CSCI 4061
Introduction to Operating Systems

Instructor: Abhishek Chandra

Outline
- Times and Timers
  - Time representation
  - Time and date functions
  - Clocks and Timers
  - Timer drifts and overruns

Times and Timers
- Why do we need times and timers?
- Users and applications:
  - Current date/time
  - Sleep and alarms
- OS:
  - Process scheduling
  - Network protocol timeouts
  - Periodic updates of system statistics

Time Representation
- Wallclock time: Absolute or "real" time
- Clock ticks: Hardware/OS clock ticks
- User time: Process time spent in user mode
- System time: Process time spent in kernel mode
Wallclock Time
- Expressed in seconds
- Time counted since the Epoch
  - 0:00 AM, Jan 1, 1970

```c
time_t time(time_t *tloc);
```
- Returns the time since the Epoch
- `time_t` usually defined as `long`
  - What if it is 32-bit?

Using Wallclock Time
- Can be used to find the time elapsed between two points in a program

```c
time_t start_time, end_time;
time_t elapsed_time;

/* Record start time */
start_time = time(NULL);
do_something_useful();
/* Record end time */
end_time = time(NULL);

elapsed_time = end_time - start_time;
```

Increasing Time Resolution
- `time()` has a resolution of seconds
- Would like a finer resolution for timing shorter program segments

```c
struct timeval {
    time_t tv_sec; /* seconds */
    time_t tv_usec; /* microseconds */
}
```
- Used by several functions to specify timeouts. E.g.: `select`
- `gettimeofday`: Returns time since Epoch in seconds and microseconds

Sleep
- `unsigned sleep(unsigned seconds);`
- Process blocks for specified period of time
- Could be interrupted by a signal
  - Returns remaining time
- `nanosleep`: Allows finer granularity
Clocks

- Hardware clock:
  - Periodic oscillator
  - Has a base resolution
- OS clock:
  - Software counter
  - Counter value expressed as clock ticks
  - E.g.: Linux uses "jiffies" (10 ms)

Measuring Process Time

- Multiple processes share the system
- Actual time spent by a process is different from the elapsed wallclock time
- Virtual time: Time spent by a process in running state
  - User time: Time spent in user space
  - System time: Time spent in kernel space

```
clock_t times(struct tms *buf);
```

Timers

- Generate a notification after a specified time interval
- Typically decrement a counter to 0
- OS uses timers for:
  - Process scheduling: Switch process after scheduler timer interrupt
  - Network protocol timers: TCP uses timers for sending messages, acknowledgements, etc.

Interval Timers

- Processes can also set and use timers
- getitimer, setitimer
  - Get and set interval timer values
- OS sends SIGALRM
- Three types of timers:
  - ITIMER_REAL: Decrements in wallclock time
  - ITIMER_VIRTUAL: Decrements in process’s user time
  - ITIMER_PROF: Decrements in both user and kernel time of process
**Timer Drifts**

- There may be delay between timer expiry and handling it
  - E.g.: process may not be running at expiry time
  - Also depends on clock resolution
- Resetting timer to same value would cause a drift
  - E.g.: timer value of 5 sec, delay of 5 msec

**Timer Overruns**

- Could use absolute time to reset timer
  - Reset to (next expiry-current time)
- Could result in missing a timer expiry completely
- Timer overrun:
  - Loss of a scheduled timer interrupt
  - Arrival of next timer interrupt before handling previous one