NAME

timeout – execute a function after a specified length of time

SYNOPSIS

#include <sys/types.h>
#include <sys/conf.h>

int timeout(void *(*func)***(caddr_t), caddr_t** *arg***, long** *ticks***);**

INTERFACE LEVEL

Architecture independent level 1 (DDI/DKI).

ARGUMENTS

func Kernel function to invoke when the time increment expires.

arg Argument to the function.

ticks Number of clock ticks to wait before the function is called.

DESCRIPTION

The **timeout( )** function schedules the specified function to be called after a specified time interval. The exact time interval over which the timeout takes effect cannot be guaranteed, but the value given is a close approximation.

The function called by **timeout( )** must adhere to the same restrictions as a driver soft interrupt handler.

The **timeout( )** function returns an identifier that may be passed to the **untimeout**(9F) function to cancel a pending request.

RETURN VALUES

Under normal conditions, **timeout( )** returns an integer timeout identifier not equal to zero. If, however, the timeout table is full, the system will panic with the following panic message:

PANIC: Timeout table overflow

CONTEXT

timeout( ) can be called from user or interrupt context.

EXAMPLE

In the following example, the device driver has issued an IO request and is waiting for the device to respond. If the device does not respond within 5 minutes, the device driver will print out an error message to the console.

```
static void
xxtimeout_handler(caddr_t arg)
{
        struct xxstate *xsp = (struct xxstate *)arg;

        mutex_enter(&xsp->lock);
        cv_signal(&xsp->cv);
        xsp->timeout_id = 0;
        xsp->flags |= TIMED_OUT;
        mutex_exit(&xsp->lock);
}
```

```
static u_int
xxintr(caddr_t arg)
{
        struct xxstate *xsp = (struct xxstate *)arg;
     .
     .
     .
        mutex_enter(&xsp->lock);
        if (xsp->timeout_id != 0) {
                (void) untimeout(xsp->timeout_id);
                xsp->timeout_id = 0;
        }

        /* Service interrupt */

        cv_signal(&xsp->cv);
        mutex_exit(&xsp->lock);

        return(DDI_INTR_CLAIMED);
}

static void
xxcheckcond(struct xxstate *xsp)
{
     .
     .
     .
        mutex_enter(&xsp->lock);
        xsp->timeout_id = timeout(xxtimeout_handler,
          (caddr_t)xsp, (5 * drv_usectohz(1000000));
        while (/* Waiting for interrupt  or timeout*/)
                cv_wait(&xsp->cv, &xsp->lock);

        if (xsp->flags & TIMED_OUT)
                cmn_err(CE_WARN, "Device not responding");
     .
     .
     .
        mutex_exit(&xsp->lock);
     .
     .
     .
}
```

SEE ALSO

bufcall(9F), **delay**(9F), **untimeout**(9F)

Writing Device Drivers