Divan van Kruiselbergen

26378396

Christiaan Markus Madeleyn

27211452

Abstract

Report about concepts, algorithms and a simulation program related to process scheduling for modern operation systems

Scheduling Report

ITRW316 Assignment 2C

Contents

[Introduction**:** 2](#_Toc3815913)

[Body: 2](#_Toc3815914)

[Our program: 2](#_Toc3815915)

[The algorithms: 3](#_Toc3815916)

[1. Shortest job first 3](#_Toc3815917)

[How we simulated it 3](#_Toc3815918)

[Screen Shot of execution: 3](#_Toc3815919)

[2. Priority Scheduling 3](#_Toc3815920)

[How we simulated it 4](#_Toc3815921)

[If statement code snippet: 4](#_Toc3815922)

[Screen Shot of execution: 4](#_Toc3815923)

[3. Round-Robin Scheduling 5](#_Toc3815924)

[How we simulate it 5](#_Toc3815925)

[While loop code snipped: 5](#_Toc3815926)

[Screen Shot of execution 6](#_Toc3815927)

[4. Multiple Queues 6](#_Toc3815928)

[How we simulate it 6](#_Toc3815929)

[Multiple queue algorithm, code snipped: 6](#_Toc3815930)

[Screen Shot of execution 7](#_Toc3815931)

[**Conclusion:** 8](#_Toc3815932)

[Bibliography 8](#_Toc3815933)

# 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.

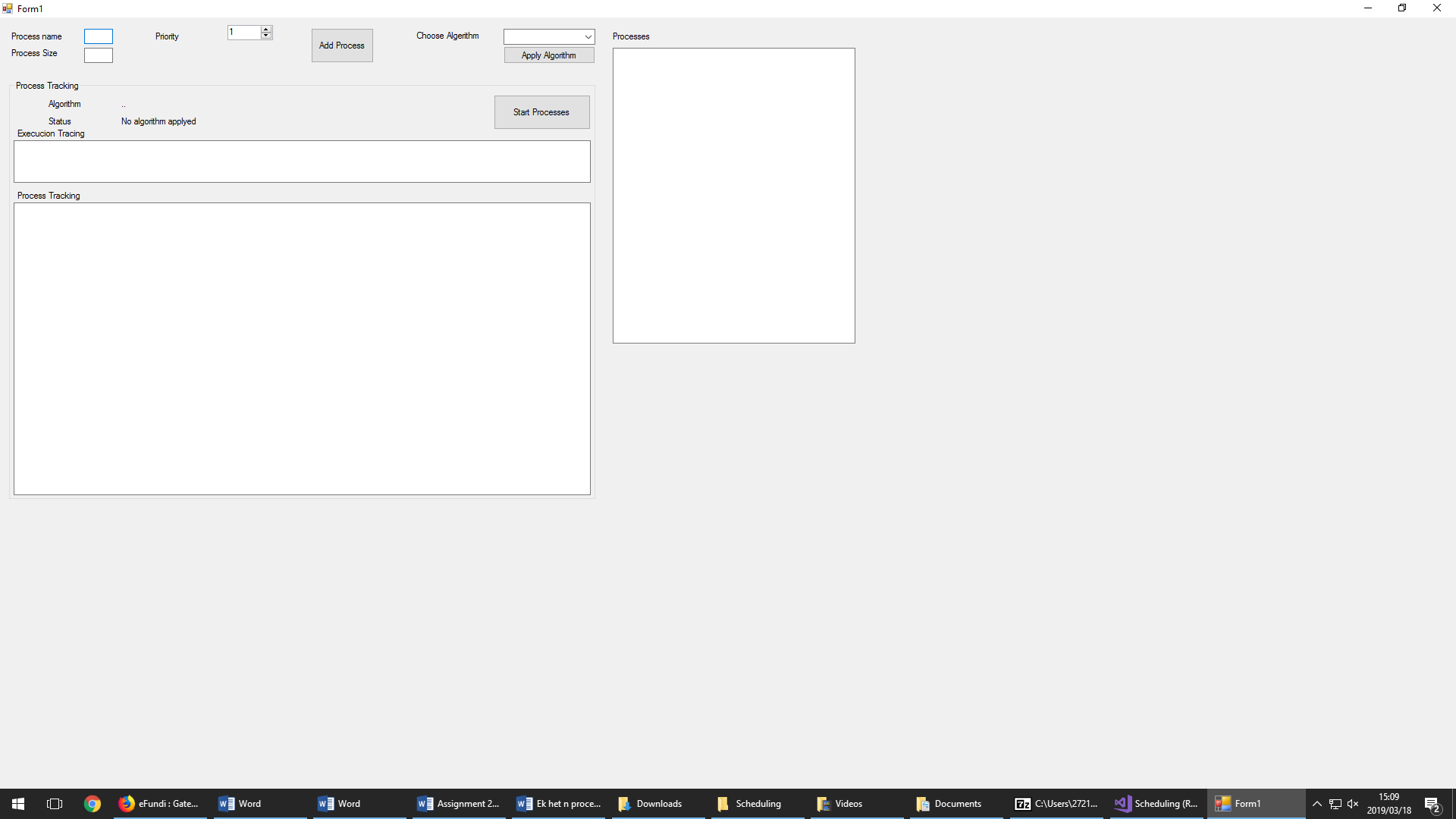
## Body:

## Our program:

On the GUI of our program you can add processes were you give the process a name, size, priority and a process type. Choose the scheduling algorithm you want to use and apply it. Clicking start processes will implement the chosen algorithm and execute the simulation.

The program keeps track of the activities of each process’s 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 or quanta it will take for the process to finish. The process priority is scale based from 1 – 10, and this will indicate to high or low priority and will be used to schedule the processes in the priority scheduling algorithm.

# The algorithms:

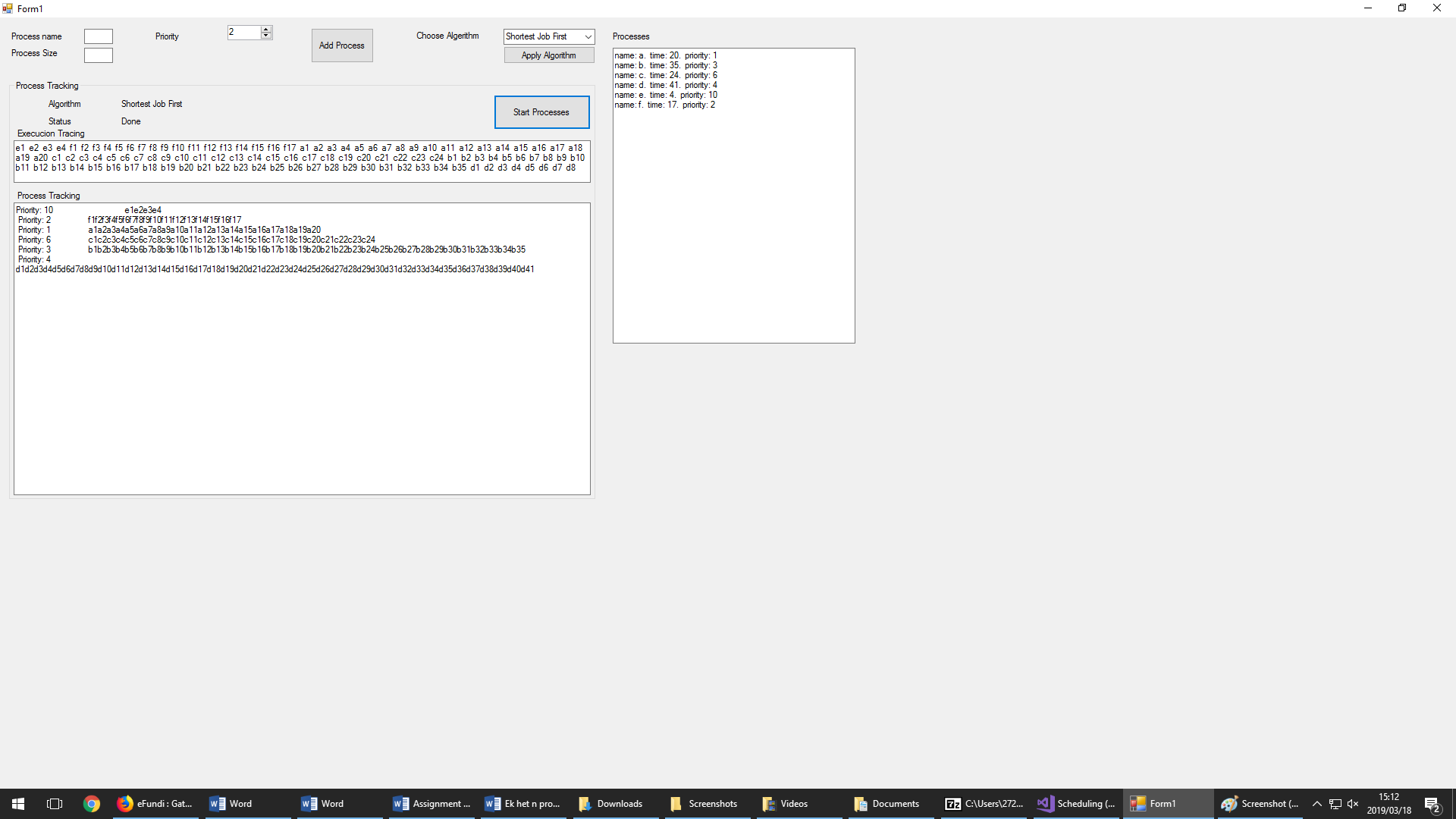
### Shortest job first

The first algorithm we simulate is the shortest job first algorithm. This algorithm is used in the batch process environment. It always run the process with the shortest runtime next and therefor it is assumed that the runtime is known. This algorithm has the shortest average turnaround time of all the algorithms.

### How we simulated it

When the user adds processes to the using the GUI he enters the process size which in this case refers to the runtime. The Shortest job first algorithm takes the processes saved on the list and sorts them from shortest runtime to longest runtime. Then each process is executed in the order of the sorted list.

### Screen Shot of execution:



### Priority Scheduling

The basic idea is straight forward: each process is assigned an priority, and t5he runnable process with the highest priority is run next.

### How we simulated it

Similarly to shortest job first this algorithm takes the processes stored in the list and sorts them using the priority attribute and then executes.

### If statement code snippet:

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++;

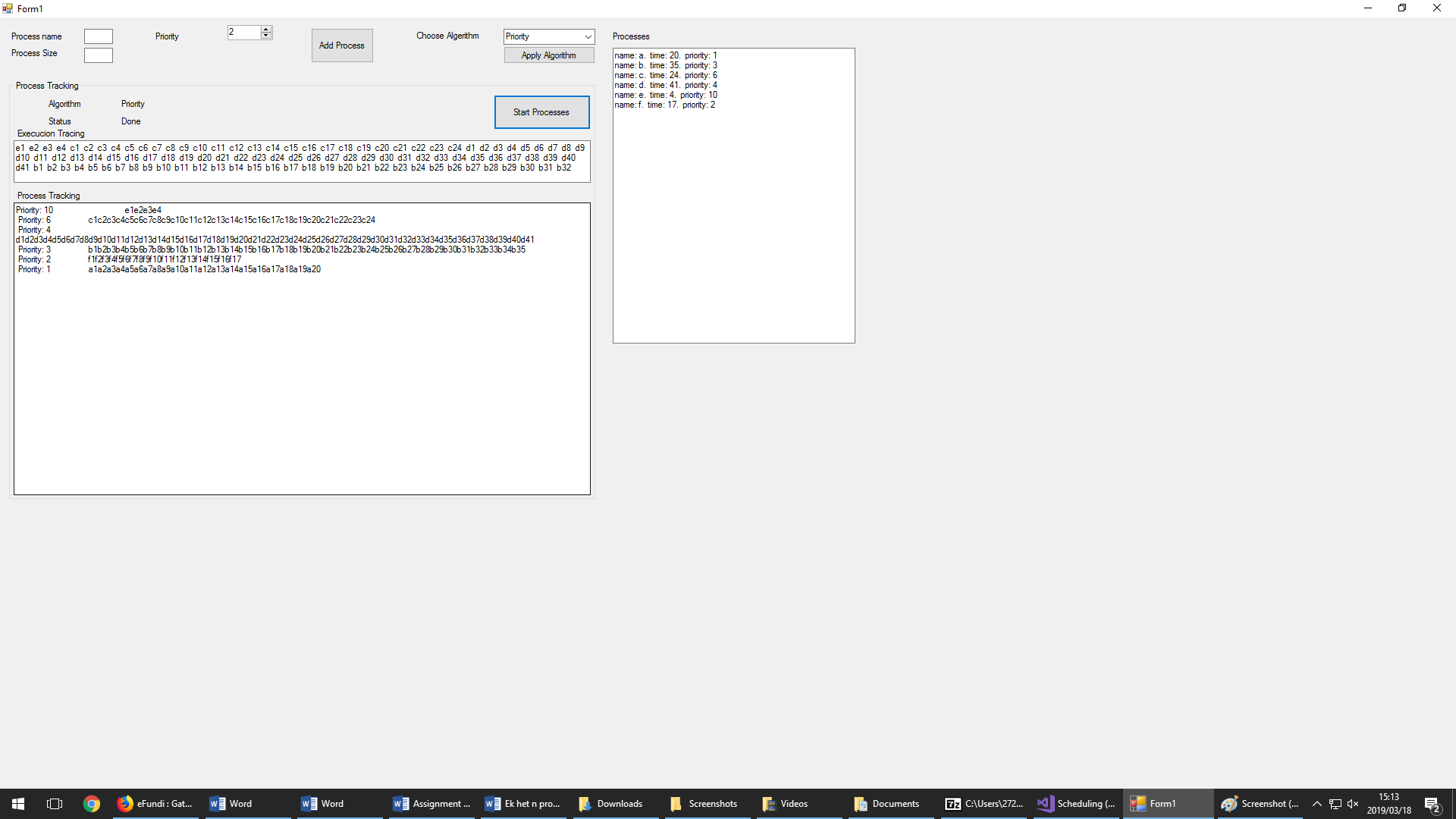
}

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

}

}

### Screen Shot of execution:



### Round-Robin Scheduling

One of the oldest, simplest, fairest and most widely used algorithms. Each process is only run for a specific time interval called a quantum. If a process is still running at the end of its quantum the CPU is preempted and given to another process.

### How we simulate it

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 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 has run, when each processes have been executed completely, the processes that have not been executed completely and the processes that are waiting their turn until they have been executed.

### While loop code snipped:

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--;

}

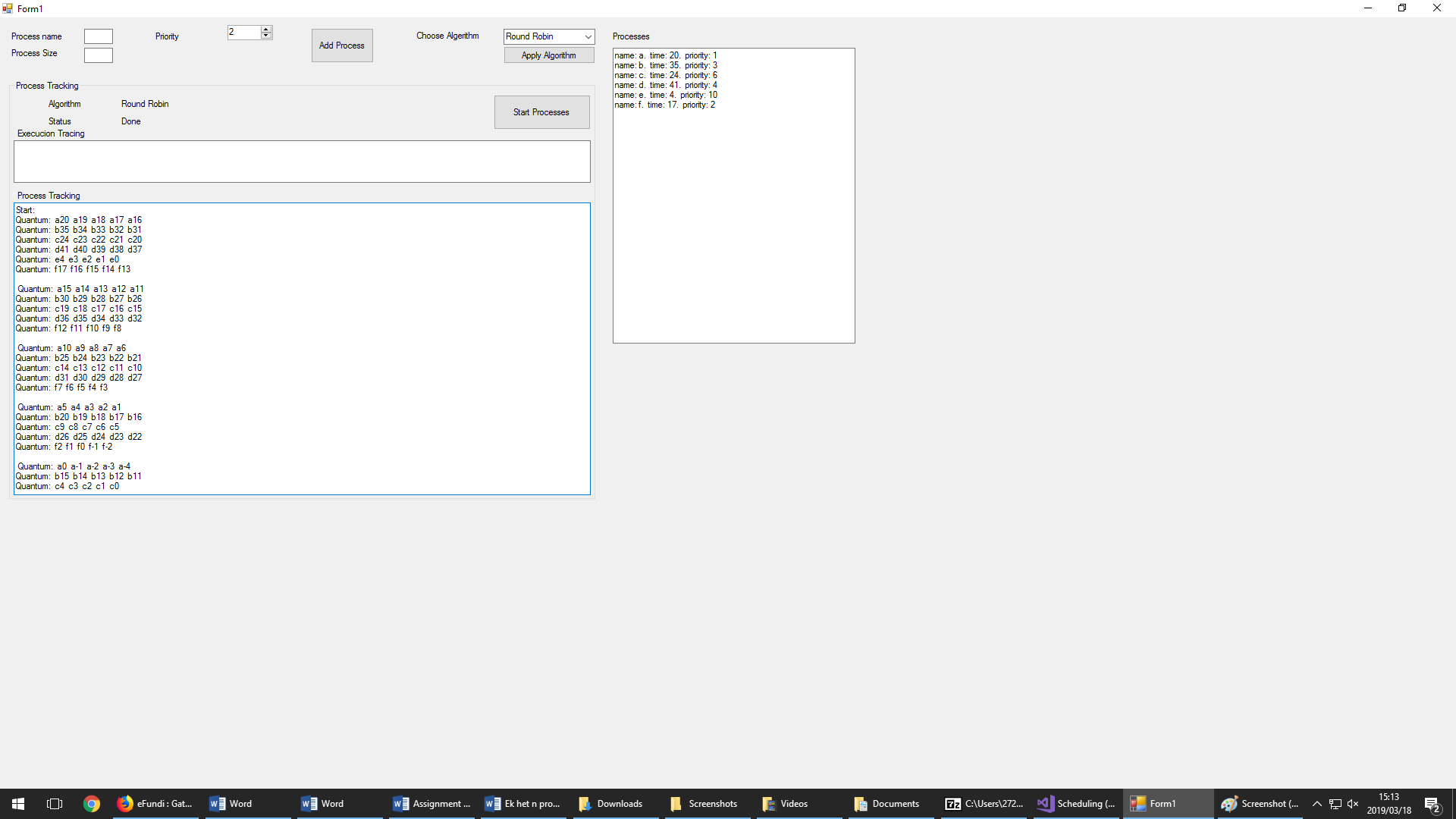
}

cycle++;

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

}

### Screen Shot of execution



### Multiple Queues

Multiple queues addresses a problem that arises in Round-Robin scheduling. The problem is that with short quantums the CPU spends a lot of time switching processes, and this waists time. The solution they found was to start each process of with small quantums and enlarge the quantums for the process each time a process is swapped without being completed. The quantum starts at one and then two and then four, continuously multiplying by two until the process is completed. In comparison with the Round-Robin algorithm in which a process of 100 quantum will have to perform 20 swaps, the same process in the multiple queues algorithm will only need 7 swaps. The process don’t completely use the seventh quantum but CPU is still saved by switching 13 time less.

### How we simulate it

Adjusting our Round-Robin algorithm by declaring a quantum variable and multiplying it with two each time a full iteration of the process list is completed and instead of 5 we use the quantum variable in the for loop( the for loop executes the amount of the quantum variables value). Each process within the iteration is execute for a quantum that equals the declared quantum variable’s value at that iteration. Which means each time a process gets swapped, the next time it is run it will have double the execution time it had the previous time it ran.

### Multiple queue algorithm, code snipped:

List<Process> temp = Hooflys;

int i = temp.Count;

int cycle = 0;

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

int quanta = 1;

while (i != 0)

{

quanta = quanta \* 2;

foreach (Process p in temp)

{

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

if (time >= 0)

{

textBox7.Text += "Quantum: ";

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

{

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

}

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

}

else if (time >= -5)

{

i--;

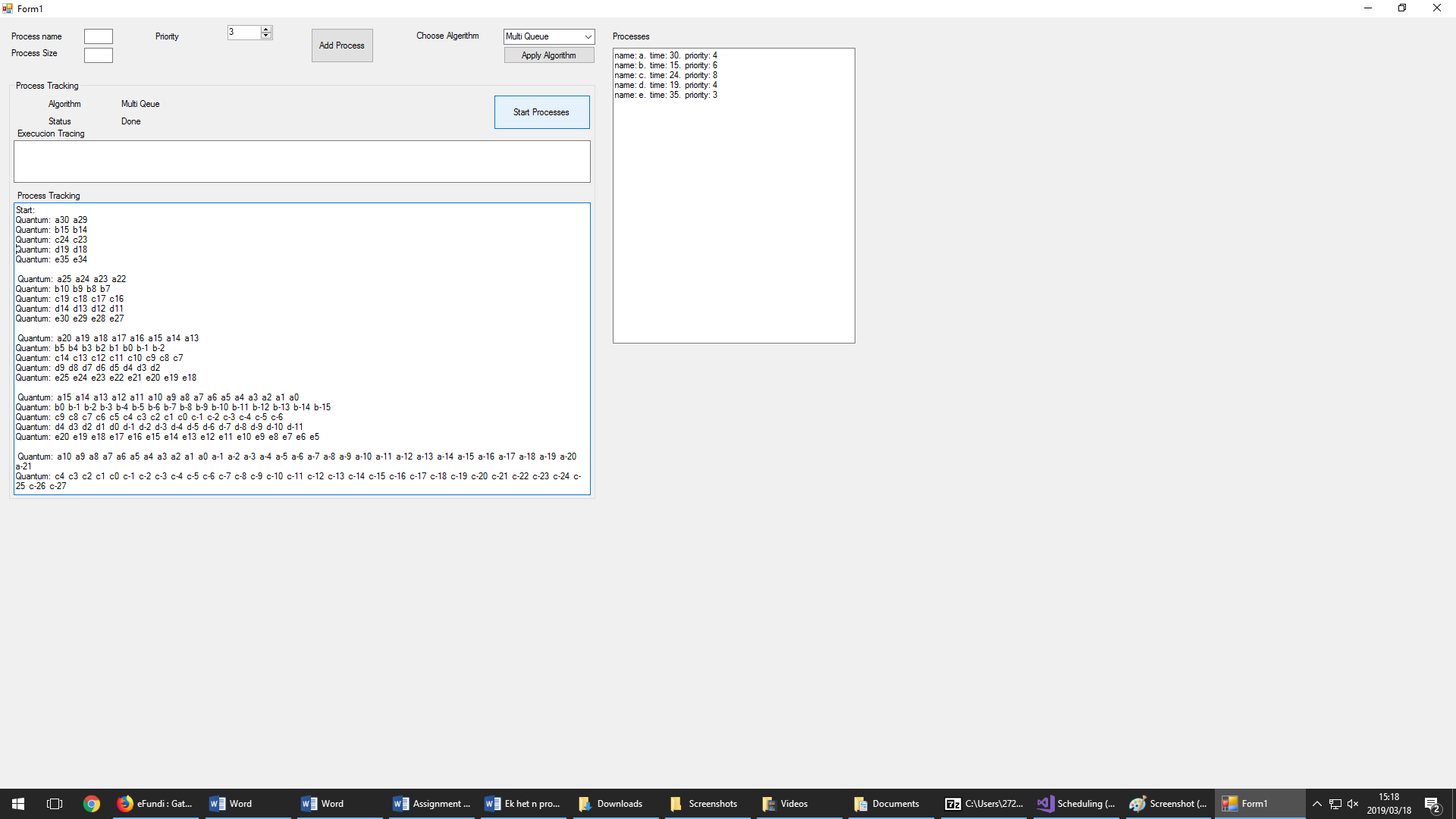
}

}

cycle++;

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

### Screen Shot of execution



# **Conclusion:**

In conclusion we acknowledge the implication of scheduling in the proper function of operation systems. Choosing the correct scheduling algorithm is vital in producing an acceptable runtime. Understanding that shortest job first scheduling is acceptable in batch system environment but will not be effective in a Real-Time System environment or interactive system environment. Priority scheduling also has a narrow region of environments where it is the best algorithm to be chosen. For example in a university there can exist a hierarchy of importance, for example the faculty dean and professors will have priority of their processes over those of students. Round –Robin scheduling is chosen if fairness is considered, and multiple queues scheduling can be used to speed up the execution Round-Robin if the execution time of round robin is not sufficient.

# Bibliography

Tenenbaum, A. S., & Bos, H. (2015). *Modern Operating Systems.* Amsterdam, The netherlands: Pearson.