NAME

swi_add, **swi_remove**, **swi_sched** - register and schedule software interrupt handlers

SYNOPSIS

#include <sys/param.h>
#include <sys/bus.h>
#include <sys/interrupt.h>

*extern struct intr_event *clk_intr_event*;

int
swi_add(*struct intr_event **eventp, const char *name, driver_intr_t handler, void *arg, int pri, enum intr_type flags, void **cookiep*);

int
swi_remove(*void *cookie*);

void
swi_sched(*void *cookie, int flags*);

DESCRIPTION

These functions are used to register and schedule software interrupt handlers. Software interrupt handlers are attached to a software interrupt thread, just as hardware interrupt handlers are attached to a hardware interrupt thread. Multiple handlers can be attached to the same thread. Software interrupt handlers can be used to queue up less critical processing inside of hardware interrupt handlers so that the work can be done at a later time. Software interrupt threads are different from other kernel threads in that they are treated as an interrupt thread. This means that time spent executing these threads is counted as interrupt time, and that they can be run via a lightweight context switch.

The **swi_add**() function is used to add a new software interrupt handler to a specified interrupt event. The *eventp* argument is an optional pointer to a *struct intr_event* pointer. If this argument points to an existing event that holds a list of interrupt handlers, then this handler will be attached to that event. Otherwise a new event will be created, and if *eventp* is not NULL, then the pointer at that address to will be modified to point to the newly created event. The *name* argument is used to associate a name with a specific handler. This name is appended to the name of the software interrupt thread that this handler is attached to. The *handler* argument is the function that will be executed when the handler is scheduled to run. The *arg* parameter will be passed in as the only parameter to *handler* when the function is executed. The *pri* value specifies the priority of this interrupt handler relative to other software interrupt handlers. If an interrupt event is created, then this value is used as the vector, and the *flags* argument is used to specify the attributes of a handler such as INTR_MPSAFE. The *cookiep* argument points to a *void **

cookie. This cookie will be set to a value that uniquely identifies this handler, and is used to schedule the handler for execution later on.

The **swi_remove**() function is used to teardown an interrupt handler pointed to by the *cookie* argument. It detaches the interrupt handler from the associated interrupt event and frees its memory.

The **swi_sched**() function is used to schedule an interrupt handler and its associated thread to run. The *cookie* argument specifies which software interrupt handler should be scheduled to run. The *flags* argument specifies how and when the handler should be run and is a mask of one or more of the following flags:

SWI_DELAY Specifies that the kernel should mark the specified handler as needing to run, but the kernel should not schedule the software interrupt thread to run. Instead, *handler* will be executed the next time that the software interrupt thread runs after being scheduled by another event.

SWI_FROMNMI Specifies that **swi_sched**() is called from NMI context and should be careful about used KPIs. On platforms allowing IPI sending from NMI context it immediately wakes *clk_intr_event* via the IPI, otherwise it works just like SWI_DELAY.

clk_intr_event is a pointer to the *struct intr_event* used to hang delayed handlers off of the clock interrupt, and is invoked directly by hardclock(9).

RETURN VALUES

The **swi_add**() and **swi_remove**() functions return zero on success and non-zero on failure.

ERRORS

The **swi_add**() 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 *flags* argument specifies INTR_ENTROPY.

[EINVAL] The *eventp* argument points to a hardware interrupt thread.

[EINVAL] Either of the *name* or *handler* arguments are NULL.

[EINVAL] The INTR_EXCL flag is specified and the interrupt event pointed to by *eventp*

already has at least one handler, or the interrupt event already has an exclusive handler.

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

[EINVAL] A software interrupt handler pointed to by *cookie* is NULL.

SEE ALSO

hardclock(9), intr_event(9), taskqueue(9)

HISTORY

The **swi_add**() and **swi_sched**() functions first appeared in FreeBSD 5.0. They replaced the **register_swi**() function which appeared in FreeBSD 3.0 and the **setsoft***(), and **schedsoft***() functions which date back to at least 4.4BSD. The **swi_remove**() function first appeared in FreeBSD 6.1.