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

C:\>man pthread_setschedparam.3

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

NAME

pthread_setschedparam, pthread_getschedparam - set/get scheduling policy and param?

eters of a thread

SYNOPSIS

```
#include <pthread.h>
int pthread_setschedparam(pthread_t thread, int policy,
                          const struct sched_param *param);
int pthread_getschedparam(pthread_t thread, int *policy,
                          struct sched_param *param);
```

Compile and link with -pthread.

DESCRIPTION

The pthread_setschedparam() function sets the scheduling policy and parameters of the thread thread.

policy specifies the new scheduling policy for thread. The supported values for policy, and their semantics, are described in sched(7).

The structure pointed to by param specifies the new scheduling parameters for thread. Scheduling parameters are maintained in the following structure:

```
struct sched_param {
    int sched_priority;     /* Scheduling priority */
};
```

As can be seen, only one scheduling parameter is supported. For details of the

permitted ranges for scheduling priorities in each scheduling policy, see sched(7).

The pthread_getschedparam() function returns the scheduling policy and parameters of the thread thread, in the buffers pointed to by policy and param, respectively.

The returned priority value is that set by the most recent pthread_setschedparam(), pthread_setschedprio(3), or pthread_create(3) call that affected thread. The re? turned priority does not reflect any temporary priority adjustments as a result of calls to any priority inheritance or priority ceiling functions (see, for example, pthread_mutexattr_setprioceiling(3) and pthread_mutexattr_setprotocol(3)).

RETURN VALUE

On success, these functions return 0; on error, they return a nonzero error number.

If pthread_setschedparam() fails, the scheduling policy and parameters of thread are not changed.

ERRORS

Both of these functions can fail with the following error:

ESRCH No thread with the ID thread could be found.

pthread_setschedparam() may additionally fail with the following errors:

EINVAL policy is not a recognized policy, or param does not make sense for the pol? icy.

EPERM The caller does not have appropriate privileges to set the specified sched? uling policy and parameters.

POSIX.1 also documents an ENOTSUP ("attempt was made to set the policy or schedul? ing parameters to an unsupported value") error for pthread_setschedparam().

ATTRIBUTES

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

Interface	Attribute	Value
pthread_setschedparam(), pthread_getschedparam()	Thread safety	MT-Safe

CONFORMING TO

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

NOTES

For a description of the permissions required to, and the effect of, changing a thread's scheduling policy and priority, and details of the permitted ranges for priorities in each scheduling policy, see sched(7).

EXAMPLE

The program below demonstrates the use of pthread_setschedparam() and pthread_getschedparam(), as well as the use of a number of other scheduling-related pthreads functions.

In the following run, the main thread sets its scheduling policy to SCHED_FIFO with a priority of 10, and initializes a thread attributes object with a scheduling pol? icy attribute of SCHED_RR and a scheduling priority attribute of 20. The program then sets (using pthread_attr_setinheritsched(3)) the inherit scheduler attribute of the thread attributes object to PTHREAD_EXPLICIT_SCHED, meaning that threads created using this attributes object should take their scheduling attributes from the thread attributes object. The program then creates a thread using the thread attributes object, and that thread displays its scheduling policy and priority.

```
$ su      # Need privilege to set real-time scheduling policies
Password:
# ./a.out -mf10 -ar20 -i e
Scheduler settings of main thread
    policy=SCHED_FIFO, priority=10
Scheduler settings in 'attr'
    policy=SCHED_RR, priority=20
    inheritsched is EXPLICIT
Scheduler attributes of new thread
    policy=SCHED_RR, priority=20
```

In the above output, one can see that the scheduling policy and priority were taken from the values specified in the thread attributes object.

The next run is the same as the previous, except that the inherit scheduler attri? bute is set to PTHREAD_INHERIT_SCHED, meaning that threads created using the thread attributes object should ignore the scheduling attributes specified in the at? tributes object and instead take their scheduling attributes from the creating thread.

```
# ./a.out -mf10 -ar20 -i i
```

```
        Scheduler settings of main thread
            policy=SCHED_FIFO, priority=10
        Scheduler settings in 'attr'
            policy=SCHED_RR, priority=20
            inheritsched is INHERIT
        Scheduler attributes of new thread
            policy=SCHED_FIFO, priority=10
```

In the above output, one can see that the scheduling policy and priority were taken from the creating thread, rather than the thread attributes object.

Note that if we had omitted the -i i option, the output would have been the same, since PTHREAD_INHERIT_SCHED is the default for the inherit scheduler attribute.

Program source

```
/* pthreads_sched_test.c */
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#define handle_error_en(en, msg) \
        do { errno = en; perror(msg); exit(EXIT_FAILURE); } while (0)
static void
usage(char *prog_name, char *msg)
{
    if (msg != NULL)
        fputs(msg, stderr);
    fprintf(stderr, "Usage: %s [options]\n", prog_name);
    fprintf(stderr, "Options are:\n");
#define fpe(msg) fprintf(stderr, "\t%s", msg);          /* Shorter */
    fpe("-a<policy><prio> Set scheduling policy and priority in\n");
    fpe("                 thread attributes object\n");
    fpe("                 <policy> can be\n");
    fpe("                     f  SCHED_FIFO\n");
    fpe("                     r  SCHED_RR\n");
```

```
    fpe("                   o  SCHED_OTHER\n");
    fpe("-A                Use default thread attributes object\n");
    fpe("-i {e|i}          Set inherit scheduler attribute to\n");
    fpe("                 'explicit' or 'inherit'\n");
    fpe("-m<policy><prio> Set scheduling policy and priority on\n");
    fpe("                 main thread before pthread_create() call\n");
    exit(EXIT_FAILURE);
}
static int
get_policy(char p, int *policy)
{
    switch (p) {
    case 'f': *policy = SCHED_FIFO;     return 1;
    case 'r': *policy = SCHED_RR;       return 1;
    case 'o': *policy = SCHED_OTHER;    return 1;
    default:  return 0;
    }
}
static void
display_sched_attr(int policy, struct sched_param *param)
{
    printf("    policy=%s, priority=%d\n",
            (policy == SCHED_FIFO)  ? "SCHED_FIFO" :
            (policy == SCHED_RR)    ? "SCHED_RR" :
            (policy == SCHED_OTHER) ? "SCHED_OTHER" :
            "???",
            param->sched_priority);
}
static void
display_thread_sched_attr(char *msg)
{
    int policy, s;
    struct sched_param param;
```

```
    s = pthread_getschedparam(pthread_self(), &policy, &param);
    if (s != 0)
        handle_error_en(s, "pthread_getschedparam");
    printf("%s\n", msg);
    display_sched_attr(policy, &param);
}
static void *
thread_start(void *arg)
{
    display_thread_sched_attr("Scheduler attributes of new thread");
    return NULL;
}
int
main(int argc, char *argv[])
{
    int s, opt, inheritsched, use_null_attrib, policy;
    pthread_t thread;
    pthread_attr_t attr;
    pthread_attr_t *attrp;
    char *attr_sched_str, *main_sched_str, *inheritsched_str;
    struct sched_param param;
    /* Process command-line options */
    use_null_attrib = 0;
    attr_sched_str = NULL;
    main_sched_str = NULL;
    inheritsched_str = NULL;
    while ((opt = getopt(argc, argv, "a:Ai:m:")) != -1) {
        switch (opt) {
        case 'a': attr_sched_str = optarg;      break;
        case 'A': use_null_attrib = 1;          break;
        case 'i': inheritsched_str = optarg;    break;
        case 'm': main_sched_str = optarg;      break;
        default:  usage(argv[0], "Unrecognized option\n");
```

```
    }
}
if (use_null_attrib &&
        (inheritsched_str != NULL || attr_sched_str != NULL))
    usage(argv[0], "Can't specify -A with -i or -a\n");
/* Optionally set scheduling attributes of main thread,
   and display the attributes */
if (main_sched_str != NULL) {
    if (!get_policy(main_sched_str[0], &policy))
        usage(argv[0], "Bad policy for main thread (-m)\n");
    param.sched_priority = strtol(&main_sched_str[1], NULL, 0);
    s = pthread_setschedparam(pthread_self(), policy, &param);
    if (s != 0)
        handle_error_en(s, "pthread_setschedparam");
}
display_thread_sched_attr("Scheduler settings of main thread");
printf("\n");
/* Initialize thread attributes object according to options */
attrp = NULL;
if (!use_null_attrib) {
    s = pthread_attr_init(&attr);
    if (s != 0)
        handle_error_en(s, "pthread_attr_init");
    attrp = &attr;
}
if (inheritsched_str != NULL) {
    if (inheritsched_str[0] == 'e')
        inheritsched = PTHREAD_EXPLICIT_SCHED;
    else if (inheritsched_str[0] == 'i')
        inheritsched = PTHREAD_INHERIT_SCHED;
    else
        usage(argv[0], "Value for -i must be 'e' or 'i'\n");
    s = pthread_attr_setinheritsched(&attr, inheritsched);
```

```
    if (s != 0)
        handle_error_en(s, "pthread_attr_setinheritsched");
}
if (attr_sched_str != NULL) {
    if (!get_policy(attr_sched_str[0], &policy))
        usage(argv[0],
              "Bad policy for 'attr' (-a)\n");
    param.sched_priority = strtol(&attr_sched_str[1], NULL, 0);
    s = pthread_attr_setschedpolicy(&attr, policy);
    if (s != 0)
        handle_error_en(s, "pthread_attr_setschedpolicy");
    s = pthread_attr_setschedparam(&attr, &param);
    if (s != 0)
        handle_error_en(s, "pthread_attr_setschedparam");
}
/* If we initialized a thread attributes object, display
   the scheduling attributes that were set in the object */
if (attrp != NULL) {
    s = pthread_attr_getschedparam(&attr, &param);
    if (s != 0)
        handle_error_en(s, "pthread_attr_getschedparam");
    s = pthread_attr_getschedpolicy(&attr, &policy);
    if (s != 0)
        handle_error_en(s, "pthread_attr_getschedpolicy");
    printf("Scheduler settings in 'attr'\n");
    display_sched_attr(policy, &param);
    s = pthread_attr_getinheritsched(&attr, &inheritsched);
    printf("    inheritsched is %s\n",
           (inheritsched == PTHREAD_INHERIT_SCHED)  ? "INHERIT" :
           (inheritsched == PTHREAD_EXPLICIT_SCHED) ? "EXPLICIT" :
           "???");
    printf("\n");
}
```

```
        /* Create a thread that will display its scheduling attributes */

        s = pthread_create(&thread, attrp, &thread_start, NULL);

        if (s != 0)

            handle_error_en(s, "pthread_create");

        /* Destroy unneeded thread attributes object */

        if (!use_null_attrib) {

          s = pthread_attr_destroy(&attr);

          if (s != 0)

              handle_error_en(s, "pthread_attr_destroy");

        }

        s = pthread_join(thread, NULL);

        if (s != 0)

            handle_error_en(s, "pthread_join");

        exit(EXIT_SUCCESS);

    }
```

SEE ALSO

getrlimit(2), sched_get_priority_min(2), pthread_attr_init(3), pthread_attr_setinheritsched(3), pthread_attr_setschedparam(3), pthread_attr_setschedpolicy(3), pthread_create(3), pthread_self(3), pthread_setschedprio(3), pthreads(7), sched(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/.