NAME

sched_setscheduler, sched_getscheduler – set/get scheduling policy and scheduling parameters

SYNOPSIS

cc [*flag* ...] *file* ... **–lposix4** [*library* ...]

#include <sched.h>

int sched_setscheduler(pid_t *pid***, int** *policy***, const struct sched_param** **param***);**

int sched_getscheduler(pid_t *pid***);**

```
struct sched_param {
        int          sched_priority;    /* process execution scheduling priority */
        ...
}
```

MT-LEVEL

MT-Safe

DESCRIPTION

sched_setscheduler( ) sets the scheduling policy and scheduling parameters of the process specified by *pid* to *policy* and the parameters specified in the **sched_param** structure pointed to by *param*, respectively. The value of *param*->**sched_priority** must be any integer within the inclusive priority range for the scheduling policy specified by *policy.*

The possible values for the *policy* parameter are defined in the header file **<sched.h>**: **SCHED_FIFO, SCHED_RR,** or **SCHED_OTHER.**

If *pid* is zero, the scheduling policy and scheduling parameters are set for the calling process. Otherwise, if a process specified by *pid* exists and if the calling process has permission, the scheduling policy and scheduling parameters are set for the process whose process ID is equal to *pid.* The real or effective user ID of the calling process must match the real or saved (from **exec**(2)) user ID of the target process unless the effective user ID of the calling process is super-user. See **intro**(2).

To change the *policy* of any process to either of the real time policies **SCHED_FIFO** or **SCHED_RR,** the calling process must either have the **SCHED_FIFO,** or **SCHED_RR** policy or have an effective user ID of **0**.

sched_getscheduler( ) returns the scheduling policy of the process specified by *pid.* If *pid* is zero, the scheduling policy is returned for the calling process. Otherwise, if a process specified by *pid* exists and if the calling process has permission, the scheduling policy is returned for the process whose process ID is equal to *pid.*

RETURN VALUES

If successful, **sched_setscheduler( )** returns the former scheduling policy of the specified process *(pid)*, which will be one of the following values:

SCHED_FIFO (realtime),
First-In-First-Out; processes scheduled to this policy, if not pre-empted by a higher priority or interrupted by a signal, will proceed until completion.

SCHED_RR (realtime),
Round-Robin; processes scheduled to this policy, if not pre-empted by a higher priority or interrupted by a signal, will execute for a time period, returned by **sched_rr_get_interval**(3R) or by the system.

or

SCHED_OTHER (time-sharing).

Otherwise, the policy and sheduling parameters remain unchanged, **sched_setscheduler( )** returns **-1**, and sets **errno** to indicate the error condition.

If successful, **sched_getscheduler( )** returns the scheduling policy of the specified processr; otherwise, it returns **-1**, and sets **errno** to indicate the error condition.

ERRORS

EINVAL The value of *policy* is invalid, or one or more of the parameters contained in **param** is outside the valid range for the specified scheduling policy.

ENOSYS **sched_setscheduler( )** and **sched_getscheduler( )** are not supported by this implementation.

EPERM **sched_setscheduler( )** does not have permission to set either or both of the scheduling parameters or the scheduling policy of the specified process.

sched_getscheduler( ) does not have permission to determine the scheduling policy of the specified process.

ESRCH No process can be found corresponding to that specified by *pid.*

SEE ALSO

priocntl(1), **intro**(2), **exec**(2), **priocntl**(2), **sched_get_priority_max**(3R), **sched_setparam**(3R)

BUGS

In Solaris 2.5, these functions always return –**1** and set **errno** to **ENOSYS,** because this release does not support the Priority Scheduling option. It is our intention to provide support for these interfaces in future releases.