Two Schedulers in PA1

Implementing Schedulers on XINU

- Solving the starvation problem

- Suggested background knowledge
  - The default scheduler (resched.c) and scheduling points (where it gets invoked)
  - Process management
    - Creation, suspension, resumption, sleep, wakeup, termination, priority, etc. …
    - Process queues (ready queue, clock queue) and their management (insert, dequeue, etc. …)

Exponential Distribution Scheduler

- Ensuring fairness by probability distribution and preemptions
- Important implementation points
  - The random number generator $X \sim \text{Exp}(\lambda = 0.1)$
  - Scheduling a **runnable** process (i.e., processes from the ready queue) whose priority is the lowest one above the outcome of X (or the highest if none is above)
  - Round robin on processes with the same priority
- Examining scheduling results against
  - Probability density function: $f(x; \lambda) = \lambda e^{-\lambda x}$
  - Cumulative distribution function: $F(x; \lambda) = 1 - e^{-\lambda x}$

Linux-like Scheduler

- Ensuring fairness by predetermined time quota and epochs
- Important concepts
  - **Epoch**: a turn that every runnable process is guaranteed a non-preemptive chance to run up to its predetermined time quantum
  - **Time quantum**: CPU ticks specifying how long a process can run within an epoch
  - **Base priority**: a static priority specified by create() or chprio()
  - **Goodness value**: a dynamic priority indicating when a process can be scheduled to run within an epoch

Determining Time Quantum and Goodness Value

- Time quantum
  - Base priority + 0.5 * unused quantum from the previous epoch
  - Determined at the beginning of each epoch
  - Computed for all processes including ones not in the ready queue
- Goodness value
  - Base priority + unused time quantum in the current epoch
  - 0 if time quantum is used up
  - Updated along with the unused time quantum whenever the scheduler is invoked

Scheduling Policies

- Starting a new epoch when all runnable processes have used up their time

- Scheduling the process with the highest goodness
  - Using the round-robin strategy if multiple processes have the same goodness
  - Be aware of the NULL process

- New processes and priority changes not in effect until the next epoch

Expected Results

- In each epoch, every runnable process will eventually be scheduled to run without being preempted until:
    - It uses up its time quantum
    - It yields (e.g., via invoking sleep) with unused time quantum
- A process with unused time quantum will be rescheduled if it becomes runnable again within the same epoch
- In other cases, half of the unused time quantum is carried to the next epoch

Other Implementation Notes

- Implementing the following functions to allow scheduler switching
  - void setschedclass (int sched_class)
    - Setting a scheduler type
    - sched_class: EXPDISTSCHED = 1, LINUXSCHED = 2
  - int getschedclass(): getting the current scheduler type

- Making your program compatible with the testmain.c that we provide