NAME

sleep − stop execution for interval

SYNOPSIS

sleep (seconds)
unsigned seconds;

DESCRIPTION

The current process is suspended from execution for the number of seconds specified by the argument. *Sleep* is implemented by using *signal*(2) to catch the alarm clock signal, then using *alarm*(2) and *pause*(2) to set up and wait for the alarm to occur after the number of seconds specified by the argument. *Sleep* returns the number of seconds left of the sleep - always zero unless the sleep is interrupted by a signal.

Sleep and *alarm* interact intelligently, i.e. the C interface to sleep is intelligent enough to preserve the alarm signal vector and alarm timer while setting up a sleep. When a sleep is terminated, either by a signal or by the sleep time expiring, the alarm vector is set back to the value it had before the sleep and the alarm timer is set to the time it would have had taking into account the time elapsed while sleeping. Of course, if the alarm is scheduled to go off before the sleep time expires, *sleep* does not change the alarm vector or alarm timer. In that case, the time returned by *sleep* will be the sleep time less the alarm time; assuming, of course, that the alarm signal is the first signal to disturb *sleep.* Alarm timers will operate for the correct number of "real" seconds no matter how long signal processing routines take.

One consequence of the above scheme that is not adequately explained is as follows; suppose you set up to ignore alarms (*signal*(2)) set an alarm timer for 60 seconds, and then execute a 70 second *sleep.* After 60 seconds of time has elapsed, you will get a return from *sleep* of 10 seconds because the alarm time has expired. It might help in your understanding of *sleep* if you considered *sleep* to be a 'timed' pause function. *Pause* relinquishes the processor until the alarm timer expires; *sleep* relinquishes the processor until the alarm timer expires or until the requested sleep time expires, whichever comes first.

SEE ALSO

sleep(1), alarm(2), pause(2), signal(2)

BUGS

Sleep does not work correctly when called recursively. Only the last *sleep* called will return the correct value. The other *sleeps* will return the same value as the last *sleep.* If you don't particularly care what *sleep* returns, then *sleep* may be used recursively.