NAME

config_intrhook - schedule a function to be run after interrupts have been enabled, but before root is mounted

SYNOPSIS

#include <sys/kernel.h>

*typedef void (*ich_func_t)(void *arg);*

int
config_intrhook_establish(*struct intr_config_hook *hook*);

void
config_intrhook_disestablish(*struct intr_config_hook *hook*);

int
config_intrhook_drain(*struct intr_config_hook *hook*);

void
config_intrhook_oneshot(*ich_func_t func, void *arg*);

DESCRIPTION

The **config_intrhook_establish**() function schedules a function to be run after interrupts have been enabled, but before root is mounted. If the system has already passed this point in its initialization, the function is called immediately.

The **config_intrhook_disestablish**() function removes the entry from the hook queue.

The **config_intrhook_drain**() function removes the entry from the hook queue in a safe way. If the hook is not currently active it removes *hook* from the hook queue and returns *ICHS_QUEUED*. If the hook is active, it waits for the hook to complete before returning *ICHS_RUNNING*. If the hook has previously completed, it returns *ICHS_DONE*. Because a *config_intrhook* is undefined prior to **config_intrhook_establish**(), this function may only be called after that function has returned.

The **config_intrhook_oneshot**() function schedules a function to be run as described for **config_intrhook_establish**(); the entry is automatically removed from the hook queue after that function runs. This is appropriate when additional device configuration must be done after interrupts are enabled, but there is no need to stall the boot process after that. This function allocates memory using M_WAITOK; do not call this while holding any non-sleepable locks.

Before root is mounted, all the previously established hooks are run. The boot process is then stalled until all handlers remove their hook from the hook queue with **config_intrhook_disestablish**(). The boot process then proceeds to attempt to mount the root file system. Any driver that can potentially provide devices they wish to be mounted as root must use either this hook, or probe all these devices in the initial probe. Since interrupts are disabled during the probe process, many drivers need a method to probe for devices with interrupts enabled.

The requests are made with the *intr_config_hook* structure. This structure is defined as follows:

```
struct intr_config_hook {
        TAILQ_ENTRY(intr_config_hook) ich_links;/* Private */
        ich_func_t      ich_func;       /* function to call */
        void            *ich_arg;       /* Argument to call */
};
```

Storage for the *intr_config_hook* structure must be provided by the driver. It must be stable from just before the hook is established until after the hook is disestablished.

Specifically, hooks are run at **SI_SUB_INT_CONFIG_HOOKS**(), which is immediately after the scheduler is started, and just before the root file system device is discovered.

RETURN VALUES

A zero return value means the hook was successfully added to the queue (with either deferred or immediate execution). A non-zero return value means the hook could not be added to the queue because it was already on the queue.

SEE ALSO

DEVICE_ATTACH(9)

HISTORY

These functions were introduced in FreeBSD 3.0 with the CAM subsystem, but are available for any driver to use.

AUTHORS

The functions were written by Justin Gibbs <*gibbs@FreeBSD.org*>. This manual page was written by M. Warner Losh <*imp@FreeBSD.org*>.