CWAIT(a) CWAIT(a)

NAME

cwait − conditional wait for event

SYNOPSIS

(cwait = 25.)
cwait(&flag)
crdblk(&flag)
cyield(&flag)
int flag;

or

cwait(0)
crdblk(0)
cyield(0)

DESCRIPTION

Cwait causes the current process to give up control (enter the road blocked state) if the value of *flag* is non-zero; an immediate return occurs if *flag* is zero. A *cwait* call with an argument of zero causes the *p_cwait* location in the PCB to be used in place of *flag*. *P_cwait* is set to one by many of the kernel EMT traps: (sleep, sendmsg, sendmsgfrom, sendcpmsg, ioqueuem, getmsg, gettype, event, and cwait), and cleared by the kernel EMT traps enevent and clrevent, as well as the occurance of any event. If *flag* or *p_cwait* is non-zero the location *p_cwait* (also in the PCB) will be set. This will cause the scheduler to keep the process in memory for the remainder of it's time slice.

Cwait should only be used if the process expects the condition causing the process to road block will be cleared up within 200 milliseconds. If a longer wait is expected use *crdblk*. *Cyield* should be used to give up control immediately.

In assembly language, r0 should point to a block of two words, the first word which is a flag to the scheduler and the second word which is the address of the synchronization flag *flag* in the caller's address space. If the address of the synchronization flag is zero, the *p_cwait* location in the PCB is used. The value of the scheduler flag is < 0 for *cyiedd,* = 0 for *crdblk* and > 0 for *cwait.*

Since event interrupts are inhibited while the kernel checks *flag,* potential timing problems between the "base line" and asynchronous event handler parts of a supervisor process can be resolved. The type of timing problem is illustrated by the buffered I/O in the UNIX supervisor: The "base line" code will set *flag* to one and initiate a buffer write then call *cwait(&flag)* waiting for the I/O to complete. If the I/O manages to complete before the "base line" completes execution of the *cwait* (preemption could occur), the event handler will mark the buffer I/O as done and clear *flag.* Base line will then complete the cwait call. The kernel will detect a zero *flag* and return from the *cwait* preventing the supervisor from road blocking for an event which has already occurred.

A value of 1 is returned from C.

SEE ALSO

DIAGNOSTICS