**Csci 432 Final Project Outline**

**Authors:**

**Devan W. Eastman-Pittam, Ben Miller**

**Problem:**

Given a data set of projects with priorities and times, compare different CPU scheduling algorithms steps/time taken to perform a full completion scheduling

**Abstract:**

**Hypothesis:**

For our hypothesis we believe that the SRT alg will have the lowest task completion time for short individual tasks but long tasks will have the highest time taken overall, the RR alg will have the longest overall task completion time for all total tasks but by its nature will complete all short tasks then medium tasks and lastly all long tasks in order. And finally the PB alg will have no particular fast completion time based on task time remaining but will complete high priority tasks faster than low priority tasks.

### **Alg1 Choice:** Shortest Remaining Time (Implementation)

**Summary of Algorithm 1:**

The Shortest Remaining Time scheduling algorithm always executes the job with the shortest remaining time. This requires some amount of knowledge about the processes involved as it is not always easy to estimate the amount of time remaining on a task. Because of this restriction, this algorithm is not used very often at the operating system level. It requires a supervisor to handle the remaining times. A great situation for this scheduling algorithm would be a web server where most of the requests are processed very quickly and sparsely, leaving time for longer requests to be processed without slowing down the faster requests.

**Pro**: Prioritizes tasks with low remaining times, so fast jobs finish quickly. Not very computationally hard to calculate the next job.

**Con**: Longer tasks are only worked on when there are no shorter tasks, so longer jobs can take a very long time to complete. Requires a way to calculate the estimated length.

### **Alg2 Choice:** Round-Robin

**Summary of Algorithm 2:**

The Round-Robin scheduling algorithm is a fairly easy to implement algorithm as it doesn’t need to know anything about any of the tasks. Each task is placed in a queue, worked on, then placed back in the queue. Completion times with algorithms depend only on the number of active tasks. If this algorithm was used in a web server, all requests would be slowed down equally if there was a surge.

**Pro**: No priorities are taken into account, so all tasks are worked on equally. The execution time is relatively stable for each type of task and is determined by the total number of active tasks.

**Con**: Significantly raises individual task time taken to completion, the least amount of control on order of completion of tasks among all 3 algorithms.

### **Alg3 Choice:** Priority Based

**Summary of Algorithm 3:**

The priority based task completion algorithm functions by assigning an integer value as a variable of each task which ranges from 0 (highest priority) to inf (lowest priority). And having the program each time unit (seconds or time units) evaluate if there exists a higher priority task which will override the current one, and repeat until all tasks are evaluated. The program will evaluate in time blocks of 1 second so if time needed to complete a task is 6.66 seconds the program will take 7 seconds to evaluate.

**Pro**: This method can quickly implement the tasks deemed the most urgent and prevent their delay in the most efficient method possible, affords more control to the judgement of which tasks should be done in which order.

**Con**: May cause low time remaining tasks to indefinitely delay completion, If you have many jobs of different priority the program might be delayed due to time taken to switch to the different tasks.

### **Metrics:**

Average task completion time. Average long task completion time. Average short task completion time, Total Steps Taken.

**Code Implementation:**

**Results:**

**Conclusion:**

**References**

*Os priority scheduling - javatpoint*. www.javatpoint.com. (n.d.). Retrieved November 23, 2021, from https://www.javatpoint.com/os-priority-scheduling.

*Os Round Robin Scheduling Algorithm - javatpoint*. www.javatpoint.com. (n.d.). Retrieved November 23, 2021, from https://www.javatpoint.com/os-round-robin-scheduling-algorithm.

*OS SRTF scheduling algorithm - javatpoint*. www.javatpoint.com. (n.d.). Retrieved November 23, 2021, from https://www.javatpoint.com/os-srtf-scheduling-algorithm.

Williams, L. (2021, October 6). *CPU scheduling algorithms in operating systems*. Guru99. Retrieved November 23, 2021, from https://www.guru99.com/cpu-scheduling-algorithms.html.

UKEssays. (November 2018). A Comparison of CPU Scheduling. Retrieved from https://www.ukessays.com/essays/computer-science/a-comparison-of-cpu-scheduling.php?vref=1

Prof. Fasy notes:

I took a quick look at your google doc and it looks like the problem you describe is a bit underspecified. In class, the objective was to find a schedule that had the shortest duration. What is the objective of the schedules that you want to consider?

The algorithms you investigate, while they don't have to be "new", need to be beyond the scope of what is covered in lectures/the course textbook. (Some can be covered in the course, but not all algorithms considered).

I’m thinking we could choose a separate metric to examine based of her feedback maybe average waiting time to completion among different algorithms or the turnaround time, here is a resource:

<https://www.notesjam.com/2018/11/scheduling-algorithms-examples.html?m=1>

<https://code-with-me.jetbrains.com/K59k7ZxWm6_Iol3usitVrg#p=IC&fp=31508E632F7153CA84A9E2E30CB4B29891AB10FDC4AE529C29FD379C94A59AD8>