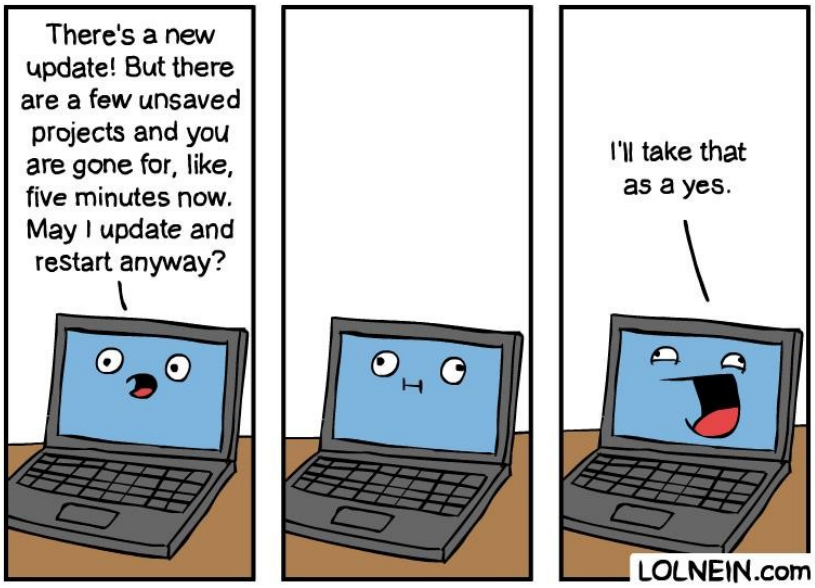
CS 370 – Project #4, Part B

Purpose: Become familiar with operating system process scheduling through **xv6**
Points: 300 (200 code / 100 final report)

Introduction

Any operating system is likely to run with more processes than the computer has CPUs, so a plan is needed to time-share the CPUs among the processes. Ideally the sharing would be transparent to user processes. A common approach is to provide each process with the illusion that it has its own virtual CPU by *multiplexing* the processes onto the hardware CPUs.


There's a new update! But there are a few unsaved projects and you are gone for, like, five minutes now. May I update and restart anyway?
I'll take that as a yes.
LOLNEIN.com

Resources

The following resources provide more in depth information regarding **xv6**. They include the **xv6** reference book, tools for installing, and guidance to better understand **xv6**.

1. xv6 Reference Book: https://pdos.csail.mit.edu/6.828/2020/xv6/book-riscv-rev1.pdf
2. Lab Tools Guide: https://pdos.csail.mit.edu/6.828/2020/tools.html
3. Lab Guidance: https://pdos.csail.mit.edu/6.828/2020/labs/guidance.html

Project

Project #4 will be done entirely in **xv6**. This part of the project is an extension of Part A. Complete the following steps.

Become familiar with **xv6** scheduling

- Complete, Project #4, part A.
- Implement First-Come-First-Server (FCFS) scheduler.
- Develop a user-level scheduler test program.
- Prepare a final summary report.

The following sections provide additional detail about the specific technical requirements.

First-Come-First-Server (FCFS) Scheduler

For this part of the project, we will modify the **xv6** scheduler from strict round-robin to a first-come-first-server (FCFS) scheduler. This will involve using the creation time entrying in the process control block that was added in part A.

We will modify the `scheduler` function (`kernel/proc.c`). Scheduler will first have to find a `RUNNABLE` process with the earliest creation time. The process with the earliest arrival time is the process with the highest priority and therefor the process that is selected for execution. Only when the currently RUNNING process terminates is another process selected to RUN.

It is suggested that you comment out the original round-robin scheduler code before adding your new version of the scheduler. This will allow you to switch back for testing in preparation for the final report.

Final Report

Prepare a final report summarizing the (1) approach to testing, (2) results of testing, (3) technical changes made to the source of the scheduler, and (4) anomalous behavior.

Approach to Testing

Develop a testing methodology and specific metrics by which to evaluate the performance of the FCFS scheduling algorithm in comparison to the existing RR. These metrics must come from the course material and textbook. After selecting the metrics, develop a methodology to collect said metrics, and provide a cohesive description of the methodology within the report.

Results of Testing

Following from the approach, present the metrics, and conclusions that can be drawn from their values. Ensure a comparison between the RR and FCFS is explicit and the impact of selecting one scheduling algorithm over the other informed by the data collected.

Technical Changes to the Scheduler

Provide an overview of the changes made to the source and their relationship to the concept of a FCSF scheduler.

Anomalous Behavior

Present any anomalous behavior with respect to process output or expected behavior. In addition, provide insight into the potential for starvation and a solution to starvation with a FCFS scheduling algorithm.

Length

The length of the final report is limited to 5 pages and a recommended length of 3 pages, single spaced, in 11 point font. The report should include appropriate titles and section headers. Additionally, spelling and grammar will be part of the final report score.

Submission

When complete, submit:

- Include a copy of the report within the zip file by including the report in the directory of your repository.
- After completing the project, type **`make clean`** and then create a ZIP file of the project as follows: **`zip -r proj4.zip .`** Which will compress your project and then submit the **`proj4.zip`** via Canvas.
- *Note*, ***the submission must include Part A and Part B code.***
- *Note*, late submissions will not be accepted.