Operating Systems

CMPT 424 • Fall 2021

*i*Project Three - 150 points

Goals

To build on the functionality of *i*Project Two (all of which is required) by adding the ability to execute **multiple** user programs at the same time.

Functional Requirements

- ☐ Allow the user to load three programs into memory at once. [5 points]
- ☐ Add the following shell commands: [15 points]
  - clearmem — clear all memory partitions
  - runall — execute all programs at once
  - ps — display the PID and state of all processes
  - kill <pid> — kill one process
  - killall — kill all process
  - quantum <int> — let the user set the Round Robin quantum (measured in cpu cycles)
- ☐ Display the Ready queue and its contents (including process state) in real time. [10 points]
- ☐ [challenge] Track turnaround time and wait time for each process. [+10 points]

Implementation Requirements

- ☐ Store multiple programs in memory, each in their own partition / segment, allocated by the client OS (which obviously needs to keep track of available and used partitions/segments). [5 points]
- ☐ Add base and limit registers to your core memory access code in the host OS and to your PCB objects in the client OS. [5 points]
- ☐ Enforce memory partition boundaries at all times. [10 points]
- ☐ Create a Resident list for the loaded processes. [5 points]
- ☐ Create a Ready queue for the running processes. [5 points]
- ☐ Instantiate a PCB for each loaded program and put it in the Resident list. [5 points]
- ☐ Develop a CPU scheduler in the client OS using Round Robin scheduling with the user-specified quantum measured in cpu cycles (default = 6). [70 points]
  - Make the client OS control the host CPU with the CPU scheduler.
  - Log all scheduling events.
- ☐ Implement context switches with software interrupts. Be sure to update the mode bit (if appropriate), the PCBs, and the Ready queue. [10 points]
- ☐ Detect and gracefully handle errors like invalid op codes, missing operands (if you can detect that), and most importantly, memory out of bounds or access violation attempts. [5 points]
- ☐ Your code must *separate structure from presentation*, be professionally formatted, use and demonstrate best practices, and make me proud to be your teacher. [−∞ if not]
- ☐ You must commit to Git early and often. I want to see many small and descriptive commits, not one or two massive ones. In fact, I will not accept projects with too few commits. [−∞ if not]

Operating Systems

CMPT 424 • Fall 2021

Hints

- Do not reset PIDs as they are used. The PID value should always increase.
- The program counter for any process should never be greater than FF (255).
- Remember that scheduling is done via context switches triggered by the scheduler and implemented through software interrupts.
- Updating your Ready Queue only on a scheduling events prevents the initial state from being displayed right away. This is bad.
- Display each processes' state in the Ready Queue.
- What should happen if the user tries to clearmem while processes are running?
- Killing a process currently running on the CPU should work and should not create a zombie.
- Speaking of killing, GLaDOS is still alive. Do not break her.

Submitting Your Work

Update GitHub with your current code and let me know that it's ready for me to grade (and which branch to grade).