Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'time.3p' command

$ man time.3p

TIME(3P) POSIX Programmer's Manual TIME(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

time ? get time

SYNOPSIS

#include <time.h>

time_t time(time_t *tloc);

DESCRIPTION

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1?2017 defers to the ISO C standard.

The time() function shall return the value of time in seconds since the Epoch.

The tloc argument points to an area where the return value is also stored. If tloc is a null pointer, no value is stored.

RETURN VALUE

Upon successful completion, time() shall return the value of time. Oth? erwise, (time_t)-1 shall be returned.

ERRORS

The time() function may fail if:

EOVERFLOW

The number of seconds since the Epoch will not fit in an object of type time_t.

The following sections are informative.

EXAMPLES

Getting the Current Time

The following example uses the time() function to calculate the time elapsed, in seconds, since the Epoch, localtime() to convert that value to a broken-down time, and asctime() to convert the broken-down time values into a printable string.

```
#include <stdio.h>
#include <time.h>
int main(void)
{
time_t result;
    result = time(NULL);
    printf("%s%ju secs since the Epoch\n",
        asctime(localtime(&result)),
            (uintmax_t)result);
    return(0);
}
```

This example writes the current time to stdout in a form like this:

```
Wed Jun 26 10:32:15 1996
835810335 secs since the Epoch
```

Timing an Event

The following example gets the current time, prints it out in the user's format, and prints the number of minutes to an event being timed.

```
#include <time.h>
#include <stdio.h>
...
```

```
time_t now;
int minutes_to_event;
...
time(&now);
minutes_to_event = ...;
printf("The time is ");
puts(asctime(localtime(&now)));
printf("There are %d minutes to the event.\n",
    minutes_to_event);
...
```

APPLICATION USAGE

None.

RATIONALE

The time() function returns a value in seconds while clock_gettime() and gettimeofday() return a struct timespec (seconds and nanoseconds) and struct timeval (seconds and microseconds), respectively, and are therefore capable of returning more precise times. The times() function is also capable of more precision than time() as it returns a value in clock ticks, although it returns the elapsed time since an arbitrary point such as system boot time, not since the epoch.

Implementations in which time_t is a 32-bit signed integer (many his? torical implementations) fail in the year 2038. POSIX.1?2008 does not address this problem. However, the use of the time_t type is mandated in order to ease the eventual fix.

On some systems the time() function is implemented using a system call that does not return an error condition in addition to the return value. On these systems it is impossible to differentiate between valid and invalid return values and hence overflow conditions cannot be reli? ably detected.

The use of the <time.h> header instead of <sys/types.h> allows compati? bility with the ISO C standard.

Many historical implementations (including Version 7) and the 1984 /usr/group standard use long instead of time_t. This volume of

POSIX.1?2017 uses the latter type in order to agree with the ISO C standard.

FUTURE DIRECTIONS

In a future version of this volume of POSIX.1?2017, time_t is likely to be required to be capable of representing times far in the future. Whether this will be mandated as a 64-bit type or a requirement that a specific date in the future be representable (for example, 10000 AD) is not yet determined. Systems purchased after the approval of this volume of POSIX.1?2017 should be evaluated to determine whether their lifetime will extend past 2038.

SEE ALSO

asctime(), clock(), clock_getres(), ctime(), difftime(), futimens(), gettimeofday(), gmtime(), localtime(), mktime(), strftime(), strp?time(), times(), utime()

The Base Definitions volume of POSIX.1?2017, <time.h>

COPYRIGHT