Released:	Oct 15th, 2023
Due:	Oct 29th, 2023, 11:59 PM EST

CS-GY 6233 - Fall 2023

Homework #3

Scheduling

---

1. Nice [20 pts]
2. Random Number Generator [20 pts]
3. Scheduling [60 pts]

General Notes

- Read the assignment requirements carefully, especially what to include in your final submission for each part.
- This assignment is to be completed in xv6.
- You will need to create a PDF where you document your design choices and explanations of all three parts. When you are done, submit this pdf document on Brightspace along with your source code and a README file.
- Refer to the submission instructions on the bottom of this document.
- For each part, you are expected to write actual test cases, run the code and take screenshots of the output and explain how they work.
- **Please include screenshots of your test cases for ALL 3 parts.**

Background

- Read Chapter 5 in the xv6 book. It gives details on how scheduling and context switching works in xv6.
- Round-robin is the default and only scheduling policy in xv6. Regardless of the priority every runnable process gets an equal time slice in the CPU.
- Read about the basic concepts of lottery scheduling in this link

Part 1 - Nice (20 Points)

Having policies is the first step in implementing a non-trivial scheduler. For the sake of simplicity, you can implement the nice UNIX API as an extra system call (in xv6). A higher nice value means lower priority. For more information type – man 2 nice on Linux system.

You should be able to store the nice value in the process structure. Xv6 stores the PCB in `struct proc` in the file `proc.h`

Note: You are not expected to enforce specific permissions on nice. On Linux systems, typically only root can increase priorities. But you can skip this part in the assignment.

Implement the nice system call in xv6. Explain your implementation in the PDF in 1~2 paragraphs. To verify that the nice value can be set and read back correctly you would need to write a few test cases. Your test cases must include the ability to handle edge cases. What happens if the values passed in are out of bounds? Provide screenshot(s) of your test cases along with an explanation. Also, mention how to run your test cases - in what file is it located and what to run.

Part 2 - Random Number Generator (20 Points)

Lottery scheduling is a randomized heuristic. Hence, we need to design a pseudo-random number generator (PRNG). Since xv6 does not include a PRNG, you need to design one. This does not require strong randomness such as in cryptographic algorithms but a decent PRNG will do.

We recommend using the Xorshift family, although you can use anything else. If you do choose to use the Xorshift values, care should be taken to pick non-zero seed values.

Implement an in-kernel PRNG and write a simple test case to verify that the values returned look evenly distributed. Explain your implementation in the PDF in 1~2 paragraphs. Provide screenshot(s) of your test cases along with an explanation. Also, mention how to run your test cases - in what file is it located and what to run.

Hints:

- For the purpose of this assignment, to test that your PRNG function works, you could run it 1000 times, and test for the max, min, and mean to see if the values are relatively evenly distributed and that the numbers are not heavily skewed. You may plot a histogram to check your even distribution and include it in your report.

Part 3 - Scheduler (60 Points)

Mentioned in the link, lottery scheduling works by assigning tickets to each of the processes in the system. Then, in each time slice, it randomly picks a 'winning' ticket. It is up to you to decide how many more tickets to assign less nice processes and vice versa.

The design problem is more open-ended. After reading the text mentioned in the link above, some of the design factors you might want to consider:

- Which data structure makes sense?
- In what way would you handle a process leaving or entering a system all while maintaining a policy?
- How would you map the winning number back to the runnable process?

Note: You should keep both the schedulers as an option. It is fine to select the scheduler (Round Robin vs. Lottery) to be a compile time option.

Implement lottery scheduling in xv6. As mentioned, the particulars of the data structure and the policies to map the number of tickets to nice values is a design choice which is open-ended. You have to convince us that your implementation handles edge cases. Provide a detailed explanation of your approach. (Questions like 'what if?' should be mentioned and handled as well)

Come up with at least 3 test cases for your lottery scheduler and provide screenshot(s) of your test cases along with an explanation. Also, mention how to run your test cases - in what file is it located and what to run.

Hints:

- You can fork several children doing the same task with different priorities. Each of the children can print the progress in incremental points.

- You may change the priority of a process dynamically and check if its progress accelerates or slows down appropriately.

Submission

On Brightspace, you must submit:

1. The source code (you can submit your entire xv6 folder as zipped file)
2. A README specifying how to run each part of the assignment.
3. A PDF containing explanation to all parts as well as screenshots.
4. partner.txt

Academic Honesty

Aside from the narrow exception for collaboration on homework, all work submitted in this course must be your own. Cheating and plagiarism will not be tolerated. If you have any questions about a specific case, please ask the Prof/TAs. We will be checking for this!

NYU Tandon's Policy on Academic Misconduct:
http://engineering.nyu.edu/academics/code-of- conduct/academic-misconduct