CAPSTONE PROJECT REPORT

(Project Term July - December 2023)

Priority Scheduling of Processes

Submitted by

Name of Student:- Pavan **Registration Nuumber:- 12218283**

Course Code:- CSE288

Under the Guidance of

Waseem ud din Wani(63869)

School of Computer Science and Engineering

DECLARATION

I hereby declare that the project work entitled (“Priority Scheduling of Processes”) is an authentic record of our own work carried out as requirements of Capstone Project for the award of B.Tech degree in Computer Science and Engineering (AI & ML) from Lovely Professional University, Phagwara, under the guidance of Waseem Ud Din Wani, during july to December 2023. All the information furnished in this capstone project report is based on our own intensive work and is genuine.

Project Group Number:11

Name of Student : Pavan
Registration Number: 12218283

Signature of Student:pavan

Date:

CERTIFICATE

This is to certify that the declaration statement made by this student is correct to the best of my knowledge and belief. He has completed this Capstone Project under my guidance and supervision. The present work is the result of their original investigation, effort and study. No part of the work has ever been submitted for any other degree at any University. The Capstone Project is fit for the submission and partial fulfillment of the conditions for the award of B.Tech degree in Computer Science and Engineering (AI & ML) from Lovely Professional University, Phagwara.

Signature and Name of the Mentor:

Designation:

School of Computer Science and Engineering,
Lovely Professional University,
Phagwara, Punjab.

Date :

ACKNOWLEDGEMENT

I would like to extend my sincere thanks to the Lovely School of Computer Science and Engineering for providing me with the opportunity to fulfill my wish and achieve my goal. I am grateful to Waseen ud din Wani sir for providing me with the opportunity to undertake this project and for providing me with all the necessary facilities. I am highly thankful to sir for his active support, valuable time and advice, whole-hearted guidance, sincere cooperation, and pain-taking involvement during the study and in completing the assignment of preparing the said project within the time stipulated. Lastly, I am thankful to all those, particularly the various friends, who have been instrumental in creating a proper, healthy, and conducive environment and including new and fresh innovative ideas for me during the project. Without their help, it would have been extremely difficult for me to prepare the project in a timebound framework.

1. INTRODUCTION

In the realm of computer science and operating systems, the efficient allocation of system resources is of paramount importance. Priority scheduling, a key component of process management, plays a crucial role in ensuring that the most critical tasks receive prompt attention from the CPU. This project delves into the world of priority scheduling of processes, elucidating its purpose and underscoring its profound significance in modern computing environments. Significance of Priority Scheduling The significance of priority scheduling in modern computing cannot be overstated:

• Real-Time Systems: In real-time systems, such as those used in autonomous vehicles, medical devices, and robotics, the timely execution of tasks can be a matter of life and death. Priority scheduling ensures that mission-critical operations are performed promptly.

• Multitasking Environments: In desktop and server operating systems, multitasking is the norm. Priority scheduling ensures that background processes do not disrupt the responsiveness of user-facing applications, thereby enhancing the overall user experience.

• Resource Management: Priority scheduling is instrumental in efficient resource management. It helps in avoiding resource starvation by allowing low-priority tasks to execute when higher-priority tasks are idle.

• Load Balancing: In distributed systems, priority scheduling can be employed to distribute tasks across multiple nodes based on their significance, leading to improved load balancing and system performance.

1.2.1. **PROJECT OVERVIEW**

In conclusion, priority scheduling is a critical component of process management that holds immense importance in ensuring the efficient operation of computer systems across various domains. This project will delve deeper into the intricacies of priority scheduling, exploring algorithms, implementation, and practical implications, all aimed at highlighting its vital role in modern computing.

1.2.2. TOOLS AND TECHNOLOGIES USED

I used the 1. IntelliJ IDEA 2. Online Java compiler. We also used Java as the programming language for developing the application.

2. ABSTRACT

This report provides an overview of a priority scheduling developed using the Java programming language. The report discusses the importance of Priority Processes and the challenges faced by them, including timming issues issues, user interface and legal issues.

The report then delves into the project of creating a priority using Java, including key components such as the user interface. The report also highlights the use of Java libraries and frameworks to facilitate the development of the app.

In conclusion, this report provides a comprehensive overview of a project using Java, highlighting the importance of priority and the challenges faced by them. The report provides valuable insights into the development process of a priority and the key components that make up such an Algorithm. By providing a detailed account of the project, this report serves as a useful resource for developers looking to create high-quality Processes Execution.

3. OBJECTIVES OF THE PROJECT

• Maximize Throughput: Priority scheduling aims to maximize the throughput of the system by giving CPU time to the processes with higher priority. This means that important and time-sensitive tasks can be executed quickly, which can be crucial in real-time systems.

• Meet Deadlines: In real-time systems, priority scheduling helps ensure that processes with deadlines are executed on time. This is especially important in applications such as industrial automation, where missing a deadline could have catastrophic consequences.

• Ensure Fairness: While priority scheduling prioritizes higher-priority tasks, it should also ensure that lower-priority tasks are not completely starved of CPU time. This balance is essential to maintain fairness in the system.

• Optimize Resource Utilization: By assigning different priorities to processes, priority scheduling can help optimize the use of system resources. It can allocate more CPU time to processes that need it, and less to those that are less critical.

• Improve Responsiveness: Priority scheduling can make the system more responsive, especially for interactive tasks. Processes that require user interaction can be given higher priority to reduce user-perceived delays.

4. DESCRIPTION OF THE PROJECT

• Preemptive and Non-Preemptive: Priority scheduling can be implemented in both preemptive and non-preemptive variants. In preemptive priority scheduling, a running process can be interrupted if a higher-priority process becomes available. In non-preemptive priority scheduling, the CPU is allocated to a process until it voluntarily relinquishes control.

• Dynamic or Static Priority: Priorities can be dynamic, where they change over time based on various factors like aging or process behavior, or they can be statically assigned and remain fixed throughout a process's lifetime.

• Multiple Queues: Some priority scheduling algorithms use multiple priority queues, each with a different priority level. Processes are placed in the queue that corresponds to their priority, and the scheduler selects processes from these queues based on their priorities.

• Real-time and General-Purpose Systems: Priority scheduling can be employed in both real-time and general-purpose operating systems. In real-time systems, it is crucial to meet deadlines and ensure predictable performance, while in general-purpose systems, it helps in optimizing overall system performance.

• Complexity: Priority scheduling can become complex when dealing with a large number of processes, and it requires a careful balance to avoid issues like priority inversion and starvation.

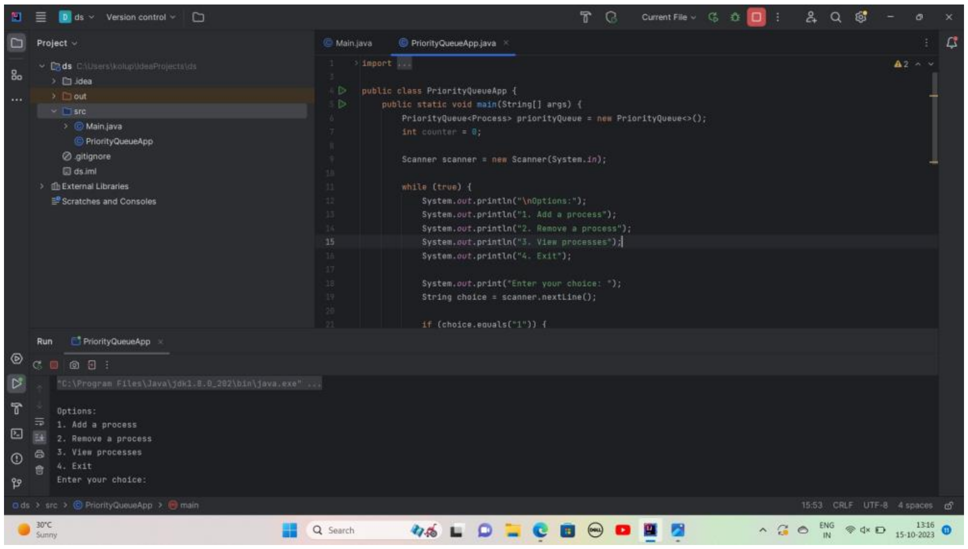
• Fairness: Ensuring fairness is a significant challenge in priority scheduling, as lower-priority processes should not be neglected, and starvation should be avoided

5.SOURCE CODE

```
 import java.util.PriorityQueue;
import java.util.Scanner;

class Process implements Comparable<Process> {
    private int pid;
    private String name;
    private int priority;

    public Process(int pid, String name, int priority) {
        this.pid = pid;
        this.name = name;
        this.priority = priority;
    }

    public int getPid() {
        return pid;
    }

    public String getName() {
        return name;
    }

    public int getPriority() {
        return priority;
    }

    @Override
    public int compareTo(Process other) {
        return Integer.compare(other.getPriority(), this.getPriority());
    }

    @Override
    public String toString() {
        return "Process ID: " + pid + ", Name: " + name + ", Priority: " + priority;
    }
}

public class PriorityQueueApp {
    public static void main(String[] args) {
        PriorityQueue<Process> priorityQueue = new PriorityQueue<>();
        int counter = 0;

        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\nOptions:");
            System.out.println("1. Add a process");
            System.out.println("2. Remove a process");
            System.out.println("3. View processes");
            System.out.println("4. Exit");

            System.out.print("Enter your choice: ");
            String choice = scanner.nextLine();

            if (choice.equals("1")) {
                System.out.print("Enter Process ID: ");
                int pid = Integer.parseInt(scanner.nextLine());
                System.out.print("Enter Process Name: ");
                String name = scanner.nextLine();
                System.out.print("Enter Priority: ");
                int priority = Integer.parseInt(scanner.nextLine());
                Process process = new Process(pid, name, priority);
                priorityQueue.add(process);
                System.out.println("Process " + name + " added to the queue.");
            } else if (choice.equals("2")) {
                Process process = priorityQueue.poll();
                if (process != null) {
                    System.out.println("Process " + process.getName() + " removed
from the queue.");
                } else {
                    System.out.println("Queue is empty.");
                }
            } else if (choice.equals("3")) {
                for (Process process : priorityQueue) {
                    System.out.println(process);
                }
            } else if (choice.equals("4")) {
                break;
            } else {
                System.out.println("Invalid choice. Please try again.");
            }
        }
    }
}
```

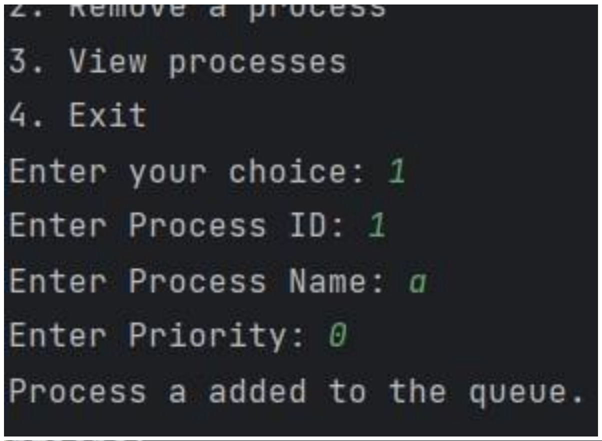
6.INPUT/OUTPUT

after clicking Run Icon

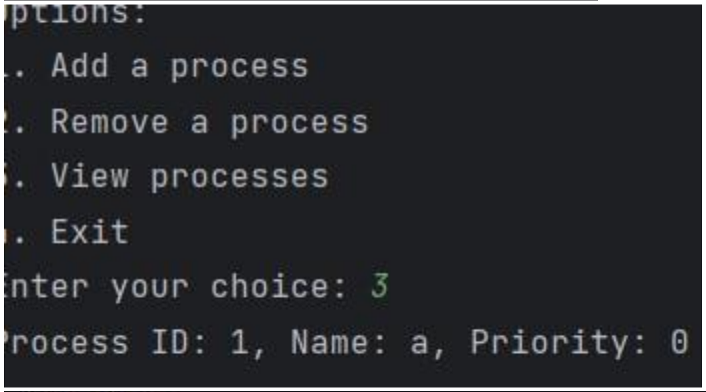


```java
import ...

public class PriorityQueueApp {
    public static void main(String[] args) {
        PriorityQueue<Process> priorityQueue = new PriorityQueue<>();
        int counter = 0;

        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\nOptions:");
            System.out.println("1. Add a process");
            System.out.println("2. Remove a process");
            System.out.println("3. View processes");
            System.out.println("4. Exit");

            System.out.print("Enter your choice: ");
            String choice = scanner.nextLine();

            if (choice.equals("1")) {
```

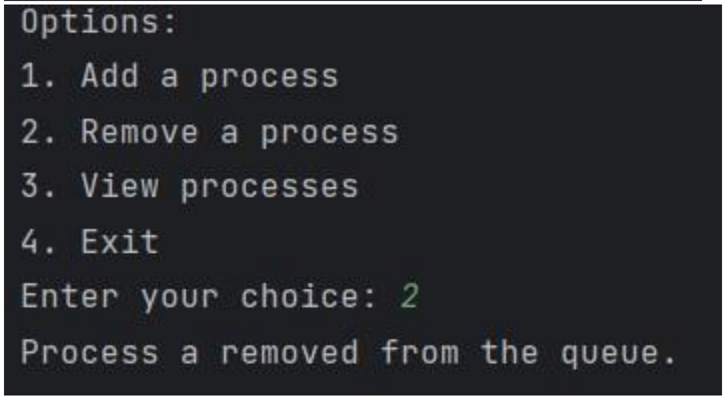
```
"C:\Program Files\Java\jdk1.8.0_202\bin\java.exe" ...

Options:
1. Add a process
2. Remove a process
3. View processes
4. Exit
Enter your choice:
```



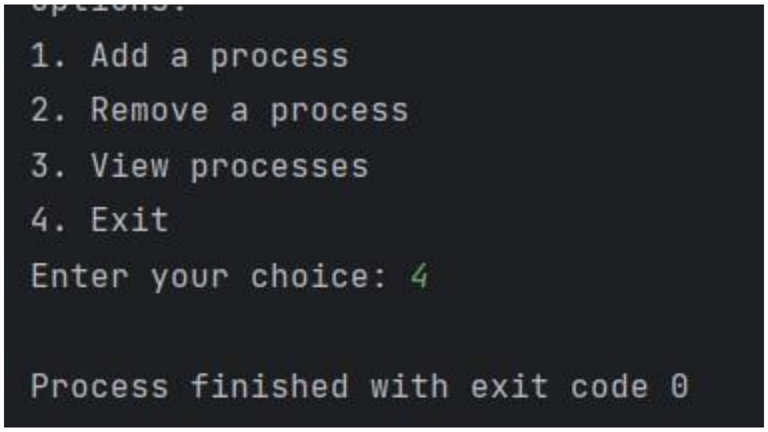
```
2. Remove a process
3. View processes
4. Exit
Enter your choice: 1
Enter Process ID: 1
Enter Process Name: a
Enter Priority: 0
Process a added to the queue.
```



```
ptions:
. Add a process
. Remove a process
. View processes
. Exit
nter your choice: 3
rocess ID: 1, Name: a, Priority: 0
```



```
Options:
1. Add a process
2. Remove a process
3. View processes
4. Exit
Enter your choice: 2
Process a removed from the queue.
```



```
Options:
1. Add a process
2. Remove a process
3. View processes
4. Exit
Enter your choice: 4

Process finished with exit code 0
```

7.SCOPE OF THE PROJECT

Priority scheduling is a scheduling algorithm used in operating systems to determine the order in which processes should be executed based on their priority. Each process is assigned a priority, and the scheduler selects the process with the highest priority to execute next. Priority scheduling has several scopes and use cases:

1. Real-Time Systems: In real-time systems, where meeting deadlines is critical, priority scheduling is often used. High-priority tasks, such as controlling hardware or handling safety-critical operations, are given priority to ensure timely execution.

2. Multitasking Operating Systems: In multitasking operating systems, priority scheduling can be used to allocate CPU time to different processes. For example, an interactive user interface process might have a higher priority to ensure responsiveness, while background tasks run at a lower priority.

3. Server Systems: In server applications, priority scheduling can be used to prioritize client requests. For example, a web server can prioritize incoming requests based on the importance or type of the request.

4. Scientific Computing: In scientific computing or simulations, priority scheduling can be used to allocate CPU time to various computational tasks. Critical simulations or computations can be given higher priority.

5. Batch Processing: In batch processing systems, priority scheduling can be used to prioritize batch jobs. High-priority jobs can be processed ahead of lower-priority jobs.

6. Embedded Systems: In embedded systems, especially those used in safety-critical applications like automotive control systems or medical devices, priority scheduling ensures that critical tasks are executed promptly.

7. Traffic Management: In network routers and switches, priority scheduling is used to determine the order in which packets should be forwarded based on their priority levels. This is crucial for Quality of Service (QoS) in networking.

8. Multithreading: In multithreaded applications, priority scheduling can be used to determine the execution order of threads. This can help in managing thread priorities in applications like gaming or multimedia processing.

9. Resource Allocation: In systems with limited resources, such as printers or shared resources, priority scheduling can be used to allocate resources to processes or users based on their priority levels.

It's important to note that priority scheduling needs to be carefully managed to avoid issues like priority inversion and starvation. Priority inversion occurs when a low-priority task holds a resource that a high-priority task needs, causing the high-priority task to wait. Starvation happens when a low-priority task never gets a chance to execute because high-priority tasks monopolize the CPU.

The effectiveness of priority scheduling depends on proper prioritization and management to meet the specific requirements and goals of the system or application.

.

8. FUTURE DEVELOPMENT OF PROJECT

There are several potential areas for future development of the Priority scheduling project using Java. These include:
The development of priority scheduling algorithms is an ongoing process, driven by advancements in computing technology and changing requirements in various domains. Here are some potential future developments of priority scheduling algorithms:

1. Energy Efficiency: As energy efficiency becomes a growing concern in computing, future priority scheduling algorithms may take into account the power consumption characteristics of processes and devices. The goal would be to optimize power usage by prioritizing low-power or energy-efficient tasks.

2. Machine Learning Integration: Machine learning techniques can be used to dynamically adjust process priorities based on historical data and patterns. This can lead to more adaptive and intelligent priority scheduling algorithms that optimize resource allocation based on the specific workload and system conditions.

3. Real-Time and Safety-Critical Systems: With the increasing use of real-time and safety-critical systems in autonomous vehicles, medical devices, and industrial control, future priority scheduling algorithms will need to guarantee stricter deadlines and safety requirements. They may incorporate advanced verification and analysis techniques to ensure predictable and reliable execution.

4. Resource Management in Cloud Computing: In cloud computing environments, where multiple tenants share resources, priority scheduling algorithms will continue to evolve to ensure fairness, performance isolation, and security. QoS-aware priority scheduling will be crucial to meet SLAs.

5. Big Data and Analytics: Priority scheduling algorithms may be designed to optimize data processing and analytics workloads, ensuring that high-priority data analysis tasks receive the necessary resources to provide real-time insights.

6. Quantum Computing: In the emerging field of quantum computing, new priority scheduling algorithms will need to be developed to manage the allocation of quantum bits (qubits) and quantum gates to different quantum algorithms. These algorithms will aim to optimize quantum resource usage.

7. Internet of Things (IoT): As IoT devices become more prevalent, priority scheduling will play a role in managing the execution of tasks on resource-constrained devices. Algorithms will need to be lightweight and energy-efficient for IoT scenarios.

8. Cross-Platform Compatibility: Priority scheduling algorithms that can work seamlessly across various platforms, including traditional computing systems, mobile devices, and edge devices, will become more important as heterogeneous computing environments become the norm.

9. Blockchain and Cryptocurrency: In blockchain networks and cryptocurrency mining, priority scheduling will continue to evolve to allocate processing power and optimize transaction verification and block creation based on factors like transaction fees, network load, and security.

10. Hybrid Scheduling: Future priority scheduling algorithms may incorporate elements of other scheduling algorithms, such as round-robin, to provide a balanced approach. Hybrid scheduling can offer the advantages of multiple scheduling strategies while mitigating their disadvantages.

11. Dynamic Priority Adjustment: Schedulers may become more adaptive, adjusting priorities in real-time based on the evolving workload and system conditions. Machine learning models can assist in this dynamic adjustment.

12. Security and Isolation: Priority scheduling will continue to be used to enhance system security by isolating sensitive tasks from less trusted ones. Advanced security and isolation techniques will be integrated into priority-based resource allocation.

The future development of priority scheduling algorithms will be closely tied to the evolving needs of various application domains and advancements in hardware and software technologies. Customized priority scheduling approaches will continue to emerge to address the unique challenges of specific use cases.

9.CONCLUSION

A binary heap is a data structure used in computer science and operating systems for implementing priority scheduling of processes in various algorithms like Dijkstra's algorithm, Prim's algorithm, and heap-based priority queues in scheduling. In summary, a binary heap is a data structure used for priority scheduling in which processes are organized based on their priority, allowing efficient retrieval and execution of the highest-priority process. It is a fundamental component in various algorithms and operating system scheduling mechanisms.

Bibliography: 1.DATA STRUCTURES AND ALGORITHMS by ALFRED V. AHO, JEFFREY D. ULLMAN AND JOHN E. HOPCROFT, PEARSON, PEARSON. 2.DATA STRUCTURES AND ALGORITHMS IN JAVA by MICHAEL T. GOODRICH AND ROBERTO TAMASSIA, JOHN WILEY & SONS.

Annexure: I would like to thank to my offline instructor mr:waseem sir, for providing good guidance in the doubts clearing section regarding the project