NAME

kthread_start, **kthread_shutdown**, **kthread_add**, **kthread_exit**, **kthread_resume**, **kthread_suspend**, **kthread_suspend_check** - kernel threads

SYNOPSIS

#include <sys/kthread.h>

void
kthread_start(*const void *udata*);

void
kthread_shutdown(*void *arg*, *int howto*);

void
kthread_exit(*void*);

int
kthread_resume(*struct thread *td*);

int
kthread_suspend(*struct thread *td*, *int timo*);

void
kthread_suspend_check(*void*);

#include <sys/unistd.h>

int
kthread_add(*void (*func)(void *), void *arg*, *struct proc *procp*, *struct thread **newtdpp*, *int flags*, *int pages*, *const char *fmt*, ...);

int
kproc_kthread_add(*void (*func)(void *), void *arg*, *struct proc **procptr*, *struct thread **tdptr*, *int flags*, *int pages*, *char * procname*, *const char *fmt*, ...);

DESCRIPTION

In FreeBSD 8.0, the older family of **kthread_***(*9*) functions was renamed to be the **kproc_*(*9*) family of functions, as they were previously misnamed and actually produced kernel processes. This new family of **kthread_*(*9*) functions was added to produce *real* kernel threads. See the kproc(9) man page for more information on the renamed calls. Also note that the **kproc_kthread_add**(*9*) function appears in both

pages as its functionality is split.

The function **kthread_start**() is used to start "internal" daemons such as **bufdaemon**, **pagedaemon**, **vmdaemon**, and the **syncer** and is intended to be called from SYSINIT(9). The *udata* argument is actually a pointer to a *struct kthread_desc* which describes the kernel thread that should be created:

```
struct kthread_desc {
        char            *arg0;
        void            (*func)(void);
        struct thread   **global_threadpp;
};
```

The structure members are used by **kthread_start**() as follows:

arg0 String to be used for the name of the thread. This string will be copied into the *td_name* member of the new threads' *struct thread*.

func The main function for this kernel thread to run.

global_threadpp A pointer to a *struct thread* pointer that should be updated to point to the newly created thread's *thread* structure. If this variable is NULL, then it is ignored. The thread will be a subthread of *proc0* (PID 0).

The **kthread_add**() function is used to create a kernel thread. The new thread runs in kernel mode only. It is added to the process specified by the *procp* argument, or if that is NULL, to *proc0*. The *func* argument specifies the function that the thread should execute. The *arg* argument is an arbitrary pointer that is passed in as the only argument to *func* when it is called by the new thread. The *newtdpp* pointer points to a *struct thread* pointer that is to be updated to point to the newly created thread. If this argument is NULL, then it is ignored. The *flags* argument may be set to RFSTOPPED to leave the thread in a stopped state. The caller must call **sched_add**() to start the thread. The *pages* argument specifies the size of the new kernel thread's stack in pages. If 0 is used, the default kernel stack size is allocated. The rest of the arguments form a printf(9) argument list that is used to build the name of the new thread and is stored in the *td_name* member of the new thread's *struct thread*.

The **kproc_kthread_add**() function is much like the **kthread_add**() function above except that if the kproc does not already exist, it is created. This function is better documented in the kproc(9) manual page.

The **kthread_exit**() function is used to terminate kernel threads. It should be called by the main function of the kernel thread rather than letting the main function return to its caller.

The **kthread_resume**(), **kthread_suspend**(), and **kthread_suspend_check**() functions are used to suspend and resume a kernel thread. During the main loop of its execution, a kernel thread that wishes to allow itself to be suspended should call **kthread_suspend_check**() in order to check if the it has been asked to suspend. If it has, it will msleep(9) until it is told to resume. Once it has been told to resume it will return allowing execution of the kernel thread to continue. The other two functions are used to notify a kernel thread of a suspend or resume request. The *td* argument points to the *struct thread* of the kernel thread to suspend or resume. For **kthread_suspend**(), the *timo* argument specifies a timeout to wait for the kernel thread to acknowledge the suspend request and suspend itself.

The **kthread_shutdown**() function is meant to be registered as a shutdown event for kernel threads that need to be suspended voluntarily during system shutdown so as not to interfere with system shutdown activities. The actual suspension of the kernel thread is done with **kthread_suspend**().

RETURN VALUES

The **kthread_add**(), **kthread_resume**(), and **kthread_suspend**() functions return zero on success and non-zero on failure.

EXAMPLES

This example demonstrates the use of a *struct kthread_desc* and the functions **kthread_start**(), **kthread_shutdown**(), and **kthread_suspend_check**() to run the **bufdaemon** process.

```
static struct thread *bufdaemonthread;

static struct kthread_desc buf_kp = {
        "bufdaemon",
        buf_daemon,
        &bufdaemonthread
};
SYSINIT(bufdaemon, SI_SUB_KTHREAD_BUF, SI_ORDER_FIRST, kthread_start,
   &buf_kp)

static void
buf_daemon()
{
        ...
        /*
         * This process needs to be suspended prior to shutdown sync.
         */
        EVENTHANDLER_REGISTER(shutdown_pre_sync, kthread_shutdown,
           bufdaemonthread, SHUTDOWN_PRI_LAST);
```

```
        ...
        for (;;) {
                kthread_suspend_check();
                ...
        }
}
```

ERRORS

The **kthread_resume**() and **kthread_suspend**() functions will fail if:

[EINVAL] The *td* argument does not reference a kernel thread.

The **kthread_add**() function will fail if:

[ENOMEM] Memory for a thread's stack could not be allocated.

SEE ALSO

kproc(9), SYSINIT(9), wakeup(9)

HISTORY

The **kthread_start**() function first appeared in FreeBSD 2.2 where it created a whole process. It was converted to create threads in FreeBSD 8.0. The **kthread_shutdown**(), **kthread_exit**(), **kthread_resume**(), **kthread_suspend**(), and **kthread_suspend_check**() functions were introduced in FreeBSD 4.0 and were converted to threads in FreeBSD 8.0. The **kthread_create**() call was renamed to **kthread_add**() in FreeBSD 8.0. The old functionality of creating a kernel process was renamed to kproc_create(9). Prior to FreeBSD 5.0, the **kthread_shutdown**(), **kthread_resume**(), **kthread_suspend**(), and **kthread_suspend_check**() functions were named **shutdown_kproc**(), **resume_kproc**(), **shutdown_kproc**(), and **kproc_suspend_loop**(), respectively.