


1.Introduction

This project is a simulation of Round Robin CPU schedule algorithm. The input is a plaintext file which stores processes and a setting time quantum. By modeling the function of CPU scheduling, we can get some standards of CPU utilization as an output. CPU scheduling is used to fit with the multiprocessing and aimed at maximum CPU utilization. The CPU scheduler selects from among the processes in ready queue and allocates a CPU core to one of them and Dispatcher module gives control of the CPU to the process selected by the short-term scheduler. This is also linked with context switching which between each transmit. One of the CPU scheduling algorithm is RR. Usually, each process gets a small unit of CPU time (time quantum q), usually 10-100 milliseconds to execute on the CPU. After this time has elapsed, the process is preempted and added to the end of the ready queue to wait. Typically, RR has higher average turnaround than SJF, but better response.

2.Implement and Running

The program uses java environment and eclipse software. The program contains five classes. The Clock class is a simulation of the clock which used to record the time change. The Main class is the start of whole program. The readyq class is a queue data Structure as a function of ready queue. The processcreat class is used to create a new process based on the given CSV files. The process class is a simulation of the CPU function.



osproj	pid,arrive,burst
> Clock.java	1,0,5
> Main.java	2,1,7
> process.java	3,0,2
> processcreat.java	4,2,6
> readyq.java	

Picture 1. The classes of program.

Picture 2. The basic form of CSV file\

The input file should base on the form above and the input is a full address of the input file and time quantum. The input is in the Main class.

```
process start= new process("C:\\Users\\dell\\Desktop\\test.txt",1);  
start.runprocess();
```

Picture 3. The input form of program.

When running the program, we will get the pid of each running process and the time it still need to finish. In the end, we will get some scheduling critical. In this program, we set the context switch using 0 time.

```

program:1 is running
remain time is 5

program:3 is running
remain time is 2

program:1 is running
remain time is 4
-----OUT PUT-----
Context switch is 19.0
CPU utiliazation is 100.0%
Average Turnaround time is 13.5
Average waiting time is 8.5
The thoughput is 21.052631578947366%

```

Picture 4. Running Step

Picture 5. The Output.

3.Result and Analysis

The last five picture is the Context switch time, CPU utilization, Average Turnaround time, Average waiting time and thoughput of five different time quantum:

1 unit, 3 units, 5 units ,7 units and 9 units. The formulation of these standard is:

CPU utilization = total burst time / total using time.

Each Turnaround time = each completion time- each arrival time

wait time = each turnaround time – burst time.

Thoughput = total processes / total time units

The average values are just simple math work. And result is below.

```

-----OUT PUT-----
Context switch is 19.0
CPU utiliazation is 100.0%
Average Turnaround time is 13.5
Average waiting time is 8.5
The thoughput is 21.052631578947366%

```

Picture 6. 1 unit timeQ

```

-----OUT PUT-----
Context switch is 7.0
CPU utiliazation is 80.95238095238095%
Average Turnaround time is 13.25
Average waiting time is 8.25
The thoughput is 19.047619047619047%

```

Picture 7. 3 units timeQ

```
-----OUT PUT-----  
Context switch is 5.0  
CPU utiliazation is 60.0%  
Average Turnaround time is 14.25  
Average waiting time is 9.25  
The throughput is 16.0%
```

Picture 8. 5 units timeQ

```
-----OUT PUT-----  
Context switch is 3.0  
CPU utiliazation is 61.904761904761905%  
Average Turnaround time is 14.75  
Average waiting time is 9.75  
The throughput is 19.047619047619047%
```

Picture 9. 7 units timeQ

```
-----OUT PUT-----  
Context switch is 3.0  
CPU utiliazation is 40.74074074074074%  
Average Turnaround time is 17.75  
Average waiting time is 12.75  
The throughput is 14.814814814814813%
```

Picture 10. 9 units timeQ

We can easily figure out that, when time quantum bigger, the context switch is small, causing lower overhead, but the efficiency is low, making it more like FCFS. If time quantum is small, the CPU utilization become bigger, but led to heavy overhead. When time Q is appropriate, in program is near 3 time units, we get best response time and lower overhead. That is what we need.

If we set the context switch with times, the efficiency of each output will drop because using resource to handle it.