NAME

set_tid_address − set pointer to thread ID

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

#include <sys/syscall.h> /* Definition of **SYS_*** constants */
#include <unistd.h>

pid_t syscall(SYS_set_tid_address, int ***tidptr*****);**

Note: glibc provides no wrapper for **set_tid_address**(), necessitating the use of **syscall**(2).

DESCRIPTION

For each thread, the kernel maintains two attributes (addresses) called *set_child_tid* and *clear_child_tid*. These two attributes contain the value NULL by default.

set_child_tid

If a thread is started using **clone**(2) with the **CLONE_CHILD_SETTID** flag, *set_child_tid* is set to the value passed in the *ctid* argument of that system call.

When *set_child_tid* is set, the very first thing the new thread does is to write its thread ID at this address.

clear_child_tid

If a thread is started using **clone**(2) with the **CLONE_CHILD_CLEARTID** flag, *clear_child_tid* is set to the value passed in the *ctid* argument of that system call.

The system call **set_tid_address**() sets the *clear_child_tid* value for the calling thread to *tidptr*.

When a thread whose *clear_child_tid* is not NULL terminates, then, if the thread is sharing memory with other threads, then 0 is written at the address specified in *clear_child_tid* and the kernel performs the following operation:

```
futex(clear_child_tid, FUTEX_WAKE, 1, NULL, NULL, 0);
```

The effect of this operation is to wake a single thread that is performing a futex wait on the memory location. Errors from the futex wake operation are ignored.

RETURN VALUE

set_tid_address() always returns the caller’s thread ID.

ERRORS

set_tid_address() always succeeds.

VERSIONS

This call is present since Linux 2.5.48. Details as given here are valid since Linux 2.5.49.

STANDARDS

This system call is Linux-specific.

SEE ALSO

clone(2), **futex**(2), **gettid**(2)