PRIORITY SCHEDULING SIMULATION

A PROJECT REPORT

Submitted by

MUKUND HARIHARAN [Reg No: RA221103010181]

RAGHUL D [Reg No: RA221103010160]

Under the Guidance of

MR.H.KARTHIKEYAN

Associate Professor, Department of Networking and Communications

In partial fulfilment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF NWC

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603203

NOV 2023

BONAFIDE CERTIFICATE

Certified that this B.Tech project report titled “ PRIORITY SCHEDULING SIMULATION” is the bonafide work of **Mr. Mukund Hariharan [Reg. No: RA2211031010181] and Mr. Raghul D [Reg. No: RA2211031010160]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Mr.H.Karthikeyan

SUPERVISOR

Assistant Professor

Department of Networking and Communications

DR. Annapurani Panaiyappan.K

HEAD OF THE DEPARTMENT

Department of Networking and Communications

SIGNATURE OF INTERNAL EXAMINER

SIGNATURE OF EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express immeasurable thanks to our Course Faculty, **Mr.H.Karthikeyan**, Associate Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under her mentorship and Supervision. He provided us with the valuable inputs during the project reviews and support at all the stages of the project work.

We sincerely thank the **Networking and Communications Department**, staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

MUKUND HARIHARAN [RA2211031010181]

RAGHUL D [RA2211031010160]

TABLE OF CONTENTS

ABSTRACT

This report introduces a Priority Scheduling Simulator designed to facilitate the understanding and analysis of priority-based process scheduling in operating systems. The simulator allows users to input process details, including priority, process ID (PID), and burst time, and executes the scheduling algorithm accordingly. The primary objective is to demonstrate the functionality of the priority scheduling algorithm through a straightforward and user-friendly interface.

The simulator features a minimalistic design, focusing on essential components such as process input, priority-based execution, and result display. Users can input multiple processes with corresponding priorities, and the simulator processes them based on their priority levels. The output provides insights into the order of process execution, emphasizing the impact of priority on scheduling decisions.

This report outlines the simulator's design and implementation, emphasizing its simplicity and ease of use. The simulator serves as an educational tool for individuals seeking to comprehend the basics of priority scheduling without unnecessary complexities. The report concludes with a brief analysis of the simulation results, offering a concise overview of the scheduler's behavior under different priority scenarios.

In summary, the Priority Scheduling Simulator provides a straightforward platform for users to input process details, experience priority-based scheduling, and observe the resulting execution order. The focus is on clarity and simplicity, making it an accessible resource for learners and educators interested in the fundamentals of priority scheduling in operating systems.

INTRODUCTION

In the realm of operating systems, effective process scheduling is paramount for optimizing system performance and resource utilization. Priority scheduling stands out as a significant algorithm, assigning priority levels to processes and determining their execution order based on these priorities. This concept plays a pivotal role in ensuring that critical tasks receive precedence, contributing to improved system responsiveness and efficiency.

This report introduces a Priority Scheduling Simulator, a tool designed to offer a practical understanding of priority-based process scheduling. Unlike complex simulators, our focus is on simplicity and clarity, providing users with a straightforward interface to input process details, including priority, process ID (PID), and burst time. The simulator then executes the priority scheduling algorithm, showcasing the impact of priority levels on the order of process execution.

Through this simulator, users gain insights into the fundamental principles of priority scheduling, witnessing firsthand how varying priority assignments influence the overall system behavior. The following sections will delve into the design, implementation, and analysis of the Priority Scheduling Simulator, highlighting its user-friendly approach and its role as an educational resource for those seeking a concise exploration of priority-based scheduling algorithms.

 Key objectives of the Scheduling Algorithms Simulator project include:

1. Visualization : The project offers a graphical representation of process scheduling, making it accessible for individuals at all levels of expertise. Users can observe how processes are allocated CPU time and the order in which they execute, providing a clear picture of scheduling dynamics.

2. Performance Analysis : Users can evaluate the efficiency of different scheduling algorithms by examining key performance metrics such as turnaround time, waiting time, and CPU utilization. This feature allows for a comprehensive assessment of scheduling algorithm effectiveness.

3. Customization : The simulator can be tailored to specific scenarios. Users can define their own processes, set priorities, and adjust time quantum parameters to create custom scheduling scenarios.

4. Educational Resource : For students and educators, the simulator serves as an invaluable tool for hands-on learning. It provides a platform to experiment with priority scheduling algorithms, reinforcing theoretical knowledge with practical experience.

5. Research Tool : Researchers and professionals can utilize the simulator to assess existing scheduling algorithms, develop new ones, and conduct experiments to gather data for optimizing system resource allocation.

METHODOLOGY

The development of the Scheduling Algorithms Simulator involves a structured approach, combining a well-defined design process and the implementation of scheduling algorithms in Python code. The methodology for building this simulator is outlined as follows:

1. Requirement Analysis :

- Define the primary objectives and user requirements for the simulator. This includes identifying the scheduling algorithms to be implemented, the user interface, and key features such as visualization and customization options.

2. System Design :

- Create a detailed system design that outlines the architecture of the simulator. Define the data structures, classes, and functions needed to represent processes, scheduling algorithms, and the simulator's core logic.

3. User Interface Design :

- Design the graphical user interface (GUI) to make the simulator userfriendly and intuitive. Implement graphical components for displaying process scheduling and user input controls.

4. Data Structures :

- Develop data structures to represent processes, scheduling queues, and performance metrics. Common data structures like arrays, linked lists, and queues will be essential for this purpose.

5. Scheduling Algorithms Implementation :

- Implement Priority scheduling algorithm. Implementing the given processes by prioritizing the one with the highest priority. The user will be prompted to enter the priority and burst time of each processes and the program will be implemented accordingly.

6. User Interaction :

   - Create user input forms to allow users to define custom processes, priorities, and time quantum parameters.

7. Simulation Logic :

   - Develop the core simulation logic that simulates the execution of processes based on the priority scheduling algorithm. Update process states, track performance metrics, and visualize the process scheduling.

8. Visualization :

   - Incorporate graphical visualization components to display the execution of processes. Use graphics libraries or built-in GUI components to represent the scheduling queues and the progress of processes.

9. Performance Metrics Calculation :

   - Implement the calculation of performance metrics, such as turnaround time, waiting time, and CPU utilization. Display these metrics to the user for analysis.

10. Testing and Debugging :

    - Thoroughly test the simulator by running simulations with different scheduling scenarios and checking the accuracy of performance metrics. Debug and fix any issues or discrepancies.

11. Documentation :

- Create comprehensive documentation that explains how to use the simulator, its features, and the underlying scheduling algorithms. Provide user guides and developer documentation.

12. User Feedback and Iteration :

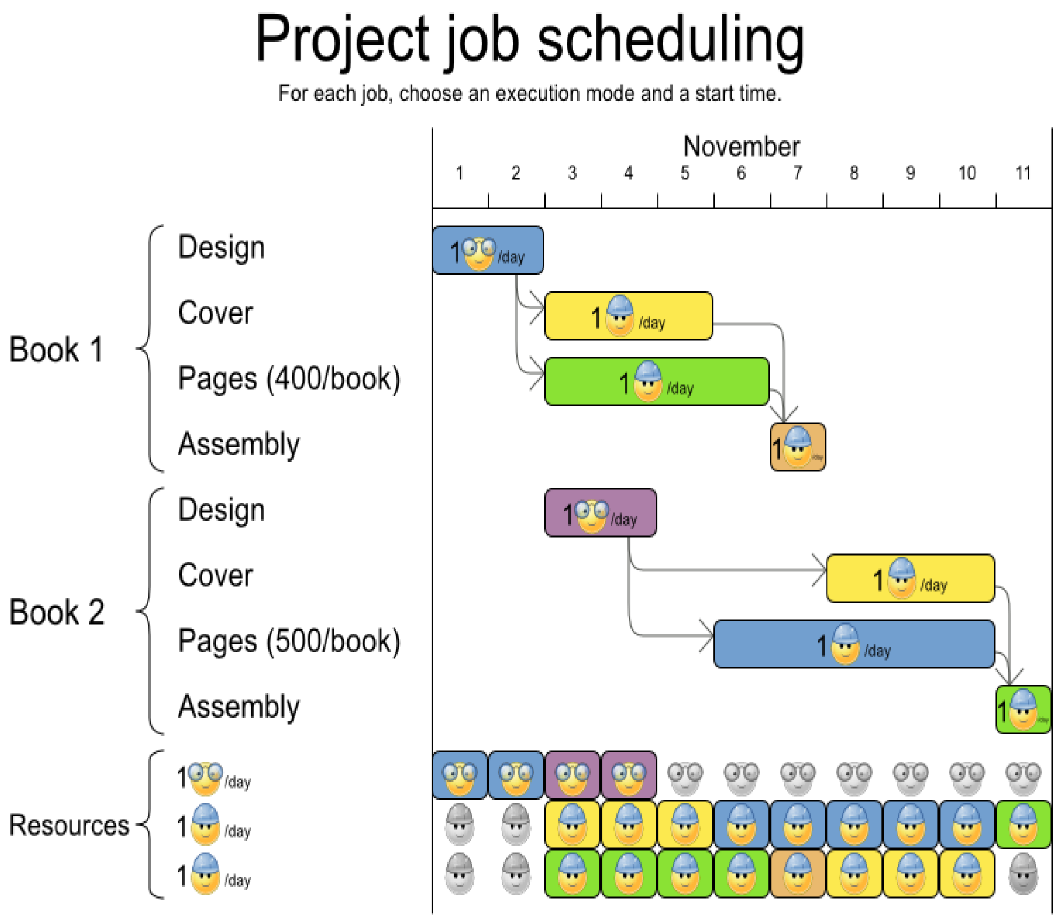
    - Gather user feedback and make necessary improvements and enhancements to the simulator. Address any issues, optimize performance, and add new features based on user suggestions.

13. Deployment :

    - Prepare the simulator for distribution, ensuring it can run on a variety of platforms. Package the simulator for easy installation and distribution.

The methodology for building the Priority Scheduling Simulator combines software engineering principles, algorithm implementation, user interface design, and thorough testing to create a robust and user-friendly tool for exploring and analyzing process scheduling in operating systems. The iterative approach ensures that the simulator can evolve to meet the changing needs of its users and the advancements in the field of computer technology.

DIAGRAM


Project job scheduling
For each job, choose an execution mode and a start time.
November
1
2
3
4
5
6
7
8
9
10
11
Book 1
Design
Cover
Pages (400/book)
Assembly
Book 2
Design
Cover
Pages (500/book)
Assembly
Resources
1 /day
1 /day
1 /day

CODE

```
import tkinter as tk
from tkinter import ttk
import heapq

class Process:
    def __init__(self, name, priority, burst_time):
        self.name = name
        self.priority = priority
        self.burst_time = burst_time

process_queue = []

def add_process():
    name = name_entry.get()
    priority = int(priority_entry.get())
    burst_time = int(burst_time_entry.get())
    process = Process(name, priority, burst_time)
    heapq.heappush(process_queue, (priority, process))
    update_display()

def run_scheduler():
    result_text.delete(1.0, tk.END)
    while process_queue:
        priority, process = heapq.heappop(process_queue)
        result_text.insert(tk.END, f"Running {process.name}...  ")
        process.burst_time -= 1
        if process.burst_time > 0:
            heapq.heappush(process_queue, (process.priority, process))
        update_display()

def update_display():
    process_listbox.delete(0, tk.END)
    for _, process in process_queue:
        process_listbox.insert(tk.END, f"{process.name} (Priority: {process.priority}, 
Burst Time: {process.burst_time})")

window = tk.Tk()
window.title("Priority Scheduling Simulation")
window.geometry("800x600")
window.configure(bg="#E0E0E0")

header_label = tk.Label(window, text="Priority Scheduling Simulation", 
font=("Helvetica", 20, "bold"), bg="#1E90FF", fg="white")
header_label.pack(fill="x", padx=20, pady=10)

input_frame = tk.Frame(window, bg="#E0E0E0")
input_frame.pack(padx=20, pady=10, fill="both")

style = ttk.Style()
style.configure("TLabel", background="#E0E0E0")
style.configure("TEntry", fieldbackground="lightgrey")
style.configure("TButton", background="#32CD32", padding=10)

name_label = ttk.Label(input_frame, text="Process Name", font=("Helvetica", 14), 
padding=10)
name_label.grid(row=0, column=0, padx=10)
name_entry = ttk.Entry(input_frame)
```

```
name_entry.grid(row=0, column=1, padx=10)

priority_label = ttk.Label(input_frame, text="Priority", font=("Helvetica", 14),
padding=10)
priority_label.grid(row=0, column=2, padx=10)
priority_entry = ttk.Entry(input_frame)
priority_entry.grid(row=0, column=3, padx=10)

burst_time_label = ttk.Label(input_frame, text="Burst Time", font=("Helvetica", 14),
padding=10)
burst_time_label.grid(row=0, column=4, padx=10)
burst_time_entry = ttk.Entry(input_frame)
burst_time_entry.grid(row=0, column=5, padx=10)

add_button = ttk.Button(input_frame, text="Add Process", command=add_process)
add_button.grid(row=1, column=0, columnspan=2, pady=20)

run_button = ttk.Button(input_frame, text="Run Scheduler", command=run_scheduler,
style="TButton")
run_button.grid(row=1, column=2, columnspan=2, pady=20)

process_listbox = tk.Listbox(window, bg="lightgrey", width=60, height=10,
font=("Helvetica", 14))
process_listbox.pack(padx=20, pady=10)

result_label = tk.Label(window, text="Execution Log", font=("Helvetica", 16, "bold"),
bg="#E0E0E0")
result_label.pack(padx=20)

result_text = tk.Text(window, height=10, width=60, bg="lightgrey", font=("Helvetica",
14))
```

```
result_text.pack(padx=20, pady=10)

window.mainloop()
```

CONCLUSION

In conclusion, the Priority Scheduling Simulator presented in this report serves as a valuable tool for comprehending the intricacies of priority-based process scheduling in operating systems. With a user-friendly interface facilitating the input of priority, process ID, and burst time, the simulator effectively demonstrates the impact of priority levels on the order of process execution.

The simulator's simplicity is intentional, providing an accessible resource for learners and educators to grasp the core concepts of priority scheduling without unnecessary complexities. By executing the priority scheduling algorithm and displaying the resulting process sequence, users can gain practical insights into how priorities influence the overall system dynamics.

As an educational instrument, the Priority Scheduling Simulator contributes to the understanding of scheduling algorithms, offering a hands-on experience that complements theoretical knowledge. Moving forward, the insights gained from this simulator can pave the way for further exploration and experimentation with more advanced scheduling strategies.

In essence, this report and the accompanying simulator aim to demystify priority scheduling, making it approachable for individuals at various levels of expertise. The simulator's intuitive design, coupled with its capacity to visually represent scheduling outcomes, enhances the learning experience and establishes it as a valuable resource in the study of operating systems.

BIBLIOGRAPHY

Books:-

- Silberschatz, Abraham, Galvin, Peter B., and Gagne, Greg. (2018). "Operating System Concepts." Wiley.
- Tanenbaum, Andrew S., and Bos, Herbert. (2015). "Modern Operating Systems." Pearson.
- Stallings, William. (2018). "Operating Systems: Internals and Design Principles." Pearson.
- Deitel, Paul J., and Deitel, Harvey M. (2017). "Operating Systems." Pearson.
- Silberschatz, Abraham, Galvin, Peter B., and Gagne, Greg. (2019). "Operating System Concepts Essentials." Wiley.

Links:-

- https://en.wikipedia.org/wiki/Operating_system
- https://www.geeksforgeeks.org/operating-systems/
- https://wiki.osdev.org/Main_Page
- https://dl.acm.org/
- https://www.google.com