Assignment No: 07

Title:
Implementation of Instruction Scheduling Algorithm.

Objectives:
- To understand OS & SCHEDULLING Concepts
- To implement Scheduling FCFS, SJF, RR & Priority algorithms
- To study about Scheduling and scheduler

Outcomes:
After completion of this assignment students will be able to:
- Knowledge Scheduling policies
- Compare different scheduling algorithms

Software Requirements: JDK toolkit

Hardware Requirements: 4 GB RAM,500 GB HDD

Theory Concepts:

CPU Scheduling:
• CPU scheduling refers to a set of policies and mechanisms built into the operating systems that govern the order in which the work to be done by a computer system is completed.
• Scheduler is an OS module that selects the next job to be admitted into the system and next process to run.

What is scheduling?
Scheduling is defined as the process that governs the order in which the work is to be done. Scheduling is done in the areas where more no. of jobs or works are to be performed. Then it requires some plan i.e. scheduling that means how the jobs are to be performed i.e. order. CPU scheduling is best example of scheduling.

What is scheduler?
1. Scheduler in an OS module that selects the next job to be admitted into the system and the next process to run.
2. Primary objective of the scheduler is to optimize system performance in accordance with the criteria deemed by the system designers. In short, scheduler is that module of OS which schedules the programs in an efficient manner.

Necessity of scheduling
• Scheduling is required when no. of jobs are to be performed by CPU.
• Scheduling provides mechanism to give order to each work to be done.
• Primary objective of scheduling is to optimize system performance.

• Scheduling provides the ease to CPU to execute the processes in efficient manner.

Types of schedulers
In general, there are three different types of schedulers which may co-exist in a complex operating system.
• Long term scheduler
• Medium term scheduler
• Short term scheduler.

Long Term Scheduler
• The long term scheduler, when present works with the batch queue and selects the next batch job to be executed.
• Batch is usually reserved for resource intensive (processor time, memory, special I/O devices) low priority programs that may be used fillers of low activity of interactive jobs.
• Batch jobs usually also contains programmer-assigned or system-assigned estimates of their resource needs such as memory size, expected execution time and device requirements.
• Primary goal of long term scheduler is to provide a balanced mix of jobs.

Medium Term Scheduler
• After executing for a while, a running process may because suspended by making an I/O request or by issuing a system call.
• When number of processes becomes suspended, the remaining supply of ready processes in systems where all suspended processes remains resident in memory may become reduced to a level that impairs functioning of schedulers.
• The medium term scheduler is in charge of handling the swapped out processes.
• It has little to do while a process is remained as suspended.

Short Term Scheduler
• The short term scheduler allocates the processor among the pool of ready processes resident in the memory.
• Its main objective is to maximize system performance in accordance with the chosen set of criteria.
• Some of the events introduced thus for that cause rescheduling by virtue of their ability to change the global system state are:
• Clock ticks
• Interrupt and I/O completions
• Most operational OS calls
• Sending and receiving of signals
• Activation of interactive programs.
• Whenever one of these events occurs ,the OS involves the short term scheduler.

Scheduling Criteria :
CPU Utilization:Keep the CPU as busy as possible. It range from 0 to 100%. In practice, it range from 40 to 90%.
Throughput: Throughput is the rate at which processes are completed per unit of time.
Turnaround time:This is the how long a process takes to execute a process. It is calculated as the time gap between the submission of a process and its completion.

Waiting time:Waiting time is the sum of the time periods spent in waiting in the ready queue.
Response time:Response time is the time it takes to start responding from submission time. It is calculated as the amount of time it takes from when a request was submitted until the first response is produced.

Non-preemptive Scheduling:
In non-preemptive mode, once if a process enters into running state, it continues to execute until it terminates or blocks itself to wait for Input/output or by requesting some operating system service.
Preemptive Scheduling:
In preemptive mode, currently running process may be interrupted and moved to the ready State by the operating system.
When a new process arrives or when an interrupt occurs, preemptive policies may incur greater overhead than non-preemptive version but preemptive version may provide better service.
It is desirable to maximize CPU utilization and throughput, and to minimize turnaround time, waiting time and response time.
Types of scheduling Algorithms
• In general, scheduling disciplines may be pre-emptive or non-pre-emptive .
• In batch, non-pre-emptive implies that once scheduled, a selected job turns to completion.
There are different types of scheduling algorithms such as:
1.FCFS(First Come First Serve)
2.SJF(Short Job First)
3.Priority scheduling
4.Round Robin scheduling algorithm
5.First Come First Serve Algorithm.

FCFS:
• FCFS is working on the simplest scheduling discipline.
• The workload is simply processed in an order of their arrival, with no pre-emption.
• FCFS scheduling may result into poor performance.

First Come, First Served

Process	Burst Time
P1	24
P2	3
P3	3

- Suppose that the processes arrive in the order:
  P1 , *P2* , *P3*
- The Gantt Chart for the schedule is:

- Waiting time for *P1* = 0; *P2* = 24; *P3* = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

Advantages:
• Better for long processes
• Simple method (i.e., minimum overhead on processor)
• No starvation
Disadvantages:
• Convoy effect occurs. Even very small process should wait for its turn to come to utilize the CPU. Short process behind long process results in lower CPU utilization.
• Throughput is not emphasized.

Shortest Job First Algorithm:
This is also known as shortest job first, or SJF
This is a non-preemptive, pre-emptive scheduling algorithm.
Best approach to minimize waiting time.
Easy to implement in Batch systems where required CPU time is known in advance.
Impossible to implement in interactive systems where required CPU time is not known.
The processer should know in advance how much time process will take.

Advantages
• It gives superior turnaround time performance to shortest process next because a short job is given immediate preference to a running longer job.
• Throughput is high.
Disadvantages
• Elapsed time (i.e., execution-completed-time) must be recorded, it results an additional overhead on the processor.
• Starvation may be possible for the longer processes.

This algorithm is divided into two types:
• Pre-emptive SJF
• Non-pre-emptive SJF

Pre-emptive SJF Algorithm:
In this type of SJF, the shortest job is executed 1st. the job having least arrival time is taken first for execution. It is executed till the next job arrival is reached.

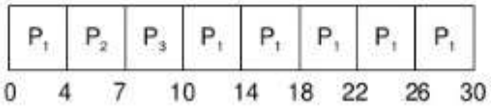
Non-pre-emptive SJF Algorithm:
In this algorithm, job having less burst time is selected 1st for execution. It is executed for its total burst time and then the next job having least burst time is selected.

Round Robin Scheduling:
• 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.

Round Robin

Process	Burst Time
P1	24
P2	3
P3	3

- Quantum time = 4 milliseconds
- The Gantt chart is:



- Average waiting time = {[0+(10-4)]+4+7}/3 = 5.6

Priority Scheduling:

• Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.

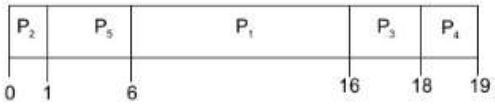
• Each process is assigned a priority. Process with highest priority is to be executed first and so on.Processes with same priority are executed on first come first served basis.

• Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Priority

Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

- Gantt Chart



- Average waiting time = (6 + 0 + 16 + 18 + 1)/5 = 8.2

Conclusion:Therefore we have studied and understand OS & Scheduling Concepts and implemented various algorithms and compared working analysis of each.

Code:

FCFS.java

```
import java.util.Scanner;
public class FCFS {
int Count=0;
int[][] processData;
Scanner scan = new Scanner(System.in);
public void scheduleFCFS(){
processData[0][2] = 0;
processData[0][3] = processData[0][1];
for(int i=1;i<Count;i++){
processData[i][2] = processData[i-1][3];
processData[i][3] = processData[i][1]+processData[i][2];
}
}
public void setValue(){
System.out.print("Enter Number Of processes : ");
Count = scan.nextInt();
processData = new int[Count][4];
for(int i=0;i<Count;i++){
System.out.print("Processes["+i+"] :-> ");
processData[i][0] = i;
processData[i][1] = scan.nextInt();
}
}
public void DisplayValue(){
System.out.println("-------------------------------------------------------------------------");
System.out.println("Process Number || Brust Time || WatingTime || Turn Around Time");
System.out.println("-------------------------------------------------------------------------");
for(int i=0;i<Count;i++){
System.out.println("Processes["+processData[i][0]+"]\t||\t"+processData[i][1]+"\t||\t"+processData[i]
[2]+"\t||\t"+processData[i][3]);
}
System.out.println("-------------------------------------------------------------------------");
}
public static void main(String a[]){
FCFS fs = new FCFS();
fs.setValue();
System.out.println("\n\n**********First Come First Server :(FCFS) :**********\n");
System.out.println("\n\nBefore Scheduling\n");
fs.DisplayValue();
fs.scheduleFCFS();
System.out.println("\n\nAfter Scheduling\n");fs.DisplayValue();
}
```

}

Output:

```
(base) dell@dell-Inspiron-15-3567:~/Downloads/Compiler/Assignment7$ javac FCFS.java
(base) dell@dell-Inspiron-15-3567:~/Downloads/Compiler/Assignment7$ java FCFS
Enter Number Of processes : 3
Processes[0] :-> 12
Processes[1] :-> 5
Processes[2] :-> 6

**********First Come First Server :(FCFS) :**********


Before Scheduling

---------------------------------------------------------------------
Process Number || Brust Time || WatingTime || Turn Around Time
---------------------------------------------------------------------
Processes[0]   ||        12       ||        0        ||        0
Processes[1]   ||        5        ||        0        ||        0
Processes[2]   ||        6        ||        0        ||        0
---------------------------------------------------------------------


After Scheduling

---------------------------------------------------------------------
Process Number || Brust Time || WatingTime || Turn Around Time
---------------------------------------------------------------------
Processes[0]   ||        12       ||        0        ||        12
Processes[1]   ||        5        ||        12       ||        17
Processes[2]   ||        6        ||        17       ||        23
---------------------------------------------------------------------
(base) dell@dell-Inspiron-15-3567:~/Downloads/Compiler/Assignment7$
```

RoundRobin.java

```
import java.util.Scanner;
public class RoundRobin {
int Count=0;
int[][] processData;
int[] CCycle; // Complete Cycle
int[] PCycle; // Partial Cycle
int[] TCycle; // Total Cycle
String[] Queue;
boolean[] Status;
int[] startQue;
int[] endQue;
int RTT = 0;
Scanner scan = new Scanner(System.in);
public void scheduleRR(){
int TotalProcess = CCycle.length;
for(int i = 0,j=0;i < Queue.length || j<Queue.length;i++){
int ProNo = i % TotalProcess;
if(CCycle[ProNo] != 0){
Queue[j] = "process["+ProNo+"]";
if(j==0)
startQue[j] = 0;
else
startQue[j] = endQue[j-1] + 1;
if(!Status[ProNo]){
processData[ProNo][2] = startQue[j];
Status[ProNo] = true;}
endQue[j] = startQue[j]+RTT-1;
j++;
CCycle[ProNo] = CCycle[ProNo] - 1;
}else{
if(Status[ProNo]){
Queue[j] = "process["+ProNo+"]";
if(j==0)
startQue[j] = 0;
else
startQue[j] = endQue[j-1] + 1;
endQue[j] = startQue[j]+PCycle[ProNo]-1;
processData[ProNo][3] = endQue[j];
PCycle[ProNo]=0;
j++;
Status[ProNo]= false;
```

```
}
}
}
for (int i = 0; i < Queue.length ; i++){
int temp = (endQue[i]-startQue[i]+1);
System.out.println(" "+Queue[i]+" "+startQue[i]+" "+endQue[i] +""+temp );
if(i%Count == Count-1)
System.out.println("");
}}
public void setValue(){
System.out.print("Enter Number Of processes : ");
Count = scan.nextInt();
System.out.print("Enter Round Trip Time : ");
RTT = scan.nextInt();
processData = new int[Count][4];
CCycle = new int[Count];
PCycle = new int[Count];
TCycle = new int[Count];
for(int i=0;i<Count;i++){
System.out.print("Processes["+i+"] :-> ");
processData[i][0] = i;
processData[i][1] = scan.nextInt();
CCycle[i] = processData[i][1]/RTT;
PCycle[i] = processData[i][1]%RTT;
if(PCycle[i]==0)
TCycle[i] = CCycle[i];
else
TCycle[i] = CCycle[i] + 1;
System.out.println("P["+i+"] CC : "+CCycle[i]+" PC : "+PCycle[i]+" TC :"+TCycle[i]);
}
Status = new boolean[Count];
for(int i=0;i<Count;i++){
Status[i]= false;
}
initQueue();
}
public void initQueue(){
int MaxQueLen = 0;
for(int i=0;i<TCycle.length;i++){
MaxQueLen = MaxQueLen + TCycle[i];
}
Queue = new String[MaxQueLen];
startQue = new int[MaxQueLen];
endQue = new int[MaxQueLen];
System.out.println("Queue Length : "+MaxQueLen);
}
public void DisplayValue(){
```

```
System.out.println("---------------------------------------------------------------------");
System.out.println("Process Number || Brust Time || Wating Time || TurnAround Time");
System.out.println("---------------------------------------------------------------------");
for(int i=0;i<Count;i++){
System.out.println("Processes["+processData[i]
[0]+"]\t||\t"+processData[i][1]+"\t||\t"+processData[i][2]+"\t||\t"+processData[i][3]);
}
System.out.println("---------------------------------------------------------------------");
}
public static void main(String a[]){RoundRobin rr = new RoundRobin();
rr.setValue();
System.out.println("\n\n**********Round Robin : (RR) :**********\n");
System.out.println("\n\nBefore Scheduling\n");rr.DisplayValue();
rr.scheduleRR();
System.out.println("\n\nAfter Scheduling\n");
rr.DisplayValue();
}
}
```

Output:

```
(base) dell@dell-Inspiron-15-3567:~/Downloads/Compiler/Assignment7$ javac RoundRobin.java
(base) dell@dell-Inspiron-15-3567:~/Downloads/Compiler/Assignment7$ java RoundRobin
Enter Number Of processes : 5
Enter Round Trip Time : 3
Processes[0] :-> 11
P[0] CC : 3 PC : 2 TC :4
Processes[1] :-> 16
P[1] CC : 5 PC : 1 TC :6
Processes[2] :-> 13
P[2] CC : 4 PC : 1 TC :5
Processes[3] :-> 5
P[3] CC : 1 PC : 2 TC :2
Processes[4] :-> 8
P[4] CC : 2 PC : 2 TC :3
Queue Length : 20


**********Round Robin : (RR) :**********



Before Scheduling

---------------------------------------------------------------------
```

Process Number || Brust Time || Wating Time || TurnAround Time
---
Processes[0] || 11 || 0 || 0
Processes[1] || 16 || 0 || 0
Processes[2] || 13 || 0 || 0
Processes[3] || 5 || 0 || 0
Processes[4] || 8 || 0 || 0
---
process[0] 0 23
process[1] 3 53
process[2] 6 83
process[3] 9 113
process[4] 12 143

process[0] 15 173
process[1] 18 203
process[2] 21 233
process[3] 24 252
process[4] 26 283

process[0] 29 313
process[1] 32 343
process[2] 35 373
process[4] 38 392
process[0] 40 412

process[1] 42 443
process[2] 45 473
process[1] 48 503
process[2] 51 511
process[1] 52 521

After Scheduling

Process Number || Brust Time || Wating Time || TurnAround Time
---
Processes[0] || 11 || 0 || 41
Processes[1] || 16 || 3 || 52
Processes[2] || 13 || 6 || 51
Processes[3] || 5 || 9 || 25
Processes[4] || 8 || 12 || 39

(base) dell@dell-Inspiron-15-3567:~/Downloads/Compiler/Assignment7$

SJF.java

```
import java.util.Scanner;
public class SJF {
int Count=0;
int[][] processData;
Scanner scan = new Scanner(System.in);
public static void main(String a[]){
System.out.println("\n\n**********Shortest Job First : (SJF) :**********\n");
SJF sjf = new SJF();
sjf.setValue();
System.out.println("\n\nBefore Scheduling\n");
sjf.DisplayValue();
sjf.scheduleSJF();
System.out.println("\n\nAfter Scheduling\n");
sjf.DisplayValue();
}
public void scheduleSJF(){
scheduleSorting();
scheduleCalculation();
}
public void scheduleSorting(){
for(int i=0;i<Count;i++){
int ele=i;
int eleData=processData[i][1];
for(int j=i+1;j<Count;j++){
if(eleData > processData[j][1]){
eleData= processData[j][1];ele=processData[j][0];
}
}
swapProcess(i,ele);
}
}
public void swapProcess(int p1,int p2){
int[] temp = processData[p1];
processData[p1] = processData[p2];
processData[p2] = temp;
}
public void scheduleCalculation(){
processData[0][2] = 0;
processData[0][3] = processData[0][1];
for(int i=1;i<Count;i++){
```

```
processData[i][2] = processData[i-1][3];
processData[i][3] = processData[i][1]+processData[i][2];
}
}
public void setValue(){
System.out.print("Enter Number Of processes : ");
Count = scan.nextInt();
processData = new int[Count][4];
for(int i=0;i<Count;i++){
System.out.println("Processes["+i+"] :-> ");
processData[i][0] = i;
System.out.print("\t\tBrust Time["+i+"] :-> ");processData[i][1] = scan.nextInt();
}
}
public void DisplayValue(){
System.out.println("-----------------------------------------------------------------------");
System.out.println("Process Number || Brust Time || Wating Time || TurnAround Time");
System.out.println("-----------------------------------------------------------------------");
for(int i=0;i<Count;i++){
System.out.println("Processes["+processData[i]
[0]+"]\t||\t"+processData[i][1]+"\t||\t"+processData[i][2]+"\t||\t"+processData[i][3]);
}
System.out.println("-----------------------------------------------------------------------");
}
}
```

o/p:

```
(base) dell@dell-Inspiron-15-3567:~/Downloads/Compiler/Assignment7$ javac SJF.java
(base) dell@dell-Inspiron-15-3567:~/Downloads/Compiler/Assignment7$ java SJF


**********Shortest Job First : (SJF) :**********

Enter Number Of processes : 5
Processes[0] :->
                Brust Time[0] :-> 15
Processes[1] :->
                Brust Time[1] :-> 6
Processes[2] :->
                Brust Time[2] :-> 9
Processes[3] :->
                Brust Time[3] :-> 18
Processes[4] :->
                Brust Time[4] :-> 20
```

Before Scheduling

Process Number	Brust Time	Wating Time	TurnAround Time
Processes[0]	15	0	0
Processes[1]	6	0	0
Processes[2]	9	0	0
Processes[3]	18	0	0
Processes[4]	20	0	0

After Scheduling

Process Number	Brust Time	Wating Time	TurnAround Time
Processes[1]	6	0	6
Processes[2]	9	6	15
Processes[0]	15	15	30
Processes[3]	18	30	48
Processes[4]	20	48	68