CS3104 (Operating Systems) Report
Practical 2

Matriculation ID: 210021783

Overview

In this practical we were tasked with implementing a process schedular for … and then editing the python script, analyse.py, to help us analyse the performance of our schedular in various scenarios. My understanding of this was that the focus of this practical should be on the analysis side of things rather than the implementation of the schedular, hence why I spent the majority of my time focusing on that. This can also be seen as to why there is little in my design and implementation section and no testing but rather an analysis and discussion section in this report.

In my implementation, I successfully completed requirement 1 and 2.

Design

In this we were given the header file of which we were supposed to implement so there were very few design choices to actually be made.

I chose to implement a round-robin (RR) with priorities, as like aforementioned, my understanding of this practical is not to choose and implement the best algorithm but rather to evaluate the performance and discuss the advantages and disadvantages of one. I chose this one because I found it to be the easiest to discuss in terms of having both good and bad consequences that comes with it.

Implementation

Scheduler.c

In this class, I made use of all the methods we were given that I felt necessary so some of them, such as *init_scheduler() and* task_ended() were unused and left empty. This was decided after I was implementing everything and testing things very gradually and realised I did not actually need them in my implementation.

I did however decide to create a helper method called *age_tasks()*. This is used to increase the priority of a task after its waiting time. I did this to reduce the amount of context switching my schedular had, this is because less context switching also reduces the amount of overhead, because each switch takes a few microseconds which does add up) which in turn increases the efficiency of the CPU by keeping it wasting time switching between processes.

Analysis

Prelude

To analyse my program, I created a few scenarios outside of the two we were given, basic.c and varied.c, which emulate various different situations that a CPU can be placed in. Each one will be analysed through 6 different graphs and a short summary of the scheduler's performance. These graphs, labeled 1 to 6 in order from top left, top right, middle left, middle right, bottom left and bottom right. Each one clearly analyses a specific field of performance which is listed later.

Prior to performing these scenarios, I expected my scheduler to balance priority handling and to still give preference to higher priority tasks, while ensuring lower priority tasks don't suffer from starvation. And for I expect for the tasks with the higher priority to always finish first, which you see. I also knew it would be a bit more complex than a simple round-robin or priority scheduler, but this would be the cost for a bit more flexibility, since it would be able to balance the need for responsiveness (need to complete high-priority tasks) and flexibility.

Scenario: Basic

Synopsis

This scenario tests the responsiveness and efficiency of my scheduler at a very basic level by utilizing only two processes. This was useful as it was quick to run and complete, and could show how the scheduler handles both CPU-intensive and time-sensitive tasks. The 'long_running' process simulated a compute-heavy task, allowing us to observe how the scheduler manages prolonged CPU usage, while the 'short_running' process provided insights into the system's ability to promptly allocate resources to newly ready tasks. This basic scenario was crucial for initial evaluations, as it laid the groundwork for identifying key performance characteristics such as the scheduler's throughput, fairness in time slice distribution, and overall CPU utilization. It also served as a straightforward benchmark against which more complex scenarios and additional features of the scheduler could be compared, ensuring that any modifications or optimizations could be assessed for their impact on these fundamental aspects of scheduler performance.

Results

Overall Mean Response Time	1.72ms
Overall Mean Run Time	8.01ms
Overall Mean Turnaround	9.73ms
Overall Mean Throughput	102.8 Processes per Second
CPU Utilization	82.33%
Idle Time	17.68%

Context Switches	40

Scenario: Varied

Synopsis

This scenario, that we were also given, is designed to simulate various types of process behaviors. It includes a "batch" process for testing the scheduler's handling of CPU-bound, long-running tasks; an "interactive" process that alternates between sleep and active states to assess responsiveness to user-interactive tasks; a "short_sleep" process, which frequently blocks and unblocks, testing the scheduler's efficiency with frequent context switches; and a "burst" process that mixes bursts of CPU activity with periods of inactivity, reflecting common real-world application patterns. Initiated by an "init" process that spawns these various tasks, this scenario comprehensively evaluates the scheduler's ability to effectively manage and balance different workloads, its responsiveness, CPU utilization efficiency, and its capability to maintain fairness and task prioritization, particularly in handling mixed workloads and fluctuating CPU demands.

Results

Overall Mean Response Time	2.4ms
Overall Mean Run Time	4.7ms
Overall Mean Turnaround	7.12ms
Overall Mean Throughput	140.5 Processes per Second
CPU Utilization	66.68%
Idle Time	33.32%
Context Switches	165

Scenario: prioScenario

Synopsis: This test is particularly useful for observing the effectiveness of the scheduler's priority mechanism. It should demonstrate whether higher-priority processes (like "*high_priority_process*") are given preferential treatment (i.e., allocated CPU time sooner and possibly more frequently) compared to lower-priority ones. It is very short compared to the others, but I had some trouble trying to increase the workload for some unknown reason and could not figure out why.

Results

Overall Mean Response Time	4.2ms
Overall Mean Run Time	2.4ms
Overall Mean Turnaround	6.5ms
Overall Mean Throughput	152 Processes per Second
CPU Utilization	36%
Idle Time	64%
Context Switches	57

Analysis

1. Task Starvation:

- **False Indications of Starvation:** My scheduler might be prone to starving some tasks, particularly evident from the high CPU idle times (around 63.49% in one scenario) and relatively low CPU utilization (36.51%). This suggests that the scheduler might be waiting excessively for high-priority tasks while leaving the CPU idle, potentially leading to lower-priority tasks being starved of CPU time. However, when you see the graphs attached, you see that this is not the case by any means and that my scheduler does switch between tasks frequently. This is still a much better alternative than others such as priority scheduling without round robin because that just goes by priority and causes serious task starvation.

2. Balancing Interactive and Batch Tasks:

- **Response and Turnaround Times:** The mean response and turnaround times (around 4.08ms and 6.43ms, respectively, in one scenario) suggest the scheduler is moderately effective in handling task switching. However, these figures also imply that interactive tasks might not be receiving immediate attention, potentially affecting the perceived responsiveness of the system.
- **Throughput:** A high throughput (155.41 processes per second) indicates the scheduler's efficiency in handling a large volume of tasks but doesn't necessarily reflect on how well it's balancing interactive versus batch tasks.

3. Pathological Scenarios:

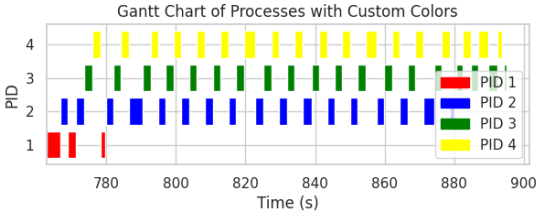
- **Heavy I/O-Bound Tasks:** Given the high idle times, the scheduler might struggle with efficiently managing I/O-bound tasks, potentially leading to increased waiting times and underutilization of the CPU.

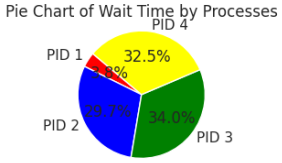
Evaluation

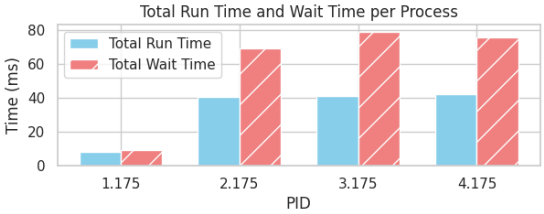
In evaluation my program is fit for purpose. And very robust, I have not been able to find clear issues that cause it to crash. I believe the report is fairly light but covers the key points I planned for it to. However, really could have been more in depth and would have liked to reference the reading more, as that's where my understanding of this came from.

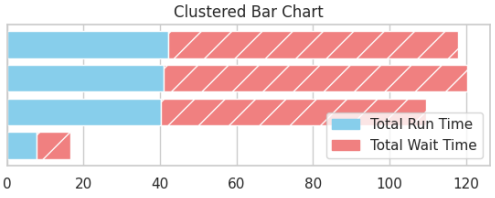
Conclusion

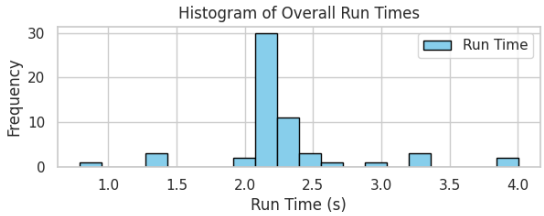
In conclusion, I am fairly happy with my submission as completed. If not for time constraints, I would have liked to go deeper in depth with each analysis/evaluation. I would have also liked to display some graphs here; I do convert them to CSV files but just don't have time to create graphs that convey what I want. I have also just realized that I planned to speak about each graph in particular but did not have time to, will still attach even though just dropping and image/data is not analysis.

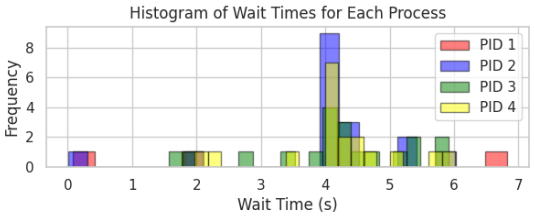

Gantt Chart of Processes with Custom Colors
PID
Time (s)
PID 1
PID 2
PID 3
PID 4

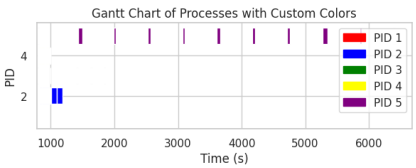

Pie Chart of Wait Time by Processes
PID 1
PID 2
PID 3
PID 4
32.5%
29.7%
34.0%

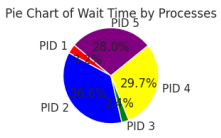

Total Run Time and Wait Time per Process
Time (ms)
PID
Total Run Time
Total Wait Time

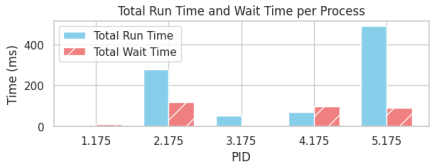

Clustered Bar Chart
Total Run Time
Total Wait Time

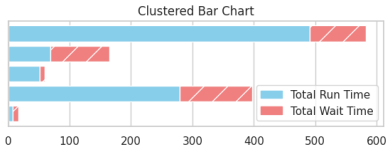

Histogram of Overall Run Times
Frequency
Run Time (s)
Run Time

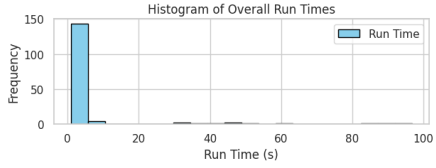

Histogram of Wait Times for Each Process
Frequency
Wait Time (s)
PID 1
PID 2
PID 3
PID 4

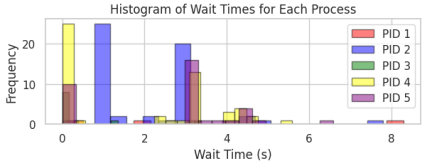

Gantt Chart of Processes with Custom Colors
PID
Time (s)
PID 1
PID 2
PID 3
PID 4
PID 5

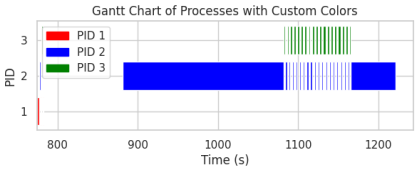

Pie Chart of Wait Time by Processes
PID 1
PID 2
PID 3
PID 4
PID 5
28.0%
29.7%

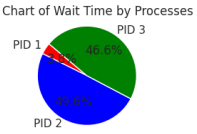

Total Run Time and Wait Time per Process
Time (ms)
PID
Total Run Time
Total Wait Time

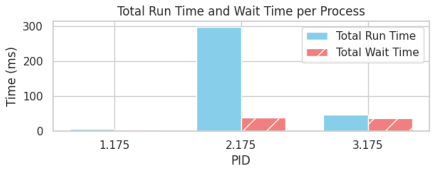

Clustered Bar Chart
Total Run Time
Total Wait Time

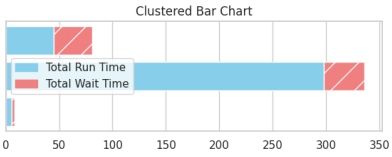

Histogram of Overall Run Times
Frequency
Run Time (s)
Run Time

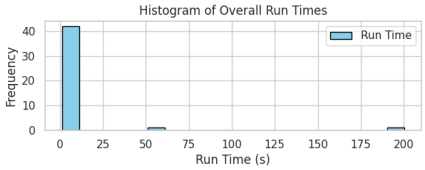

Histogram of Wait Times for Each Process
Frequency
Wait Time (s)
PID 1
PID 2
PID 3
PID 4
PID 5

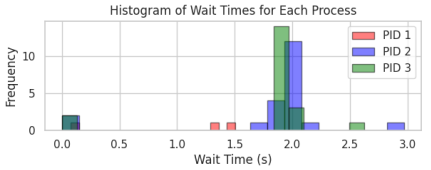

Gantt Chart of Processes with Custom Colors
PID 1
PID 2
PID 3
PID
Time (s)


Pie Chart of Wait Time by Processes
PID 1
PID 3
46.6%
49.6%
PID 2


Total Run Time and Wait Time per Process
Total Run Time
Total Wait Time
Time (ms)
PID


Clustered Bar Chart
Total Run Time
Total Wait Time


Histogram of Overall Run Times
Run Time
Frequency
Run Time (s)


Histogram of Wait Times for Each Process
PID 1
PID 2
PID 3
Frequency
Wait Time (s)