



Rocky Enterprise Linux 9.2 Manual Pages on command 'pthread_getcpuclockid.3'

C:\>man pthread_getcpuclockid.3

PTHREAD_GETCPULOCKID(3) Linux Programmer's Manual PTHREAD_GETCPULOCKID(3)

NAME

pthread_getcpuclockid - retrieve ID of a thread's CPU time clock

SYNOPSIS

```
#include <pthread.h>
```

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

```
int pthread_getcpuclockid(pthread_t thread, clockid_t *clock_id);
```

Compile and link with -pthread.

DESCRIPTION

The pthread_getcpuclockid() function returns the clock ID for the CPU time clock of the thread thread.

RETURN VALUE

On success, this function returns 0; on error, it returns a nonzero error number.

ERRORS

ENOENT Per-thread CPU time clocks are not supported by the system.

ESRCH No thread with the ID thread could be found.

VERSIONS

This function is available in glibc since version 2.2.

ATTRIBUTES

For an explanation of the terms used in this section, see attributes(7).

??

?Interface ? Attribute ? Value ?

??

?pthread_getcpuclockid() ? Thread safety ? MT-Safe ?

??

CONFORMING TO

POSIX.1-2001, POSIX.1-2008.

NOTES

When thread refers to the calling thread, this function returns an identifier that refers to the same clock manipulated by clock_gettime(2) and clock_settime(2) when given the clock ID CLOCK_THREAD_CPUTIME_ID.

EXAMPLE

The program below creates a thread and then uses clock_gettime(2) to retrieve the total process CPU time, and the per-thread CPU time consumed by the two threads.

The following shell session shows an example run:

```
$ ./a.out
Main thread sleeping
Subthread starting infinite loop
Main thread consuming some CPU time...
Process total CPU time:  1.368
Main thread CPU time:   0.376
Subthread CPU time:     0.992
```

Program source

```
/* Link with "-lrt" */
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <string.h>
#include <errno.h>

#define handle_error(msg) \
    do { perror(msg); exit(EXIT_FAILURE); } while (0)

#define handle_error_en(en, msg) \
    do { errno = en; perror(msg); exit(EXIT_FAILURE); } while (0)
```

```

static void *
thread_start(void *arg)
{
    printf("Subthread starting infinite loop\n");
    for (;;)
        continue;
}

static void
pclock(char *msg, clockid_t cid)
{
    struct timespec ts;
    printf("%s", msg);
    if (clock_gettime(cid, &ts) == -1)
        handle_error("clock_gettime");
    printf("%4ld.%03ld\n", ts.tv_sec, ts.tv_nsec / 1000000);
}

int
main(int argc, char *argv[])
{
    pthread_t thread;
    clockid_t cid;
    int j, s;
    s = pthread_create(&thread, NULL, thread_start, NULL);
    if (s != 0)
        handle_error_en(s, "pthread_create");
    printf("Main thread sleeping\n");
    sleep(1);
    printf("Main thread consuming some CPU time...\n");
    for (j = 0; j < 2000000; j++)
        getppid();
    pclock("Process total CPU time: ", CLOCK_PROCESS_CPUTIME_ID);
    s = pthread_getcpuclockid(pthread_self(), &cid);
    if (s != 0)

```

```
    handle_error_en(s, "pthread_getcpuclockid");
pclock("Main thread CPU time: ", cid);
/* The preceding 4 lines of code could have been replaced by:
    pclock("Main thread CPU time: ", CLOCK_THREAD_CPUTIME_ID); */
s = pthread_getcpuclockid(thread, &cid);
if (s != 0)
    handle_error_en(s, "pthread_getcpuclockid");
pclock("Subthread CPU time: 1  ", cid);
exit(EXIT_SUCCESS);    /* Terminates both threads */
}
```

SEE ALSO

clock_gettime(2), clock_settime(2), timer_create(2), clock_getcpuclockid(3),
pthread_self(3), pthreads(7), time(7)

COLOPHON

This page is part of release 5.05 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at <https://www.kernel.org/doc/man-pages/>.