CECS 326
Project 4: CPU Scheduling

Mark Garcia

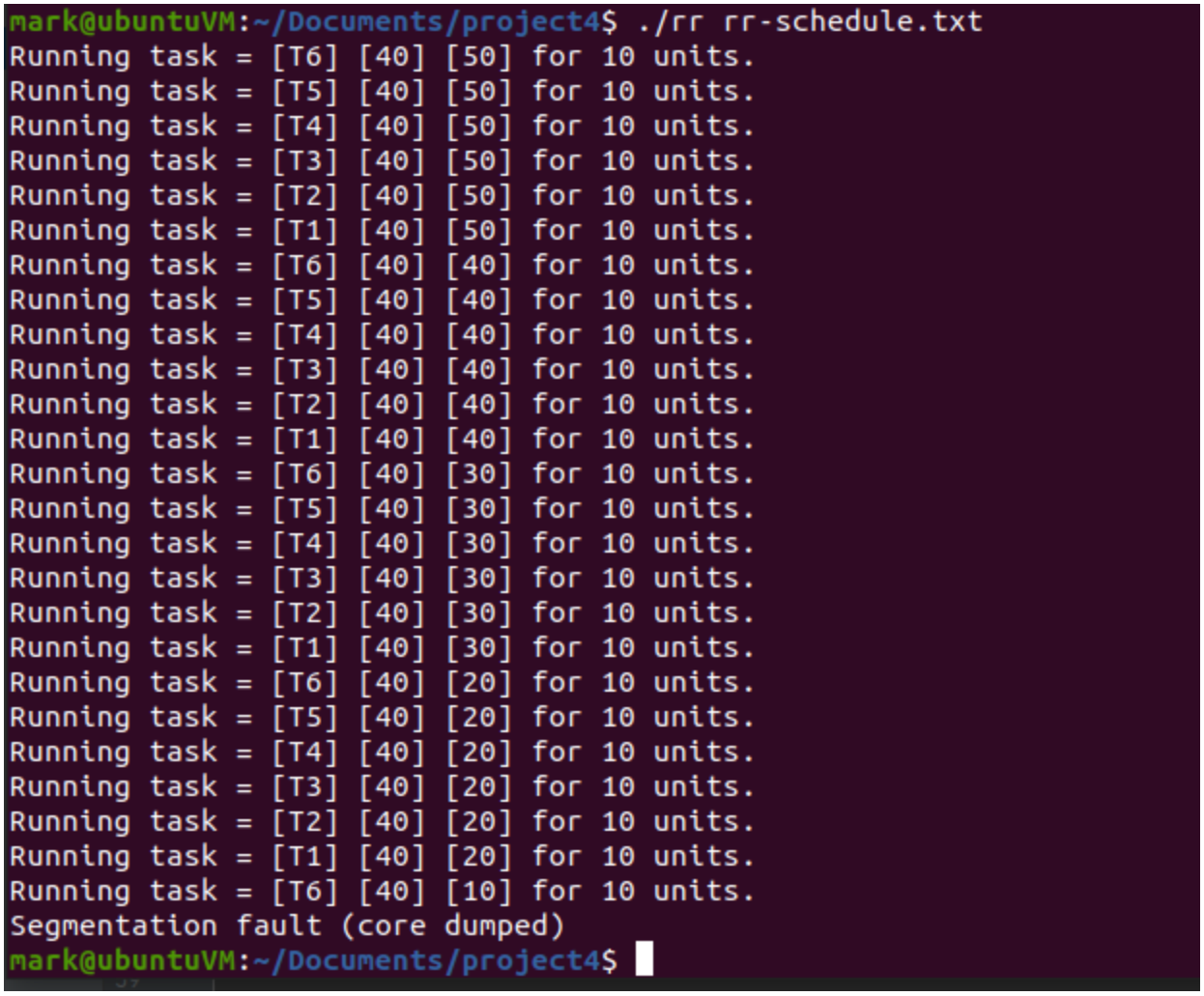
Intro:

The purpose of this project is to demonstrate different methods for CPU scheduling of tasks. The First-Come-First-Serve method schedules tasks base on the order in which the CPU requests them. The first ones get executed first followed by other tasks in that order. The Priority method schedules tasks based on their priority with higher priority numbers being executed first. In the scope of the project, priorities were represented as integers from 1 to 10 where 10 holds the greatest priority and 1 holds the lowest priority. For the Round-Robin scheduling method, tasks were run based on their burst time compared to time quantum. In the scope of this project, time quantum was a constant of 10 ms. If their burst time was greater than time quantum then we reduced the tasks burst time by quantum (10 ms).

Results:

```
mark@ubuntuVM:~/Documents/project4$ ./fcfs schedule.txt
Running task = [T1] [4] [20] for 20 units.
Running task = [T2] [3] [25] for 25 units.
Running task = [T3] [3] [25] for 25 units.
Running task = [T4] [5] [15] for 15 units.
Running task = [T5] [5] [20] for 20 units.
Running task = [T6] [1] [10] for 10 units.
Running task = [T7] [3] [30] for 30 units.
Running task = [T8] [10] [25] for 25 units.
mark@ubuntuVM:~/Documents/project4$ make priority
gcc -Wall -c schedule_priority.c
gcc -Wall -o priority driver.o schedule_priority.o list.o CPU.o
mark@ubuntuVM:~/Documents/project4$ ./priority pri-schedule.txt
Running task = [T6] [1] [50] for 50 units.
Running task = [T5] [1] [50] for 50 units.
Running task = [T4] [1] [50] for 50 units.
Running task = [T3] [1] [50] for 50 units.
Running task = [T2] [1] [50] for 50 units.
Running task = [T1] [1] [50] for 50 units.
```

Output for FCFS and Priority scheduling:



Output for RR scheduling.

Conclusion:

In conclusion, I was able to successfully demonstrate FCFS and Priority scheduling. RR scheduling resulted in a segmentation fault that I was not able to fix. In my recording for the project, I thought that it ran properly so i did not address the issue there. I believe that this issue stems from the part of the code after I ran the last task. I had a node that pointed to the next node in the queue but I did not take into consideration if I was at the last node in the list. I believe that the segfault error occurs because I am pointing to some arbitrary location in memory and trying to run whatever is there.