

NAME

times - get process and waited-for child process times

SYNOPSIS

```
#include <sys/times.h>
```

```
clock_t times(struct tms *buffer);
```

DESCRIPTION

The *times()* function shall fill the **tms** structure pointed to by *buffer* with time-accounting information. The **tms** structure is defined in [<sys/times.h>](#).

All times are measured in terms of the number of clock ticks used.

The times of a terminated child process shall be included in the *tms_cutime* and elements of the parent when [wait\(\)](#), [waitid\(\)](#), or [waitpid\(\)](#) returns the process ID of this terminated child. If a child process has not waited for its children, their times shall not be included in its times.

- The structure member is the CPU time charged for the execution of user instructions of the calling process.
- The structure member is the CPU time charged for execution by the system on behalf of the calling process.
- The structure member is the sum of the *tms_utime* and *tms_cutime* times of the child processes.
- The structure member is the sum of the *tms_stime* and *tms_cstime* times of the child processes.

RETURN VALUE

Upon successful completion, *times()* shall return the elapsed real time, in clock ticks, since an arbitrary point in the past (for example, system start-up time). This point does not change from one invocation of *times()* within the process to another. The return value may overflow the possible range of type **clock_t**. If *times()* fails, (**clock_t**)-1 shall be returned and *errno* set to indicate the error.

ERRORS

The *times()* function shall fail if:

[EOVERFLOW]

The return value would overflow the range of **clock_t**.

The following sections are informative.

EXAMPLES

Timing a Database Lookup

The following example defines two functions, *start_clock()* and *end_clock()*, that are used to time a lookup. It also defines variables of type **clock_t** and **tms** to measure the duration of transactions. The *start_clock()* function saves the beginning times given by the *times()* function. The *end_clock()* function gets the ending times and prints the difference between the two times.

```
#include <sys/times.h>
#include <stdio.h>

...
void start_clock(void);
void end_clock(char *msg);
...
static clock_t st_time;
static clock_t en_time;
static struct tms st_cpu;
static struct tms en_cpu;
...
void
start_clock()
{
    st_time = times(&st_cpu);
}

/* This example assumes that the result of each subtraction
   is within the range of values that can be represented in
   an integer type. */
void
end_clock(char *msg)
{
    en_time = times(&en_cpu);

    fputs(msg, stdout);
    printf("Real Time: %jd, User Time %jd, System Time %jd\n",
        (intmax_t)(en_time - st_time),
        (intmax_t)(en_cpu.tms_utime - st_cpu.tms_utime),
        (intmax_t)(en_cpu.tms_stime - st_cpu.tms_stime));
}
```

APPLICATION USAGE

Applications should use *sysconf(_SC_CLK_TCK)* to determine the number of clock ticks per second as it may vary from system to system.

RATIONALE

The accuracy of the times reported is intentionally left unspecified to allow implementations flexibility in design, from uniprocessor to multi-processor networks.

The inclusion of times of child processes is recursive, so that a parent process may collect the total times of all of its descendants. But the times of a child are only added to those of its parent when its parent successfully waits on the child. Thus, it is not guaranteed that a parent process can always see the total times of all its descendants; see also the discussion of the term ``realtime'' in [alarm](#).

If the type `clock_t` is defined to be a signed 32-bit integer, it overflows in somewhat more than a year if there are 60 clock ticks per second, or less than a year if there are 100. There are individual systems that run continuously for longer than that. This volume of POSIX.1-2017 permits an implementation to make the reference point for the returned value be the start-up time of the process, rather than system start-up time.

The term ``charge'' in this context has nothing to do with billing for services. The operating system accounts for time used in this way. That information must be correct, regardless of how that information is used.

FUTURE DIRECTIONS

None.

SEE ALSO

[alarm](#), [exec](#), [fork](#), [sysconf](#), [time](#), [wait](#), [waitid](#)

XBD [<sys/times.h>](#)

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

Issue 7

POSIX.1-2008, Technical Corrigendum 2, XSH/TC2-2008/0371 [644] is applied.

End of informative text.

[return to top of page](#)

UNIX ® is a registered Trademark of The Open Group.
POSIX ™ is a Trademark of The IEEE.
Copyright © 2001-2018 IEEE and The Open Group, All Rights Reserved
[[Main Index](#) | [XBD](#) | [XSH](#) | [XCU](#) | [XRAT](#)]

[<<< Previous](#)

[Home](#)

[Next >>>](#)
