Time Management

Hardware Clocks

A *real-time clock* (or *line clock*) pulses at an integral number of times each second, interrupting the CPU each time a pulse occurs. If the CPU fails to service an interrupt before the next pulse, it will miss the interrupt.

The *clock rate* refers to how fast the clock pulses. Some clock rates are 10Hz, 60Hz, 100Hz, and 1000Hz.

A *programmable clock* uses a holding register to maintain a starting counter. A quartz crystal then generates a periodic signal (up to 1000MHz or more).

On each pulse the counter is decremented. When the counter reaches zero an interrupt is generated and the counter can be reset from holding register.

The holding counter allows the operating system can control the interrupt frequency by modifying the holding register.

See Fig. 5-31.

Uses of Clocks

Real-time clocks are used to schedule events:

- *preemption events* guarantee that equal priority processes receive round robin service and to break infinite loops
- *wakeup events* allow a process to suspend itself (or another) for an amount of time specified by the process. Or to get an interrupt at a future time.

  In Linux/Unix, the routine *sleep(sec)* suspends the current process for *sec* seconds.
- may have to keep track of time-of-day
- the OS kernel might support a timeout mechanism that calls a kernel procedure at some future time. Watchdog timers.
- performance monitoring: at each event, record which process was running to determine percentage of cpu time used by each process
- measure percentage of time spent in user mode and privileged mode
- profiling: at each event, record the region in which the program is executing
- accounting: charge processes for resource usage (e.g., memory)

Delta List Processing

How should timer events be stored to minimize work on each clock tick? They could be stored in a list, with:

- an absolute time of wakeup stored in the key field
- the time relative to the current time stored in the key field
- the *delta* (difference) relative to other (earlier) events stored in key field

Because of the expense of searching through arbitrarily long lists of events to find those that occur at a given clock tick, operating systems typically store timer events in *delta lists*.

Events in delta lists are ordered by the time at which they will occur; an event's key gives the number of clock ticks that must elapse relative to the preceding element of the list. The event at the front of the list stores the number of ticks relative to "now".

For instance, if processes A-E had wakeup times scheduled at:

1017, 1027, 1028, 1032, and 1032

and the current time is 1000, the delta list would contain:

17, 10, 1, 4, and 0