NAME

EVENTHANDLER - kernel event handling functions

SYNOPSIS

#include <sys/eventhandler.h>

EVENTHANDLER_DECLARE(*name*, *type*);

EVENTHANDLER_DEFINE(*name*, *func*, *arg*, *priority*);

EVENTHANDLER_INVOKE(*name*, ...);

eventhandler_tag
EVENTHANDLER_REGISTER(*name*, *func*, *arg*, *priority*);

EVENTHANDLER_DEREGISTER(*name*, *tag*);

EVENTHANDLER_DEREGISTER_NOWAIT(*name*, *tag*);

EVENTHANDLER_LIST_DECLARE(*name*);

EVENTHANDLER_LIST_DEFINE(*name*);

EVENTHANDLER_DIRECT_INVOKE(*name*);

eventhandler_tag
eventhandler_register(*struct eventhandler_list *list*, *const char *name*, *void *func*, *void *arg*, *int priority*);

void
eventhandler_deregister(*struct eventhandler_list *list*, *eventhandler_tag tag*);

void
eventhandler_deregister_nowait(*struct eventhandler_list *list*, *eventhandler_tag tag*);

*struct eventhandler_list **
eventhandler_find_list(*const char *name*);

void
eventhandler_prune_list(*struct eventhandler_list *list*);

DESCRIPTION

The **EVENTHANDLER** mechanism provides a way for kernel subsystems to register interest in kernel events and have their callback functions invoked when these events occur.

Callback functions are invoked in order of priority. The relative priority of each callback among other callbacks associated with an event is given by argument *priority*, which is an integer ranging from EVENTHANDLER_PRI_FIRST (highest priority), to EVENTHANDLER_PRI_LAST (lowest priority). The symbol EVENTHANDLER_PRI_ANY may be used if the handler does not have a specific priority associated with it.

The normal way to use this subsystem is via the macro interface. For events that are high frequency it is suggested that you additionally use **EVENTHANDLER_LIST_DEFINE**() so that the event handlers can be invoked directly using **EVENTHANDLER_DIRECT_INVOKE**() (see below). This saves the invoker from having to do a locked traversal of a global list of event handler lists.

EVENTHANDLER_DECLARE()

This macro declares an event handler named by argument *name* with callback functions of type *type*.

EVENTHANDLER_DEFINE()

This macro uses SYSINIT(9) to register a callback function *func* with event handler *name*. When invoked, function *func* will be invoked with argument *arg* as its first parameter along with any additional parameters passed in via macro **EVENTHANDLER_INVOKE**() (see below).

EVENTHANDLER_REGISTER()

This macro registers a callback function *func* with event handler *name*. When invoked, function *func* will be invoked with argument *arg* as its first parameter along with any additional parameters passed in via macro **EVENTHANDLER_INVOKE**() (see below). If registration is successful, **EVENTHANDLER_REGISTER**() returns a cookie of type *eventhandler_tag*.

EVENTHANDLER_DEREGISTER()

This macro removes a previously registered callback associated with tag *tag* from the event handler named by argument *name*. It waits until no threads are running handlers for this event before returning, making it safe to unload a module immediately upon return from this function.

EVENTHANDLER_DEREGISTER_NOWAIT()

This macro removes a previously registered callback associated with tag *tag* from the event handler named by argument *name*. Upon return, one or more threads could still be running the removed function(s), but no new calls will be made. To remove a handler function from within that function, use this version of deregister, to avoid a deadlock.

EVENTHANDLER_INVOKE()
This macro is used to invoke all the callbacks associated with event handler *name*. This macro is a variadic one. Additional arguments to the macro after the *name* parameter are passed as the second and subsequent arguments to each registered callback function.

EVENTHANDLER_LIST_DEFINE()
This macro defines a reference to an event handler list named by argument *name*. It uses SYSINIT(9) to initialize the reference and the eventhandler list.

EVENTHANDLER_LIST_DECLARE()
This macro declares an event handler list named by argument *name*. This is only needed for users of **EVENTHANDLER_DIRECT_INVOKE**() which are not in the same compilation unit of that list's definition.

EVENTHANDLER_DIRECT_INVOKE()
This macro invokes the event handlers registered for the list named by argument *name*. This macro can only be used if the list was defined with **EVENTHANDLER_LIST_DEFINE**(). The macro is variadic with the same semantics as **EVENTHANDLER_INVOKE**().

The macros are implemented using the following functions:

eventhandler_register()
The **eventhandler_register**() function is used to register a callback with a given event. The arguments expected by this function are:

list A pointer to an existing event handler list, or NULL. If *list* is NULL, the event handler list corresponding to argument *name* is used.

name The name of the event handler list.

func A pointer to a callback function. Argument *arg* is passed to the callback function *func* as its first argument when it is invoked.

priority The relative priority of this callback among all the callbacks registered for this event. Valid values are those in the range EVENTHANDLER_PRI_FIRST to EVENTHANDLER_PRI_LAST.

The **eventhandler_register**() function returns a *tag* that can later be used with **eventhandler_deregister**() to remove the particular callback function.

eventhandler_deregister()

The **eventhandler_deregister**() function removes the callback associated with tag *tag* from the event handler list pointed to by *list*. If *tag* is *NULL*, all callback functions for the event are removed. This function will not return until all threads have exited from the removed handler callback function(s). This function is not safe to call from inside an event handler callback.

eventhandler_deregister_nowait()

The **eventhandler_deregister**() function removes the callback associated with tag *tag* from the event handler list pointed to by *list*. This function is safe to call from inside an event handler callback.

eventhandler_find_list()

The **eventhandler_find_list**() function returns a pointer to event handler list structure corresponding to event *name*.

eventhandler_prune_list()

The **eventhandler_prune_list**() function removes all deregistered callbacks from the event list *list*.

RETURN VALUES

The macro **EVENTHANDLER_REGISTER**() and function **eventhandler_register**() return a cookie of type *eventhandler_tag*, which may be used in a subsequent call to **EVENTHANDLER_DEREGISTER**() or **eventhandler_deregister**().

The **eventhandler_find_list**() function returns a pointer to an event handler list corresponding to parameter *name*, or NULL if no such list was found.

HISTORY

The **EVENTHANDLER** facility first appeared in FreeBSD 4.0.

AUTHORS

This manual page was written by Joseph Koshy <*jkoshy@FreeBSD.org*> and
Matt Joras <*mjoras@FreeBSD.org*>.