International Islamic University Chittagong

Department of Computer Science and Engineering

Visual Operating System Scheduler

Course Code: CSE-3632
Course Name: Operating System Lab

Submitted By:
Lamisa Mashiat (C201249)
Sohana Tasneem (C201266)
Sadia Haque Chowdhury (C201270)
Tahura Tasmin Chowdhury (C201272)
Nusrat Jahan (C201261)

Submitted To:
Shefayatuj Johara Chowdhury
Assistant Lecturer
Dept. of CSE, IIUC

***Date of Submission**: June 14, 2023*

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 Problem Statement:

The comprehending process scheduling algorithms, which are vital for the functionality of our system, use some really complicated algorithms. Users find it challenging to understand the complexities of these algorithms because they are frequently obscured and complicated. Users' abilities to optimize system resources and solve scheduling problems are hampered by this ignorance. To address this, our project intends to create a visualizer that makes process scheduling algorithms easier to comprehend, empowering users to make knowledgeable decisions and improve system efficiency through logical visual representations and interactive features

1.2 Proposed Solution:

The Visual-OS-Scheduler project aims to enhance understanding and analysis of process scheduling algorithms through visualizations. It facilitates effective comparison of different algorithms, identifies optimization opportunities through visual analysis of scheduling graphs, and provides a user-friendly interface for intuitive exploration and interaction. By visualizing the complexities of scheduling algorithms, the project empowers users to make informed decisions, optimize system performance, and gain a deeper understanding of the impact of scheduling decisions.

1.3 Objectives:

- ✓ Enhance understanding of process scheduling algorithms through visualization.
- ✓ Facilitate effective comparison of different scheduling algorithms.
- ✓ Identify optimization opportunities based on visual analysis of scheduling graphs.
- ✓ Provide a user-friendly interface for intuitive exploration and interaction with the scheduling algorithms.

CHAPTER 2 ALGORITHMS

Implemented Scheduling Algorithms: *Non-Preemptive Highest Priority First (HPF), First Come First Served (FCFS), Round Robin (RR) with fixed time quantum, and Preemptive Shortest Remaining Time Next (SRTN).*

2.1 Round Robin (RR)

Round Robin (RR) is a popular process scheduling algorithm used in operating systems. It is designed to provide fair and balanced CPU time allocation among multiple processes. The RR algorithm works by assigning each process a fixed time quantum, allowing it to execute for that duration before moving to the next process in the queue. This preemptive approach ensures that no single process monopolizes the CPU for an extended period, promoting fairness. With its simple and straightforward nature, the RR algorithm is widely utilized in scenarios where time-sharing and equal distribution of resources are crucial, making it an essential component of modern operating systems.

2.2 Highest Priority First (HPF)

Non-Preemptive Highest Priority First (HPF) is a process scheduling algorithm widely used in operating systems. In HPF, each process is assigned a priority value, and the process with the highest priority is selected for execution first. Unlike preemptive algorithms, HPF allows a process to run until it voluntarily relinquishes the CPU, such as when it completes its execution or enters a waiting state. This non-preemptive nature ensures that higher-priority processes receive immediate attention, potentially leading to better response times for critical tasks. HPF is suitable for scenarios where prioritizing important processes is vital, such as real-time systems or time-sensitive applications.

2.3 First Come First Served (FCFS)

First Come First Served (FCFS) is a basic process scheduling algorithm commonly used in operating systems. In FCFS, processes are executed in the order they arrive in the system's ready queue. The first process that arrives is the first to be executed, followed by the subsequent processes in the order of their arrival. This non-preemptive algorithm operates on a simple principle of fairness, as it ensures that processes are served in the same sequence, they entered the system. FCFS is easy to understand and implement but may suffer from the "convoy effect," where a long-running process delays the execution of other processes waiting in the queue. It is suitable for scenarios without strict performance requirements or time-sensitive tasks.

2.4 Shortest Remaining Time Next (SRTN)

Shortest Remaining Time Next (SRTN) is a preemptive process scheduling algorithm utilized in operating systems. In SRTN, the process with the smallest remaining execution time is selected for execution. If a new process with an even smaller execution time arrives, the currently running process is preempted, and the new process takes over. This dynamic nature ensures that processes with shorter execution times are prioritized, reducing overall waiting time and improving system responsiveness. SRTF is effective in scenarios where process execution times vary significantly, as it optimizes CPU utilization and minimizes turnaround time. However, the frequent context switches associated with preemption can introduce some overhead.

CHAPTER 3
SYSTEM REQUIREMENTS AND ANALYSIS

3.1 Language & Tools

Programming Language: Python

IDE: Pycharm

Required Libraries:

⇨ **Matplotlib -** Matplotlib is used for generating plots and computing the scheduling results and comparing the performance. Matplotlib is a low level graph plotting library in python that serves as a visualization utility. It is open source and we can use it freely. Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

⇨ **Tkinter -** Tkinter is used in the project for creating the graphical user interface (GUI) to visualize the process scheduling algorithms. Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and macOS installs of Python. The name Tkinter comes from Tk interface.

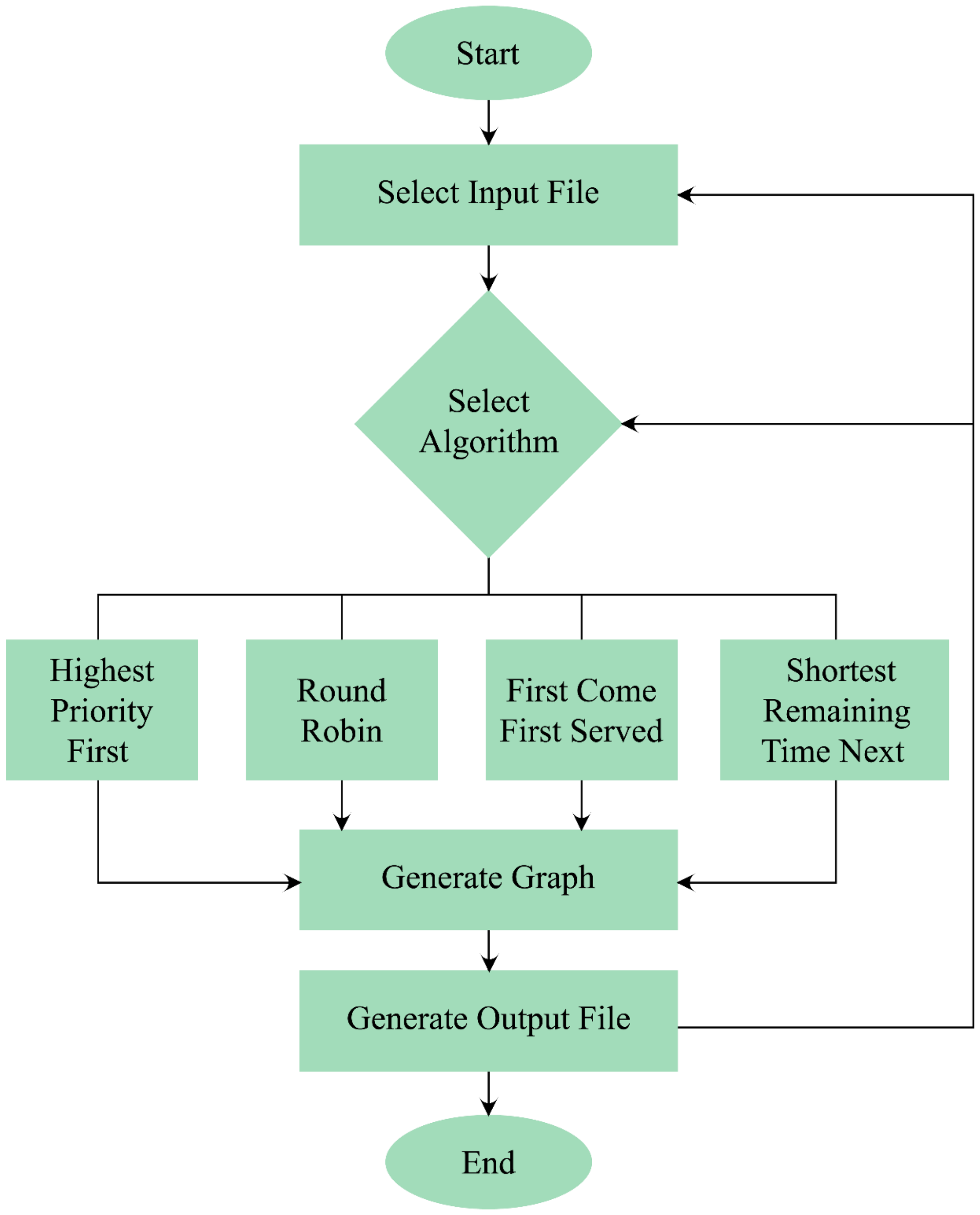
⇨ **Numpy -** NumPy is used in the project to handle numerical operations and data manipulation. It is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

3.2 Input/Output Format

Input will be taken as text file. The first line contains a single integer n- the number of processes. Next n lines contains P1, A1, R1, P1 the process number, arrival time, running time and priority respectively.

CHAPTER 4
SYSTEM DESIGN & MODELING

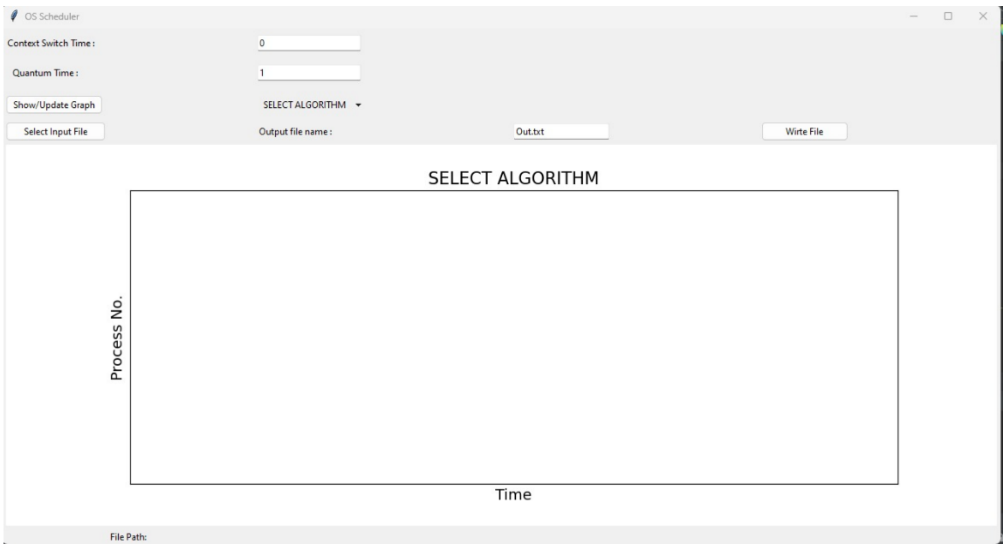
4.1 Process Flowchart



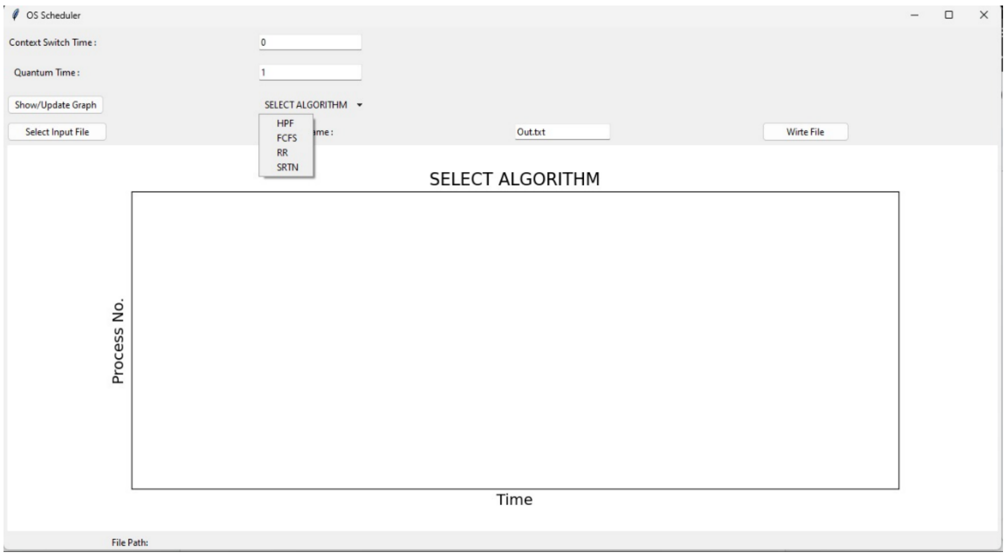
CHAPTER 5
INTERFACE & FUNCTIONALITIES

5.1 Interfaces of the System

⇨ **Homepage:**


Context Switch Time : 0
Quantum Time : 1
Output file name : Out.txt

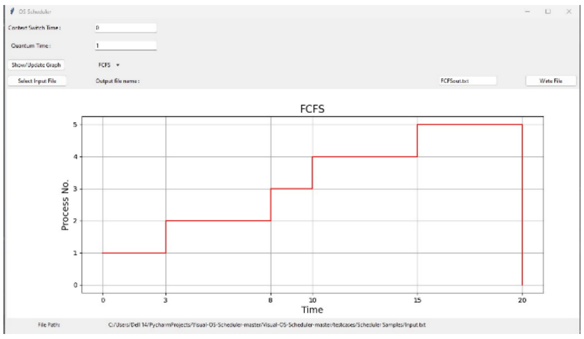
⇨ **Input:** Through a simple GUI the user should be able to do the following:
- Enter the input file name
- Choose one of the implemented scheduling algorithms torun.
- Specify the “Context Switching”time.
- Specify the “Time Quantum” in case of choosing RoundRobin.


Context Switch Time : 0
Quantum Time : 1

⇨ **Output:** For the given input, outputs for four different Algorithms are given below

First Come First Served:

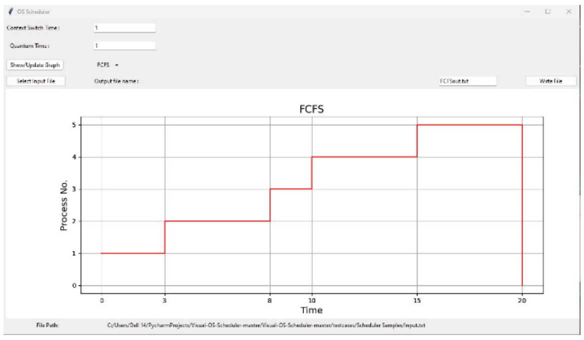
The First-Come-First-Served (FCFS) graph interface displays the sequential execution order of processes based on their arrival time, providing a visual representation of the First Come First Served scheduling algorithm.



Context Switch Time : 0
Quantum Time : 1
FCFS
Output file name : FCFSout.txt

First Come First Served Context Switching:

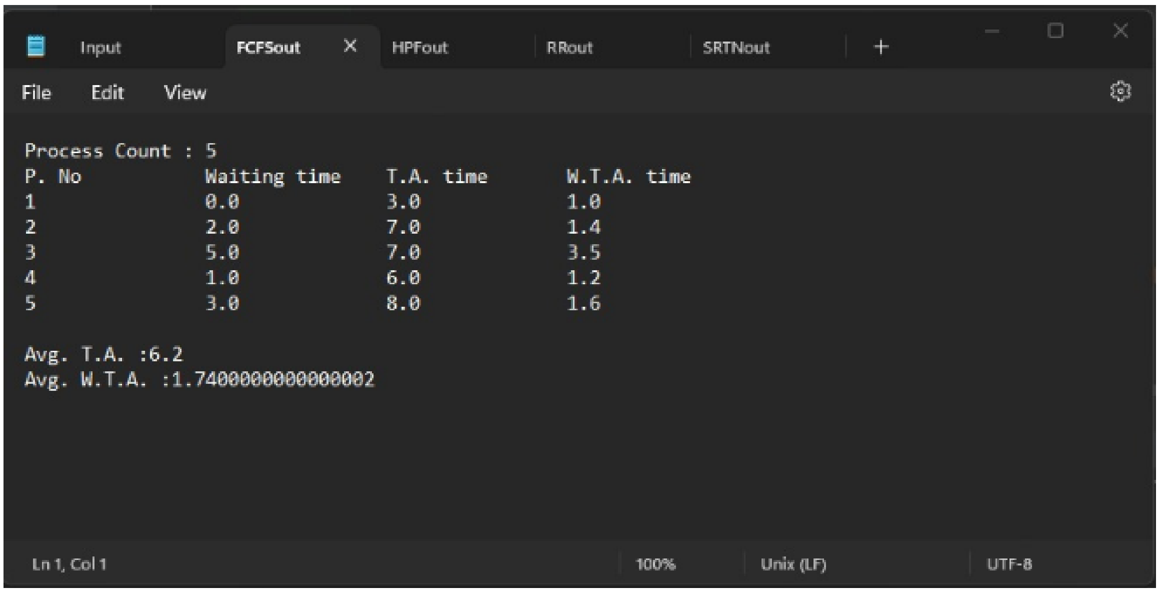
In the First-Come-First-Served (FCFS) scheduling algorithm, context switching does not occur within the execution of a single process. FCFS follows a non-preemptive approach, meaning that once a process starts executing, it continues until it finishes or voluntarily yields the CPU. Therefore, there is no need for a context switch in FCFS scheduling until the running process completes its execution. So, the graph seems same for context switching time 1.



Context Switch Time : 1
Quantum Time : 1
FCFS
Output file name : FCFSout.txt

First Come First Served Output:

The generated output file provides information on the waiting time, turnaround time, and weighted turnaround time for each process, along with average values for turnaround time and weighted turnaround time.



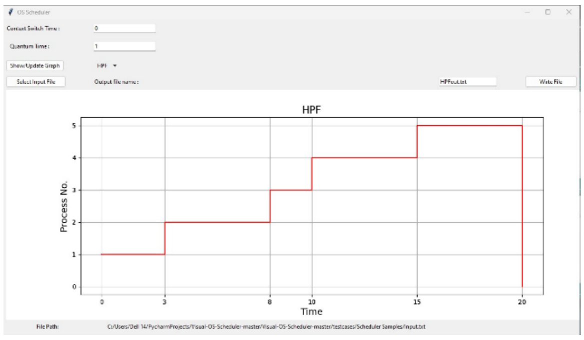
```
Process Count : 5
P. No           Waiting time    T.A. time       W.T.A. time
1               0.0             3.0             1.0
2               2.0             7.0             1.4
3               5.0             7.0             3.5
4               1.0             6.0             1.2
5               3.0             8.0             1.6

Avg. T.A. :6.2
Avg. W.T.A. :1.7400000000000002
```

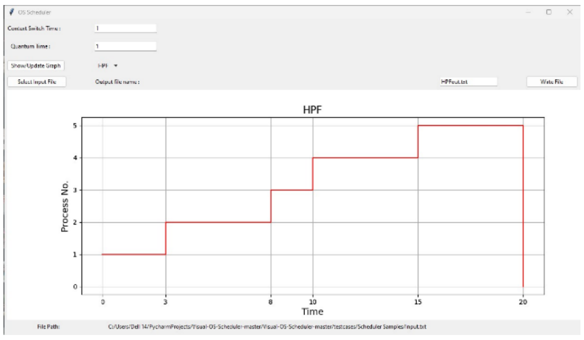
Highest Priority First:

The Highest Priority First (HPF) graph interface showcases the execution order of processes based on their priority levels, visualizing the Non-Preemptive Highest Priority First scheduling algorithm

Highest Priority First Context Switching:

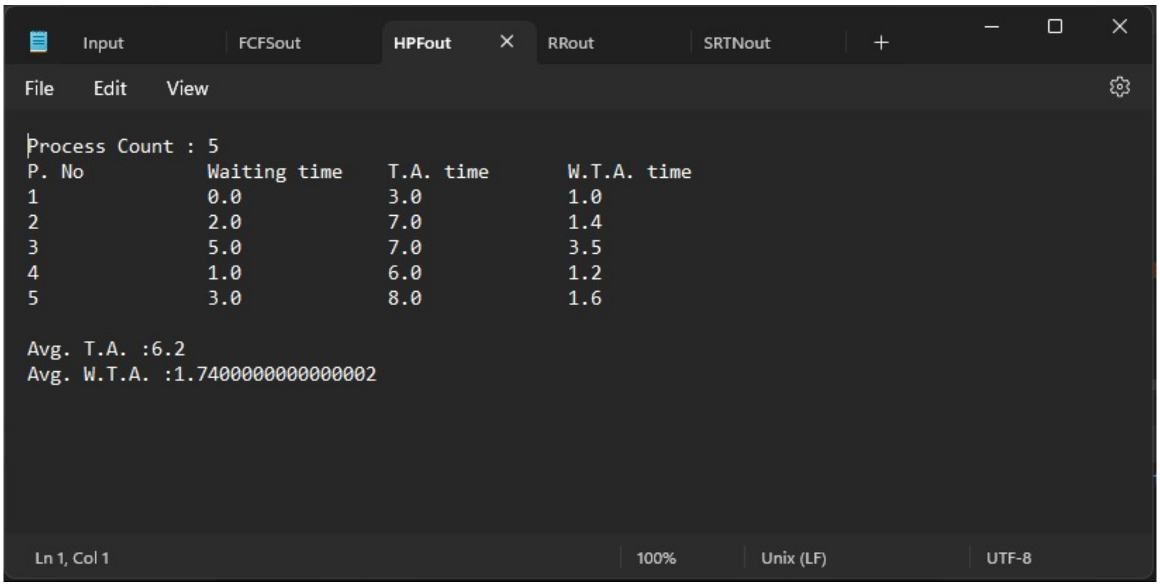
In the Highest Priority First (HPF) scheduling algorithm, context switching occurs when a higher-priority process arrives and takes precedence over the currently executing process. The graph for HPF includes context switching indicators that highlight the moments when the CPU switches from a lower-priority process to a higher-priority process.


Context Switch Time : 0
Quantum Time : 1
HPF
Output file name : HPFout.txt
File Path: C:/Users/Dell 14/PycharmProjects/Visual-OS-Scheduler-master/Visual-OS-Scheduler-master/testcases/Scheduler Samples/input.txt


Context Switch Time : 1
Quantum Time : 1
HPF
Output file name : HPFout.txt
File Path: C:/Users/Dell 14/PycharmProjects/Visual-OS-Scheduler-master/Visual-OS-Scheduler-master/testcases/Scheduler Samples/input.txt

Highest Priority First Output:

For HPF (Highest Priority First), the generated output file includes the process number, waiting time, turnaround time, and weighted turnaround time for each process, along with the average turnaround time and average weighted turnaround time.



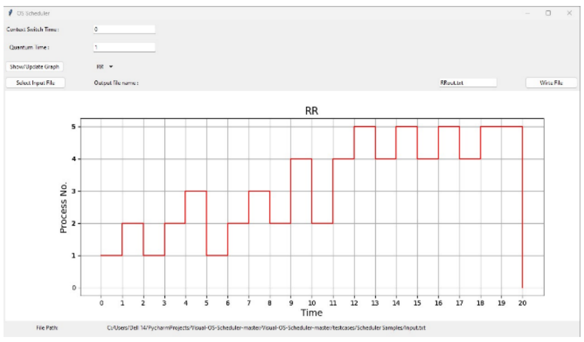
```
Process Count : 5
P. No           Waiting time    T.A. time       W.T.A. time
1               0.0             3.0             1.0
2               2.0             7.0             1.4
3               5.0             7.0             3.5
4               1.0             6.0             1.2
5               3.0             8.0             1.6

Avg. T.A. :6.2
Avg. W.T.A. :1.7400000000000002
```

Round Robin:

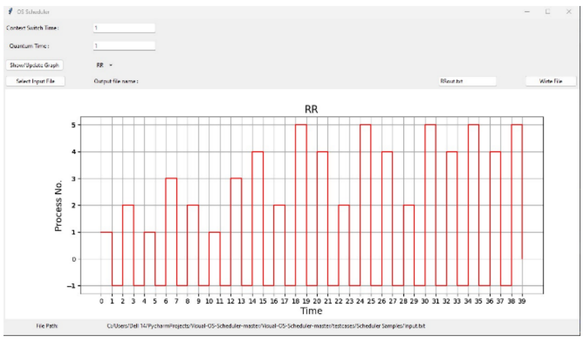
Round Robin (RR) graph interface illustrates the cyclic execution of processes with a fixed time quantum, demonstrating the Round Robin scheduling algorithm.

Round Robin Context Switching::

In the Round Robin (RR) scheduling algorithm, context switching happens at the end of each time quantum. The graph for RR incorporates context switching indicators to signify the points where the CPU switches from one process to another within the fixed time quantum.



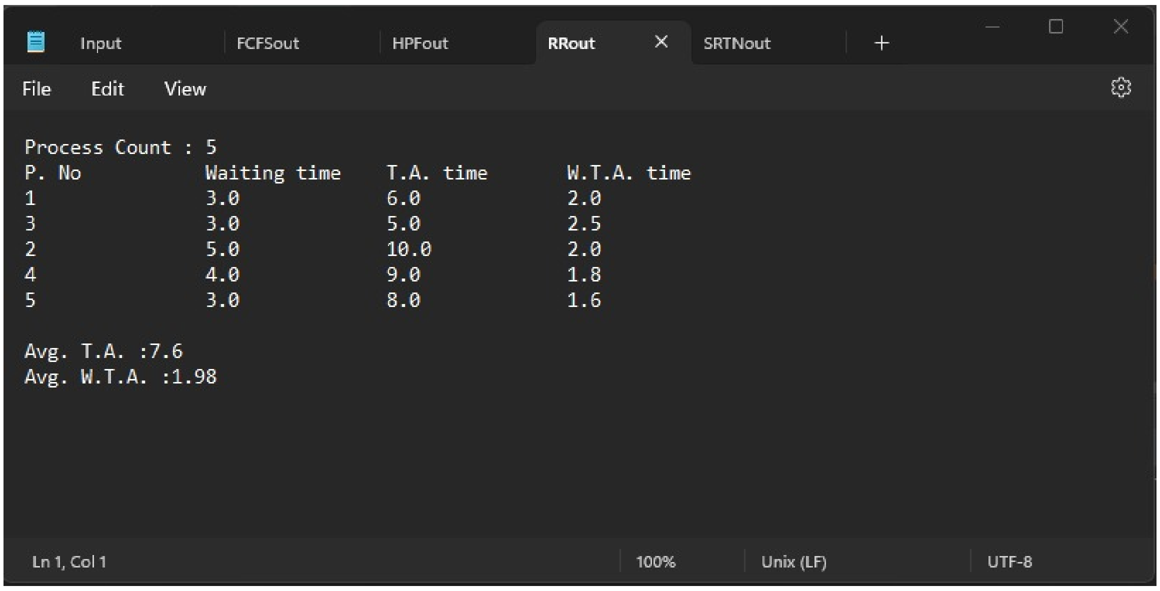
```
OS Scheduler
Context Switch Time :   0
Quantum Time :          1
Show/Update Graph       RR
Select Input File       Output file name :                RRout.txt      Write File
File Path:   C:/Users/Dell 14/PycharmProjects/Visual-OS-Scheduler-master/Visual-OS-Scheduler-master/testcases/Scheduler Samples/Input.txt
```



```
OS Scheduler
Context Switch Time :   1
Quantum Time :          1
Show/Update Graph       RR
Select Input File       Output file name :                RRout.txt      Write File
File Path:   C:/Users/Dell 14/PycharmProjects/Visual-OS-Scheduler-master/Visual-OS-Scheduler-master/testcases/Scheduler Samples/Input.txt
```

Round Robin Output:

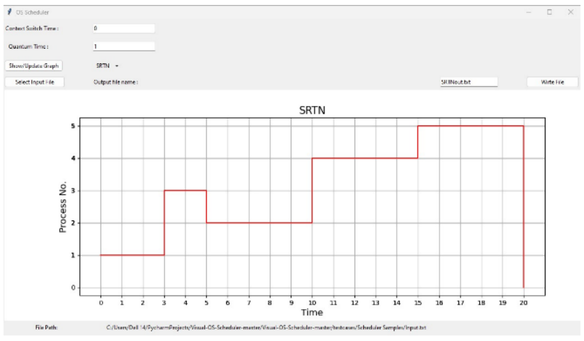
For RR (Round Robin), the output file presents the process number, waiting time, turnaround time, and weighted turnaround time for each process, as well as the average turnaround time and average weighted turnaround time.



```
Process Count : 5
P. No           Waiting time    T.A. time       W.T.A. time
1               3.0             6.0             2.0
3               3.0             5.0             2.5
2               5.0             10.0            2.0
4               4.0             9.0             1.8
5               3.0             8.0             1.6

Avg. T.A. :7.6
Avg. W.T.A. :1.98
```

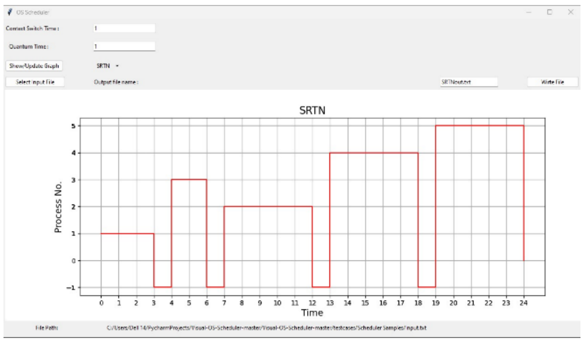
Shortest Remaining Time Next:

The Shortest Remaining Time Next (SRTN) graph interface displays the dynamic execution order of processes based on their remaining execution time, showcasing the Preemptive Shortest Remaining Time Next scheduling algorithm.


Context Switch Time : 0
Quantum Time : 1
SRTN
Output file name : SRTNout.txt

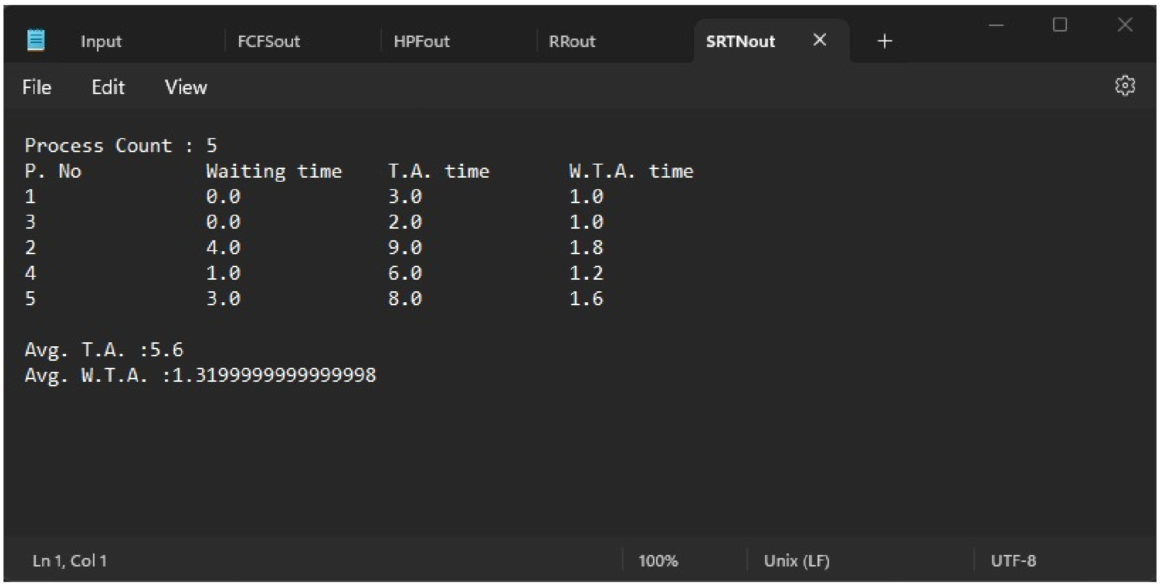
Shortest Remaining Time Next Context Switching:

In the Shortest Remaining Time Next (SRTN) scheduling algorithm, context switching occurs when a process with a shorter remaining execution time arrives and preempts the currently running process. The graph for SRTN includes context switching indicators to represent these instances of preemption and context switching between processes with varying remaining execution times.


Context Switch Time : 1
Quantum Time : 1
SRTN
Output file name : SRTNout.txt

Shortest Remaining Time Output:

For SRTN (Shortest Remaining Time Next), the output file provides the process number, waiting time, turnaround time, and weighted turnaround time for each process, along with the average turnaround time and average weighted turnaround time.



```
Process Count : 5
P. No           Waiting time    T.A. time       W.T.A. time
1               0.0             3.0             1.0
3               0.0             2.0             1.0
2               4.0             9.0             1.8
4               1.0             6.0             1.2
5               3.0             8.0             1.6

Avg. T.A. :5.6
Avg. W.T.A. :1.3199999999999998
```

CHAPTER 6
FUTURE ENHANCEMENTS AND DISCUSSION

6.1 Potential Future Enhancements

⇨ **Update GUI:**
Enhancements can be made to the graphical user interface (GUI) to improve its aesthetics, usability, and functionality. This may include refining the layout, adding tooltips or help sections for better user guidance, and optimizing the overall user experience.

⇨ **Addition Of More Algorithms:**
Introducing additional process scheduling algorithms expands the capabilities of the visualizer. Users can explore and compare a broader range of algorithms, such as Priority Scheduling, Multilevel Queue Scheduling, or Shortest Job Next (SJN), to gain a better understanding of their characteristics and performance.

⇨ **Real Time Visualization:**
Implementing real-time visualization allows users to observe the dynamic behavior of processes and scheduling decisions as they occur. This feature can provide a more immersive experience and accurately represent system performance by updating the visualization in real-time based on actual events and changes in the scheduling process.

⇨ **More Comprehensive Analysis:**
Enhancing the analysis capabilities of the visualizer involves incorporating advanced performance metrics. Metrics like turnaround time (the time from process arrival to completion), waiting time (the time spent in the ready queue), and response time (the time from process arrival to first CPU allocation) provide a more comprehensive evaluation of scheduling algorithms and their impact on system performance.

⇨ **Resource Allocation Graph:**
Extending the visualizer to include resource allocation visualization enhances its ability to analyze the scheduling process. This can involve creating additional graphs or visual representations to depict the utilization of system resources, such as memory or I/O devices, alongside the process scheduling graph. This enables users to identify potential resource bottlenecks and optimize resource allocation for improved overall system performance.

By incorporating these future enhancements, the Visual-OS-Scheduler can provide a more versatile, informative, and interactive platform for studying, analyzing, and optimizing process scheduling algorithms, catering to a wider range of user needs and scenarios.

6.2 Conclusion

In conclusion, the **Visual-Operating-System-Scheduler project** has successfully developed a user-friendly tool that enhances the understanding of process scheduling algorithms through visualization. The project provides an intuitive interface for users to select input files, choose scheduling algorithms, and visualize the CPU usage and context switching events. The inclusion of four scheduling algorithms, namely FCFS, HPF, RR, and SRTN, allows for effective comparison and analysis. The use of Tkinter for GUI and Matplotlib for plotting ensures an interactive and informative experience. Future work could include expanding the algorithm repertoire, real-time visualization, advanced performance metrics, customizable time quantum, resource allocation visualization, and integration with real operating systems. Overall, the Visual-OS-Scheduler project offers a valuable tool for studying, analyzing, and optimizing process scheduling algorithms, aiding students and researchers in gaining a deeper understanding of these complex concepts.

REFERENCES

[1] https://www.researchgate.net/publication/351330519_A_Review_on_the_CPU_Scheduling_Algorithms_Comparative_Study
[2] https://afteracademy.com/blog/process-scheduling-algorithms-in-the-operating-system/
[3] https://www.grin.com/document/267035
[4] https://www.scaler.com/topics/operating-system/scheduling-algorithms-in-os/
[5] https://quescol.com/operating-system/goal-of-cpu-scheduling-algorithm