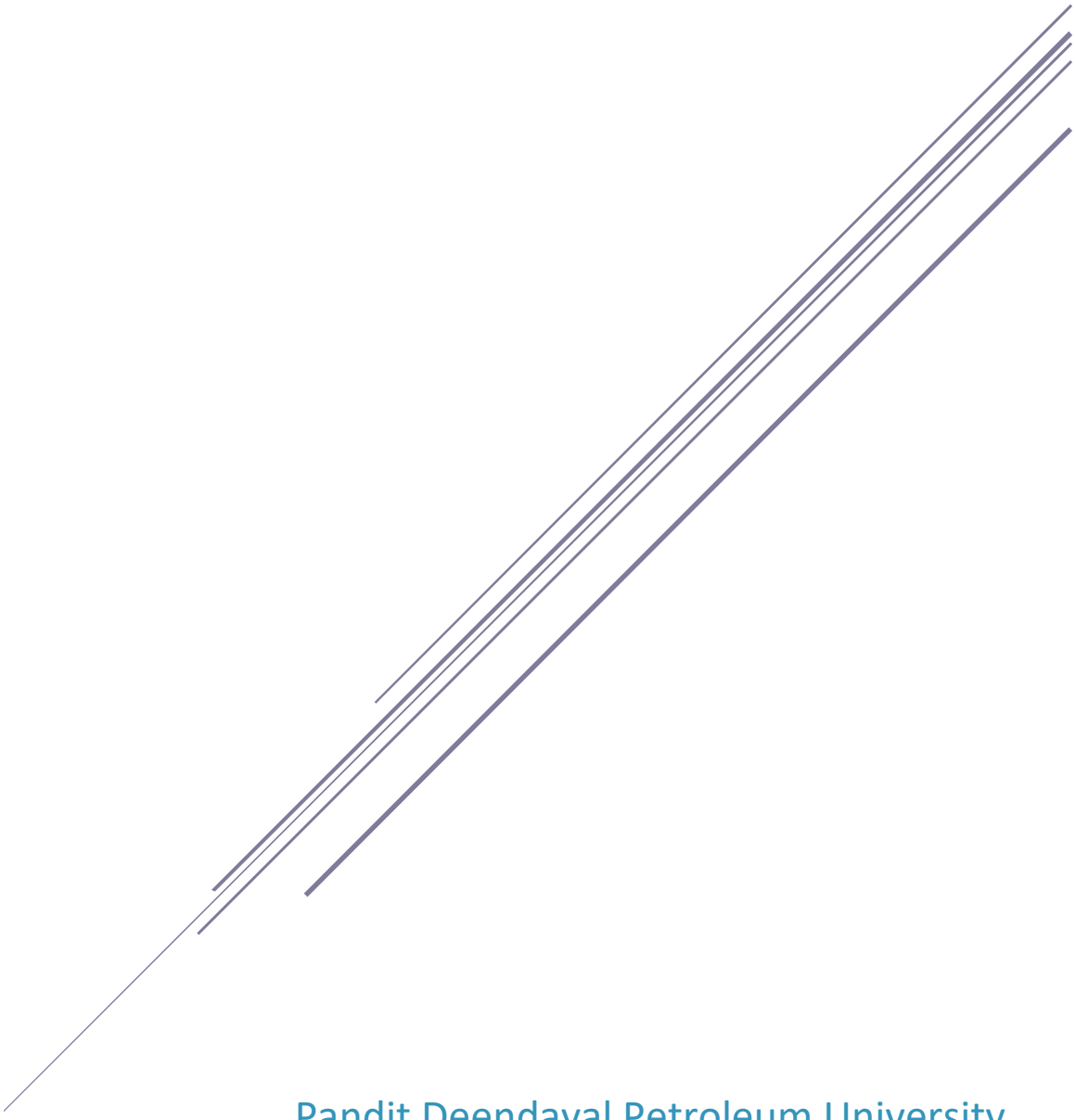
OPERATING SYSTEMS PROJECT

Scheduling Algorithms GUI



Pandit Deendayal Petroleum University
CE'18

PROJECT DETAILS

Submission to: Mr. Chintan Patel

Date: 12th December, 2020

Group 15

18BCP029	Divam Kachoria
18BCP096	Samip Sharma
18BCP108	Shristi Darbar
18BCP121	Vaidehi Shah

Content

Introduction

Central Processing Unit of a computer system is known as the heart of the system. All the processes and activities or tasks are carried out by this unit. In order to improve the overall efficiency with which we can carry out various computations we need to ensure that we fully utilize the resources and features of CPU available to us at any given time. Hence we need to ensure that efficiency of CPU utilization while carrying out various processes simultaneously must be as high as possible for better and faster computation.

In order to improve the CPU utilizations we need to have ways or methods that ensure maximum usage of CPU and minimum time for CPU idleness. In order to utilize CPU to maximum extent certain algorithms are developed. These algorithms are known as scheduling algorithms as they help us to schedule all the process in some particular way that would eventually help us to meet some specific requirements.

The GUI interface we have created basically provides an easy visualization platform for this algorithms in order to understand how this algorithms work.

But before being fully able to grasp the crux of CPU scheduling we must be aware of the theory behind these algorithms. A brief theory and implementation details of CPU scheduling, I/O process, types of scheduling algorithm etc. is provided in this report.

CPU Scheduling

CPU scheduling is a process which allows one process to use the CPU while the execution of another process is on hold (in waiting state) due to unavailability of any resource like I/O etc., thereby making full use of CPU. The aim of CPU scheduling is to make the system efficient, fast and fair.

Whenever the CPU becomes idle, the operating system must select one of the processes in the ready queue to be executed. The selection process is carried out by the short-term scheduler (or CPU scheduler). The scheduler selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them.

NON-PREEMPTIVE SCHEDULING

Under non-preemptive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state.

It is the only method that can be used on certain hardware platforms, because It does not require the special hardware (for example: a timer) needed for preemptive scheduling.

PREEMPTIVE SCHEDULING

In this type of Scheduling, the tasks are usually assigned with priorities. At times it is necessary to run a certain task that has a higher priority before another task although it is running. Therefore, the running task is interrupted for some time and resumed later when the priority task has finished its execution.

SCHEDULING CRITERIA

There are certain criteria on the basis of which we can decide whether a given algo. Will meet a system's particular requirement or not. They are :

- CPU Utilization

To make out the best use of CPU and not to waste any CPU cycle, CPU would be working most of the time(Ideally 100% of the time). Considering a real system, CPU usage should range from 40% (lightly loaded) to 90% (heavily loaded.)

- Throughput

It is the total number of processes completed per unit time or rather say total amount of work done in a unit of time. This may range from 10/second to 1/hour depending on the specific processes.

- Turnaround Time

It is the amount of time taken to execute a particular process, i.e. The interval from time of submission of the process to the time of completion of the process(Wall clock time).

- Waiting Time

The sum of the periods spent waiting in the ready queue amount of time a process has been waiting in the ready queue to acquire get control on the CPU.

- Load Average

It is the average number of processes residing in the ready queue waiting for their turn to get into the CPU.

- Response Time

Amount of time it takes from when a request was submitted until the first response is produced. Remember, it is the time till the first response and not the completion of process execution (final response).

In general CPU utilization and Throughput are maximized and other factors are reduced for proper optimization.

I/O Scheduling

Now process can be basically of two types.

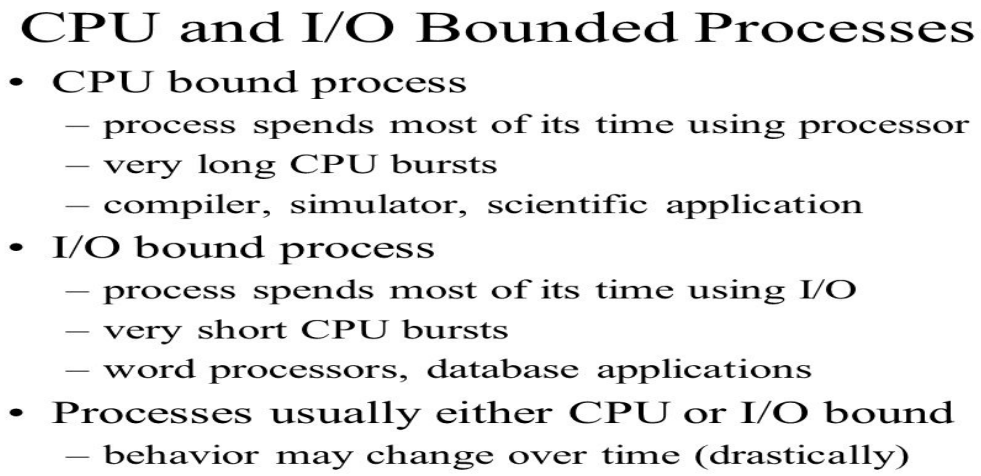
1. CPU bound: when a process has to perform some kind of computation and mathematical or logical operations all it requires is CPU time. During this time the CPU does not sit idle and instead performs the operations for the process and is hence considered busy at that particular moment.
2. I/O bound: Some processes are a bit complex in nature. Along with some computations, these processes also require to perform some I/O interactive tasks as well. For example some process might require taking input for an I/O devices for example keyboard and then perform computation on it. In such cases CPU has no role to play while the process is preforming I/O activity or waiting for a input or output response. So we can say the CPU is free when a process is busy in I/O activities and it can be utilize for other processes. The processes which have I/O demands can be said to require both CPU time as well as I/O time.

When several processes with various I/O requirement time arrives, scheduling must take in such a way that CPU utilization is maximize. Scheduling must ensure that I/O activities should not hinder CPU usage.

Hence all the scheduling algorithms are designed for both the kind of processes- process which requires only CPU time and processes which require both CPU and I/O time.

The interface designed by us takes care of both possible types of process.

In broad terms differences between both types of processes are given below:



FCFS-FIRST COME FIRST SERVE

First in, first out (FIFO), also known as first come, first served (FCFS), is the simplest scheduling algorithm. FIFO simply queues processes in the order that they arrive in the ready queue.

In this, the process that comes first will be executed first and next process starts only after the previous gets fully executed.

SOME FEATURES:

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Easy to understand and implement.
- Its implementation is based on FIFO queue.
- Poor in performance as average wait time is high.

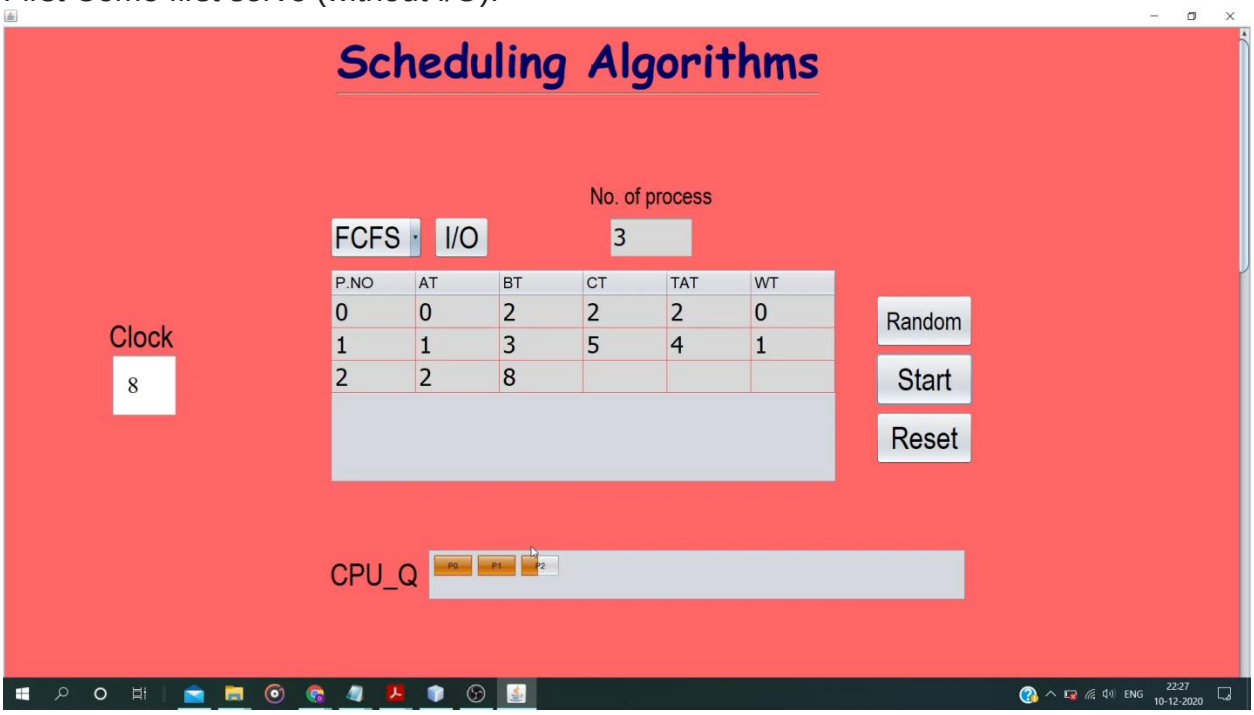
Advantages of FCFS

- The simplest form of a CPU scheduling algorithm
- Easy to program
- First come first served

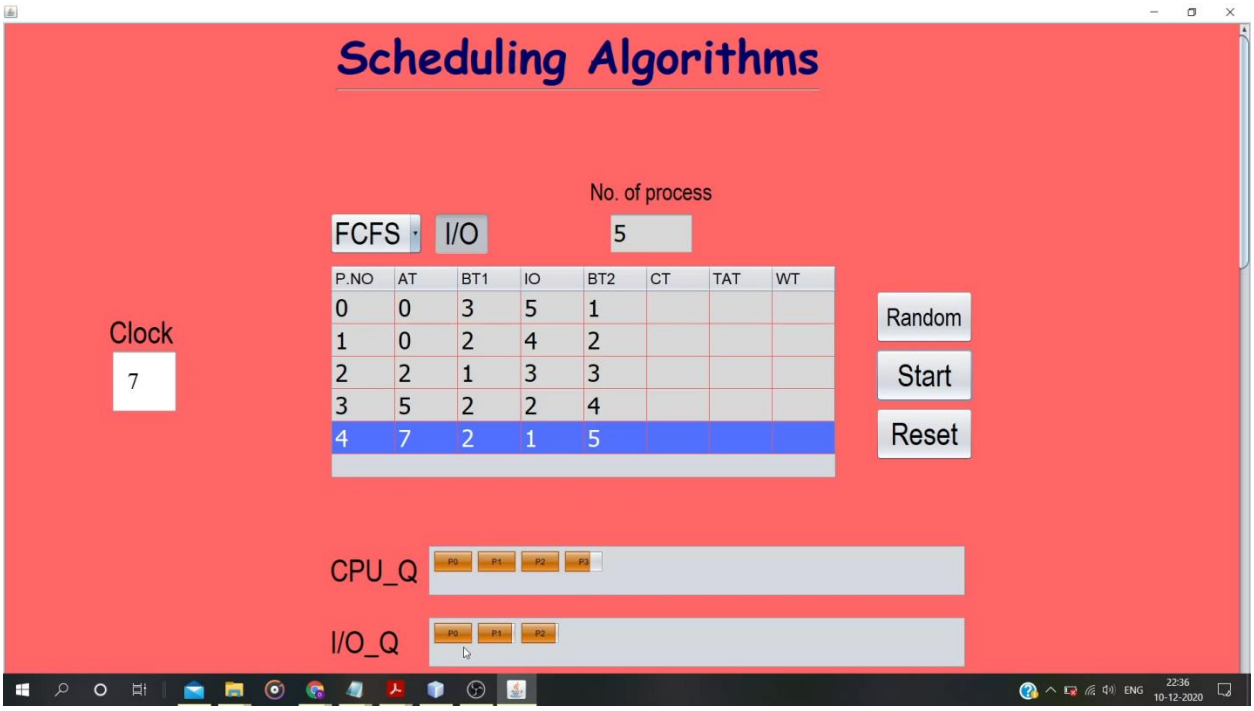
Disadvantages of FCFS

- It is a Non-Preemptive CPU scheduling algorithm, so after the process has been allocated to the CPU, it will never release the CPU until it finishes executing.
- The Average Waiting Time is high.
- Short processes that are at the back of the queue have to wait for the long process at the front to finish.
- Not an ideal technique for time-sharing systems.
- Because of its simplicity, FCFS is not very efficient.

First Come first serve (without I/O):



First come first serve (with I/O):



SJF - SHORTEST JOB FIRST

Shortest Job First (SJF) is an algorithm in which the process having the smallest execution time is chosen for the next execution.

Some Features:

- It is associated with each job as a unit of time to complete.
- This algorithm method is helpful for batch-type processing, where waiting for jobs to complete is not critical.
- It can improve process throughput by making sure that shorter jobs are executed first, hence possibly have a short turnaround time.
- It improves job output by offering shorter jobs, which should be executed first, which mostly have a shorter turnaround time.
- It is non-preemptive in nature (the preemptive version also exists but we call it SRTF)

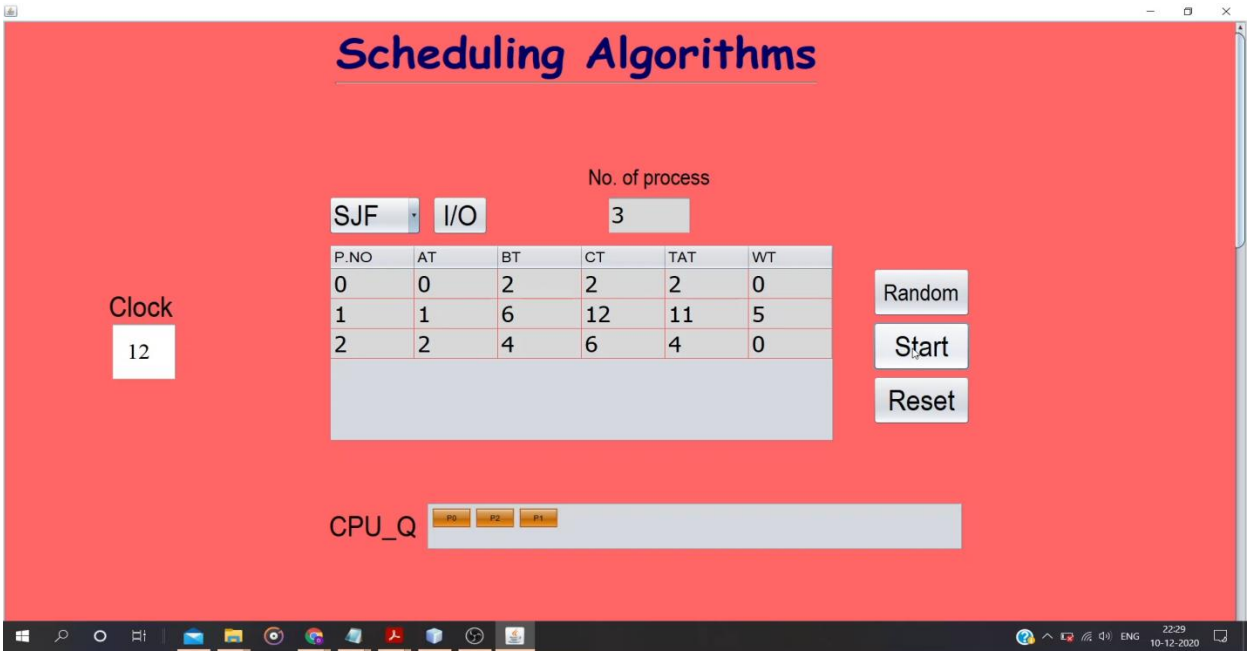
Advantages:

Reduces average wait time and average response time

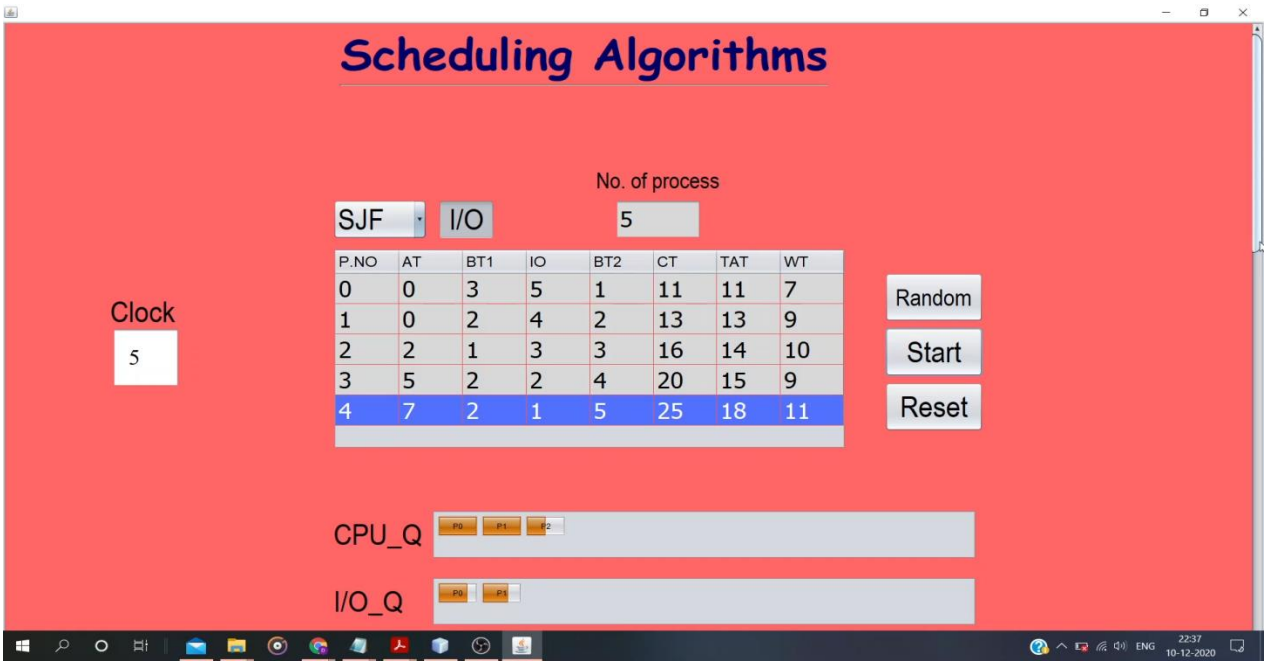
Disadvantages:

- May starve long jobs
- Not practical difficult to predict burst time

Shortest Job First (without I/O):



Shortest Job First (with I/O):



LJF – LONGEST JOB FIRST

Longest Job First (LJP) is a non-preemptive scheduling algorithm. This algorithm is based upon the burst time of the processes. The processes are put into the ready queue based on their burst times i.e., in a descending order of the burst times. As the name suggests this algorithm is based upon the fact that the process with the largest burst time is processed first. The burst time of only those processes is considered that have arrived in the system until that time. Its preemptive version is called Longest Remaining Time First (LRTF) algorithm.

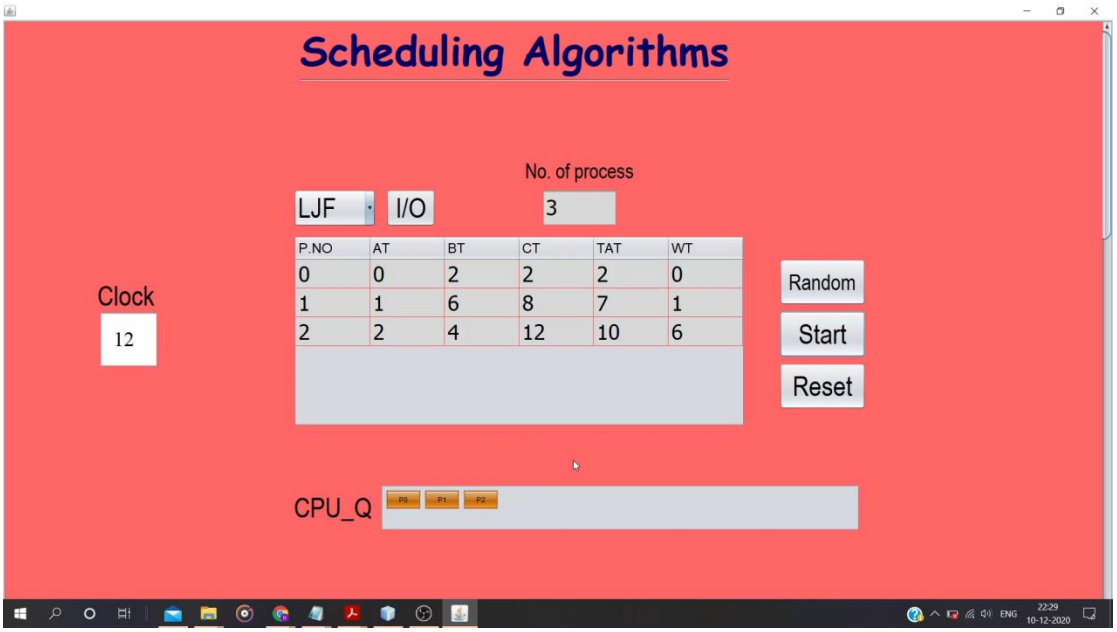
Some Features:

- This is non-preemptive
- Job with longest burst time is processed first

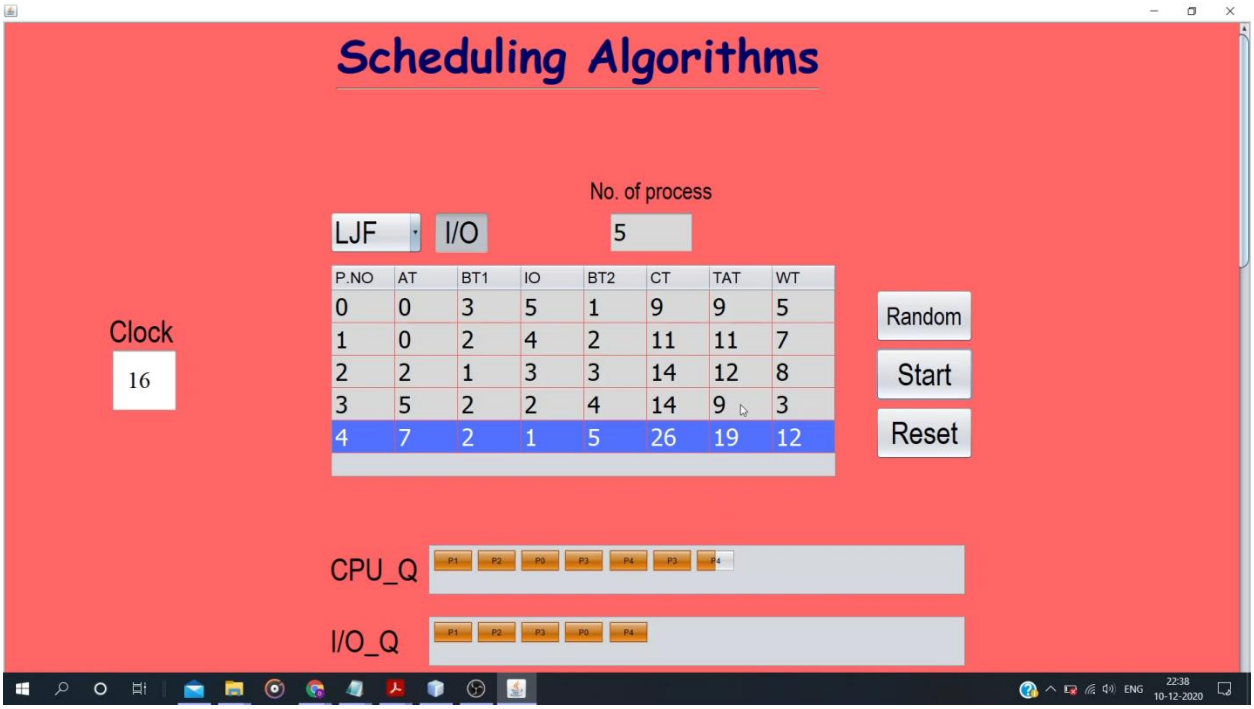
Disadvantages:

- This algorithm gives very high average waiting time and average turn-around time for a given set of processes.
- This may lead to convoy effect.
- It may happen that a short process may never get executed and the system keeps on executing the longer processes.
- It reduces the processing speed and thus reduces the efficiency and utilization of the system.

Longest Job First (without I/O):



Longest Job First (with I/O):



RR – ROUND ROBIN

The name of this algorithm comes from the round-robin principle, where each person gets an equal share of something in turns. It is the oldest, simplest scheduling algorithm, which is mostly used for multitasking.

In Round-robin scheduling, each ready task runs turn by turn only in a cyclic queue for a limited time slice. This algorithm also offers starvation free execution of processes.

Special Features:

- Round robin is a pre-emptive algorithm
- The CPU is shifted to the next process after fixed interval time, which is called time quantum/time slice.
- The process that is preempted is added to the end of the queue.
- Round robin is a hybrid model which is clock-driven
- Time slice should be minimum, which is assigned for a specific task that needs to be processed. However, it may differ OS to OS.
- It is a real time algorithm which responds to the event within a specific time limit.
- Round robin is one of the oldest, fairest, and easiest algorithms.
- Widely used scheduling method in traditional OS.

Advantages:

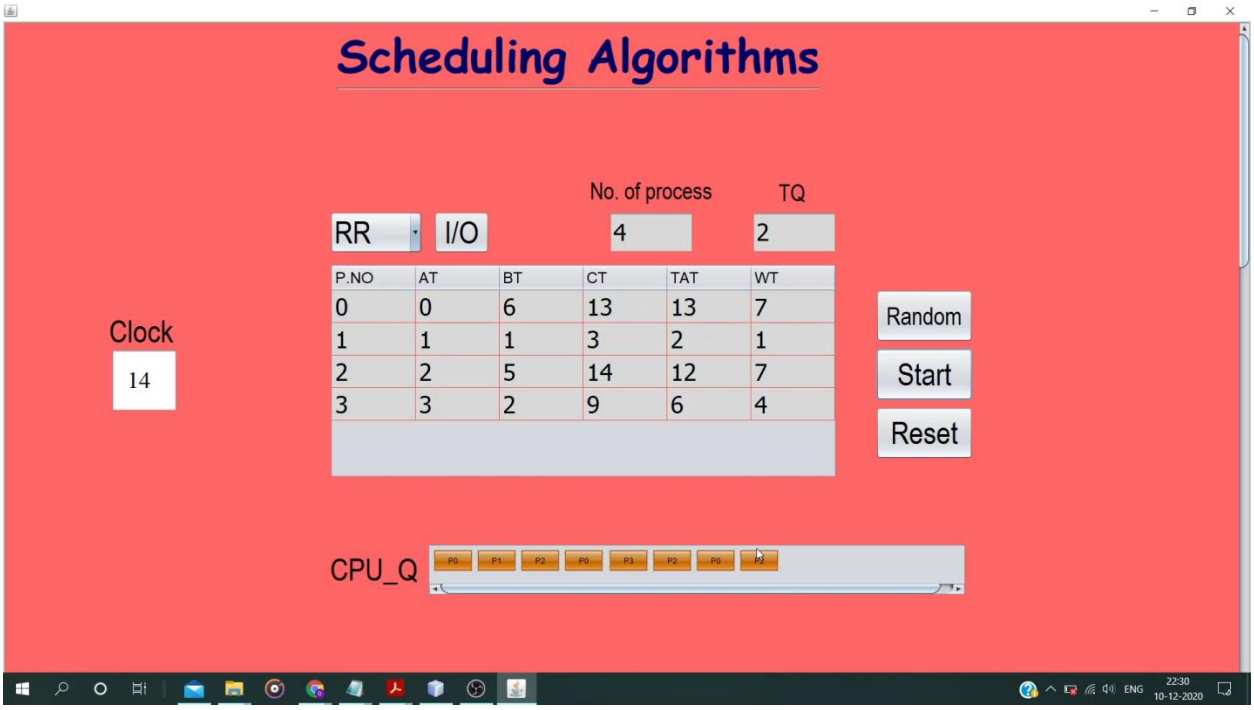
- It doesn't face the issues of starvation or convoy effect.
- All the jobs get a fair allocation of CPU.
- It deals with all process without any priority
- If you know the total number of processes on the run queue, then you can also assume the worst-case response time for the same process.
- This scheduling method does not depend upon burst time. That's why it is easily implementable on the system.
- Once a process is executed for a specific set of the period, the process is preempted, and another process executes for that given time period.
- Allows OS to use the Context switching method to save states of preempted processes.
- It gives the best performance in terms of average response time.

Disadvantages:

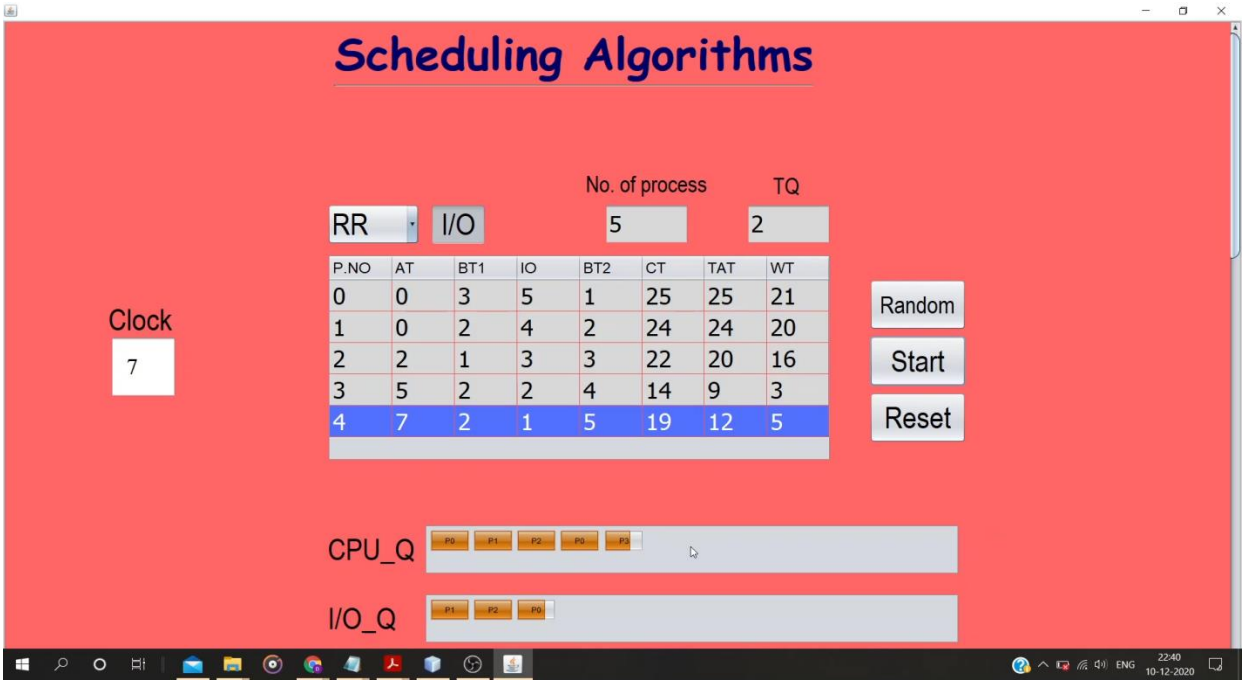
- If slicing time of OS is low, the processor output will be reduced.
- This method spends more time on context switching
- Its performance heavily depends on time quantum.
- Priorities cannot be set for the processes.
- Round-robin scheduling doesn't give special priority to more important tasks.

- Decreases comprehension
- Lower time quantum results in higher the context switching overhead in the system.
- Finding a correct time quantum is a quite difficult task in this system.

Round Robin (without I/O):



Round Robin (with I/O):



SRTF – SHORTEST REMAINING TIME FIRST

This Algorithm is the preemptive version of SJF scheduling. In SRTF, the execution of the process can be stopped after certain amount of time. At the arrival of every process, the short term scheduler schedules the process with the least remaining burst time among the list of available processes and the running process.

Once all the processes are available in the ready queue, No preemption will be done and the algorithm will work as SJF scheduling.

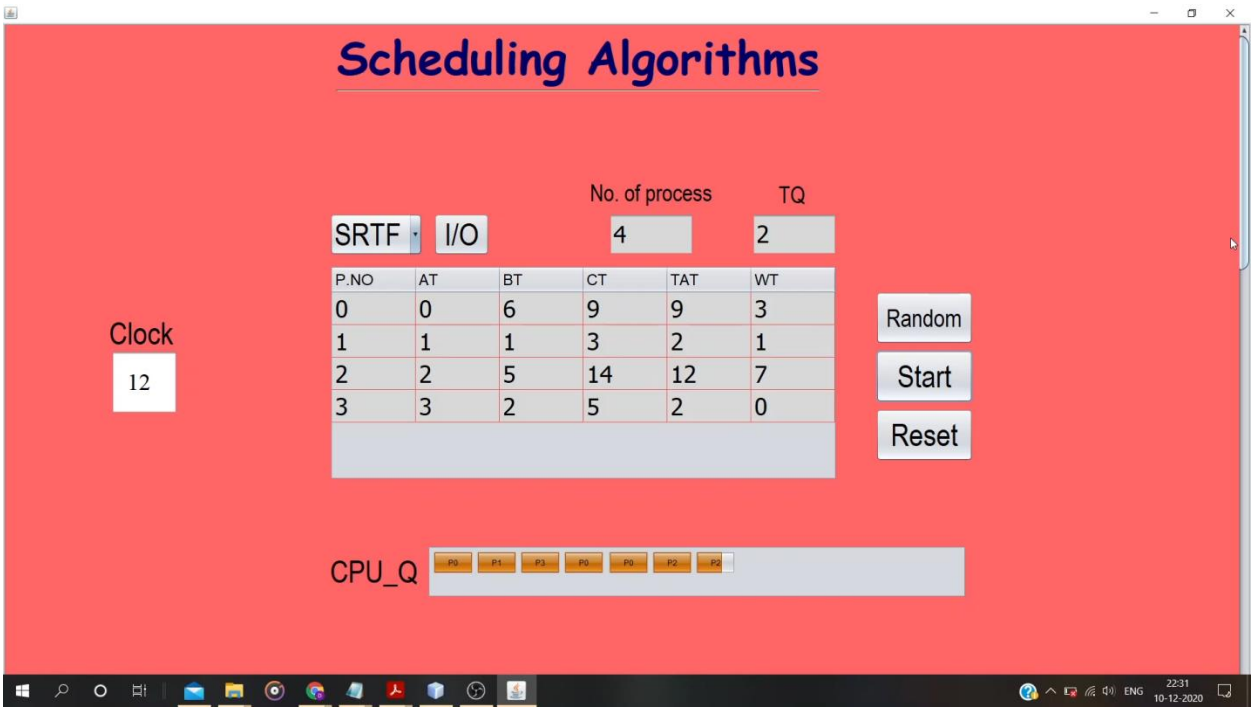
Advantage:

SRTF algorithm makes the processing of the jobs faster than SJN algorithm, given it's overhead charges are not counted.

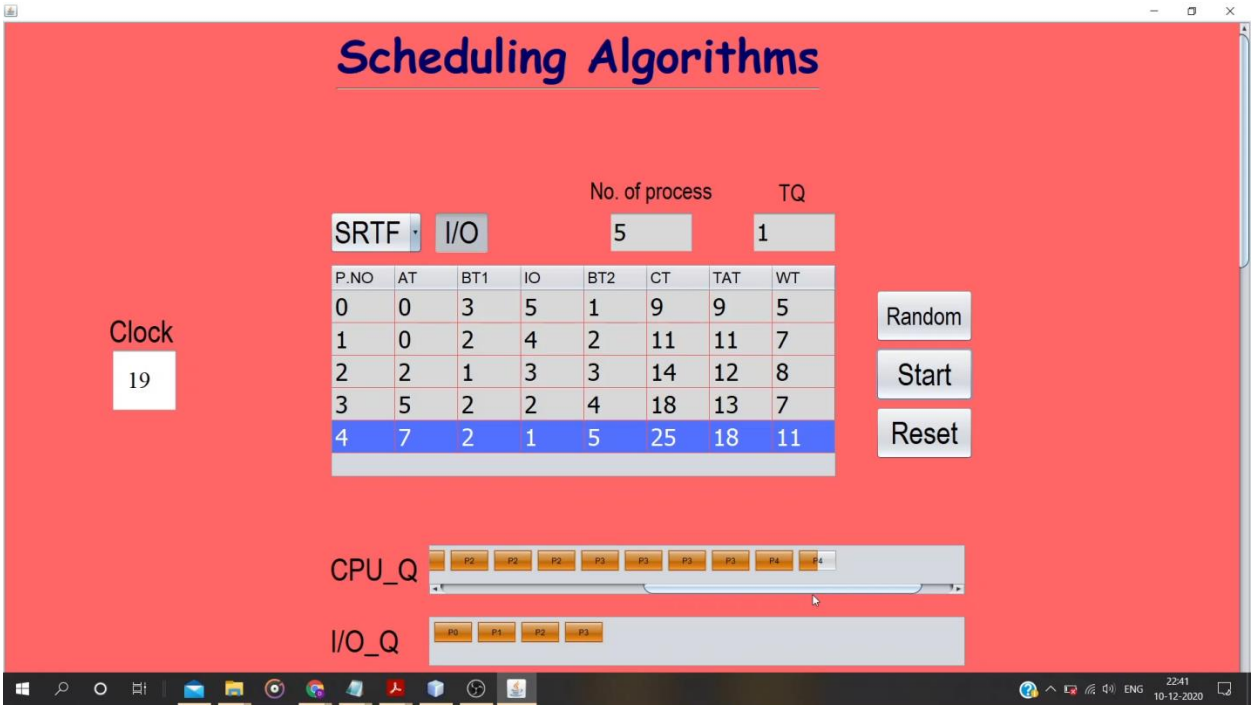
Disadvantages:

The context switch is done a lot more times in SRTF than in SJN, and consumes CPU's valuable time for processing. This adds up to it's processing time and diminishes it's advantage of fast processing.

Shortest Remaining Time First (without I/O):



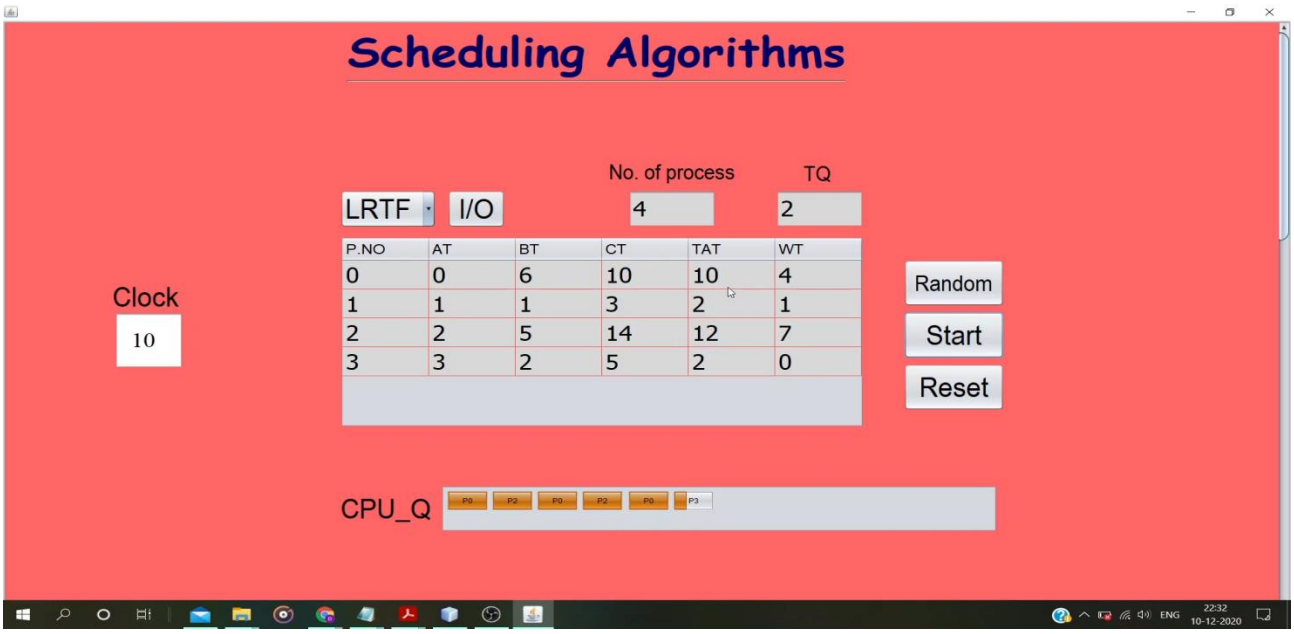
Shortest Remaining Time First (with I/O):



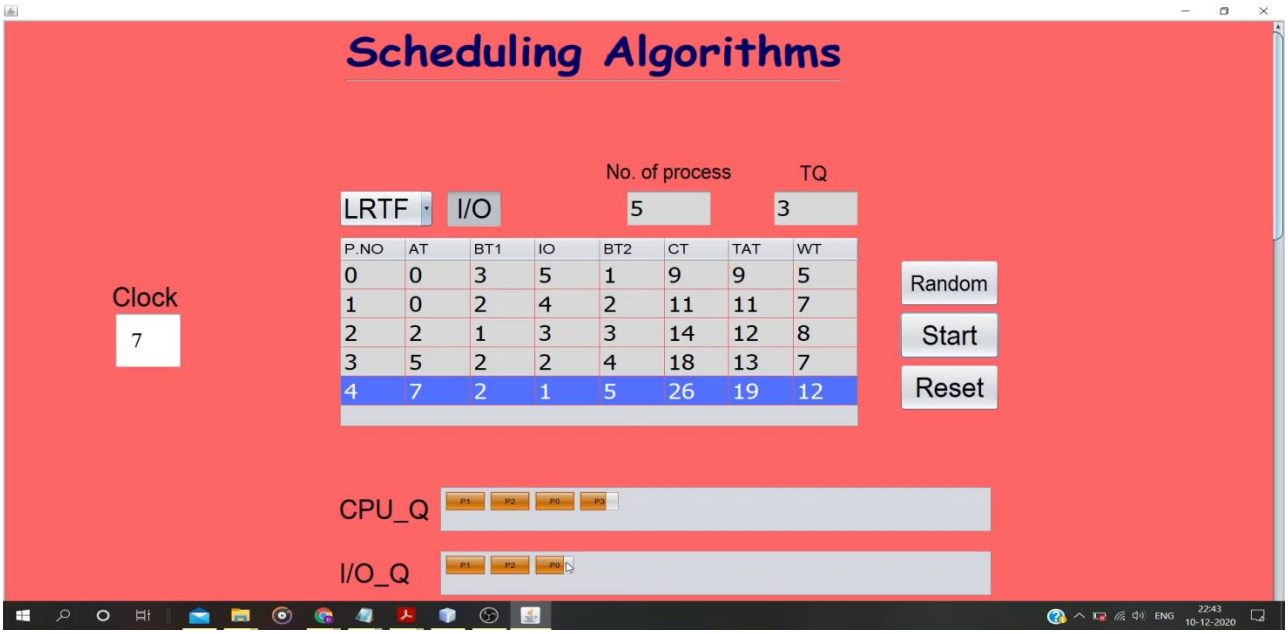
LRTF – LONGEST REMAINING TIME FIRST

This is a pre-emptive version of Longest Job First (LJF) scheduling algorithm. In this scheduling algorithm, we find the process with the maximum remaining time and then process it. We check for the maximum remaining time after some interval of time (say 1 unit each) to check if another process having more Burst Time arrived up to that time.

Longest Remaining Time First (without I/O):



Longest Remaining Time First (with I/O):



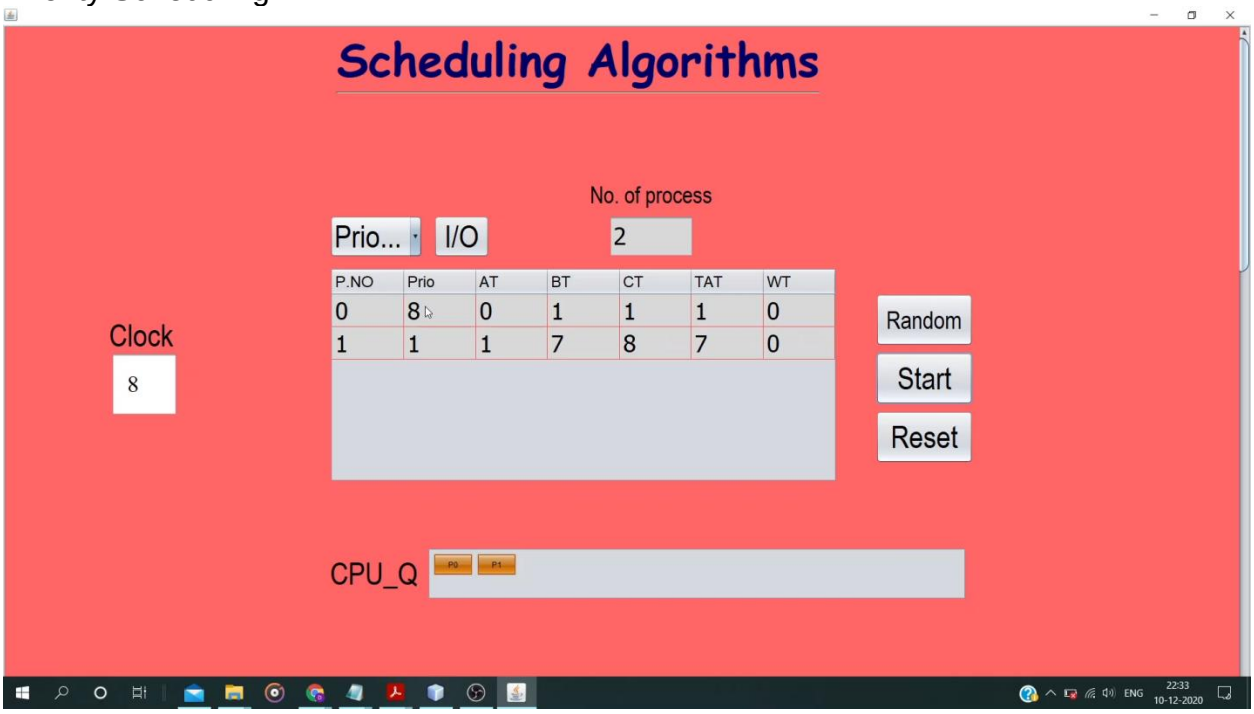
PRIORITY

In priority scheduling, each process has a priority which is an integer value assigned to it. The smallest integer is considered as the highest priority and the largest integer is considered as the lowest priority. The process with the highest priority gets the CPU first.

In rare systems, the largest number might be treated as the highest priority also so it all depends on the implementation.

If priorities are internally defined then some measurable quantities such as time limits, memory requirements, the number of open files and the ratio of average I/O burst to average CPU burst are used to compute priorities.
External priorities are assigned on the basis of factors such as the importance of process, the type and amount of funds been paid for computer use, the department sponsoring of the work, etc.

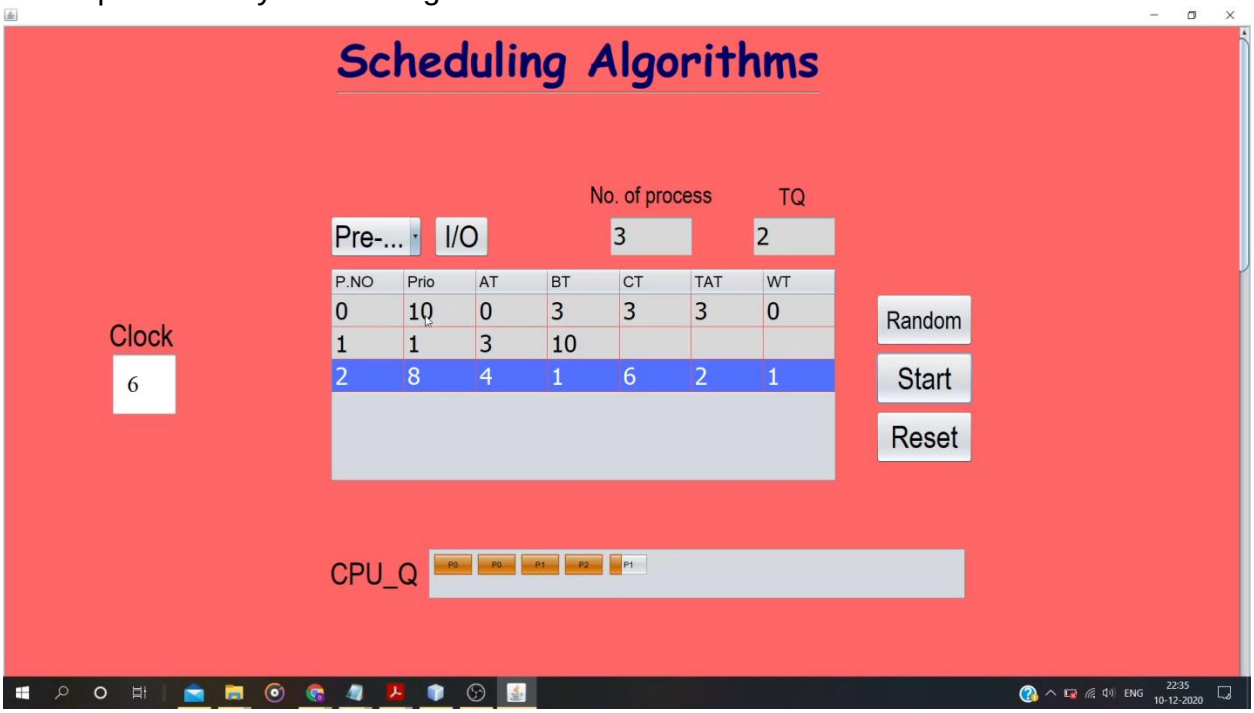
Priority Scheduling:



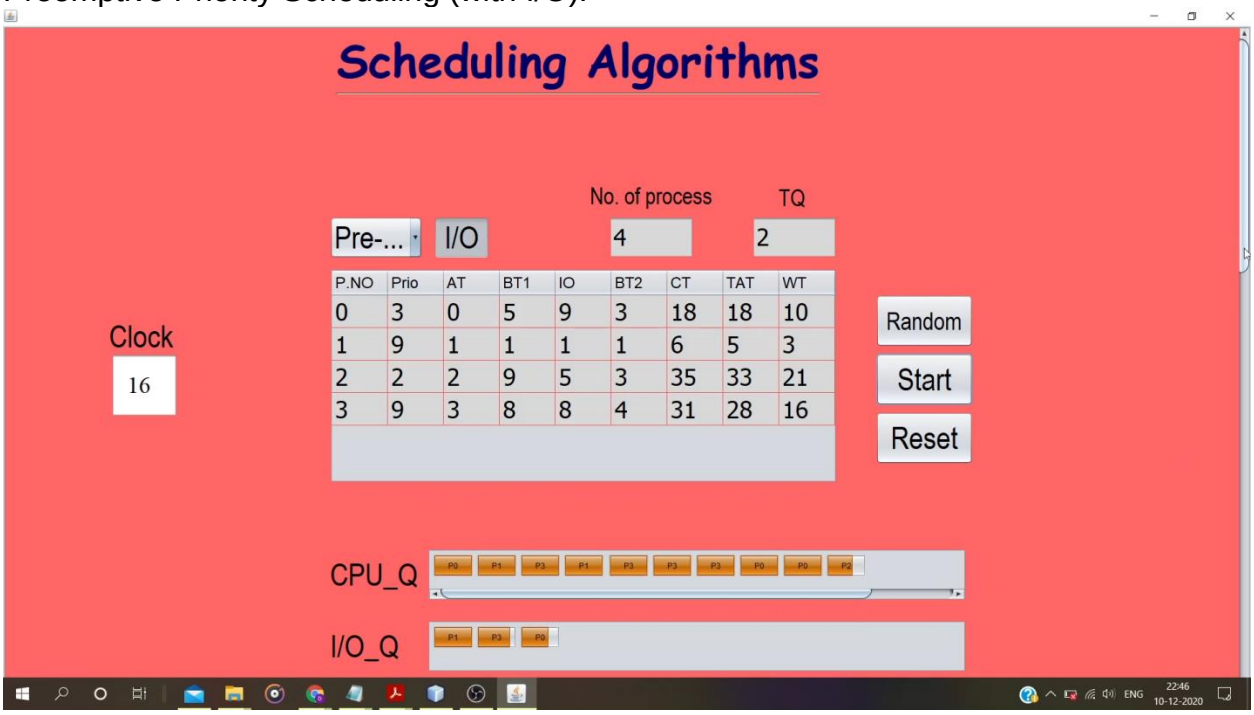
PREEMPTIVE PRIORITY

Sometimes it is important to execute higher priority tasks immediately even when a task is currently being executed. For example, when a phone call is received, the CPU is immediately assigned to this task even if some other application is currently being used. This is because the incoming phone call has a higher priority than other tasks. This is a perfect example of priority preemptive scheduling. If a task with higher priority than the current task being executed arrives then the control of the CPU is taken from the current task and given to the higher priority task.

Preemptive Priority Scheduling:



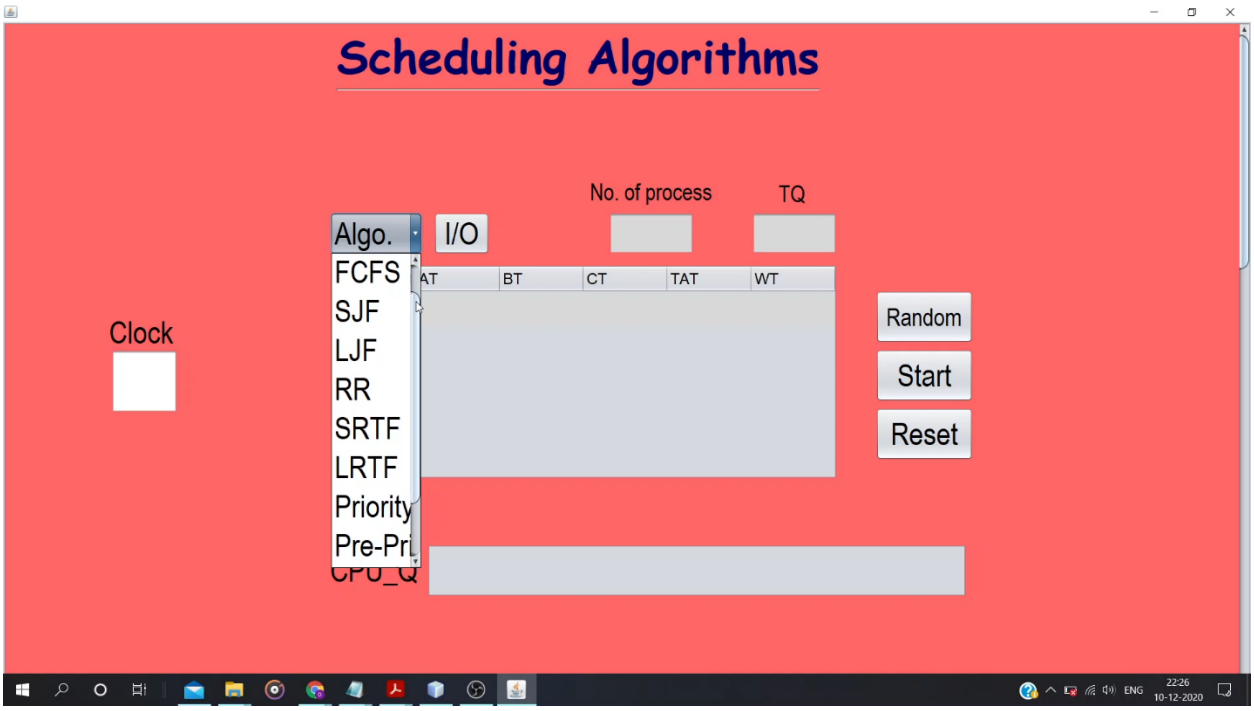
Preemptive Priority Scheduling (with I/O):



ADDITIONAL FEATURES

In the GUI that we created several other additional features are also included. These features have their own significance. Brief detail about each one of them is included in this section:

1. **Easy Selection Of Algorithms:**
   With the help of algorithm selector in GUI algorithms can be selected hassle free, even for the same data if we want to compare performance of different algorithms, we can easily do that by resetting animation and starting again without having to fill the data again.



2. **Detailed Performance Report:**
   After the end of animation by scrolling down, you can find detailed report of performance of the selected algorithm on given data.
   Details provided: -
   - Average Turn-Around Time
   - Average Wait Time
   - Total Context Switches
   - Total Stall Pulses
   - CPU Efficiency

3. **Gant Chart:**

   Scrolling further down from here, you will see Gant chart (used for detailed exploration of the performance of selected algorithm) of all the Processes with time axis.