NAME

kproc_start, **kproc_shutdown**, **kproc_create**, **kproc_exit**, **kproc_resume**, **kproc_suspend**, **kproc_suspend_check** - kernel processes

SYNOPSIS

#include <sys/kthread.h>

void
kproc_start(*const void *udata*);

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

int
kproc_create(*void (*func)(void *)*, *void *arg*, *struct proc **newpp*, *int flags*, *int pages*, *const char *fmt*, *...*);

void
kproc_exit(*int ecode*);

int
kproc_resume(*struct proc *p*);

int
kproc_suspend(*struct proc *p*, *int timo*);

void
kproc_suspend_check(*struct proc *p*);

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 **kthread***(*9*) family of functions was renamed to be the **kproc***(*9*) family of functions, as they were misnamed and actually produced kernel processes. A new family of *different* **kthread_***(*9*) functions was added to produce *real* kernel *threads*. See the kthread(9) man page for more information on those calls. Also note that the **kproc_kthread_add**(*9*) function appears in both pages as its functionality is split.

The function **kproc_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 kproc_desc* which describes the kernel process that should be created:

```
struct kproc_desc {
        char            *arg0;
        void            (*func)(void);
        struct proc     **global_procpp;
};
```

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

arg0	String to be used for the name of the process. This string will be copied into the *p_comm* member of the new process’ *struct proc*.
func	The main function for this kernel process to run.
global_procpp	A pointer to a *struct proc* pointer that should be updated to point to the newly created process’ process structure. If this variable is NULL, then it is ignored.

The **kproc_create**() function is used to create a kernel process. The new process shares its address space with process 0, the **swapper** process, and runs in kernel mode only. The *func* argument specifies the function that the process 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 process. The *newpp* pointer points to a *struct proc* pointer that is to be updated to point to the newly created process. If this argument is NULL, then it is ignored. The *flags* argument specifies a set of flags as described in rfork(2). The *pages* argument specifies the size of the new kernel process’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 process and is stored in the *p_comm* member of the new process’s *struct proc*.

The **kproc_exit**() function is used to terminate kernel processes. It should be called by the main function of the kernel process rather than letting the main function return to its caller. The *ecode* argument specifies the exit status of the process. While exiting, the function exit1(9) will initiate a call to wakeup(9) on the process handle.

The **kproc_resume**(), **kproc_suspend**(), and **kproc_suspend_check**() functions are used to suspend and resume a kernel process. During the main loop of its execution, a kernel process that wishes to allow itself to be suspended should call **kproc_suspend_check**() passing in *curproc* as the only argument. This function checks to see if the kernel process has been asked to suspend. If it has, it will tsleep(9) until it is told to resume. Once it has been told to resume it will return allowing execution of the kernel process to continue. The other two functions are used to notify a kernel process of a suspend or resume request. The *p* argument points to the *struct proc* of the kernel process to suspend or resume. For **kproc_suspend**(), the *timo* argument specifies a timeout to wait for the kernel process to acknowledge the suspend request and suspend itself.

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

The **kproc_kthread_add**() function is much like the **kproc_create**() function above except that if the kproc already exists, then only a new thread (see kthread(9)) is created on the existing process. The *func* argument specifies the function that the process 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 process. The *procptr* pointer points to a *struct proc* pointer that is the location to be updated with the new proc pointer if a new process is created, or if not NULL, must contain the process pointer for the already existing process. If this argument points to NULL, then a new process is created and the field updated. If not NULL, the *tdptr* pointer points to a *struct thread* pointer that is the location to be updated with the new thread pointer. The *flags* argument specifies a set of flags as described in rfork(2). 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 procname argument is the name the new process should be given if it needs to be created. It is *NOT* a printf style format specifier but a simple string. 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*.

RETURN VALUES

The **kproc_create**(), **kproc_resume**(), and **kproc_suspend**() functions return zero on success and non-zero on failure.

EXAMPLES

This example demonstrates the use of a *struct kproc_desc* and the functions **kproc_start**(), **kproc_shutdown**(), and **kproc_suspend_check**() to run the **bufdaemon** process.

```
static struct proc *bufdaemonproc;

static struct kproc_desc buf_kp = {
        "bufdaemon",
        buf_daemon,
        &bufdaemonproc
};
SYSINIT(bufdaemon, SI_SUB_KTHREAD_BUF, SI_ORDER_FIRST, kproc_start,
    &buf_kp)

static void
buf_daemon()
{
        ...
        /*
         * This process needs to be suspended prior to shutdown sync.
         */
        EVENTHANDLER_REGISTER(shutdown_pre_sync, kproc_shutdown,
            bufdaemonproc, SHUTDOWN_PRI_LAST);
        ...
        for (;;) {
                kproc_suspend_check(bufdaemonproc);
                ...
        }
}
```

ERRORS

The **kproc_resume**() and **kproc_suspend**() functions will fail if:

[EINVAL]	The *p* argument does not reference a kernel process.

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

[EAGAIN]	The system-imposed limit on the total number of processes under execution would be exceeded. The limit is given by the sysctl(3) MIB variable KERN_MAXPROC.
[EINVAL]	The RFCFDG flag was specified in the *flags* parameter.

SEE ALSO

rfork(2), exit1(9), kthread(9), SYSINIT(9), wakeup(9)

HISTORY

The **kproc_start**() function first appeared in FreeBSD 2.2. The **kproc_shutdown**(), **kproc_create**(), **kproc_exit**(), **kproc_resume**(), **kproc_suspend**(), and **kproc_suspend_check**() functions were introduced in FreeBSD 4.0. Prior to FreeBSD 5.0, the **kproc_shutdown**(), **kproc_resume**(), **kproc_suspend**(), and **kproc_suspend_check**() functions were named **shutdown_kproc**(), **resume_kproc**(), **shutdown_kproc**(), and **kproc_suspend_loop**(), respectively. Originally they had the names **kthread_***() but were changed to

kproc_*() when real kthreads became available.