

EXPERIMENT NO : 4

Name: Fale Manish Dinkar

Class: TE

Div: A

Roll No: T211036

Batch: A2

Problem Statement Write a program to simulate CPU Scheduling Algorithms:

1. FCFS
2. SJF(Preemptive)
3. Priority(Non- Preemptive)
4. Round Robin(Preemptive)

1. FCFS

```
package FCFS;
import java.util.*;
class FCFS {

    public static void main(String[] args)
    {
        int id[]=new int[20];
        int etime[]=new int[20];
        int stime[]=new int[20];
        int wtime[]=new int[20];
        int tat[]=new int[20];
        int total=0,total1=0;
        float avg,avg1;
        Scanner sn=new Scanner(System.in);
        System.out.print("\nEnter the number of processes :");
        int n=sn.nextInt();
        for(int i=0;i<n;i++)
        {
            System.out.println();
            System.out.print("\nEnter the process ID of process"+ i+1+":");
            id[i]=sn.nextInt();
            System.out.print("Enter the execution time of process"+(i+1)+":");
            etime[i]=sn.nextInt();
        }
        stime[0]=0;
        for (int i=1;i<n;i++)
        {
            stime[i]=stime[i-1]+etime[i-1];
        }

        wtime[0]=0;
        for(int i=0;i<n;i++)
        {
```

```

        wtime[i]=stime[i]-id[i];
        total=total+wtme[i];
    }
    for(int i=0;i<n;i++)
    {
        tat[i]=wtme[i]+etime[i];
        total1=total1+tat[i];
    }
    avg=(float)total/n;
    avg1=(float)total1/n;

    System.out.println("\nArrival_time\tExecution_time\tService_time\tWait_time\tturn_around
time");

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

        System.out.println(id[i]+"\\t\\t"+etime[i]+"\\t\\t"+stime[i]+"\\t\\t"+wtme[i]+"\\t\\t"+tat[i]);
    }
    System.out.println("\nAverage turn around time:"+avg1+"\nAverage wait time:"+ avg);
}
}

```

Output:

The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Shows the project structure with 'SPOS1' containing 'src' and 'module-info.java'.
- Editor:** Displays the code for 'FCFS.java'.
- Console:** Shows the execution output, including prompts for process IDs and execution times, a table of results, and average calculations.

Console Output:

```

-terminated> FCFS [Java Application] /snap/eclipse/62/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_17.0.4.v20220903-1038/jre/bin/java (14-Nov-2022, 11:13:17 am - 11:14:05 am) [pid: 36930]

Enter the number of processes :4

Enter the process ID of process1:0
Enter the execution time of process1:8

Enter the process ID of process2:1
Enter the execution time of process2:4

Enter the process ID of process3:2
Enter the execution time of process3:9

Enter the process ID of process4:3
Enter the execution time of process4:5

Arrival_time    Execution_time  Service_time    Wait_time       turn_around time
0                4              8               0               8
1                9              12              7               11
2                5              21              10              19
3                5              21              18              23

Average turn around time:15.25
Average wait time:8.75

```

2. SJF(Preemptive)

```
import java.util.*;

public class SJFP {
    public static void main (String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.println ("enter no of process:");
        int n= sc.nextInt();
        int pid[] = new int[n]; // it takes pid of process
        int at[] = new int[n]; // at means arrival time
        int bt[] = new int[n]; // bt means burst time
        int ct[] = new int[n]; // ct means complete time
        int ta[] = new int[n]; // ta means turn around time
        int wt[] = new int[n]; // wt means waiting time
        int f[] = new int[n]; // f means it is flag it checks process is completed or not
        int k[]= new int[n]; // it is also stores burst time
        int i, st=0, tot=0;
        float avgwt=0, avgta=0;

        for (i=0;i<n;i++)
        {
            pid[i]= i+1;
            System.out.println ("enter process " +(i+1)+ " arrival time:");
            at[i]= sc.nextInt();
            System.out.println("enter process " +(i+1)+ " burst time:");
            bt[i]= sc.nextInt();
            k[i]= bt[i];
            f[i]= 0;
        }

        while(true){
            int min=99,c=n;
            if (tot==n)
                break;

            for ( i=0;i<n;i++)
            {
                if ((at[i]<=st) && (f[i]==0) && (bt[i]<min))
                {
                    min=bt[i];
                    c=i;
                }
            }

            if (c==n)
                st++;
            else
            {
                bt[c]--;
                st++;
            }
        }
    }
}
```

```

        if (bt[c]==0)
        {
            ct[c]= st;
            f[c]=1;
            tot++;
        }
    }
}

for(i=0;i<n;i++)
{
    ta[i] = ct[i] - at[i];
    wt[i] = ta[i] - k[i];
    avgwt+= wt[i];
    avgta+= ta[i];
}

System.out.println("pid arrival burst complete turn waiting");
for(i=0;i<n;i++)
{
    System.out.println(pid[i] + "\t" + at[i] + "\t" + k[i] + "\t" + ct[i] + "\t" + ta[i] + "\t" + wt[i]);
}

System.out.println("\naverage tat is " + (float)(avgta/n));
System.out.println("average wt is " + (float)(avgwt/n));
sc.close();
}
}

```

Output:

```

"C:\Program Files\Java\jdk-18.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.2\lib\idea_rt.jar=53717:C:\Program Files\JetBrains
enter no of process:
4
enter process 1 arrival time:
2
enter process 1 burst time:
1
enter process 2 arrival time:
4
enter process 2 burst time:
5
enter process 3 arrival time:
7
enter process 3 burst time:
8
enter process 4 arrival time:
8
enter process 4 burst time:
3

pid arrival burst complete turn waiting
1 2 1 3 1 0
2 4 5 9 5 0
3 7 8 9 2 2
4 8 3 12 4 1

average tat is 3.0
average wt is 0.75

```

3. Priority(Non- Preemptive)

```
package priority;

import java.util.*;

class Process
{
    int pid; // Process ID
    int bt; // CPU Burst time required
    int priority; // Priority of this process
    Process(int pid, int bt, int priority)
    {
        this.pid = pid;
        this.bt = bt;
        this.priority = priority;
    }
    public int prior()
    {
        return priority;
    }
}

public class priority
{
    //Function to finding waiting time for all processes
    public void findwaitingTime(Process proc[], int n, int wt[])
    {
        //waiting time for first process is 0
        wt[0] = 0;

        //calculating waiting time
        for (int i = 1; i < n; i++)
            wt[i] = proc[i-1].bt + wt[i-1];
    }

    //Function to calculate turn around time
    public void findTurnArounTime(Process proc[], int n, int wt[], int tat[])
    {
        // calculating turn around time by adding bt[i] + wt[i]
        for (int i = 0; i < n; i++)
            tat[i] = proc[i].bt + wt[i];
    }

    //Function to calculate average time
    public void findavgTime(Process proc[], int n)
    {
        int wt[] = new int[n], tat[] = new int[n], total_wt = 0, total_tat = 0;

        //function to find waiting time of all processes
        findwaitingTime(proc, n, wt);

        //Function to find turn around time for all process
        findTurnArounTime(proc, n, wt, tat);

        //Display processes along with all details
        System.out.print("\nProcesses Burst time Waiting time Turn around time\n");

        //Calculate total waiting time and toatl turnaround time
        for (int i = 0; i < n; i++)
```

```

        {
            total_wt = total_wt + wt[i];
            total_tat = total_tat + tat[i];
            System.out.print(" " + proc[i].pid + "\t\t" + proc[i].bt + "\t\t" + wt[i] + "\t\t" + tat[i] + "\n");
        }

        System.out.print("\nAverage waiting time = " + (float)total_wt / (float)n);
    }

public void priorityScheduling(Process proc[], int n)
{
    // Sort process by priority
    Arrays.sort(proc, new Comparator<Process>()
    {
        @Override
        public int compare(Process a, Process b)
        {
            return b.prior() - a.prior();
        }
    });

    System.out.print("Order in which processes gets executed \n");
    for (int i=0; i<n; i++)
        System.out.print(proc[i].pid + "");

    findavgTime(proc, n);
}

public static void main(String[] args)
{
    priority ob = new priority();
    int n = 3;
    Process proc[] = new Process[n];
    proc[0] = new Process(1, 10, 2);
    proc[1] = new Process(2, 5, 0);
    proc[2] = new Process(3, 8, 1);
    ob.priorityScheduling(proc, n);
}
}

```

Output:

The screenshot shows the Eclipse IDE with the following code and output:

```

package priority;
import java.util.*;
class Process
{
    int pid; // Process ID
    int bt; // CPU Burst time required
    int priority; // Priority of this process
    Process(int pid, int bt, int priority)
    {
        this.pid = pid;
        this.bt = bt;
        this.priority = priority;
    }
    public int prior()
    {
        return priority;
    }
}

public class priority
{
    //Function to finding waiting time for all processes
    public void findWaitingTime(Process proc[], int n, int wt[])
    {
        //waiting time for first process is 0
        wt[0] = 0;

        //calculating waiting time
        for (int i = 1; i < n; i++)
            wt[i] = proc[i-1].bt + wt[i-1];
    }

    //Function to calculate turn around time
    public void findTurnArounTime(Process proc[], int n, int wt[], int tat[])
    {
        // calculating turn around time by adding bt[i] + wt[i]
        for (int i = 0; i < n; i++)
            tat[i] = proc[i].bt + wt[i];
    }

    //Function to calculate average time
    public void findavgTime(Process proc[], int n)
    {
        int wt[] = new int[n], tat[] = new int[n], total_wt = 0, total_tat = 0;

        //function to find waiting time of all processes
        findWaitingTime(proc, n, wt);
    }
}

```

Output:

```

Order in which processes gets executed
1      10      0      10
2       5      10      15
3       8      18      23

Average waiting time = 9.333333

```

4. Round Robin(Preemptive)

```
package RoundRobin;
import java.util.Scanner;
public class RoundRobin
{
    public static void main(String args[])
    {
        int n,i,qt,count=0,temp,sq=0,bt[],wt[],tat[],rem_bt[];
        float awt=0,atat=0;
        bt = new int[10];
        wt = new int[10];
        tat = new int[10];
        rem_bt = new int[10];
        Scanner s=new Scanner(System.in);
        System.out.print("Enter the number of process (maximum 10) = ");
        n = s.nextInt();
        System.out.print("Enter the burst time of the process\n");
        for (i=0;i<n;i++)
        {
            System.out.print("P"+i+" = ");
            bt[i] = s.nextInt();
            rem_bt[i] = bt[i];
        }
        System.out.print("Enter the quantum time: ");
        qt = s.nextInt();
        while(true)
        {
            for (i=0,count=0;i<n;i++)
            {
                temp = qt;
                if(rem_bt[i] == 0)
                {
                    count++;
                    continue;
                }
                if(rem_bt[i]>qt)
                    rem_bt[i]= rem_bt[i] - qt;
                else
                if(rem_bt[i]>=0)
                {
                    temp = rem_bt[i];
                    rem_bt[i] = 0;
                }
            }
            sq = sq + temp;
            tat[i] = sq;
        }
        if(n == count)
            break;
    }
    System.out.print("-----");
```

```

System.out.print("\nProcess\t    Burst Time\t    Turnaround Time\t    Waiting
Time\n");
System.out.print("-----");
for(i=0;i<n;i++)
{
    wt[i]=tat[i]-bt[i];
    awt=awt+wt[i];
    atat=atat+tat[i];
    System.out.print("\n "+(i+1)+"\t "+bt[i]+" \t\t "+tat[i]+" \t\t "+wt[i]+" \n");
}
awt=awt/n;
atat=atat/n;
System.out.println("\nAverage waiting Time = "+awt+"\n");
System.out.println("Average turnaround time = "+atat);
}
}

```

OUTPUT:-

The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Shows the project structure with 'Pass2_of_two_pass' and its sub-packages like 'src', 'JRE System Library [Java]', 'dinning', 'FCFS', 'pass1macro', 'pass2', 'ReaderWriter', 'RoundRobin', 'SIFNP', 'SIFP', and 'module-info.java'.
- Editor:** Displays the 'RoundRobin.java' file with the following code:


```

1 package RoundRobin;
2
3 <terminated> RoundRobin [Java Application] C:\Users\prera\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe
4 Enter the number of process (maximum 10) = 7
5 Enter the burst time of the process
6 P0 = 8
7 P1 = 10
8 P2 = 15
9 P3 = 19
10 P4 = 35
11 P5 = 20
12 P6 = 28
13 Enter the quantum time: 4
14
15 Process      Burst Time      Turnaround Time      Waiting Time
16 -----
17 1          8          32          24
18 2         10          58          48
19 3         15          81          66
20 4         19         100          81
21 5         35         135         100
22 6         20         108          88
23 7         28         128         100
24
25 Average waiting Time = 72.42857
26 Average turnaround time = 91.71429

```
- Console:** Shows the execution output, including the input values and the table of process metrics.
- Outline:** Shows the 'main(String[]) : void' method.