Operating Systems

Course 5 - Time management

Adrien Krähenbühl

Master of Computer Science
PUF - Hồ Chí Minh

2016/2017

Interest

Time notion critical for the global system execution

- ✓ System is event-driven and time-driven at a time

Require to know when execute the tasks

- ✓ Regular tasks
  - Preempt and balance the scheduling queues
  - refresh screen
- ✓ Differed tasks
  - Disk I/Os

Provide indications to the user

- ✓ Current date
- ✓ Dates of modification/access to files
- ✓ Functions with timeout
- ✓ Alarms
- ✓ Usage statistics
  - Also useful for scheduling algorithms
- ✓ Event logs

How to know the time?

Using the clock tics:

- ✓ Programmable Interval Timer (PIT)
- ✓ The linux “jiffies”
- ✓ Light and regular but coarse
  - 250Hz with recent Linux 2.6/x86
  - 4ms of absolute precision
  - Duration of interrupt handling
- ✓ Increase the frequency to increase the precision?
  - Prohibitive cost of interruptions

How to know the time... more precisely?

- ✓ Count the cycles
  - In the majority of modern processors
  - Instruction `rdtsc` in x86
  - Very accurate in the short term, not expensive
- ✓ Hard to connect to actual time
  - Frequency changing
  - De-synchronization between processors (or between cores of a same processor)

How to know the absolute time?

- ✓ Ticks and TSC = relative time
- ✓ Absolute time given by the clock
  - RTC = Real Time Clock
  - Works even if the computer is turned off (as long as the motherboard battery is alive)
  - High granularity
  - Expensive to program and consult

Combine origins

- ✓ RTC read at boot
  - See from time to time
- ✓ Used as basic interrupts to keep a software time
  - Average granularity (ms)
- ✓ Other hardware-specific features to increase accuracy
  - TSC (less and less often)

Delete clock tics?

- ✓ Clock ticks may be unnecessary
  - Idle system
  - Nothing to do in the interruption handler
  - Unique process performing calculations without I/Os
- ✓ The OS deletes ticks by programming a tick only for the next event if necessary
  - Energy saving

Measurement of the scheduling duration

- ✓ At each clock tick, assign the scheduling duration to the current process
- ✓ Not very accurate
  - Other intermediate interruptions not properly counting
  - Intermediate preemptions ignored
- ✓ Sufficient for the scheduler
  - Time slice often at least of the order of 10ms
  - And not very expensive

Program an alarm

- ✓ Save a function to be executed after a certain time
  - To wake up a process (sleep, timeout, . . .)
  - Or send a SIGALRM
- ✓ How to run it?
  - The kernel must be activated at the right time to execute it
  - Same problem as for preemption
- ✓ Check periodically
  - If the delay is exceeded, execute

Program an alarm

- ✓ Active waiting until the expiration of the delay
  - Wastage of processor time
- ✓ Check whether the timeout has expired at each clock interruption
  - Not very accurate
- ✓ Modern processors can program interrupts more precisely (ex: HPET on x86)
  - Ex: ualarm (8100 µs) if tick = 4ms?
  - After 2 normal interrupt ticks, program an HPET alarm in 0.1 ms