NAME

thr_new - create new thread of execution

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/thr.h>

int
thr_new(*struct thr_param *param, int param_size*);

DESCRIPTION

This function is intended for implementing threading. Normal applications should call pthread_create(3) instead.

The **thr_new**() system call creates a new kernel-scheduled thread of execution in the context of the current process. The newly created thread shares all attributes of the process with the existing kernel-scheduled threads in the process, but has private processor execution state. The machine context for the new thread is copied from the creating thread's context, including coprocessor state. FPU state and specific machine registers are excluded from the copy. These are set according to ABI requirements and syscall parameters. The FPU state for the new thread is reinitialized to clean.

The *param* structure supplies parameters affecting the thread creation. The structure is defined in the <*sys/thr.h*> header as follows

```
struct thr_param {
    void        (*start_func)(void *);
    void        *arg;
    char        *stack_base;
    size_t      stack_size;
    char        *tls_base;
    size_t      tls_size;
    long        *child_tid;
    long        *parent_tid;
    int         flags;
    struct rtprio *rtp;
};
```

and contains the following fields:

start_func Pointer to the thread entry function. The kernel arranges for the new thread to start executing the function upon the first return to userspace.

arg Opaque argument supplied to the entry function.

stack_base
Stack base address. The stack must be allocated by the caller. On some architectures, the ABI might require that the system put information on the stack to ensure the execution environment for *start_func*.

stack_size
Stack size.

tls_base TLS base address. The value of TLS base is loaded into the ABI-defined machine register in the new thread context.

tls_size TLS size.

child_tid Address to store the new thread identifier, for the child's use.

parent_tid Address to store the new thread identifier, for the parent's use.

Both *child_tid* and *parent_tid* are provided, with the intent that *child_tid* is used by the new thread to get its thread identifier without issuing the thr_self(2) syscall, while *parent_tid* is used by the thread creator. The latter is separate from *child_tid* because the new thread might exit and free its thread data before the parent has a chance to execute far enough to access it.

flags Thread creation flags. The *flags* member may specify the following flags:

THR_SUSPENDED Create the new thread in the suspended state. The flag is not currently implemented.

THR_SYSTEM_SCOPE Create the system scope thread. The flag is not currently implemented.

rtp Real-time scheduling priority for the new thread. May be NULL to inherit the priority from the creating thread.

The *param_size* argument should be set to the size of the *param* structure.

After the first successful creation of an additional thread, the process is marked by the kernel as multi-threaded. In particular, the P_HADTHREADS flag is set in the process' p_flag (visible in the ps(1) output), and several operations are executed in multi-threaded mode. For instance, the execve(2) system call terminates all threads but the calling one on successful execution.

RETURN VALUES

If successful, **thr_new**() will return zero, otherwise -1 is returned, and *errno* is set to indicate the error.

ERRORS

The **thr_new**() operation returns the following errors:

[EFAULT]	The memory pointed to by the *param* argument is not valid.
[EFAULT]	The memory pointed to by the *param* structure *child_tid*, *parent_tid* or *rtp* arguments is not valid.
[EFAULT]	The specified stack base is invalid, or the kernel was unable to put required initial data on the stack.
[EINVAL]	The *param_size* argument specifies a negative value, or the value is greater than the largest *struct param* size the kernel can interpret.
[EINVAL]	The *rtp* member is not NULL and specifies invalid scheduling parameters.
[EINVAL]	The specified TLS base is invalid.
[EPERM]	The caller does not have permission to set the scheduling parameters or scheduling policy.
[EPROCLIM]	Creation of the new thread would exceed the RACCT_NTHR limit, see rctl_get_racct(2).
[EPROCLIM]	Creation of the new thread would exceed the kern.threads.max_threads_per_proc sysctl(3) limit.
[ENOMEM]	There was not enough kernel memory to allocate the new thread structures.

SEE ALSO

ps(1), _umtx_op(2), execve(2), rctl_get_racct(2), thr_exit(2), thr_kill(2), thr_kill2(2), thr_self(2), thr_set_name(2), pthread_create(3)

STANDARDS

The **thr_new**() system call is non-standard and is used by the 1:1 Threading Library (libthr, -lthr) to implement IEEE Std 1003.1-2001 ("POSIX.1") pthread(3) functionality.

HISTORY

The **thr_new**() system call first appeared in FreeBSD 5.2.