NAME

sched_yield − yield the processor

SYNOPSIS

#include <sched.h>

int sched_yield(void);

DESCRIPTION

sched_yield() causes the calling thread to relinquish the CPU. The thread is moved to the end of the queue for its static priority and a new thread gets to run.

RETURN VALUE

On success, **sched_yield**() returns 0. On error, −1 is returned, and *errno* is set appropriately.

ERRORS

In the Linux implementation, **sched_yield**() always succeeds.

CONFORMING TO

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

NOTES

If the calling thread is the only thread in the highest priority list at that time, it will continue to run after a call to **sched_yield**().

POSIX systems on which **sched_yield**() is available define **_POSIX_PRIORITY_SCHEDULING** in *<unistd.h>*.

Strategic calls to **sched_yield**() can improve performance by giving other threads or processes a chance to run when (heavily) contended resources (e.g., mutexes) have been released by the caller. Avoid calling **sched_yield**() unnecessarily or inappropriately (e.g., when resources needed by other schedulable threads are still held by the caller), since doing so will result in unnecessary context switches, which will degrade system performance.

sched_yield() is intended for use with real-time scheduling policies (i.e., **SCHED_FIFO** or **SCHED_RR**). Use of **sched_yield**() with nondeterministic scheduling policies such as **SCHED_OTHER** is unspecified and very likely means your application design is broken.

SEE ALSO

sched(7)

COLOPHON

This page is part of release 5.02 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/.