

A project report on

IMPROVED SCHEDULING FOR BATCH OS USING INTELLIGENT TIME-SLICING

submitted in partial fulfilment for the award of the degree of

CSE2005 – OPERATING SYSTEMS

in

***B. Tech. in COMPUTER SCIENCE AND
ENGINEERING***

by

WILSON VIDYUT DOLOY (19BCE1603)

RISHANK PRATIK (19BCE1606)

BHAVAY CHOPRA (19BCE1411)



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING,
VELLORE INSTITUTE OF TECHNOLOGY,
CHENNAI - 600127**

June 2021

ABSTRACT

In today's world, there are different kinds of users prevalent. Different users require different systems with varying demands. As indicated by the necessities of the batch type system, we have made a program to evaluate the requirements and utilize the appropriate scheduling algorithm for maximum efficiency. The CPU scheduling algorithms present today are not appropriate for a wide range of systems. Different algorithms are required by different systems. For example, Batch operating system mostly uses priority CPU scheduling. A comparative analysis of our algorithm for batch operating systems with the algorithms for other systems like fcfs, priority and round robin will be presented on the basis of average turnaround time, average waiting time, varying time quantum and number of context switches.

CONTENTS

Title	Page No.
1. Introduction	4
1.1 About Batch Operating System	5
1.2 Objective	5
1.3 Our Approach	6
1.4 Advantages	6
2. Architecture	7
2.1 Algorithm	7
2.2 Step Wise Algorithm	7
2.3 Input/ Output	8
3. Results	9
3.1 Analysis	9
3.2 Comparison	9
4. Conclusion	10
5. References	10
5.1 Citations	10
5.2 Links	10
6. Appendix	13
6.1 Code	13
6.2 Screenshots	17

INTRODUCTION

The method by which processes are given access to system resources like processor cycles and communications bandwidth is known as scheduling. To execute multitasking and multiplexing, scheduling algorithms are required by the computer systems. Fundamentally, scheduling determines which process runs, in the presence of multiple run-able processes. Resource utilisation and other performance parameters are affected by the method of CPU scheduling. Some assumptions are considered in CPU scheduling such as:

- ❖ Job pool has run-able processes that wait for the CPU.
- ❖ Each and every process is independent. They compete for resources.
- ❖ Distributing the limited resources of the CPU among different processes optimize certain performance criteria is the primary aim of CPU scheduling.

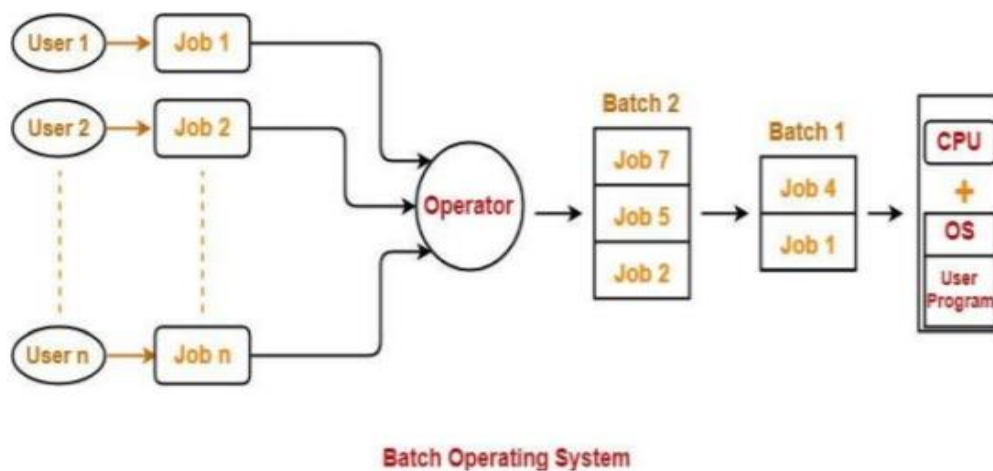
The component of the kernel which selects processes to run next is the scheduler. Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. There are three types of schedulers as follows:

Long term Scheduler	Mid term Scheduler	Short term scheduler
The long term scheduler or job scheduler selects processes from the job pool and loads them into memory for execution.	The medium term scheduler removes processes from memory and reduces the degree of multiprogramming results in the scheme of swapping.	The short term or CPU scheduler selects from among the processes that are ready to execute, and allocates CPU to one of them.

1.1 About Batch OS

In batch operating system,

1. Firstly, the user prepares his job on an offline device instead of interacting with the computer directly using punch cards.
2. Then, this is submitted to the computer operator.
3. Operators collect the jobs from different users and sort the jobs into batches with similar requirements.
4. Then, the operator submits the batches to the processor one by one.
5. All the jobs of one batch are executed together in order to speed up processing.



1.2 Objective

To implement improved algorithm for batch operating systems which works on a dynamic time quantum, that is, it changes after every round of execution. This will reduce the average waiting time, average turnaround time and reduce the number of context switches that take place which in turn enhances the working of the computer system.

1.3 Our Approach

Our approach to this problem of indefinite postponement is to use dynamic priorities. At the expiration of each quantum, the scheduler can decrease the priority of the current running process.

The scheduler will also keep track of low priority processes that do not get a chance to run and increase their priority so that eventually the priority will be high enough so that the processes will get scheduled to run.

Our algorithm is based on a round-robin algorithm and priority algorithm which takes the advantage from both the algorithms.

Thus, we get less average burst time and average turn-around time than the generic round-robin algorithm and generic priority algorithm.

1.4 Advantages

1. It saves the time that was being wasted earlier for each individual process in context switching from one environment to another environment.
2. No manual intervention is needed.

ARCHITECTURE

2.1 Algorithm

It is generally considered that priority scheduling is best for batch operating systems for optimal use of processor time and enhanced efficiency of the system. In Priority Scheduling, a priority is assigned to each process, which is decided on the basis of memory and other resource requirements. The process with the highest priority is executed first, followed by the processes in the same order. If two processes have the same priority, the First Come First Serve method is used to remove ambiguity. This results in high priority jobs to have low waiting time and so the problem of deadlines being not met can be eradicated. Disadvantage of this method is that of starvation of lower priority processes, i.e. the low priority process to wait indefinitely for its turn. This is possible when a large number of higher priority processes continuously arrive.

2.2 Step-by-Step Algorithm

1. Input the data (Processes, Burst times, Priorities, Initial time slice)
2. We sorted the process two times. Bubble sort using burst time. We calculated new priority i.e. $\text{priority} += \text{index}$, and sorted the processes again according to the new priorities
3. For each round we calculate the new priority based on their shortness components and priority components which is calculated on the basis of the number of processes and their new priority (which is just one greater than their initial value)
 - $\text{PC} = 0$, if new priority $= 2n/3$ (Not important)
 - $\text{PC} = 1$, if new priority $= < 2n/3$ but $> n/3$ (Moderately important)
 - $\text{PC} = 2$, if new priority $= \geq 1$ (Important)
4. For each round Shortness component is calculated on the basis of burst times of the previous process in execution.
 - $\text{SC} = 0$, if $\text{bt}(i) > \text{bt}(i-1)$
 - $\text{SC} = 1$, if $\text{bt}(i) \leq \text{bt}(i-1)$
5. For each round we Calculate the Intelligent Time Slice (ITS) for each process as the sum of the initial time slice, burst time, priority component and shortness component.

6. For each round we are Calculating Time Quanta utilizing the ITS and SC for each process which have been calculated before. Repeating this process for each round of execution.
 - For the first round ($j = 0$)
 - i. $TQ_{j,i} = ITS_i$ if $SC_i = 1$
 - ii. $TQ_{j,i} = ITS_i/2$ if $SC_i = 0$
 - For the subsequent rounds of execution ($j > 0$)
 - i. $TQ_{j,i} = TQ_{j-1,i} * 2$, if $SC_i = 1$
 - ii. $TQ_{j,i} = TQ_{j-1,i} * 1.5$, if $SC_i = 0$
7. Finally, calculate average waiting time and average turnaround time

2.3 Input/Output

INPUT:

- No. of Processes
- Burst times of each Process
- Priorities of each Process
- Initial time slice

OUTPUT:

Calculate and display average waiting time and average turnaround time, which will be much less than the generic round-robin algorithm and generic priority algorithm.

RESULTS

Inputs Given

Process No.	Arrival Time	Burst Time	Priority
1	0	5	3
2	0	3	5
3	0	4	4
4	0	8	1
5	0	2	6
6	0	6	2

3.1 Analysis

Priority Scheduling:

Average Waiting Time = 15.0

Average Turn-Around Time = 19.67

Round Robin:

Average Waiting Time = 13.83

Average Turn-Around Time = 18.50

Improved Batch OS:

Average Waiting Time = 9.33

Average Turn-Around Time = 14.0

3.2 Comparison

OS/Time	Avg WAT	Avg TAT
Improved Batch OS	9.33	14.0
Priority Scheduling	15.0	19.67
Round Robin	13.83	18.50

CONCLUSION

The main issue of Batch Operating systems was:

- If some job takes too much time i.e. if error occurs in job then other jobs will wait for unknown time.
- To speed up the processing, jobs with similar need to be batched together and run as a group.
- It is difficult to provide the desired priority.

As indicated by the necessities of the Batch type system, we developed a program based on Round Robin and Priority Algorithm which will take advantage from both algorithms to evaluate the requirements and utilize the appropriate scheduling algorithm for maximum efficiency.

A comparative analysis of our algorithm for batch operating systems with the algorithms for other systems like FCFS, priority and round robin was presented on the basis of average turnaround time, average waiting time, varying time quantum and number of context switches to prove our concept.

Thus, we have enhanced the working of the computer system by reducing the average waiting time, average turnaround time and reducing the number of context switches from generic algorithms.

REFERENCES

- 1] Mohanty, Rakesh, et al. "Design and performance evaluation of a new proposed shortest remaining burst round robin (SRBRR) scheduling algorithm." *Proceedings of International Symposium on Computer Engineering & Technology (ISCET)*. Vol. 17. 2010.
- 2] Sabha, Saqib Ul. "A novel and efficient round robin algorithm with intelligent time slice and shortest remaining time first." *Materials Today: Proceedings* 5.5 (2018): 12009- 12015.
- 3] Aboalama, Mohammed, and Adil Yousif. "Enhanced job scheduling algorithm for cloud computing using the shortest remaining job first." *International Journal of Computer Science & Management Studies* 15.6 (2015): 65-68.
- 4] Varma, P. Surendra. "Improved Shortest Remaining Burst Round Robin (ISRBR) Using RMS as its time quantum." *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)* 1.8 (2012).
- 5] Aboalama, Mohammed, and Adil Yousif. "Enhanced job scheduling algorithm for cloud computing using the shortest remaining job first." *International Journal of Computer Science & Management Studies* 15.6 (2015): 65-68.
- 6] Prabhakaran, Suraj, et al. "A batch system with efficient adaptive scheduling for malleable and evolving applications." *2015 IEEE international parallel and distributed processing symposium*. IEEE, 2015.
- 7] Lichen Zhang, "Scheduling algorithm for real-time applications in grid environment," *IEEE International Conference on Systems, Man and Cybernetics, Yasmine Hammamet, Tunisia, 2002*, pp. 6 pp. vol.5-, doi: 10.1109/ICSMC.2002.1176337.
- 8] García-Muñoz, Salvador, et al. "Optimization of batch operating policies. Part I. Handling multiple solutions#." *Industrial & engineering chemistry research* 45.23 (2006): 7856-7866.
- 9] J. P. Lehoczky, "Fixed priority scheduling of periodic task sets with arbitrary deadlines," [1990] *Proceedings 11th RealTime Systems Symposium, Lake Buena Vista, FL, USA, 1990*, pp. 201-209, doi:10.1109/REAL.1990.128748.
- 10] Andersson, Björn, Sanjoy Baruah, and Jan Jonsson. "Static-priority scheduling on multiprocessors." *Proceedings 22nd IEEE Real-Time Systems Symposium (RTSS 2001)*(Cat. No. 01PR1420). IEEE, 2001.
- 11] Leung, Joseph Y-T., and Jennifer Whitehead. "On the complexity of fixed-priority scheduling of periodic, real-time tasks." *Performance evaluation* 2.4 (1982): 237-250.
- 12] S. Prabhakaran, M. Iqbal, S. Rinke, C. Windisch and F. Wolf, "A Batch System with Fair Scheduling for Evolving Applications," *2014 43rd International Conference on Parallel Processing, Minneapolis MN, 2014*, pp. 351-360, doi: 10.1109/ICPP.2014.44.
- 13] Davis, Robert, and Andy Wellings. "Dual priority scheduling." *Proceedings 16th IEEE Real-Time Systems Symposium*. IEEE, 1995.

- 14] Rami Abielmona, Scheduling Algorithmic Research, Department of Electrical and Computer Engineering OttawaCarleton Institute, 2000.
- 15] Tarek Helmy, Abdelkader Dekdouk, "Burst Round Robin: As a Proportional-Share Scheduling Algorithm", IEEE Proceedings of the fourth IEEE-GCC Conference on towards Techno-Industrial Innovations, pp. 424-428, 11- 14 November, 2007.
- 16] Silberschatz, A., P.B. Galvin and G.Gagne, Operating Systems Concepts.7th Edn., John Wiley and Sons, USA ISBN:13:978- 0471694663, pp: 944, 2004.
- 17] A.S. Tanenbaun, Modern Operating Systems.3rd Edn, Prentice Hall, ISBN:13: 9780136006633, pp: 1104, 2008
- 18] Rami J. Matarneh , Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the now Running Processes, Department of Management Information Systems, American Journal of Applied Sciences 6 (10):1831- 1837, 2009, ISSN 1546-9239.
- 19]Padhy, R.P., Load Balancing in cloud computing systems, 2011, National Institute of Technology, Rourkela.
- 20] Computing, C., a practical Approach, Anthony T. Velte, Toby J. Velte, Robert Elsenpeter, McGraw Hill, 2010.
- 21] Zhang, H., et al., A PSO-Based Hierarchical Resource Scheduling Strategy on Cloud Computing, in Trustworthy Computing and Services. 2013, Springer. p. 325-332.
- 22] The CLOUDS (2012) Lab: CloudSim:.2012.
- 23] Ercan, T. Effective use of cloud computing in educational institutions. Procedia-Social and Behavioral Sciences, 2010. 2(2): p. 938-942.
- 24] Mishra, R. and A. Jaiswal, Ant colony optimization: A solution of load balancing in the cloud. International Journal of Web & Semantic Technology (IJWest), 2012. 3(2): p. 33- 50.
- 25] GOYAL, T. and A. AGRAWAL, Host Scheduling Algorithm Using Genetic Algorithm In Cloud Computing Environment. International Journal of Research in Engineering & Technology (IJRET) Vol. 1.
- 26]Classic Operating Systems: From Batch Processing to Distributed Systems edited by Per Brinch Hansen

APENDIX

CODE

```
#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;
int main()
{
    cout << "-----Time Scheduling for Batch OS-----" << endl;
    int n; //no. of processes
    cout << "\nEnter the number of processes: ";
    cin >> n;
    int bt[n]; //burst times for each process
    int p[n]; //priority for each process
    int s[n]; //shortness component for each process
    int wt[n], tat[n]; //wait time and TAT for each process
    int ts; //initial time slice
    int its[n]; //intelligent time slice
    int tq[n][n]; //tq[j][i] -> time quantum for process 'i' at round 'j'
    int rbt[n]; //REMAINING burst time for each consequent round
    int ord[n]; //stores the process numbers in whichever order we sort
    for (int i = 0; i < n; i++)
    {
        wt[i] = tat[i] = 0; //initialize variables to 0
        s[i] = 1; //initialize shortness component to 1
        for (int j = 0; j < n; j++)
            tq[i][j] = 0; //initialize variable to 0
    }
    cout << "\nEnter the initial time slice: ";
    cin >> ts;
    for (int i = 0; i < n; i++)
    {
        ord[i] = i + 1; //storing process numbers [Numbering from 1 to n]
        cout << "\nProcess " << i + 1 << endl;
        cout << "Burst time: ";
        cin >> bt[i]; //accept Burst Time
        cout << "Priority: ";
        cin >> p[i]; //accept priority
    }
    int flag = 0, j = 0; //initialize flag=0 and round number 'j'=0
    //j=0 means first round
    /* sort processes in order of shortness component
    doing this will ensure that shortness RANK of the
    process will be the index no. of that process in ord[]
    e.g.: If P5 is at index 3 in ord[], then shortness rank of P5 is 3
    */
    for (int i = 0; i < n - 1; i++)
    for (int j = 0; j < n - 1; j++)
```

```

if (bt[j] > bt[j + 1])
{
int t = bt[j];
bt[j] = bt[j + 1];
bt[j + 1] = t;
t = p[j];
p[j] = p[j + 1];
p[j + 1] = t;
t = ord[j];
ord[j] = ord[j + 1];
ord[j + 1] = t;
}
//finished sorting processes according to shortness component,
//so now, new priority of a process at index 'i' -> p[i] = p[i] + i
for (int i = 0; i < n; i++)
p[i] += i; //setting new priority for process
//now using Bubble sort to sort processes according to their new
//priority [ascending order]
for (int i = 0; i < n - 1; i++)
for (int j = 0; j < n - 1 - i; j++)
if (p[j] > p[j + 1])
{
int t = bt[j];
bt[j] = bt[j + 1];
bt[j + 1] = t;
t = p[j];
p[j] = p[j + 1];
p[j + 1] = t;
t = ord[j];
ord[j] = ord[j + 1];
ord[j + 1] = t;
}
for (int i = 0; i < n; i++)
rbt[i] = bt[i]; //for first round, burst times are assigned as is
//setting PCi=0 or 1 or 2 and SCi=0 or 1 [Sci = 1 by default at
//initialization] at every round
//starting execution!!
while (!flag)
{
for (int i = 0; i < n; i++)
{
// Setting PCi
if (p[i] > 0.67 * n)
p[i] = 0;
else if (p[i] > 0.33 * n)
p[i] = 1;
else
p[i] = 2;
//PCi has been set
//for the first process(i=0), there is no process before it,so
//its shortness = 1 which is already set by default

```

```

if (i != 0) //for subsequent processes, setting value for
//shortness component
if ((bt[i] - bt[i - 1]) > 0)
s[i] = 0;
its[i] = ts + bt[i] + s[i] + p[i]; //setting Intelligent Time
//Slice
/* tq[j][i] -> time quantum for process 'i' at round 'j' */
//round number(j) simply signifies which iteration or 'round'
// of execution we are on
if (j == 0) //for first round we set our conditions separately
{
if (s[i] == 1)
tq[j][i] = its[i];
else
tq[j][i] = ceil(0.5 * (float)its[i]);
if (rbt[i] < tq[j][i]) //incase process does not take up the
//entire time slice allotted to it
tq[j][i] = rbt[i];
rbt[i] = rbt[i] - tq[j][i]; //remaining burst time for the
//next round.
}
else //for subsequent rounds we set conditions according to
//the algorithm.
{
if (rbt[i] <= 0)
tq[j][i] = 0;
else if (s[i] == 1)
tq[j][i] = 2 * tq[j - 1][i];
else
tq[j][i] = 1.5 * tq[j - 1][i];
if (rbt[i] < tq[j][i]) //incase process does not take up the
//entire time slice allotted to it
tq[j][i] = rbt[i];
rbt[i] = rbt[i] - tq[j][i];
}
}
j++; //incrementing round number (Next iteration of execution)
flag = -1; //Take an assumption that all processes have finished
for (int i = 0; i < n; i++)
if (rbt[i] > 0) //if atleast one process is yet to execute, we
flag = 0; //need to proceed for another round, set flag=0
}
cout << "\n\nProcess no. :";
for (int i = 0; i < n; i++)
cout << setw(5) << ord[i];
cout << "\n\nBurst Times :";
for (int i = 0; i < n; i++)
cout << setw(5) << bt[i];
cout << "\n\nDynamic Time Quantums :";
for (int x = 0; x < j; x++)
{

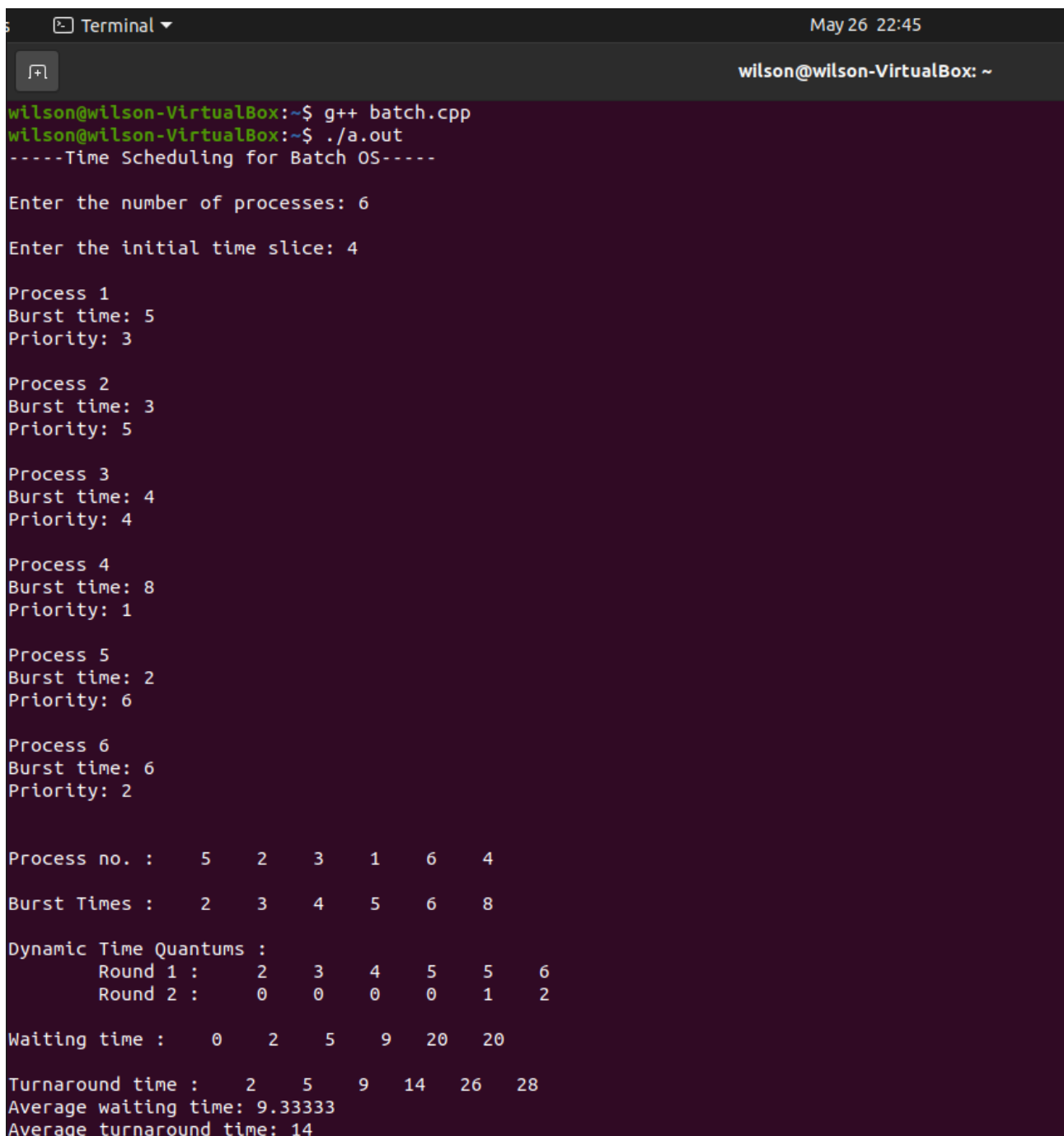
```

```

cout << "\n\tRound " << x + 1 << " : ";
for (int y = 0; y < n; y++)
cout << setw(5) << tq[x][y];
}
for (int x = 0; x < n; x++)
{
flag = -1;
for (int y = 0; y < j; y++)
{
for (int z = 0; z < n; z++)
{
if (z != x)
wt[x] += tq[y][z];
else if (z == x && tq[y + 1][z] == 0)
{
flag = 0;
break;
}
}
tat[x] += tq[y][x];
if (flag == 0)
break;
}
tat[x] += wt[x];
}
cout << "\n\nWaiting time :";
for (int i = 0; i < n; i++)
cout << setw(5) << wt[i];
cout << "\n\nTurnaround time :";
for (int i = 0; i < n; i++)
cout << setw(5) << tat[i];
float avwt = 0, avtat = 0;
for (int i = 0; i < n; i++)
{
avwt += wt[i];
avtat += tat[i];
}
avwt /= n;
avtat /= n;
cout << "\nAverage waiting time: " << avwt << endl;
cout << "Average turnaround time: " << avtat << endl;
}

```


SCREENSHOTS



```
wilson@wilson-VirtualBox: ~  
$ g++ batch.cpp  
$ ./a.out  
-----Time Scheduling for Batch OS-----  
  
Enter the number of processes: 6  
Enter the initial time slice: 4  
  
Process 1  
Burst time: 5  
Priority: 3  
  
Process 2  
Burst time: 3  
Priority: 5  
  
Process 3  
Burst time: 4  
Priority: 4  
  
Process 4  
Burst time: 8  
Priority: 1  
  
Process 5  
Burst time: 2  
Priority: 6  
  
Process 6  
Burst time: 6  
Priority: 2  
  
Process no. :    5    2    3    1    6    4  
Burst Times :    2    3    4    5    6    8  
  
Dynamic Time Quanta :  
Round 1 :      2    3    4    5    5    6  
Round 2 :      0    0    0    0    1    2  
  
Waiting time :    0    2    5    9   20   20  
  
Turnaround time :    2    5    9   14   26   28  
Average waiting time: 9.33333  
Average turnaround time: 14
```

(Improved Batch Operating System, for same set of inputs)

```
wilson@wilson-VirtualBox:~$ cc priority.c
wilson@wilson-VirtualBox:~$ ./a.out
Enter the number of process:6
Enter process name,arrival time,burst time & priority:1 0 5 3
Enter process name,arrival time,burst time & priority:2 0 3 5
Enter process name,arrival time,burst time & priority:3 0 4 4
Enter process name,arrival time,burst time & priority:4 0 8 1
Enter process name,arrival time,burst time & priority:5 0 2 6
Enter process name,arrival time,burst time & priority:6 0 6 2
```

Pname	arrivaltime	bursttime	priority	waitingtime	tatime
4	0	8	1	0	8
6	0	6	2	8	14
1	0	5	3	14	19
3	0	4	4	19	23
2	0	3	5	23	26
5	0	2	6	26	28

```
Average waiting time is:15.000000
Average turnaroundtime is:19.666666wilson@wilson-VirtualBox:~$
```

(Priority Scheduling, for same set of inputs)

```
wilson@wilson-VirtualBox:~$ g++ round.c
wilson@wilson-VirtualBox:~$ ./a.out
Enter Total Number of Processes: 6

Enter Details of Process[1]Arrival Time: 0
Burst Time: 5

Enter Details of Process[2]Arrival Time: 0
Burst Time: 3

Enter Details of Process[3]Arrival Time: 0
Burst Time: 4

Enter Details of Process[4]Arrival Time: 0
Burst Time: 8

Enter Details of Process[5]Arrival Time: 0
Burst Time: 2

Enter Details of Process[6]Arrival Time: 0
Burst Time: 6

Enter Time Quantum: 4

Process IDttBurst Timet Turnaround Timet Waiting Time
Process[2]          3          7          4
Process[3]          4          11         7
Process[5]          2          17        15
Process[1]          5          22        17
Process[4]          8          26        18
Process[6]          6          28        22
Average Waiting Time: 13.833333
Avg Turnaround Time: 18.500000
```

(Round Robin, for same set of inputs)
