

Operating systems (cse 316)

Student Name: Narayan Chandravanshi Guided by: Baljit singh saini

**Student ID:** 11803932

**Email Address:** [narayanchandra952@gmail.com](mailto:narayanchandra952@gmail.com)

**GitHub Link:** https://github.com/Narayan1180

Problem: design a scheduling programme to implement a Queue with two levels:

Level 1:Fixed priority preemptive Scheduling

Level 2:Round Robin Scheduling

For a fixed priority scheduling(Queue1),the priority 0is highest priority.If one process P1 is scheduled and running,another process P2 with higher priority comes.The new process(high priority ) preempts currently running process P1 and process P1will go to second level queue .Time for which process will strictly excute must be considered in multiple of 2.

All processes in second level queue will complete their execution according to round robin scheduling.

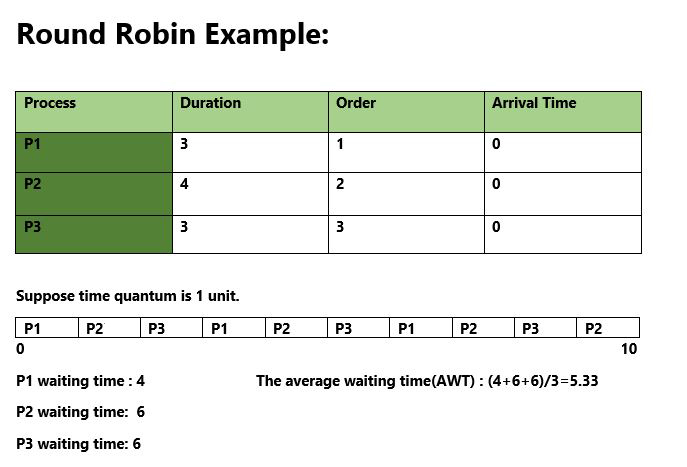
Concider :1.Queue 2 will be processed after Queue 1 become empty.

2. Priority of Queue 2 has lower priority than in queue1.

**Ans:**

**Round-robin** (RR) is one of the algorithms employed by [process](https://en.wikipedia.org/wiki/Process_scheduler) and [network schedulers](https://en.wikipedia.org/wiki/Network_scheduler) in [computing](https://en.wikipedia.org/wiki/Computing).[[1]](https://en.wikipedia.org/wiki/Round-robin_scheduling#cite_note-ostep-1-1)[[2]](https://en.wikipedia.org/wiki/Round-robin_scheduling#cite_note-Zander-2) As the term is generally used, [time slices](https://en.wikipedia.org/wiki/Preemption_(computing)#Time_slice) (also known as time quanta)[[3]](https://en.wikipedia.org/wiki/Round-robin_scheduling#cite_note-3) are assigned to each process in equal portions and in circular order, handling all processes without [priority](https://en.wiktionary.org/wiki/priority) (also known as [cyclic executive](https://en.wikipedia.org/wiki/Cyclic_executive)). Round-robin scheduling is simple, easy to implement, and [starvation](https://en.wikipedia.org/wiki/Resource_starvation)-free. Round-robin scheduling can be applied to other scheduling problems, such as data packet scheduling in computer networks. It is an [operating system](https://en.wikipedia.org/wiki/Operating_system) concept.

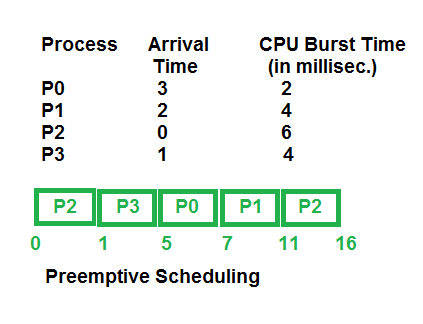
The name of the algorithm comes from the [round-robin](https://en.wikipedia.org/wiki/Round-robin_(disambiguation)) principle known from other fields, where each person takes an equal share of something in turn.



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

**1. Preemptive Scheduling:**  
Preemptive scheduling is used when a process switches from running state to ready state or from waiting state to ready state. The resources (mainly CPU cycles) are allocated to the process for the limited amount of time and then is taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining. That process stays in ready queue till it gets next chance to execute.

Algorithms based on preemptive scheduling are: [Round Robin (RR)](https://www.geeksforgeeks.org/program-round-robin-scheduling-set-1/),[Shortest Remaining Time First (SRTF)](https://www.geeksforgeeks.org/program-shortest-job-first-scheduling-set-2srtf-make-changesdoneplease-review/), [Priority (preemptive version)](https://www.geeksforgeeks.org/program-for-preemptive-priority-cpu-scheduling/), etc.



Algorithm:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queuealongwith arrival time and burst time.

Step 3: For each process in the ready Q, assign the process with a time slice of X unit infirst iteration.

Step 4: For each process again assign the process with a time slice of Y unit insecond iteration.

Step 5: Assign Processes with least execution time i.e CPU is assigned to process with less CPU burst time.

Step 6: Calculate

1. Waiting time for process(n) = waiting time of process(n-1)+ burst time of process(n-1 )+ the time difference in getting the CPU fromprocess(n-1)
2. Turn around time for process(n) = waiting time of process(n) + burst time ofprocess(n)+ the time difference in getting CPU fromprocess(n).

Step 7: Calculate

1. Average waiting time = Total waiting Time / Number ofprocess
2. Average Turnaround time = Total Turnaround Time / Number ofprocess Step 8: Stop theprocess

Description:

To implement the above problem we have to make three iterations in which first iteration with a time slice of X units reduce the burst time of each process by X. In second iteration ------- --- --- ------ ----- ---- waiting time of each process.

Code:

//Question no -4 Narayan Chandravanshi (k18vq53 )

#include<stdio.h>

#include<string.h>

#include<conio.h>

int main()

{

char p[10][5],temp[5];

int i,j,pt[10],wt[10],totwt=0,pr[10],temp1,n;

float avgwt;

printf("enter no of processes:");

scanf("%d",&n);

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

{

printf("enter process%d name:",i+1);

scanf("%s",&p[i]);

printf("enter process time:");

scanf("%d",&pt[i]);

printf("enter priority:");

scanf("%d",&pr[i]);

}

for(i=0;i<n-1;i++)

{

for(j=i+1;j<n;j++)

{

if(pr[i]>pr[j])

{

temp1=pr[i];

pr[i]=pr[j];

pr[j]=temp1;

temp1=pt[i];

pt[i]=pt[j];

pt[j]=temp1;

strcpy(temp,p[i]);

strcpy(p[i],p[j]);

strcpy(p[j],temp);

}

}

}

wt[0]=0;

for(i=1;i<n;i++)

{

wt[i]=wt[i-1]+wt[i-1];

totwt=totwt+wt[i];

}

avgwt=(float)totwt/n;

printf("p\_name\t p\_time\t priority\t w\_time\n");

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

{

printf(" %s\t %d\t %d\t %d\n" ,p[i],pt[i],pr[i],wt[i]);

}

printf("total waiting time=%d\n avg waiting time=%f",totwt,avgwt);

int ts,pid[10],need[10],wt1[10],tat[10],i1,j1,n2,n1;

int bt[10],flag[10],ttat=0,twt=0;

float awt,atat;

printf("\nEnter the number of Processors \n");

scanf("%d",&n);

n1=n;

printf("\n Enter the Timeslice \n");

scanf("%d",&ts);

for(i=1;i<=n;i++)

{

printf("\n Enter the process ID %d",i);

scanf("%d",&pid[i]);

printf("\n Enter the Burst Time for the process");

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

need[i]=bt[i];

}

for(i=1;i<=n;i++)

{

flag[i]=1;

wt[i]=0;

}

while(n!=0)

{

for(i=1;i<=n;i++)

{

if(need[i]>=ts)

{

for(j=1;j<=n;j++)

{

if((i!=j)&&(flag[i]==1)&&(need[j]!=0))

wt[j]+=ts;

}

need[i]-=ts;

if(need[i]==0)

{

flag[i]=0;

n--;

}

}

else

{

for(j=1;j<=n;j++)

{

if((i!=j)&&(flag[i]==1)&&(need[j]!=0))

wt[j]+=need[i];

}

need[i]=0;

n--;

flag[i]=0;

}

}

}

for(i=1;i<=n1;i++)

{

tat[i]=wt[i]+bt[i];

twt=twt+wt[i];

ttat=ttat+tat[i];

}

awt=(float)twt/n1;

atat=(float)ttat/n1;

printf("\n\n Process \t Process ID \t BurstTime \t Waiting Time \t TurnaroundTime \n ");

for(i=1;i<=n1;i++)

{

printf("\n %5d \t %5d \t\t %5d \t\t %5d \t\t %5d \n", i,pid[i],bt[i],wt[i],tat[i]);

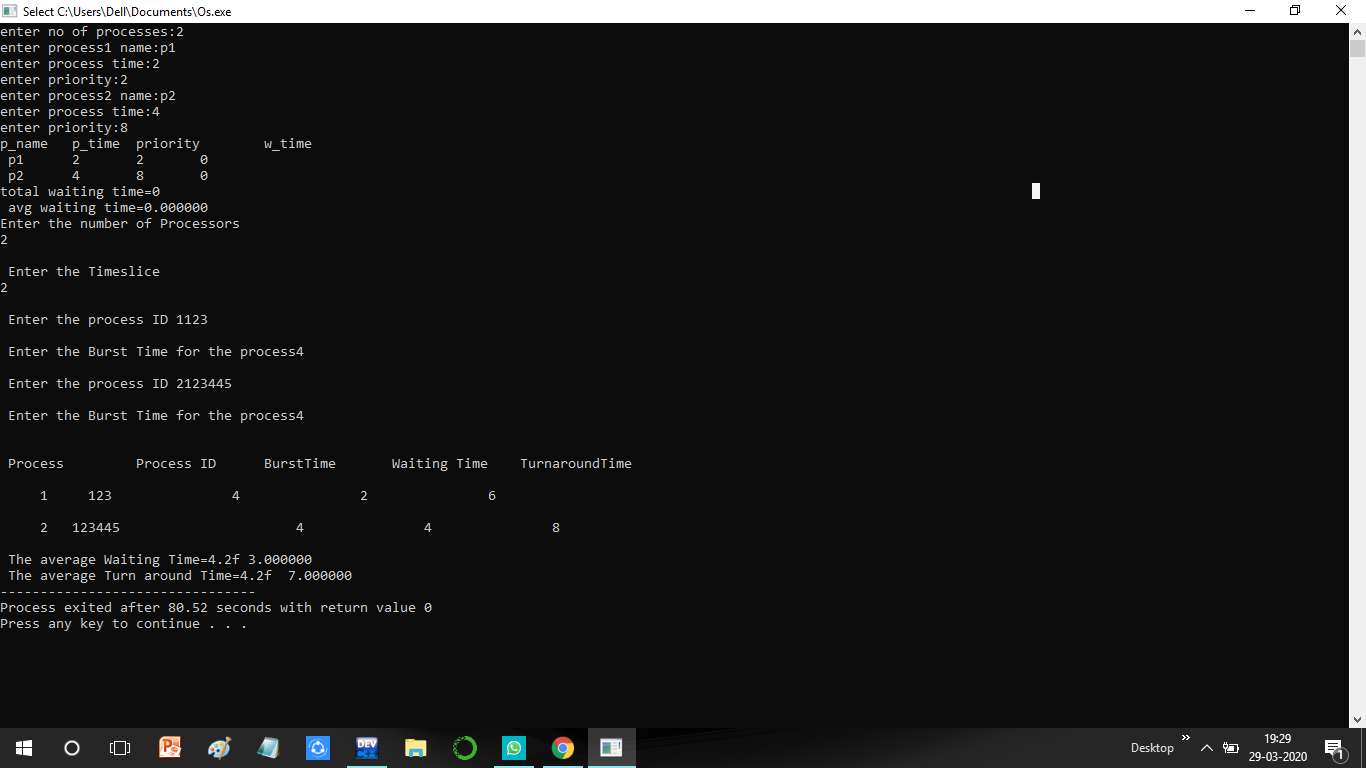
}

printf("\n The average Waiting Time=4.2f",awt);

printf("\n The average Turn around Time=4.2f",atat);

}

Output:



**For preemptive**

**In the above programme ask no of process input**

**After that it ask for name of process then priority**

**And then pocess time .**

**For Round robin**

**It ask for no of processors then process id then time slice(time quantum)**

**When u enter then u gives output as avg waiting time and turn around time.**

**Complexity of above programme is O(n^2).**