Name: Aaditya khandal
Reg. No. : 11705679
Roll No. : 23
Email Address: saytoaditya@gmail.com
GitHub Link: https://github.com/saytoaditya/os-project.git

Description:

Round Robin Scheduling

Round Robin is the preemptive process scheduling algorithm. Each process is provided a fix time slot to execute, it is called as time quantum. Once a process is executed for a given time period, it preempts and other process executes for a given time period.

Round Robin Example:

Process	Duration	Order	Arrival Time
P1	3	1	0
P2	4	2	0
P3	3	3	0

Suppose time quantum is 1 unit.

P1	P2	P3	P1	P2	P3	P1	P2	P3	P2

0 10

P1 waiting time : 4

P2 waiting time: 6

P3 waiting time: 6

The average waiting time(AWT) : (4+6+6)/3=5.33

How to compute below times in Round Robin using a program?

1. Completion Time: Time at which process completes its execution.
2. Turn Around Time: Time Difference between completion time and arrival time. Turn Around Time = Completion Time – Arrival Time
3. Waiting Time (W.T): Time Difference between turn around time and burst time. Waiting Time = Turn Around Time – Burst Time

Name: Aaditya khandal
Reg. No. : 11705679
Roll No. : 23
Email Address: saytoaditya@gmail.com
GitHub Link: https://github.com/saytoaditya/os-project.git

Steps to find waiting times of all processes:

1- Create an array **rem_bt[]** to keep track of remaining
burst time of processes. This array is initially a
copy of bt[] (burst times array)
2- Create another array **wt[]** to store waiting times
of processes. Initialize this array as 0.
3- Initialize time : t = 0
4- Keep traversing the all processes while all processes
are not done. Do following for i'th process if it is
not done yet.
a- If rem_bt[i] > quantum
(i) t = t + quantum
(ii) bt_rem[i] -= quantum;
c- Else // Last cycle for this process
(i) t = t + bt_rem[i];
(ii) wt[i] = t - bt[i]
(ii) bt_rem[i] = 0; // This process is over

Algorithm:

ROUND ROBIN SCHEDULING

Step 1: Start the process
Step 2: Accept the number of processes in the ready Queue and time quantum (or) time slice
Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time
Step 4: Calculate the no. of time slices for each process where
No. of time slice for process(n) = burst time process(n)/time slice
Step 5: If the burst time is less than the time slice then the no. of time slices =1.
Step 6: Consider the ready queue is a circular Q, calculate
(a) Waiting time for process(n) = waiting time of process(n-1)+ burst time of process(n-1) + the time difference in getting the CPU from process(n-1)
(b) Turn around time for process(n) = waiting time of process(n) + burst time of process(n)+ the time difference in getting CPU from process(n).
Step 7: Calculate
(a) Average waiting time = Total waiting Time / Number of process
(b) Average Turnaround time = Total Turnaround Time / Number of process
Step 8: Stop the process

Name: Aaditya khandal
Reg. No. : 11705679
Roll No. : 23
Email Address: saytoaditya@gmail.com
GitHub Link: https://github.com/saytoaditya/os-project.git

Question 2: Considering the arrival time and burst time requirement of the process the scheduler schedules the processed by interrupting the processor after every 6 units of time and does consider the completion of the process in the iteration. The scheduler than checks for the number of processor every 10 unit of time and considers the completion of the processed in the iteration. the scheduler check the number of processes waiting in the queue for the processor after the second iteration and gives the processor to the process which needs more time to complete than the other processes to go in the terminated state. The inputs for the number of reqirements, arrical time and burst time should be provided by the user.

Consider the bollowing units for refrence.

Process	Arrival time	Burst time
P1	**0**	**20**
P2	**5**	**36**
P3	**13**	**19**
P4	**26**	**42**

Implementation:

```
#include<stdio.h>

                    int main()

                    {

                    int count,j,n,time,remain,flag=0,time_quantum;

                    int wait_time=0,turnaround_time=0,at[10],bt[10],rt[10];

                    printf("Enter Total Process:\t ");

                    scanf("%d",&n);
```

Name: Aaditya khandal
Reg. No. : 11705679
Roll No. : 23
Email Address: saytoaditya@gmail.com
GitHub Link: https://github.com/saytoaditya/os-project.git

```
remain=n;

for(count=0;count<n;count++)

{

printf("Enter Arrival Time and Burst Time for Process Process Number %d
:",count+1);

scanf("%d",&at[count]);

scanf("%d",&bt[count]);

rt[count]=bt[count];

}

printf("Enter Time Quantum:\t");

scanf("%d",&time_quantum);

printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");

for(time=0,count=0;remain!=0;)

{

if(rt[count]<=time_quantum && rt[count]>0)

{

time+=rt[count];

rt[count]=0;

flag=1;

}
```

Name: Aaditya khandal
Reg. No. : 11705679
Roll No. : 23
Email Address: saytoaditya@gmail.com
GitHub Link: https://github.com/saytoaditya/os-project.git

```
else if(rt[count]>0)

{

rt[count]-=time_quantum;

time+=time_quantum;

}

if(rt[count]==0 && flag==1)

{

remain--;

printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-
bt[count]);

wait_time+=time-at[count]-bt[count];

turnaround_time+=time-at[count];

flag=0;

}

if(count==n-1)

count=0;

else if(at[count+1]<=time)

count++;

else

count=0;
```

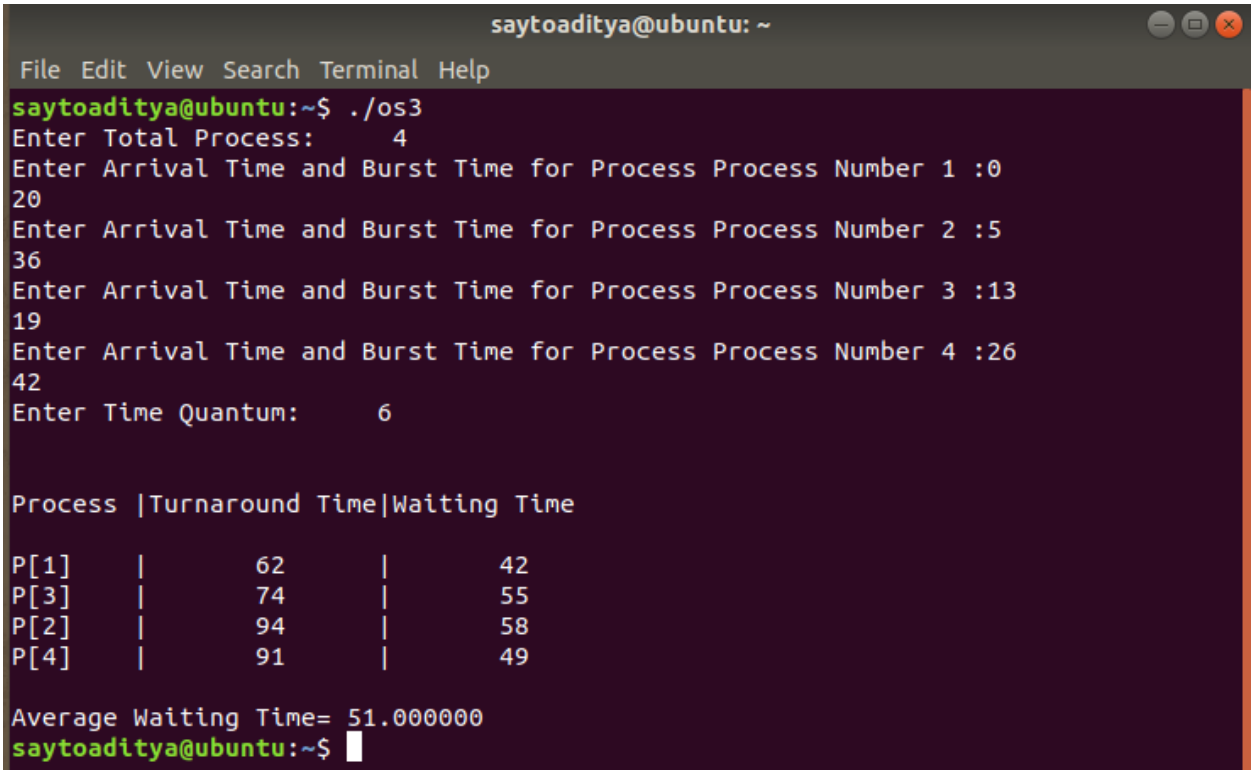
Name: Aaditya khandal
Reg. No. : 11705679
Roll No. : 23
Email Address: saytoaditya@gmail.com
GitHub Link: https://github.com/saytoaditya/os-project.git

```
}

printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);

printf("Avg Turnaround Time = %f",turnaround_time*1.0/n);

return 0;

}
```

Name: Aaditya khandal
Reg. No. : 11705679
Roll No. : 23
Email Address: saytoaditya@gmail.com
GitHub Link: https://github.com/saytoaditya/os-project.git

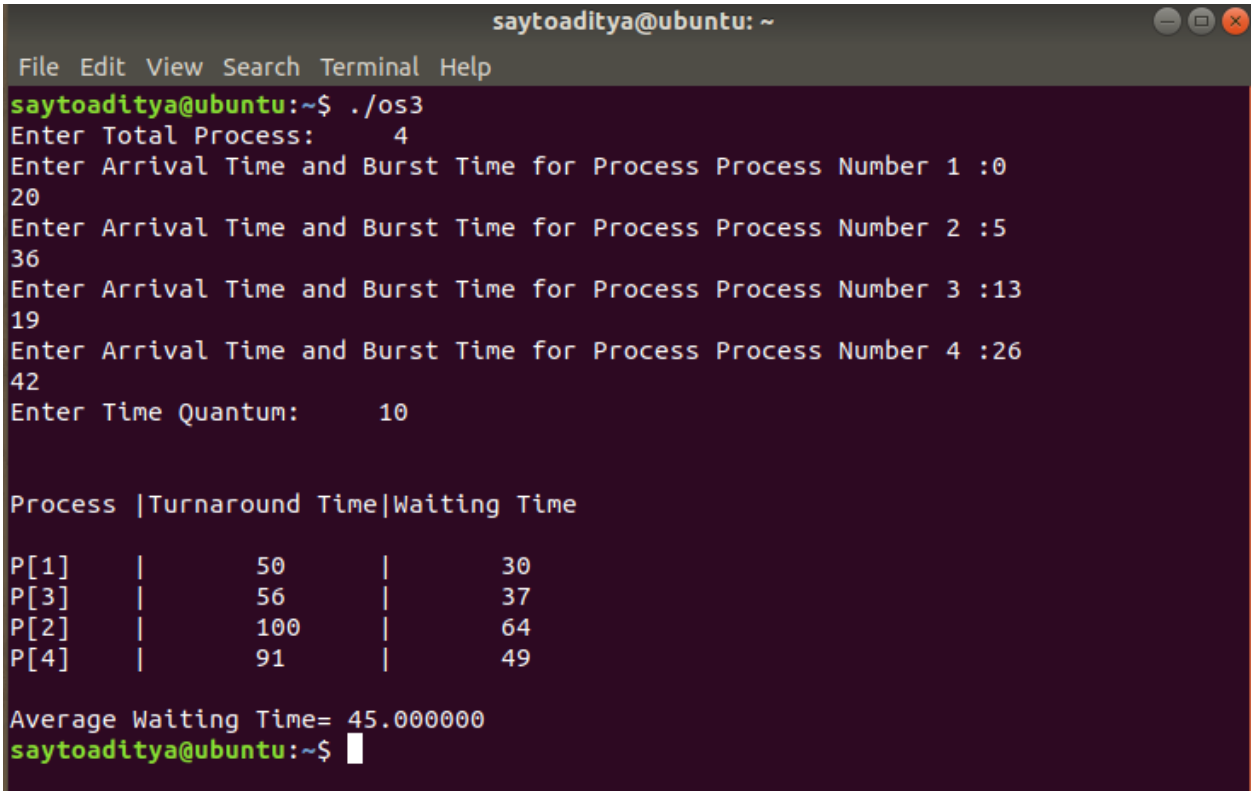
Output:

→Time Quantum=6



Name: Aaditya khandal
Reg. No. : 11705679
Roll No. : 23
Email Address: saytoaditya@gmail.com
GitHub Link: https://github.com/saytoaditya/os-project.git

→Time Quantum=10



Name: Aaditya khandal
Reg. No. : 11705679
Roll No. : 23
Email Address: saytoaditya@gmail.com
GitHub Link: https://github.com/saytoaditya/os-project.git

Advantages:

Round Robin is a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way.

- It is simple, easy to implement, and starvation-free as all process get fair share of CPU.
- One of the most commonly used technique in CPU scheduling as a core.
- It is preemptive as processes are assigned CPU only fixed slice of time at most.