Scheduler Report

Team 10 (CIE 302)
Aml Tarek Ismail 202200126
Youssef Mohamed Allam 202200286
Mohammad Mahmoud Abd Elrahman 202200438

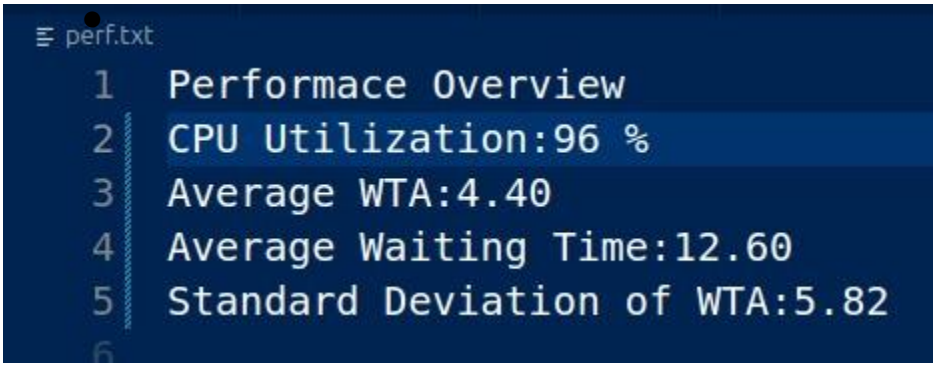
Data Structures

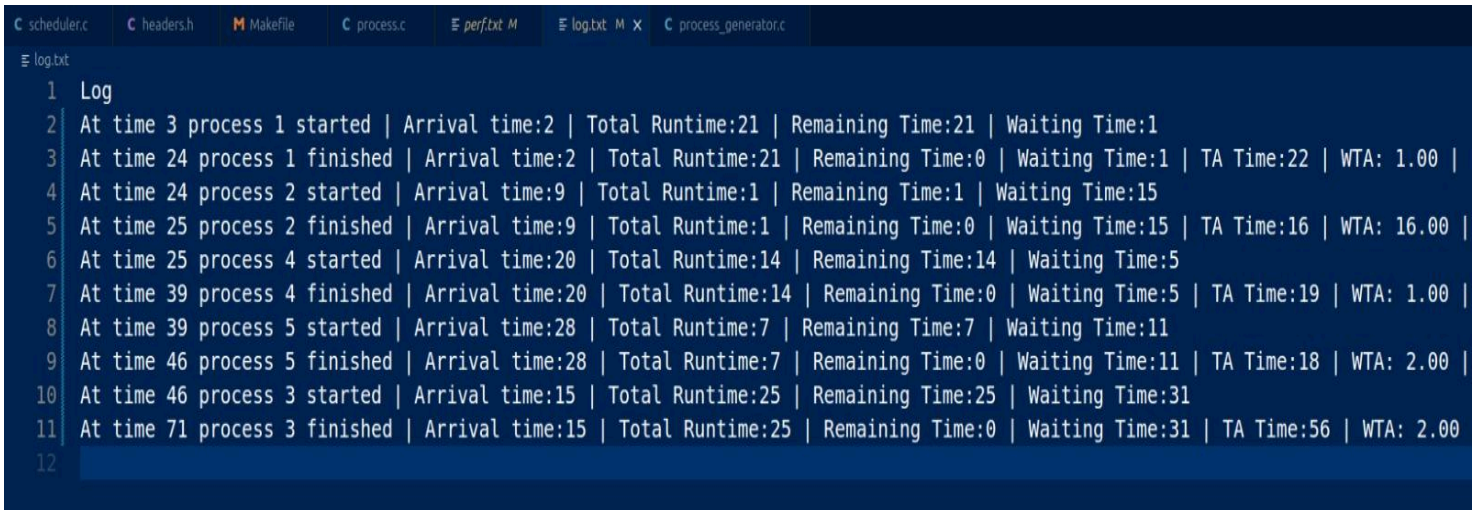
1)Priority Queue: We make it to store processes for inter-process communication, enabling the scheduler to receive process requests in an orderly way. Each process is assigned a priority level, and the scheduler retrieves processes based on these priorities, selecting the highest priority task available.We used it also in SJF and PHPF.

2)Circular Queue: allows us to manage the processes in a circular way, where each process is allocated a fixed time slice (quantum) for execution in the Round Robin Algorithm. Once a process's time slice expires, it is moved to the back of the queue, and the next process in line is given CPU access.

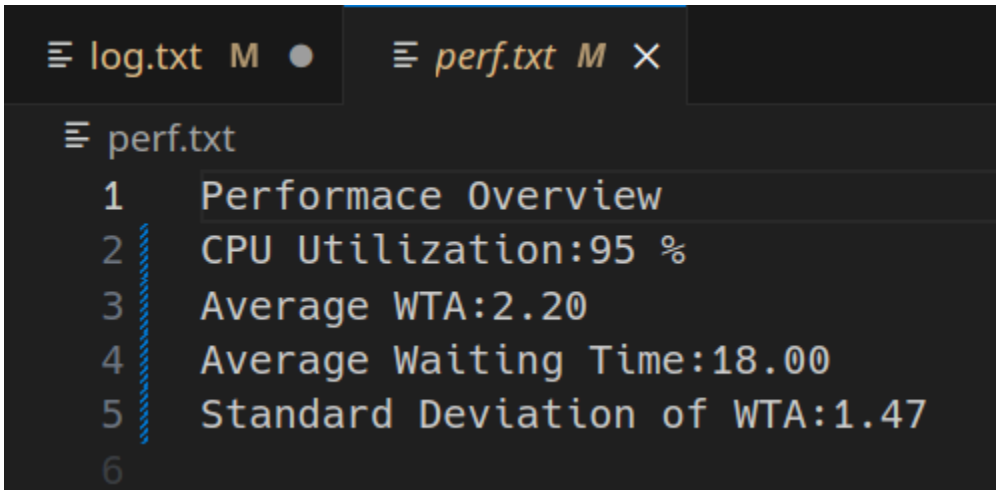
Algorithms and Explanation

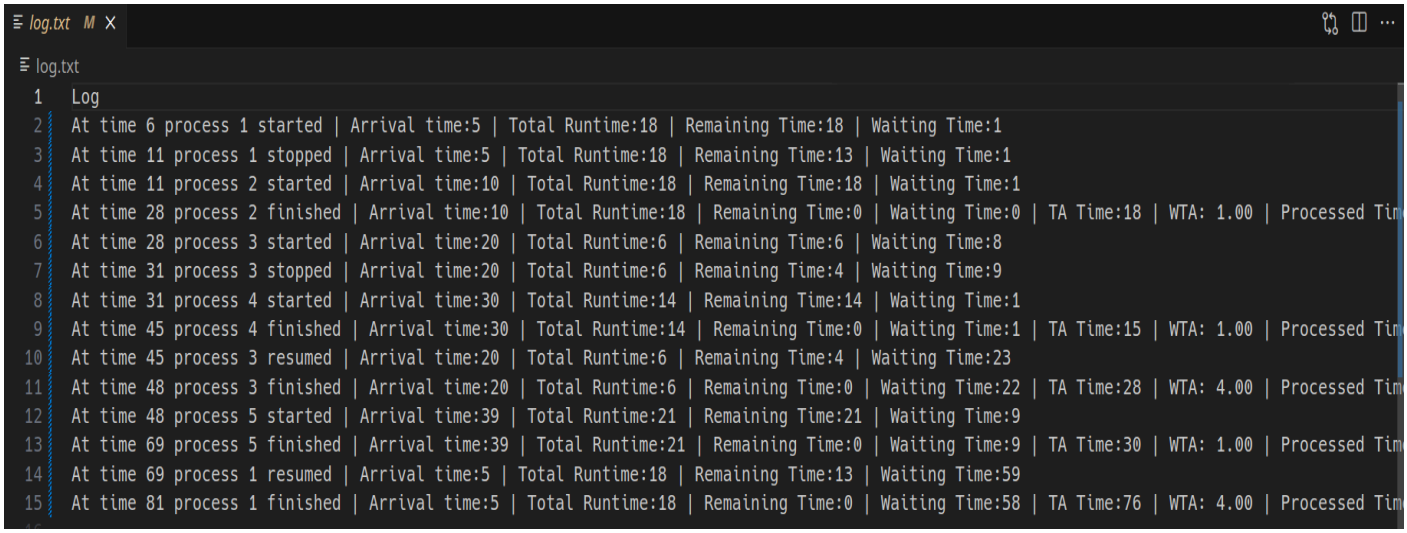
1)Short Job First (SJF): We implement using a priority queue, where processes are prioritized based on their execution time. In the code, when SJF is activated, it continuously checks for new messages from a process generator using message requests.Upon receiving a message, it creates a new process and either starts it immediately if no other processes are running or enqueues it in the priority queue based on its execution time. This approach ensures that the process with the shortest duration is always executed next, optimizing overall wait times and improving system efficiency.



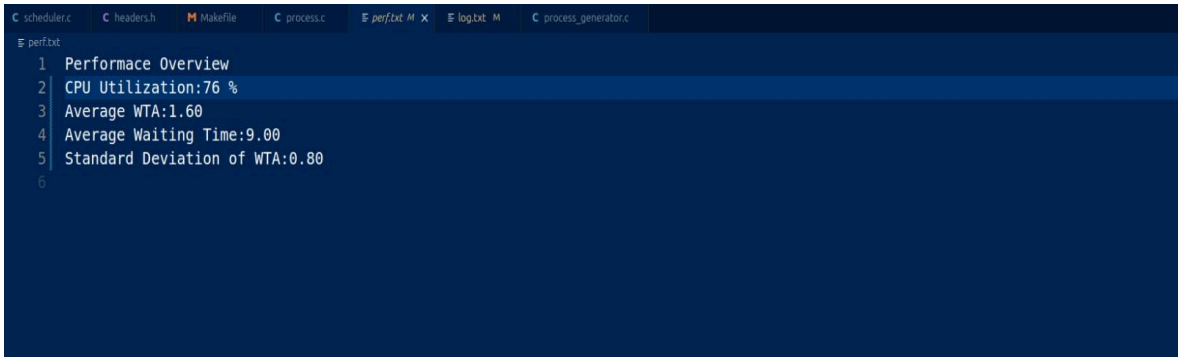


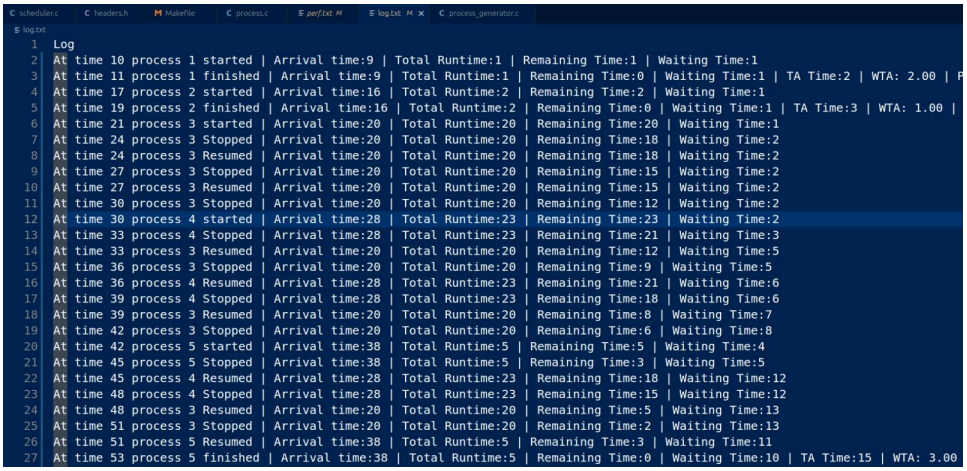
2)Preemptive Highest Priority First (PHPF): We implement a priority queue to manage processes based on their priority levels. In the code, when PHPF is activated, it continuously checks for new messages from a process generator while ensuring that processes are either running or queued. Upon receiving a message, it creates a new process and, if the queue is empty, starts it immediately; otherwise, it preempts the currently running process, queues it, and then selects the highest priority process to run next. This approach allows for dynamic scheduling, ensuring that the process with the highest priority is always executed, thereby optimizing responsiveness and performance.

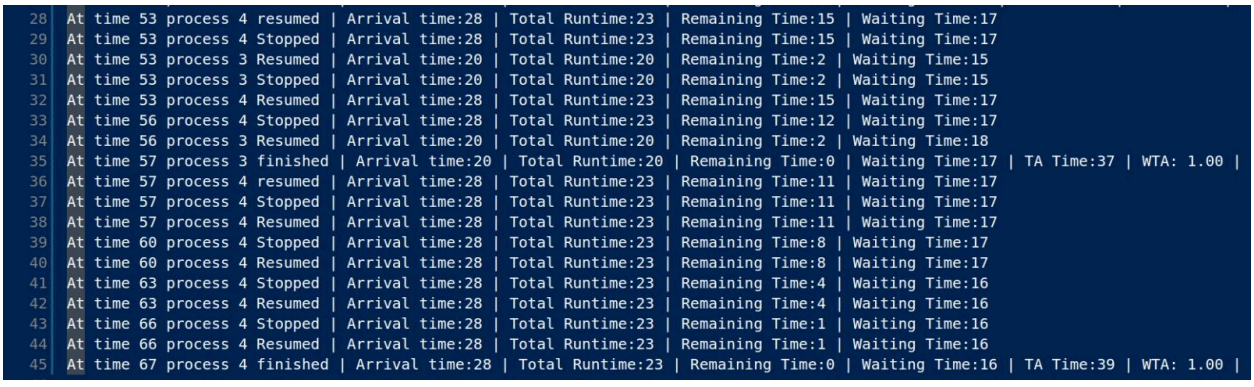




3)Round Robin (RR): The Round Robin (RR) scheduling algorithm allocates CPU time slices to processes in a cyclic manner, ensuring fair resource distribution. When activated, it reads a time quantum and continuously checks for new processes, adding them to a circular queue while managing the currently running process by stopping it after ending the quantum and re-queueing it. The algorithm dequeues the next process to run for the specified quantum, allowing for efficient time-sharing and enabling processes to continue execution or be re-enqueued as needed.







Assumptions

1)Short Job First (SJF)

- If two or more processes arrive with the same execution time, we use first-come, first-served (FCFS) order.
-

2)Preemptive Highest Priority First (PHPF)

- If two or more processes arrive with the same priority, we use first-come, first-served (FCFS) order.

3)Round Robin (RR)

- If two processes arrive at the same time, we will run the lowest id process first.

Memory Allocation (Phase 2)

The memory allocation system divides memory into blocks of sizes that are powers of two. It allocates the smallest available block for each process and merges freed blocks for reuse later.

processes.txt

```
#id arrival runtime priority memsize
1   6   28  0   83
2   14  20  6   12
3   18  16  4   15
4   20  4   3   14
5   21  17  5   35
```

memory.txt

```
Memory
At time 7 process 1 allocated a memory of 128 from 0 to 128
At time 15 process 2 allocated a memory of 16 from 128 to 144
At time 19 process 3 allocated a memory of 16 from 144 to 160
At time 21 process 4 allocated a memory of 16 from 160 to 176
At time 22 process 5 allocated a memory of 64 from 192 to 256
At time 35 process 1 freed a memory of 128 from 0 to 128
At time 39 process 4 freed a memory of 16 from 160 to 176
At time 55 process 3 freed a memory of 16 from 144 to 160
At time 72 process 5 freed a memory of 64 from 192 to 256
At time 92 process 2 freed a memory of 16 from 128 to 144
```

Work Load Distribution

We do not focus on workload distribution for this project, as it is part of sensitive projects where each step is dependent on the successful completion of the previous one. Therefore, we are working on each step collaboratively to ensure seamless integration and execution.

Tasks Management

File	Time taken	Status
Process generator	**12.5 hours, we spend a lot of time here to understand how the project works.**	**Ended**
SJF	**2.5 hours.**	**Ended**
PHPF	**1.5 hours.**	**Ended**
RR	**9 hours, we spent a lot of time here because the algorithms had many special cases that needed to be resolved.**	**Ended**
output/input	**3.5 hours.**	**Ended**
Statistics/clearing resources	**12 hours.**	**Ended**
Memory Allocation	**3 hours.**	**Ended**