a programming model with an **int** type using at least 32 bits and **long**, **pointer**, and **off_t** types using at least 64 bits.

_CS_POSIX_V6_WIDTH_RESTRICTED_ENVS
: This value is a <newline>-separated list of names of programming environments supported by the implementation in which the widths of the **blksize_t**, **cc_t**, **mode_t**, **nfds_t**, **pid_t**, **ptrdiff_t**, **size_t**, **speed_t**, **ssize_t**, **suseconds_t**, **tcflag_t**, **useconds_t**, **wchar_t**, and **wint_t** types are no greater than the width of type **long**.

The following symbolic constants are reserved for compatibility with Issue 5:

_CS_XBS5_ILP32_OFF32_CFLAGS (**LEGACY**)
_CS_XBS5_ILP32_OFF32_LDFLAGS (**LEGACY**)
_CS_XBS5_ILP32_OFF32_LIBS (**LEGACY**)
_CS_XBS5_ILP32_OFF32_LINTFLAGS (**LEGACY**)
_CS_XBS5_ILP32_OFFBIG_CFLAGS (**LEGACY**)
_CS_XBS5_ILP32_OFFBIG_LDFLAGS (**LEGACY**)
_CS_XBS5_ILP32_OFFBIG_LIBS (**LEGACY**)
_CS_XBS5_ILP32_OFFBIG_LINTFLAGS (**LEGACY**)
_CS_XBS5_LP64_OFF64_CFLAGS (**LEGACY**)
_CS_XBS5_LP64_OFF64_LDFLAGS (**LEGACY**)
_CS_XBS5_LP64_OFF64_LIBS (**LEGACY**)
_CS_XBS5_LP64_OFF64_LINTFLAGS (**LEGACY**)
_CS_XBS5_LPBIG_OFFBIG_CFLAGS (**LEGACY**)
_CS_XBS5_LPBIG_OFFBIG_LDFLAGS (**LEGACY**)
_CS_XBS5_LPBIG_OFFBIG_LIBS (**LEGACY**)

_CS_XBS5_LPBIG_OFFBIG_LINTFLAGS (**LEGACY**)

The following symbolic constants shall be defined for the *lseek*() and *fcntl*() functions and shall have distinct values:

SEEK_CUR
: Set file offset to current plus *offset*.

SEEK_END
: Set file offset to EOF plus *offset*.

SEEK_SET
: Set file offset to *offset*.

The following symbolic constants shall be defined as possible values for the *function* argument to the *lockf*() function:

F_LOCK
: Lock a section for exclusive use.

F_TEST
: Test section for locks by other processes.

F_TLOCK
: Test and lock a section for exclusive use.

F_ULOCK
: Unlock locked sections.

The following symbolic constants shall be defined for *pathconf*():

_PC_ALLOC_SIZE_MIN
_PC_ASYNC_IO
_PC_CHOWN_RESTRICTED
_PC_FILESIZEBITS
_PC_LINK_MAX
_PC_MAX_CANON
_PC_MAX_INPUT
_PC_NAME_MAX
_PC_NO_TRUNC
_PC_PATH_MAX
_PC_PIPE_BUF
_PC_PRIO_IO
_PC_REC_INCR_XFER_SIZE
_PC_REC_MIN_XFER_SIZE
_PC_REC_XFER_ALIGN
_PC_SYMLINK_MAX
_PC_SYNC_IO
_PC_VDISABLE

The following symbolic constants shall be defined for *sysconf*():

_SC_2_C_BIND
_SC_2_C_DEV
_SC_2_C_VERSION
_SC_2_CHAR_TERM
_SC_2_FORT_DEV
_SC_2_FORT_RUN
_SC_2_LOCALEDEF

_SC_2_PBS
_SC_2_PBS_ACCOUNTING
_SC_2_PBS_CHECKPOINT
_SC_2_PBS_LOCATE
_SC_2_PBS_MESSAGE
_SC_2_PBS_TRACK
_SC_2_SW_DEV
_SC_2_UPE
_SC_2_VERSION
_SC_ADVISORY_INFO
_SC_ARG_MAX
_SC_AIO_LISTIO_MAX
_SC_AIO_MAX
_SC_AIO_PRIO_DELTA_MAX
_SC_ASYNCHRONOUS_IO
_SC_ATEXIT_MAX
_SC_BARRIERS
_SC_BC_BASE_MAX
_SC_BC_DIM_MAX
_SC_BC_SCALE_MAX
_SC_BC_STRING_MAX
_SC_CHILD_MAX
_SC_CLK_TCK
_SC_CLOCK_SELECTION
_SC_COLL_WEIGHTS_MAX
_SC_CPUTIME
_SC_DELAYTIMER_MAX
_SC_EXPR_NEST_MAX
_SC_FILE_LOCKING
_SC_FSYNC
_SC_GETGR_R_SIZE_MAX
_SC_GETPW_R_SIZE_MAX
_SC_HOST_NAME_MAX
_SC_IOV_MAX
_SC_IPV6
_SC_JOB_CONTROL
_SC_LINE_MAX
_SC_LOGIN_NAME_MAX
_SC_MAPPED_FILES
_SC_MEMLOCK
_SC_MEMLOCK_RANGE
_SC_MEMORY_PROTECTION
_SC_MESSAGE_PASSING
_SC_MONOTONIC_CLOCK
_SC_MQ_OPEN_MAX
_SC_MQ_PRIO_MAX
_SC_NGROUPS_MAX
_SC_OPEN_MAX
_SC_PAGE_SIZE
_SC_PAGESIZE
_SC_PRIORITIZED_IO
_SC_PRIORITY_SCHEDULING
_SC_RAW_SOCKETS
_SC_RE_DUP_MAX

_SC_READER_WRITER_LOCKS
_SC_REALTIME_SIGNALS
_SC_REGEXP
_SC_RTSIG_MAX
_SC_SAVED_IDS
_SC_SEMAPHORES
_SC_SEM_NSEMS_MAX
_SC_SEM_VALUE_MAX
_SC_SHARED_MEMORY_OBJECTS
_SC_SHELL
_SC_SIGQUEUE_MAX
_SC_SPAWN
_SC_SPIN_LOCKS
_SC_SPORADIC_SERVER
_SC_STREAM_MAX
_SC_SYMLOOP_MAX
_SC_SYNCHRONIZED_IO
_SC_THREAD_ATTR_STACKADDR
_SC_THREAD_ATTR_STACKSIZE
_SC_THREAD_CPUTIME
_SC_THREAD_DESTRUCTOR_ITERATIONS
_SC_THREAD_KEYS_MAX
_SC_THREAD_PRIO_INHERIT
_SC_THREAD_PRIO_PROTECT
_SC_THREAD_PRIORITY_SCHEDULING
_SC_THREAD_PROCESS_SHARED
_SC_THREAD_SAFE_FUNCTIONS
_SC_THREAD_SPORADIC_SERVER
_SC_THREAD_STACK_MIN
_SC_THREAD_THREADS_MAX
_SC_TIMEOUTS
_SC_THREADS
_SC_TIMER_MAX
_SC_TIMERS
_SC_TRACE
_SC_TRACE_EVENT_FILTER
_SC_TRACE_INHERIT
_SC_TRACE_LOG
_SC_TTY_NAME_MAX
_SC_TYPED_MEMORY_OBJECTS
_SC_TZNAME_MAX
_SC_V6_ILP32_OFF32
_SC_V6_ILP32_OFFBIG
_SC_V6_LP64_OFF64
_SC_V6_LPBIG_OFFBIG
_SC_VERSION
_SC_XBS5_ILP32_OFF32 (**LEGACY**)
_SC_XBS5_ILP32_OFFBIG (**LEGACY**)
_SC_XBS5_LP64_OFF64 (**LEGACY**)
_SC_XBS5_LPBIG_OFFBIG (**LEGACY**)
_SC_XOPEN_CRYPT
_SC_XOPEN_ENH_I18N
_SC_XOPEN_LEGACY
_SC_XOPEN_REALTIME

_SC_XOPEN_REALTIME_THREADS
_SC_XOPEN_SHM
_SC_XOPEN_STREAMS
_SC_XOPEN_UNIX
_SC_XOPEN_VERSION
_SC_XOPEN_XCU_VERSION

The two constants _SC_PAGESIZE and _SC_PAGE_SIZE may be defined to have the same value.

The following symbolic constants shall be defined for file streams:

STDERR_FILENO
File number of *stderr*; 2.

STDIN_FILENO
File number of *stdin*; 0.

STDOUT_FILENO
File number of *stdout*; 1.

Type Definitions

The **size_t**, **ssize_t**, **uid_t**, **gid_t**, **off_t**, **pid_t**, and **useconds_t** types shall be defined as described in *<sys/types.h>*.

The **intptr_t** type shall be defined as described in *<inttypes.h>*.

Declarations

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
int     access(const char *, int);
unsigned    alarm(unsigned);
int     chdir(const char *);
int     chown(const char *, uid_t, gid_t);
int     close(int);
size_t    confstr(int, char *, size_t);


char     *crypt(const char *, const char *);
char     *ctermid(char *);

int     dup(int);


int     dup2(int, int);

void     encrypt(char[64], int);

int     execl(const char *, const char *, ...);
int     execle(const char *, const char *, ...);
int     execlp(const char *, const char *, ...);
int     execv(const char *, char *const []);
int     execve(const char *, char *const [], char *const []);
int     execvp(const char *, char *const []);
void     _exit(int);
int     fchown(int, uid_t, gid_t);
```

```
int         fchdir(int);


int         fdatasync(int);

pid_t        fork(void);
long         fpathconf(int, int);

int         fsync(int);

int         ftruncate(int, off_t);
char         *getcwd(char *, size_t);
gid_t        getegid(void);
uid_t        geteuid(void);
gid_t        getgid(void);
int         getgroups(int, gid_t []);

long         gethostid(void);

int         gethostname(char *, size_t);
char         *getlogin(void);
int         getlogin_r(char *, size_t);
int         getopt(int, char * const [], const char *);

pid_t        getpgid(pid_t);

pid_t        getpgrp(void);
pid_t        getpid(void);
pid_t        getppid(void);

pid_t        getsid(pid_t);

uid_t        getuid(void);

char         *getwd(char *); (LEGACY )

int         isatty(int);

int         lchown(const char *, uid_t, gid_t);

int         link(const char *, const char *);

int         lockf(int, int, off_t);

off_t        lseek(int, off_t, int);

int         nice(int);

long         pathconf(const char *, int);
int         pause(void);
int         pipe(int [2]);

ssize_t      pread(int, void *, size_t, off_t);
ssize_t      pwrite(int, const void *, size_t, off_t);
```

```
ssize_t     read(int, void *, size_t);
ssize_t     readlink(const char *restrict, char *restrict, size_t);
int         rmdir(const char *);
int         setegid(gid_t);
int         seteuid(uid_t);
int         setgid(gid_t);


int         setpgid(pid_t, pid_t);

pid_t       setpgrp(void);
int         setregid(gid_t, gid_t);
int         setreuid(uid_t, uid_t);

pid_t       setsid(void);
int         setuid(uid_t);
unsigned    sleep(unsigned);

void        swab(const void *restrict, void *restrict, ssize_t);
int         symlink(const char *, const char *);
void        sync(void);

long        sysconf(int);
pid_t       tcgetpgrp(int);
int         tcsetpgrp(int, pid_t);

int         truncate(const char *, off_t);

char        *ttyname(int);
int         ttyname_r(int, char *, size_t);

useconds_t  ualarm(useconds_t, useconds_t);

int         unlink(const char *);

int         usleep(useconds_t);
pid_t       vfork(void);

ssize_t     write(int, const void *, size_t);
```

Implementations may also include the *pthread_atfork*() prototype as defined in <*pthread.h*> .

The following external variables shall be declared:

```
extern char  *optarg;
extern int   optind, opterr, optopt;
```

The following sections are informative.

APPLICATION USAGE

IEEE Std 1003.1-2001 only describes the behavior of systems that claim conformance to it. However, application developers who want to write applications that adapt to other versions of IEEE Std 1003.1 (or to systems that do not conform to any POSIX standard) may find it useful to code them so as to conditionally compile different code depending on the value of _POSIX_VERSION, for example:

```
#if _POSIX_VERSION >= 200112L
/* Use the newer function that copes with large files. */
off_t pos=ftello(fp);
#else
/* Either this is an old version of POSIX, or _POSIX_VERSION is
   not even defined, so use the traditional function. */
long pos=ftell(fp);
#endif
```

Earlier versions of IEEE Std 1003.1 and of the Single UNIX Specification can be identified by the following macros:

POSIX.1-1988 standard
: _POSIX_VERSION==198808L

POSIX.1-1990 standard
: _POSIX_VERSION==199009L

ISO POSIX-1:1996 standard
: _POSIX_VERSION==199506L

Single UNIX Specification, Version 1
: _XOPEN_UNIX and _XOPEN_VERSION==4

Single UNIX Specification, Version 2
: _XOPEN_UNIX and _XOPEN_VERSION==500

IEEE Std 1003.1-2001 does not make any attempt to define application binary interaction with the underlying operating system. However, application developers may find it useful to query _SC_VERSION at runtime via *sysconf*() to determine whether the current version of the operating system supports the necessary functionality as in the following program fragment:

```
if (sysconf(_SC_VERSION) < 200112L) {
   fprintf(stderr, "POSIX.1-2001 system required, terminating \n");
   exit(1);
}
```

New applications should not use _XOPEN_SHM or _XOPEN_ENH_I18N.

RATIONALE

As IEEE Std 1003.1-2001 evolved, certain options became sufficiently standardized that it was concluded that simply requiring one of the option choices was simpler than retaining the option. However, for backwards-compatibility, the option flags (with required constant values) are retained.

Version Test Macros

The standard developers considered altering the definition of _POSIX_VERSION and removing _SC_VERSION from the specification of *sysconf*() since the utility to an application was deemed by some to be minimal, and since the implementation of the functionality is potentially problematic. However, they recognized that support for existing application binaries is a concern to manufacturers, application developers, and the users of implementations conforming to IEEE Std 1003.1-2001.

While the example using _SC_VERSION in the APPLICATION USAGE section does not provide the greatest degree of imaginable utility to the application developer or user, it is arguably better than a **core** file or some other equally obscure result. (It is also possible for implementations to encode and recognize application binaries compiled in various POSIX.1-conforming environments, and modify the semantics of the underlying system to conform to the expectations of the application.) For the reasons outlined in the preceding paragraphs and in the APPLICATION USAGE section, the standard developers elected to retain the _POSIX_VERSION and _SC_VERSION functionality.

Compile-Time Symbolic Constants for System-Wide Options

IEEE Std 1003.1-2001 now includes support in certain areas for the newly adopted policy governing options and stubs.

This policy provides flexibility for implementations in how they support options. It also specifies how conforming applications can adapt to different implementations that support different sets of options. It allows the following:

1. If an implementation has no interest in supporting an option, it does not have to provide anything associated with that option beyond the announcement that it does not support it.
2. An implementation can support a partial or incompatible version of an option (as a non-standard extension) as long as it does not claim to support the option.
3. An application can determine whether the option is supported. A strictly conforming application must check this announcement mechanism before first using anything associated with the option.

There is an important implication of this policy. IEEE Std 1003.1-2001 cannot dictate the behavior of interfaces associated with an option when the implementation does not claim to support the option. In particular, it cannot require that a function associated with an unsupported option will fail if it does not perform as specified. However, this policy does not prevent a standard from requiring certain functions to always be present, but that they shall always fail on some implementations. The *setpgid*() function in the POSIX.1-1990 standard, for example, is considered appropriate.

The POSIX standards include various options, and the C-language binding support for an option implies that the implementation must supply data types and function interfaces. An application must be able to discover whether the implementation supports each option.

Any application must consider the following three cases for each option:

1. Option never supported.

The implementation advertises at compile time that the option will never be supported. In this case, it is not necessary for the implementation to supply any of the data types or function interfaces that are provided only as part of the option. The implementation might provide data types and functions that are similar to those defined by IEEE Std 1003.1-2001, but there is no guarantee for any particular behavior.

2. Option always supported.

The implementation advertises at compile time that the option will always be supported. In this case, all data types and function interfaces shall be available and shall operate as specified.

3. Option might or might not be supported.

Some implementations might not provide a mechanism to specify support of options at compile time. In addition, the implementation might be unable or unwilling to specify support or non-support at compile time. In either case, any application that might use the option at runtime must be able to compile and execute. The implementation must provide, at compile time, all data types and function interfaces that are necessary to allow this. In this situation, there must be a mechanism that allows the application to query, at runtime, whether the option is supported. If the application attempts to use the option when it is not supported, the result is unspecified unless explicitly specified otherwise in IEEE Std 1003.1-2001.

FUTURE DIRECTIONS

None.

SEE ALSO

<inttypes.h>, *<limits.h>*, *<sys/socket.h>*, *<sys/types.h>*, *<termios.h>*, *<wctype.h>*, the System Interfaces volume of IEEE Std 1003.1-2001, *access*(), *alarm*(), *chdir*(), *chown*(), *close*(), *crypt*(), *ctermid*(), *dup*(), *encrypt*(), *environ*, *exec*, *exit*(), *fchdir*(), *fchown*(), *fcntl*(), *fork*(), *fpathconf*(), *fsync*(), *ftruncate*(), *getcwd*(), *getegid*(), *geteuid*(), *getgid*(), *getgroups*(), *gethostid*(), *gethostname*(), *getlogin*(), *getpgid*(), *getpgrp*(), *getpid*(), *getppid*(), *getsid*(), *getuid*(), *isatty*(), *lchown*(), *link*(), *lockf*(), *lseek*(), *nice*(), *pathconf*(), *pause*(), *pipe*(), *read*(), *readlink*(), *rmdir*(), *setgid*(), *setpgid*(), *setpgrp*(), *setregid*(), *setreuid*(), *setsid*(), *setuid*(), *sleep*(), *swab*(), *symlink*(), *sync*(), *sysconf*(), *tcgetpgrp*(), *tcsetpgrp*(), *truncate*(), *ttyname*(), *ualarm*(),

unlink(), *usleep*(), *vfork*(), *write*()

COPYRIGHT