CSC3002F 2025, Operating Systems Part 1 Assignment

Lecturer: Michelle Kuttel

Sarah the Zombie Barman

Version 1, 14 April 2025

Introduction

The aim of this assignment is to increase your understanding of **CPU process scheduling,** particularly the different **metrics** by which schedulers are compared, and to give you some experience evaluating and comparing basic scheduling algorithms using a **simulation**. Computer simulation is used to model the behaviour of a complex system; the theory is that the simplified model has *predictive properties* and can tell us something about a real world system.

You are given a Java package for simulating a version of process scheduling. This is simple simulation (the output is just text) involves a scenario of a busy bar where patrons arrive at random times throughout the evening and place orders for 1-5 drinks (the drink orders are the "jobs" to be processed). The drinks orders are filled by **Sarah the Zombie Barman** (who here is the CPU and scheduler in one). In the code you are given, **Sarah the Zombie Barman** uses one of three possible scheduling algorithms: **First Come First Served** (FCFS), **Shortest Job First** (SJF) with no pre-emption and **Round Robin** (RR).

Assignment Specifications

You are provided with a simple Java package. You will need to inspect it carefully, run it with different inputs (you can use the Makefile) and figure out what it does. Note carefully the **synchronisation mechanisms** used in the code to keep it thread safe (we discussed thread safety last year and will do so again this year).

Your task is to extend the code supplied so that you can compare the performance of the **FCFS**, **SJF** and **RR** schedules for **Sarah the Zombie Barman**. To do this, you will need to add **system calls for timing** to the code, and to write their values to files for analysis. You should also experiment with shell scripts to automate the testing for different scenarios. Note that, in order to compare the same simulation across all the scheduling algorithms, you should **use the same random seed** for each run to be compared.

compare performance, adding system calls for timing scripts

First, you will choose and justify a **suitable context switching time**. Then you will experiment to **find an optimal value** for the **time quantum (q)** for the **RR** algorithm. Then, **using this value of q,** you must compare the performance of the **FCFS**, **SJF** and **RR** schedules for this scenario across **all the metrics** discussed in the course (pages 9 and 10 of the second slide deck in this section). I am happy to clarify what these mean for **Sarah the Zombie Barman** in class or on the MS Team, if asked well ahead of the deadline.

find a context switching time
find q for RR. Then using q - compare FCFS, SJF, RR across all metrics

main criteria: CPU utilization
throughput
turnaround time
waiting time
response time

additional criteria:
predictability
fairness
lack of starvation

CPU utilisation - time barman is making the drink, more context switches less utilisation.
IO time included in turnaround time

fixed number of drinks -> 5

End

assume barman takes same amount of time in making the drink

Note that you will have to decide how best to test the algorithms, bearing in mind that **experiments** must be run **multiple times** and with a **big enough sample** for reliable data.

Then must submit a **short report** containing **graphs of your data** that show the results of your experiments. The report must **explain** how you determined a good value for the **context switch** and the **optimal value of q** for **RR**, using the data to back you up. You need to **show graphs** of the scheduling algorithm behaviours across all five metrics and **draw conclusions** on the basis of the data. You must compare and contrast the **average values, the median values** and the **distribution** of each metric for each algorithm. (A reasonable and predictable wait for a drink may more desirable than a faster average, but with some patrons waiting for a very long time.) **Do the algorithms behave as expected**? You should also comment on the **predictability, fairness** and **possibility of starvation** in this scenario for the three algorithms, using your experimental data to back up your claims.

Finally you need conclude by **making a recommendation** for the best scheduling algorithm for **Sarah the Zombie Barman**, and **justify your recommendation**.

CONSTRAINTS

- You have to profile the code provided – you may not re-do it and you may not change the fundamental way it works.
- You may not change the command-line arguments for the code (but you may add a few more, as long as the original ones still work).
- The list of drinks is fixed – you may not change it.
- You may not change the arrival times for the patrons.

SUBMISSION

1. Submit the **code** to the automarker as a zipped **archive.** Your submission archive must contain the following.
    - **All the files** needed to run your solution, including any shell scripts.
    - A **Makefile** so that the following command **must run your solution** correctly.

      ```
      make run ARGS="30 2 5 30"
      ```

    - **a GIT usage log (**as a .txt file, use `git log –all` to display all commits and save).
- Submit a separate document in **PDF format** containing your report. It should be about 3-10 pages in length, and report on all the aspects listed above.

CPU burst time - barman is the CPU and schedular. Time to make the drink is CPU burst. input output is waiting time??

Time units of seconds
Turnaround time,

Figure out how to compare the different algorithm. e.g x axis is number of patients
can show distribution per patron. find a nice number 5 is not good data, 100 is slow - can figure out average - then each patron figure our turnaround time. one graph or separate. orefer on one page.

run a more than 3, but run a few 10 times - mostly all 10 times. use maybe a shell script or you have to babysit code. use shellscript - you can set random seed. each iteration has all algorithms run with the same seed.