**Christiaan Markus Madeleyn 27211452**

**Divan van Kruiselbergen 26378396**

**ITRW316**

**Assignment 2C**

**2019/03/17**

**Index:**

Introduction 1

Body 1

Conclusion 5

References 6

**Introduction:**

The program is a simulation of different types of scheduling algorithms, namely: Shortest job first, Round Robin, Priority and Multi Queue.

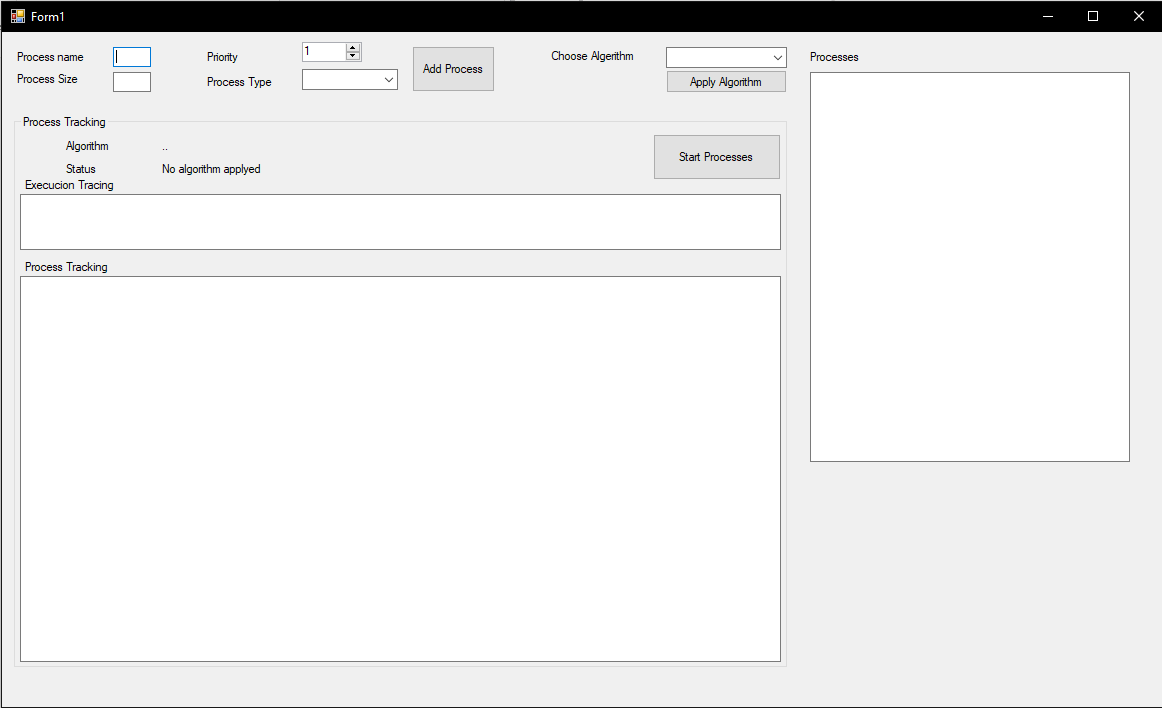
**\*/**Scheduling is when a computer is multiprogrammed, it frequently has multiple processes or threads competing for the CPU at the same time. This situation occurs whenever two or more of them are simultaneously in the ready state. If only one CPU is available, a choice has to be made to be made which process to run next. The part of the operating system that makes the choice is called the scheduler, and the algorithm it uses is called the scheduling algorithm (Tanenbaum, n.d.).

C# was the chosen programming language as it comes with pre added GUI forms and as we were more experienced with this language.

**Body:**

On the GUI of our program you can add process were you give the process a name, size, priority and a process type. You can then apply an Algorithm to the process. When both are applied you can start processes that will do the scheduling simulation.

It will also keep track of the processing and execution.

The GUI looks as follows:

The process size is the amount of activities that is in the process itself, it represents the amount of time it will take for the process to finish. The process priority is scaled base from 1 – 10, and this will indicate to high or low level of the process.

Process type has four different scheduling environments, namely: Batch, Game, Primary, Secondary. (Only to categorize the process, they are only for simulation purposes). Different environments need different scheduling algorithms. This is because different application areas have different goals. So the scheduler optimizations are not the same in all systems (Tanenbaum, n.d.).

Each process that is added on the GUI is saved on a list.

As said the program can perform four different algorithms, namely: Shortest job first, Round Robin, Priority and Multi Queue.

The Shortest job first algorithm takes the processes saved on the list and sorts them from shortest process to longest process. After words it iterates the list and executes each process.

The Priority algorithm takes the processes stored in the list and sorts them using priority attribute and then executes.

Priority If statement:

if (comboBox1.Text == "Priority")

{

foreach (Process p in Priotitylys)

{

int i = 1;

String name = p.getName();

textBox7.Text += "Priority: " + p.getPriority() + "\t\t";

while (i <= p.getTime())

{

textBox6.Text += name + i.ToString() + " ";

textBox7.Text += name + i.ToString();

i++;

}

textBox7.Text += "\n\r\n ";

}

}

The Round Robin algorithm uses a while loop with two for loops. The while loop continues when there is a process in the list that has not yet been properly executed. Within the while loop there is a for loop that run through each process in the list and iterates them. Within that for loop there is another for loop that executes five time. It runs five times since this is my chosen quantum size is for the Round Robin algorithm. By using an if statement it is constantly keeping track of the amount of times the for loops and while loop runs, the amount of times the processes have been executed successfully, the processes that have not been executed successfully and the processes that are waiting their turn until they have been executed.

Round Robin While Loop:

while (i != 0)

{

foreach (Process p in temp)

{

int time = p.getTime() - (5 \* cycle);

if(time>=0)

{

textBox7.Text += "Quantum: ";

for (int x =0;x<5; x++)

{

textBox7.Text += p.getName()+((p.getTime()-(5\*cycle))-x).ToString()+" ";

}

textBox7.Text += "\n\r\n\r";

}

else if(time>=-5)

{

i--;

}

}

cycle++;

textBox7.Text +=" \n\r\n\r ";

}

These three algorithms uses the class Process.cs that saves the process name, process size, process priority and process type values.

The Multi Queue Algorithm makes use of the class typeClass.cs. Within this class there are four objects namely: Batch, Game, Primary and Secondary. These objects represent the type processes that are input by the user (Process Type on GUI). The processes are placed at their type object, an algorithm breaks the processes into their activities and stores it into their succession activities. With the four objects that has each own a series of activities (the series owns all the activities of each process of their type). The objects gets placed into a list that gets sorted in order of priority. The game object gets executed first followed by the primary object, then the secondary object and then finally the batch object activities. A quantum of 20 and an algorithm that differs a little form the round robin algorithm is used. There is a while loop that iterates until each object’s activities are fully executed. A for loop within the while loop iterates by each object in the list, another for loop within that for loop runs through 20 quanta’s.

Multi Queue While Loop:

while (i <= 3)

{

foreach (TypeClass t in newList)

{

String disp =String.Format("{0,10} \t\t",t.getName());

String[] arr = t.getPro();

int j = 0;

if (arr.Length > 20 \* (cycle + 1))

{

for (int x = 20 \* cycle; x < 20 \* (cycle + 1); x++)

{

textBox6.Text +=arr[x].ToString()+ " ";

disp += arr[x].ToString();

j++;

}

textBox7.Text +=disp;

}

else if(arr.Length-(20 \* (cycle + 1)) <=-20)

{ }

else

{

for (int x = 20 \* cycle; x < 20 \* (cycle + 1); x++)

{

try

{

textBox6.Text +=arr[x].ToString()+ " ";

disp += arr[x].ToString();

j++;

}

catch { break; }

}

textBox7.Text += disp;

i++;

}

textBox7.Text += "\n\r\n\r ";

}

cycle++;

}

Each quantum represents one activity that belongs to a process of the type of object were the first for loop currently is. This means that the processes that belongs to the game object executes 20 activities, these activities can belong to more than one process or only a part of a process that has more than 20 activities. If the 20 quanta’s have finished executing, then the for loop moves on to the next object in the type list and runs another 20 quanta’s. This get repeated until all the activities in each type object has been fully executed.

The shortest job first and priority algorithms were fairly easy to program.

Problems that raised during the coding of this program was as follows. With the Round Robin algorithm is was hard keeping track of the processes that was not yet fully executed and the processes that were fully executed. In other word it was hard to continue with different processes after one was successfully executed.

The same problem came with the Multi Queue algorithm, but here the dividing of the process activities and putting them into typeClass.cs was especially hard. This was due to pore planning. His was solved by going to pen and paper and trying to plan wat to do, we discovered that we needed an extra class to get the algorithm to perform best.

**Conclusion:**

**References:**

Tanenbaum, A. (n.d.). *Modern operating systems*. 3rd ed. p.143 - 150.

