Scheduling Simulation Programming Project - Total Points 100

Scheduling

The goal of this project is to implement a process scheduling simulator for the non-preemptive scheduling algorithms discussed in the textbook. Your code will simulate the *First Come First Served* (FCFS), *Shortest Process Next* (SPN), and *Highest Response Ratio Next* (HRRN) algorithms. Your simulator will allow the user to compare the performance of the different algorithms in terms of average *Wait Time* and average *Turnaround Time*.
In addition, you will be required to submit a 2-4 page report explaining the project and addressing the following items:

- Introduction — describe the project and goals
- Scheduling Algorithms — describe/define the three scheduling algorithms you are simulating. Provide at least one advantage and one disadvantage of each scheduling algorithm in comparison to the the other two. Construct and provide one example of scheduling at least 5 processes using each respective algorithm. Use a table to keep track of the finish time, turnaround time, and wait time, similar to what the author does in the textbook. Compute the average wait time and average turnaround time. You cannot use the author's examples — create your own! NOTE: your examples are also great for testing.
- Performance Metrics — describe/define the two performance metrics we are using.
- Simulation Algorithm — describe in English your simulation algorithm including what data structures you use, what information you track, and how you keep track of simulation time
- Results — summarize the results of your simulator by comparing the average wait time and average turnaround time for each of the three algorithms given 1000 processes and using 5 different random seeds. Succinctly show these results graphically using bar graphs.

Provided Files, Details, and Constraints

The file, *sched_compare.c*, contains a stubbed-out main() that you must use as described in the comments. Any functions you require for your solution can easily be implemented within this file.
In addition, the following files are required by the stubbed-out main(). **Don't** modify these files:

- **poisson.c** and **poisson.h**: These files are used by *procs.c* to generate an array of random process arrival times and service times. If you are curious about the Poisson distribution, google it!
- **procs.c** and **procs.h**: These files contain routines to randomly generate process arrival times and service times, read a list of arrival times and service times from a file, and print an array of process arrival times and service times. See comments in the main() for how to call them.
- **procheap.c** and **procheap.h**: These files are *optional* in your solution, but recommended. It is useful to be able to sort arriving process according to some comparison criterion. For example, with *SPN*, you will need to choose the minimal service time process among a set of processes that arrive within an interval of time. A heap (priority queue) can be useful in making this determination.

  In order to use the heap, you will have to initialize it with a *size* and value function, value_func. An example value function might be something like this:

```
double myvalue(proc_t *proc)
{
   return proc->arrival_time;
}
```

The process structure, *proc_t*, is defined in *procs.h*. Processes inserted into the heap initialized with the value function above, will be prioritized (sorted) based on minimum process arrival time.

Collaboration and Plagiarism

This is an **team assignment**. Collaboration **is only permitted** between team members on the same team. Plagiarism will not be tolerated. Submitted solutions that are very similar (determined by the instructor) will be given a grade of zero. Please do your own work, and everything will be OK.

Submission

Create a compressed tarball, i.e. *tar.gz*, that includes your report in PDF format, the Makefile and **all** the .h and .c files needed to compile and execute your program.
The name of the compressed tarball **must only be the last name of one team member**. For example, *ritchie.tar.gz* would be correct if the original co-developers of UNIX (Dennis Ritchie and Ken Thompson) submitted the assignment. Only assignments submitted in the correct format will be accepted (no exceptions). **Only one designated team member** will submit the compressed tarball to the Dropbox project folder on OAKS. Your team may resubmit the compressed tarball as needed. I will only use the most recent submission.
Late assignments will not be accepted. Exceptions will only be made for extenuating circumstances, i.e. death in the family, health related problems, etc. You will be given two weeks to complete this assignment. Poor time management is not excuse. Do not email the assignment after the due date, it will not be accepted.
Please feel free to setup an appointment to discuss this project. I am more than happy to listen to your approach and make suggestions. However, I cannot tell you how to code the solution. Additionally, debugging is your job. You may use the debugger (gdb) or print statements to help understand why your solution is not working correctly.

Grading Rubric

For this assignment the grading rubric is provided in the table shown below.

program compiles	10 points
program runs with no errors	10 points
report	25 points
implementation of FCFS algorithm	15 points
implementation of SPN algorithm	15 points
implementation of HRRN algorithm	15 points
implementation of the main function	10 points

Of course, if your program does not compile, you will miss points for many of the other items.