**National University of Computer & Emerging Sciences**

**Karachi Campus**

**Project Report**

**Operating System**

**Section: BCS-4F**



**Preemptive Scheduler Using Threads**

**Group Members:**

**21k-3263 Uzair Shahzad**

**21k-3259 Talal Abdullah**

# Objective:

The objective of this project is to implement and evaluate Remaining Time First (SRTF) preemptive scheduling algorithm. Implement the scheduling algorithm to understand their underlying mechanisms and functionality. This includes coding the algorithm in a suitable programming language and designing the necessary algorithm to support process management and scheduling. This report presents the findings of our study, including the methodology, results, and conclusions.

By achieving these objectives, this project aims to enhance our understanding of preemptive scheduling algorithm and its impact on process management in operating systems. The findings and recommendations from this project can assist system administrators and developers in making informed decisions when selecting scheduling algorithms for their systems, thereby improving overall system performance and efficiency.

# Introduction:

In the realm of operating systems, scheduling algorithms play a critical role in managing and optimizing the execution of processes or tasks. This algorithm determine how the CPU resources are allocated to different processes, affecting important performance metrics such as turnaround time, waiting time, and response time. Preemptive scheduling algorithms, in particular, allow for the interruption and resumption of processes based on specific criteria.

# Background:

Each scheduling algorithm has its own unique characteristics and prioritization criteria. SRTF focus on the remaining execution time of a process, where it selects the process with the shortest remaining time.

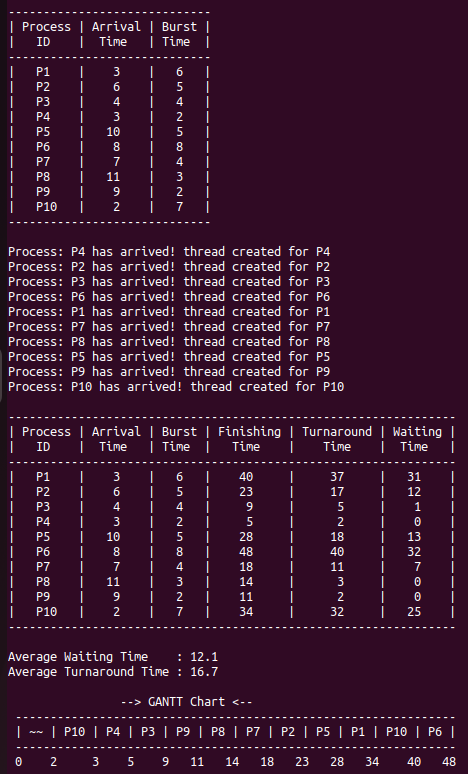
# Platform and Languages:

For the implementation of the preemptive scheduling algorithms, we utilized a programming platform that supports process management and scheduling. We chose to implement the algorithms in C++, a versatile and widely used programming language. C++ provides a range of libraries (like pthreads.h and semaphore.h) and modules that facilitate the creation and analysis of scheduling algorithms.

# Methodology:

* Data Preparation:
  + We asked the user for a set of processes with arrival times, burst times and then test the algorithm.
* SRTF Algorithm:
  + We implemented the Shortest Remaining Time First algorithm, where processes are selected based on the shortest remaining burst time.
* Performance Evaluation:
  + We measured and compared the turnaround time and waiting time of the algorithm using the generated workload.

Shortest Remaining Time First Algorithm



Advantages of SRTF Algorithm:

Advantages of the Shortest Remaining Time First (SRTF) scheduling algorithm:

* **Reduced Average Waiting Time:** SRTF minimizes the average waiting time for processes. By selecting the process with the shortest remaining burst time, it prioritizes the execution of shorter tasks, leading to faster completion and reduced waiting times overall.
* **Increased System Throughput:** SRTF can improve the system throughput by executing shorter processes more quickly. This allows for a higher number of processes to be completed within a given time, increasing overall system efficiency.
* **Responsive to Interactive Tasks:** SRTF is particularly suitable for interactive systems where responsiveness is crucial. By prioritizing short tasks, it ensures that interactive tasks, such as user input processing or system responsiveness, are handled promptly, enhancing the user experience.
* **Efficient Utilization of CPU:** SRTF utilizes the CPU efficiently by always selecting the process with the shortest remaining burst time. It minimizes idle CPU time by keeping the CPU busy with executing tasks as much as possible.
* **No Need for Prioritization:** SRTF inherently gives priority to short processes without requiring explicit priority assignments. This simplifies the scheduling algorithm and eliminates the need for manual prioritization.
* **Better Resource Utilization:** SRTF optimizes the utilization of system resources by executing short processes quickly and freeing up resources for subsequent tasks. This improves overall resource utilization within the system.

Disadvantages of SRTF Algorithm:

Disadvantages of the Shortest Remaining Time First (SRTF) scheduling algorithm:

* **Starvation:** SRTF is prone to causing starvation, particularly for processes with longer burst times. When a process with a shorter burst time arrives continuously, longer processes may never get a chance to execute, leading to starvation. This can result in decreased fairness and overall system performance.
* **High overhead:** SRTF incurs high overhead due to frequent context switches between processes with the shortest remaining burst time. This can degrade system performance by increasing the cost of saving and restoring process states.
* **Difficulty in predicting burst times:** The accurate prediction of the remaining burst time for each process is challenging. If the estimated burst times are inaccurate, the scheduler may make poor decisions and impact overall scheduling efficiency.
* **Preemption frequency:** SRTF involves frequent preemption due to new processes with shorter burst times arriving. This disrupts execution flow and introduces overhead for saving and restoring process states.
* **Complexity:** Implementing SRTF requires additional bookkeeping to maintain the remaining burst times of processes accurately. This complexity can make the implementation and maintenance of the algorithm more challenging.
* **Sensitivity to process arrival patterns**: SRTF's performance can vary depending on the arrival pattern of processes. If processes with similar burst times arrive at the same time, the algorithm may result in higher context switching overhead and decreased efficiency.

Conclusion:

The choice of preemptive scheduling algorithm depends on the specific requirements of the system and the workload characteristics. Factors such as responsiveness, fairness, and throughput should be considered when selecting an appropriate algorithm for a given scenario.