# Simulation for scheduling Algorithm

## Introduction

We are assigned to develop a simulation test programme to understand the scheduling of Operating systems.

## Our Basic idea to develop the algorithm

We have developed an OS scheduling simulation using visual C#. This program will demonstrate how a scheduling algorithm works by an OS.

We choose Round Robin scheduling algorithm for our simulation.

That means a specific time quantum will be allocated to each process and every process will get a chance to be processed. Thus we can avoid starvation.

Even though we created our simulation with round robin concept we included some extra features to increase the performance of round robin algorithm.

We included priority to processes and we change priority regarding process age in queues so that we can give better performance for the larger processes in the suspended queue.

This is a pre-emptive type of algorithm it will break processes when running regarding to time out or IO requirement of the process.

Our algorithm will arrange all queues regarding to process priority. Priority will be set regarding to user desire and aging in queues.

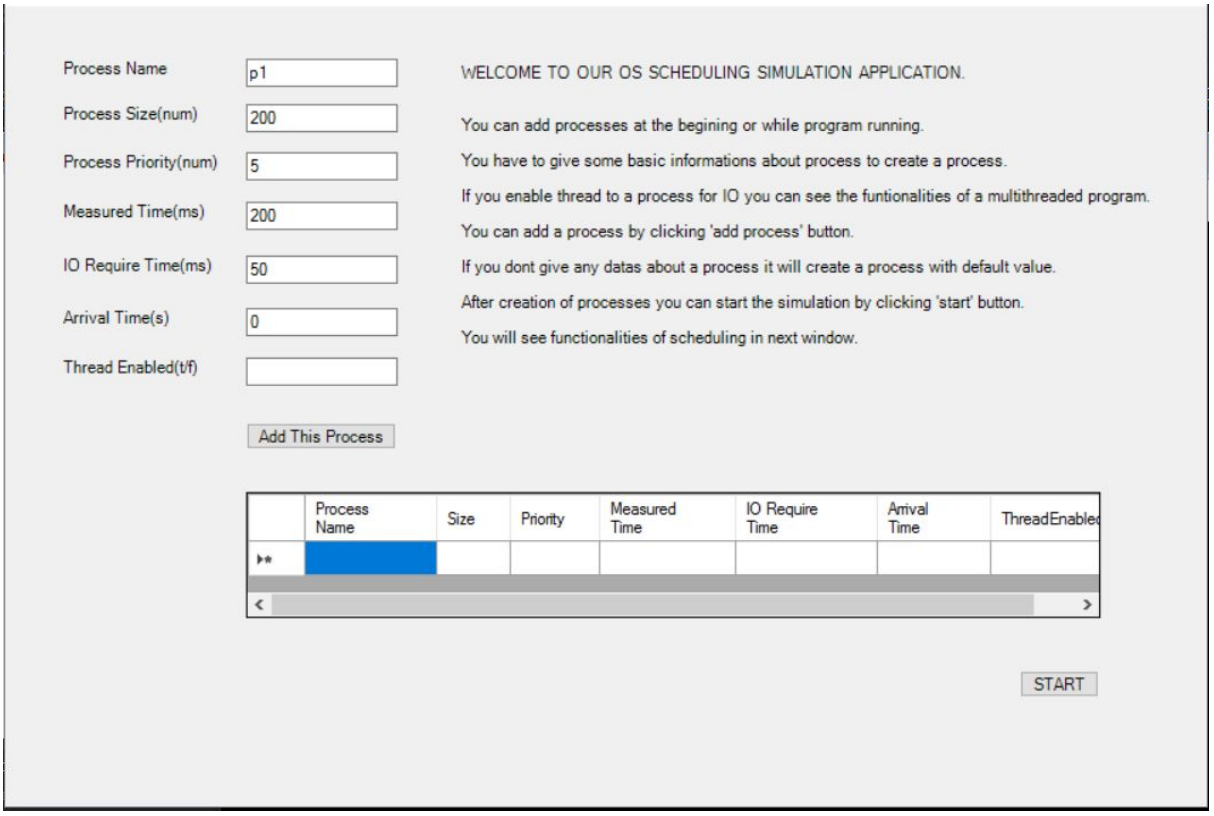
When we create a process, we have the right to give that process is thread enabled or not.

If our process created is ‘ thread enabled’ it means this process will act as a multithreaded process it will create a thread when it requires an IO and that thread will wait in the waiting Queue while mother process has ability to run in CPU at the same time.

We created GUI for demonstrate the algorithm and we designed this to show processes in the ready queue, suspended ready queue, waiting queue, suspended waiting queue, new queue and activities in the CPU.

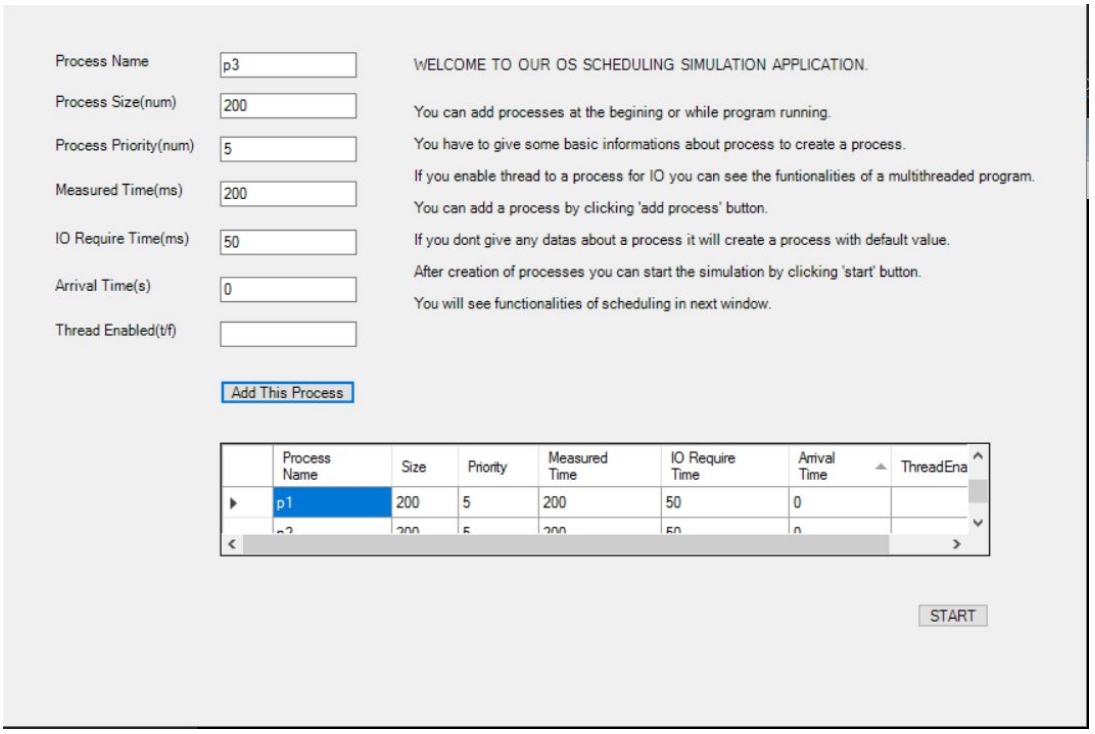
This algorithm can be used in modern OS to increase concurrency of the processes. We created ready queue and waiting queue with limited size and other queues are unlimited size because others are in secondary memory.

## Project Demonstration.



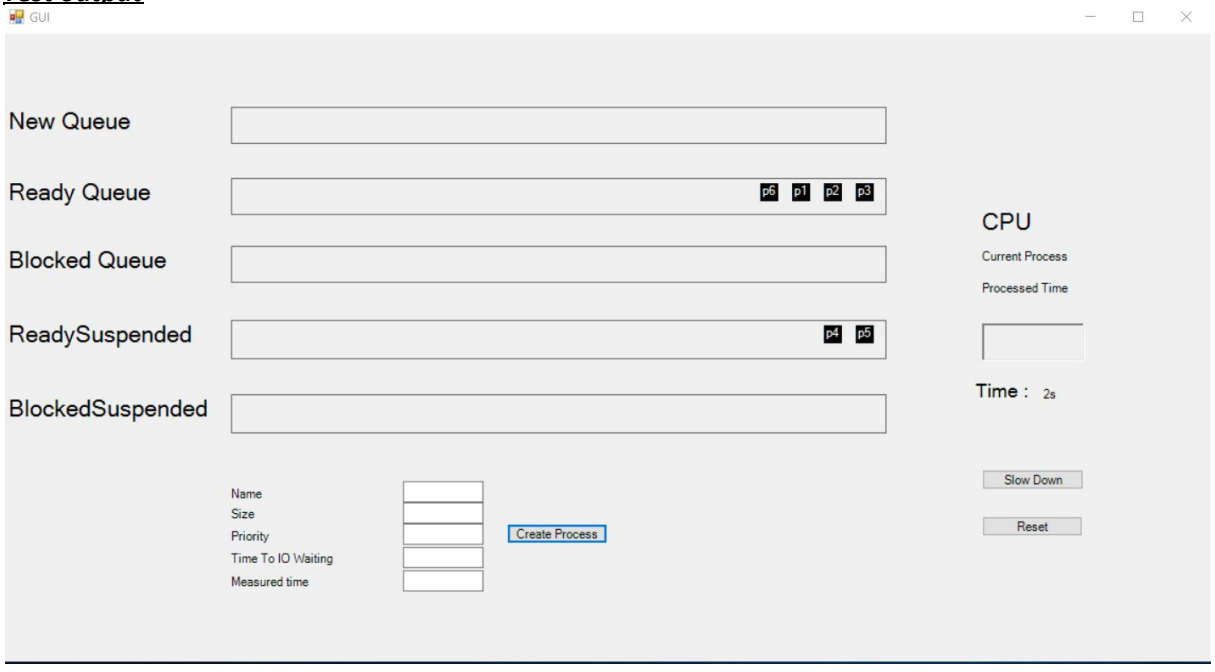

We can include process name, size, priority, measured time, IO require time and arrival time as we wish.

And also we should include ‘t’ or ‘f’ to determine whether the process is thread enabled or not. That means if thread is enabled the main process will create a thread to get the IO.



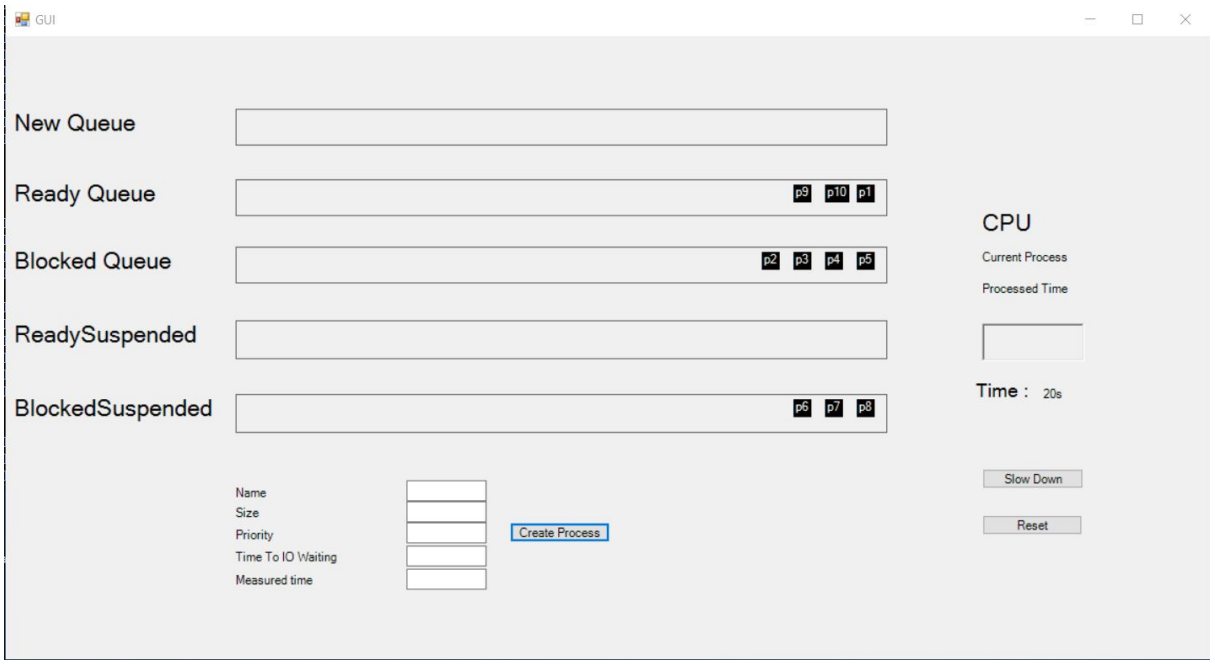

Like this we add so many process to demonstrate our algorithm

***Test output***




After we run our test this graphical demonstration will be appeared.It contains the details of Newly created process queue,Ready queue,Blocked queue,ready and suspended queue and blocked and suspended queue.The process which can not accommodate in main memory will be suspended and

pushed in to ready suspended queue to the secondary memory.So the process p4 and p5 here are shown as suspended here are in secondary memory as the ready queue in main memory is full.




This shows that processes p2, p3 , p4, p5,p6,p7 and p8 are blocked as they wait for an IO. And if that blocked queue is full, then those processes will be pushed to secondary memory into a blocked suspended queue.

**<u>Our System benefits</u>**

We can add another process while the test is running.

We can slow down the processing and back to the normal speed. As it is a simulation program this slowing down attribute helps a lot to study and understand the concepts.

We used AGING so every process will run even though they created with low priority.

If a big size process is created it has no room in main memory to be in ready queue. So it will be suspended. But due to aging it also get a chance to enter in to a ready queue after a while thus improves the concurrency.

If big processes are in ready queue the system will reduce its priority according to its size.( The big sized processes will be allocated with low priority ).We should consider that big process also should run ,but they should not interrupt other small processes which be in ready queue. So aging is used to increase the priority and sized priority technique is used to decrease its priority. It’s a trade-off.