**Name: Shardul Vikram Dharmadhikari**

**Roll no: 8016**

ASSIGNMENT NO: 08

**AIM:** Write a Java program (using OOP features) to implement following scheduling algorithms: FCFS, SJF (Preemptive), Priority (Non-Preemptive) and Round Robin (Preemptive)

**OBJECTIVES:**

* To study the basics of scheduling algorithms and learn concept of Preemptive and Non-Preemptive scheduling.
* To understand aggregation functions.

**PRE-REQUISITES:**

1. Java basics.
2. Basics of Operating System.

**Schedular:-**

Schedular is an Operaing System module that selects the next job to be  admitted into the system & next process to run.There are major three types of schedular  basically as follows:**-**

**1. Short Term Schedular**

**2. Mid Term Schedular**

**3. Long Term Schedular**

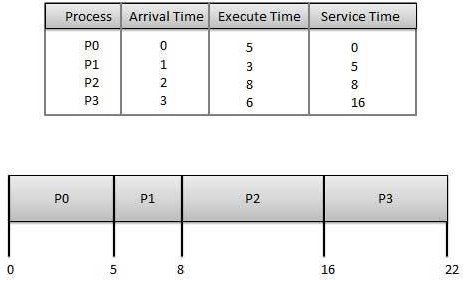
**Scheduling :-**

Schduling is the method specified by some means is assigned to resources  that complete the work; work may be either virtual computation elements like thread ,  processes , data flows etc. There is Major two types of scheduling algorithm to solve any sort  of operations.

**THEORY:**

|  |  |
| --- | --- |
| **Preemptive Scheduling**  Processor can be preempted to execute  a different process in the middle of  execution of any current process.  CPU utilization is more compared to  Non-Preemptive Scheduling.  Waiting time and Response time is less.  The preemptive scheduling is  prioritized. The highest priority process  should always be the process that is  currently utilized.  If a high priority process frequently  arrives in the ready queue, low priority  process may starve.  Preemptive scheduling is flexible. Ex:- SRTF, Priority, Round Robin, etc. Ex:- FCFS, SJF, Priority, etc. | **Non-Preemptive Scheduling**  Once Processor starts to execute a  process it must finish it before  executing the other. It cannot be  paused in middle.  CPU utilization is less compared to  Preemptive Scheduling.  Waiting time and Response time is  more.  When a process enters the state of  running, the state of that process is not  deleted from the scheduler until it  finishes its service time.  If a process with long burst time is  running CPU, then another process  with less CPU burst time may starve. Non-preemptive scheduling is rigid. |

* 1. **irst Come First Serve**
     + Jobs are executed on first come, first serve basis.
     + It is a non-preemptive, pre-emptive scheduling algorithm.
     + Easy to understand and implement.
     + Its implementation is based on FIFO queue.



Wait time of each process is as follows –

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Average Wait Time: (0+4+6+13) / 4 = 5.75

1. **Shortest Job First (Preemptive)**

For SJF scheduling algorithm, read the number of processes/jobs in the system, Their CPU burst times arrange all the jobs in order with respect to their burst times.  There may be two jobs in queue with the same execution time, and then FCFS approach  is to be performed. Each process will be executed according to the length of its burst  time. Then calculate the waiting time and turnaround time of each of the processes accordingly

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

P0 3 - 0 = 3

P1 0 - 0 = 0

P2 16 - 2 = 14

P3 8 - 3 = 5

**Average Wait Time: (3+0+14+5) / 4 = 5.50**

1. **Priority (Non-Preemptive)**

Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems. Each process is assigned a priority. Process with the highest priority is to be executed first and so on.

Processes with the same priority are executed on first come first served basis. Priority can be decided based on memory requirements, time requirements or any other resource requirement.

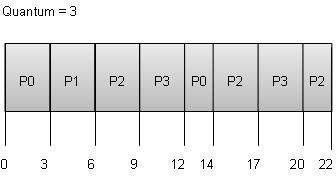
|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

# Round Robin (Preemptive)

* + Round Robin is the preemptive process scheduling algorithm.
  + Each process is provided a fix time to execute, it is called a quantum.
  + Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
  + Context switching is used to save states of preempted processes.





|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Average Wait Time: (9+2+12+11) / 4 = 8.5

**CONCLUSION:**

Thus we learned different Scheduling algorithms.

**CODE:**

/\* Problem Statement: Write a JAVA program (using oop features) to implement following

1. FCFS

2. SJF(Preemptive)

3. Priority(Non- Preemptive)

4. Round Robin(Preemptive)

1.FCFS

\*/

import java.io.\*;

import java.util.Scanner;

public class FCFS

{

public static void main(String args[])

{

int i,no\_p,burst\_time[],TT[],WT[];

float avg\_wait=0,avg\_TT=0;

burst\_time=new int[50];

TT=new int[50];

WT=new int[50];

WT[0]=0;

Scanner s=new Scanner(System.in);

System.out.println("Enter the number of process: ");

no\_p=s.nextInt();

System.out.println("\nEnter Burst Time for processes:");

for(i=0;i<no\_p;i++)

{

System.out.print("\tP"+(i+1)+": ");

burst\_time[i]=s.nextInt();

}

for(i=1;i<no\_p;i++)

{

WT[i]=WT[i-1]+burst\_time[i-1];

avg\_wait+=WT[i];

}

avg\_wait/=no\_p;

for(i=0;i<no\_p;i++)

{

TT[i]=WT[i]+burst\_time[i];

avg\_TT+=TT[i];

}

avg\_TT/=no\_p;

System.out.println("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("\tProcesses:");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println(" Process\tBurst Time\tWaiting Time\tTurn Around Time");

for(i=0;i<no\_p;i++)

{

System.out.println("\tP"+(i+1)+"\t "+burst\_time[i]+"\t\t "+WT[i]+"\t\t "+TT[i]);

}

System.out.println("\n----------------------------------------------------------------");

System.out.println("\nAverage waiting time : "+avg\_wait);

System.out.println("\nAverage Turn Around time : "+avg\_TT+"\n");

}

}

/\*Output:

Enter the number of process:

3

Enter Burst Time for processes:

P1: 24

P2: 3

P3: 3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Processes:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Process Burst Time Waiting Time Turn Around Time

P1 24 0 24

P2 3 24 27

P3 3 27 30

----------------------------------------------------------------

Average waiting time : 17.0

Average Turn Around time : 27.0 \*/

/\*Round Robin(Preemptive)\*/

import java.util.\*;

import java.io.\*;

class RoundR

{

public static void main(String args[])

{

int Process[]=new int[10];

int a[]=new int[10];

int Arrival\_time[]=new int[10];

int Burst\_time[]=new int[10];

int WT[]=new int[10];

int TAT[]=new int[10];

int Pno,sum=0;;

int TimeQuantum;

System.out.println("\nEnter the no. of Process::");

Scanner sc=new Scanner(System.in);

Pno=sc.nextInt();

System.out.println("\nEnter each process::");

for(int i=0;i<Pno;i++)

{

Process[i]=sc.nextInt();

}

System.out.println("\nEnter the Burst Time of each process::");

for(int i=0;i<Pno;i++)

{

Burst\_time[i]=sc.nextInt();

}

System.out.println("\nEnter the Time Quantum::");

TimeQuantum=sc.nextInt();

do{

for(int i=0;i<Pno;i++)

{

if(Burst\_time[i]>TimeQuantum)

{

Burst\_time[i]-=TimeQuantum;

for(int j=0;j<Pno;j++)

{

if((j!=i)&&(Burst\_time[j]!=0))

WT[j]+=TimeQuantum;

}

}

else

{

for(int j=0;j<Pno;j++)

{

if((j!=i)&&(Burst\_time[j]!=0))

WT[j]+=Burst\_time[i];

}

Burst\_time[i]=0;

}

}

sum=0;

for(int k=0;k<Pno;k++)

sum=sum+Burst\_time[k];

} while(sum!=0);

for(int i=0;i<Pno;i++)

TAT[i]=WT[i]+a[i];

System.out.println("process\t\tBT\tWT\tTAT");

for(int i=0;i<Pno;i++)

{

System.out.println("process"+(i+1)+"\t"+a[i]+"\t"+WT[i]+"\t"+TAT[i]);

}

float avg\_wt=0;

float avg\_tat=0;

for(int j=0;j<Pno;j++)

{

avg\_wt+=WT[j];

}

for(int j=0;j<Pno;j++)

{

avg\_tat+=TAT[j];

}

System.out.println("average waiting time "+(avg\_wt/Pno)+"\n Average turn around time"+(avg\_tat/Pno));

}

}

/\*OUTPUT::

unix@unix-HP-280-G1-

MT:~/TEA33$ java RoundR

Enter the no. of Process::

5

Enter each process::

1

2

3

4

5

Enter the Burst Time of each process::

2

1

8

4

5

Enter the Time Quantum::

2

process BT WT TAT

process1 0 0 0

process2 0 2 2

process3 0 12 12

process4 0 9 9

process5 0 13 13

average waiting time 7.2

Average turn around time7.2 \*/

/\* 2. SJF(Non-Preemptive) \*/

import java.util.Scanner;

class SJF1{

public static void main(String args[]){

int burst\_time[],process[],waiting\_time[],tat[],i,j,n,total=0,pos,temp;

float wait\_avg,TAT\_avg;

Scanner s = new Scanner(System.in);

System.out.print("Enter number of process: ");

n = s.nextInt();

process = new int[n];

burst\_time = new int[n];

waiting\_time = new int[n];

tat = new int[n];

System.out.println("\nEnter Burst time:");

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

{

System.out.print("\nProcess["+(i+1)+"]: ");

burst\_time[i] = s.nextInt();;

process[i]=i+1; //Process Number

}

//Sorting

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

{

pos=i;

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

{

if(burst\_time[j]<burst\_time[pos])

pos=j;

}

temp=burst\_time[i];

burst\_time[i]=burst\_time[pos];

burst\_time[pos]=temp;

temp=process[i];

process[i]=process[pos];

process[pos]=temp;

}

//First process has 0 waiting time

waiting\_time[0]=0;

//calculate waiting time

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

{

waiting\_time[i]=0;

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

waiting\_time[i]+=burst\_time[j];

total+=waiting\_time[i];

}

//Calculating Average waiting time

wait\_avg=(float)total/n;

total=0;

System.out.println("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");

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

{

tat[i]=burst\_time[i]+waiting\_time[i]; //Calculating Turnaround Time

total+=tat[i];

System.out.println("\n p"+process[i]+"\t\t "+burst\_time[i]+"\t\t "+waiting\_time[i]+"\t\t "+tat[i]);

}

//Calculation of Average Turnaround Time

TAT\_avg=(float)total/n;

System.out.println("\n\nAverage Waiting Time: "+wait\_avg);

System.out.println("\nAverage Turnaround Time: "+TAT\_avg);

}

}

/\* 2. SJF(Preemptive)\*/

import java.util.Scanner;

class sjf\_swap1{

public static void main(String args[])

{

int burst\_time[],process[],waiting\_time[],tat[],arr\_time[],completion\_time[],i,j,n,total=0,total\_comp=0,pos,temp;

float wait\_avg,TAT\_avg;

Scanner s = new Scanner(System.in);

System.out.print("Enter number of process: ");

n = s.nextInt();

process = new int[n];

burst\_time = new int[n];

waiting\_time = new int[n];

arr\_time=new int[n];

tat = new int[n];

completion\_time=new int[n];

//burst time

System.out.println("\nEnter Burst time:");

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

{

System.out.print("\nProcess["+(i+1)+"]: ");

burst\_time[i] = s.nextInt();;

process[i]=i+1; //Process Number

}

//arrival time

System.out.println("\nEnter arrival time:");

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

{

System.out.print("\nProcess["+(i+1)+"]: ");

arr\_time[i] = s.nextInt();;

process[i]=i+1; //Process Number

}

//Sorting

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

{

pos=i;

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

{

if(burst\_time[j]<burst\_time[pos])

pos=j;

}

temp=burst\_time[i];

burst\_time[i]=burst\_time[pos];

burst\_time[pos]=temp;

temp=process[i];

process[i]=process[pos];

process[pos]=temp;

System.out.println("process"+process[i]);

}

//completion

time new

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

{

completion\_time[i]=0;

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

completion\_time[i]+=burst\_time[j];

total\_comp+=completion\_time[i];

}

//First process has 0 waiting

time

waiting\_time[0]=0;

//calculate

waiting time

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

{

waiting\_time[i]=0;

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

waiting\_time[i]+=burst\_time[j];

total+=waiting\_time[i];

}

//Calculating Average waiting time

wait\_avg=(float)total/n;

total=0;

System.out.println("\nPro\_number\t Burst Time \tcompletion\_time\tWaiting Time\tTurnaround Time");

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

{

tat[i]=burst\_time[i]+waiting\_time[i];

//Calculating Turnaround Time

total+=tat[i];

System.out.println("\n"+process[i]+"\t\t "+burst\_time[i]+"\t\t "+completion\_time[i]+"\t\t"+waiting\_time[i]+"\t\t "+tat[i]);

}

//Calculation of Average Turnaround Time

TAT\_avg=(float)total/n;

System.out.println("\n\nAWT: "+wait\_avg);

System.out.println("\nATAT: "+TAT\_avg);

}

}