NAME

time.h − time types

SYNOPSIS

#include <time.h>

DESCRIPTION

Some of the functionality described on this reference page extends the ISO C standard. Applications shall define the appropriate feature test macro (see the System Interfaces volume of IEEE Std 1003.1-2001, Section 2.2, The Compilation Environment) to enable the visibility of these symbols in this header.

The <*time.h*> header shall declare the structure **tm**, which shall include at least the following members:

int tm_sec Seconds [0,60].
int tm_min Minutes [0,59].
int tm_hour Hour [0,23].
int tm_mday Day of month [1,31].
int tm_mon Month of year [0,11].
int tm_year Years since 1900.
int tm_wday Day of week [0,6] (Sunday =0).
int tm_yday Day of year [0,365].
int tm_isdst Daylight Savings flag.

The value of *tm_isdst* shall be positive if Daylight Savings Time is in effect, 0 if Daylight Savings Time is not in effect, and negative if the information is not available.

The <*time.h*> header shall define the following symbolic names:

NULL Null pointer constant.

CLOCKS_PER_SEC
A number used to convert the value returned by the *clock*() function into seconds.

CLOCK_PROCESS_CPUTIME_ID

The identifier of the CPU-time clock associated with the process making a *clock*() or *timer**() function call.

CLOCK_THREAD_CPUTIME_ID

The identifier of the CPU-time clock associated with the thread making a *clock*() or *timer**() function call.

The <*time.h*> header shall declare the structure **timespec**, which has at least the following members:

time_t tv_sec Seconds.
long tv_nsec Nanoseconds.

The <*time.h*> header shall also declare the **itimerspec** structure, which has at least the following members:

struct timespec it_interval Timer period.
struct timespec it_value Timer expiration.

The following manifest constants shall be defined:

CLOCK_REALTIME
The identifier of the system-wide realtime clock.

TIMER_ABSTIME

Flag indicating time is absolute. For functions taking timer objects, this refers to the clock associated with the timer.

CLOCK_MONOTONIC

The identifier for the system-wide monotonic clock, which is defined as a clock whose value cannot be set via *clock_settime*() and which cannot have backward clock jumps. The maximum possible clock jump shall be implementation-defined.

The **clock_t**, **size_t**, **time_t**, **clockid_t**, and **timer_t** types shall be defined as described in <*sys/types.h*> .

Although the value of CLOCKS_PER_SEC is required to be 1 million on all XSI-conformant systems, it may be variable on other systems, and it should not be assumed that CLOCKS_PER_SEC is a compile-time constant.

The <*time.h*> header shall provide a declaration for *getdate_err*.

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

```
char     *asctime(const struct tm *);

char     *asctime_r(const struct tm *restrict, char *restrict);

clock_t   clock(void);

int       clock_getcpuclockid(pid_t, clockid_t *);


int       clock_getres(clockid_t, struct timespec *);
int       clock_gettime(clockid_t, struct timespec *);


int       clock_nanosleep(clockid_t, int, const struct timespec *,
            struct timespec *);


int       clock_settime(clockid_t, const struct timespec *);

char     *ctime(const time_t *);

char     *ctime_r(const time_t *, char *);

double    difftime(time_t, time_t);

struct tm *getdate(const char *);

struct tm *gmtime(const time_t *);

struct tm *gmtime_r(const time_t *restrict, struct tm *restrict);

struct tm *localtime(const time_t *);

struct tm *localtime_r(const time_t *restrict, struct tm *restrict);
```

```
time_t     mktime(struct tm *);

int        nanosleep(const struct timespec *, struct timespec *);

size_t     strftime(char *restrict, size_t, const char *restrict,
           const struct tm *restrict);

char       *strptime(const char *restrict, const char *restrict,
           struct tm *restrict);

time_t     time(time_t *);

int        timer_create(clockid_t, struct sigevent *restrict,
           timer_t *restrict);
int        timer_delete(timer_t);
int        timer_gettime(timer_t, struct itimerspec *);
int        timer_getoverrun(timer_t);
int        timer_settime(timer_t, int, const struct itimerspec *restrict,
           struct itimerspec *restrict);


void       tzset(void);
```

The following shall be declared as variables:

```
extern int    daylight;
extern long   timezone;


extern char  *tzname[];
```

Inclusion of the <*time.h*> header may make visible all symbols from the <*signal.h*> header.

The following sections are informative.

APPLICATION USAGE

The range [0,60] for *tm_sec* allows for the occasional leap second.

tm_year is a signed value; therefore, years before 1900 may be represented.

To obtain the number of clock ticks per second returned by the *times*() function, applications should call *sysconf*(_SC_CLK_TCK).

RATIONALE

The range [0,60] seconds allows for positive or negative leap seconds. The formal definition of UTC does not permit double leap seconds, so all mention of double leap seconds has been removed, and the range shortened from the former [0,61] seconds seen in previous versions of POSIX.

FUTURE DIRECTIONS

None.

SEE ALSO

<*signal.h*>, <*sys/types.h*>, the System Interfaces volume of IEEE Std 1003.1-2001, *asctime*(), *clock*(), *clock_getcpuclockid*(), *clock_getres*(), *clock_nanosleep*(), *ctime*(), *difftime*(), *getdate*(), *gmtime*(), *localtime*(), *mktime*(), *nanosleep*(), *strftime*(), *strptime*(), *sysconf*(), *time*(), *timer_create*(), *timer_delete*(), *timer_getoverrun*(), *tzname*, *tzset*(), *utime*()

COPYRIGHT