NAME

times − get process times

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

#include <sys/times.h>

clock_t times(struct tms *****buf***);**

DESCRIPTION

times() stores the current process times in the *struct tms* that *buf* points to. The *struct tms* is as defined in *<sys/times.h>*:

```
struct tms {
    clock_t tms_utime;  /* user time */
    clock_t tms_stime;  /* system time */
    clock_t tms_cutime; /* user time of children */
    clock_t tms_cstime; /* system time of children */
};
```

The *tms_utime* field contains the CPU time spent executing instructions of the calling process. The *tms_stime* field contains the CPU time spent executing inside the kernel while performing tasks on behalf of the calling process.

The *tms_cutime* field contains the sum of the *tms_utime* and *tms_cutime* values for all waited-for terminated children. The *tms_cstime* field contains the sum of the *tms_stime* and *tms_cstime* values for all waited-for terminated children.

Times for terminated children (and their descendants) are added in at the moment **wait**(2) or **waitpid**(2) returns their process ID. In particular, times of grandchildren that the children did not wait for are never seen.

All times reported are in clock ticks.

RETURN VALUE

times() returns the number of clock ticks that have elapsed since an arbitrary point in the past. The return value may overflow the possible range of type *clock_t*. On error, *(clock_t) −1* is returned, and *errno* is set to indicate the error.

ERRORS

EFAULT

tms points outside the process's address space.

STANDARDS

POSIX.1-2001, POSIX.1-2008, SVr4, 4.3BSD.

NOTES

The number of clock ticks per second can be obtained using:

```
sysconf(_SC_CLK_TCK);
```

In POSIX.1-1996 the symbol **CLK_TCK** (defined in *<time.h>*) is mentioned as obsolescent. It is obsolete now.

Before Linux 2.6.9, if the disposition of **SIGCHLD** is set to **SIG_IGN**, then the times of terminated children are automatically included in the *tms_cstime* and *tms_cutime* fields, although POSIX.1-2001 says that this should happen only if the calling process **wait**(2)s on its children. This nonconformance is rectified in Linux 2.6.9 and later.

On Linux, the *buf* argument can be specified as NULL, with the result that **times**() just returns a function result. However, POSIX does not specify this behavior, and most other UNIX implementations require a non-NULL value for *buf*.

Note that **clock**(3) also returns a value of type *clock_t*, but this value is measured in units of

CLOCKS_PER_SEC, not the clock ticks used by **times**().

On Linux, the "arbitrary point in the past" from which the return value of **times**() is measured has varied across kernel versions. On Linux 2.4 and earlier, this point is the moment the system was booted. Since Linux 2.6, this point is *(2^32/HZ) − 300* seconds before system boot time. This variability across kernel versions (and across UNIX implementations), combined with the fact that the returned value may overflow the range of *clock_t*, means that a portable application would be wise to avoid using this value. To measure changes in elapsed time, use **clock_gettime**(2) instead.

Historical

SVr1-3 returns *long* and the struct members are of type *time_t* although they store clock ticks, not seconds since the Epoch. V7 used *long* for the struct members, because it had no type *time_t* yet.

BUGS

A limitation of the Linux system call conventions on some architectures (notably i386) means that on Linux 2.6 there is a small time window (41 seconds) soon after boot when **times**() can return −1, falsely indicating that an error occurred. The same problem can occur when the return value wraps past the maximum value that can be stored in **clock_t**.

SEE ALSO

time(1), **getrusage**(2), **wait**(2), **clock**(3), **sysconf**(3), **time**(7)