**Bahria University, Lahore Campus**

Department of Computer Sciences

Lab Journal 07

**(SPRING 2024)**

|  |  |  |
| --- | --- | --- |
| Course: | **Operating System Lab** | Date: 05/02/2024 |
| Course Code: | CSL - 320 | Max Marks: 20 |
| Faculty’s Name: | ABDULLAH |  |

Name:\_Affan Ahmad \_\_\_\_\_\_\_\_\_\_ Enroll No: \_03-134221-003\_\_\_\_\_

Objective(s) :

To write a C program to implement the CPU scheduling algorithm for FIRST COME FIRST SERVE. To write a C program to implement the CPU scheduling algorithm for Shortest job first.

## Lab Tasks :

**Task 1 :** Calculate the Average Time using FCFS Algorithm.

**Task 2:** Write the program for First Come First Serve scheduling algorithm.

**Task 3 :** Calculate the Average Time using SJF Algorithm.

**Task 4 :** Write the program for Shortest Job First scheduling algorithm.

**Lab Grading Sheet :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Max Marks** | **Obtained Marks** | **Comments(*if any*)** |
| 1. | 5 |  |  |
| 2. | 5 |  |  |
| 3. | 5 |  |  |
| 4. | 5 |  |  |
| **Total** | **20** |  | **Signature** |

**Note : Attempt all tasks and get them checked by your Lab. Instructor**.

# Lab 07: Scheduling

**Objective(s):**

* To write a C program to implement the CPU scheduling algorithm for FIRST COME FIRST SERVE.
* To write a C program to implement the CPU scheduling algorithm for Shortest job first.

**Tool(s) used:**

Ubuntu, VIM Editor

**First Come First Serve**

CPU scheduler will decide which process should be given to the CPU for its execution. For this it uses different algorithms to choose among the process. One among that algorithm is FCFS algorithm. In this algorithm, the process which arrive first is given to the CPU after finishing its request only it will allow CPU to execute other process.

Task 1 :Calculate the Average Time using FCFS Algorithm.

|  |  |  |  |
| --- | --- | --- | --- |
| **Process** | **Duration** | **Order** | **Arrival Time** |
| P1 | 24 | 1 | 0 |
| P2 | 3 | 2 | 0 |
| P3 | 4 | 3 | 0 |

**Gantchart:**

|  |  |  |
| --- | --- | --- |
| P1 | P2 | P3 |

## 

## P1 waiting time : 0

**P2 waiting Time : 24**

## P3 waiting Time : 27

Average waiting Time is :

## (0+24+27)/3=17

**Task 2 :** Write the program for First Come First Serve scheduling algorithm.

Step1: Create the number of process taken from user.

Step2: Get the ID and Service time for each process from user.

Step3: Initially, Waiting time of first process is zero and Total time for the first process is equal to its service time.

Step4: Calculate the Total time and processing (service) time for the remaining processes.

Step5: Waiting time of one process is the total service time of all the previous processes.

Step6: Total time of process is calculated by adding its Waiting time and its Service time.

Step7: Total waiting time of all processes is calculated by adding the waiting time for all process.

Step8: Total turnaround time is calculated by adding total time of all processes.

Step9: Now Calculate Average waiting time by dividing the total waiting time by total number of process.

Step10: And Calculate Average turnaround time by dividing the total time by the number of process.

Step11: Display the result in the following format for each process. printf("Id \t Service Time \t Waiting Time \t Total Time");

printf(“%d \t %d \t %d \t %d \t \n”, p[i].id, p[i].serviceTime, p[i].waitingTime, p[i].totalTime);

**Program**

## #include<iostream>

## using namespace std;

## 

## void findWaitingTime(int processes[], int n,

## int bt[], int wt[])

## {

## 

## wt[0] = 0;

## 

## 

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

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

## }

## 

## void findTurnAroundTime( int processes[], int n,

## int bt[], int wt[], int tat[])

## {

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

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

## }

## 

## void findavgTime( int processes[], int n, int bt[])

## {

## int wt[n], tat[n], total\_wt = 0, total\_tat = 0;

## 

## 

## findWaitingTime(processes, n, bt, wt);

## 

## 

## findTurnAroundTime(processes, n, bt, wt, tat);

## 

## 

## cout << "Processes "<< " Burst time "

## << " Waiting time " << " Turn around time\n";

## 

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

## {

## total\_wt = total\_wt + wt[i];

## total\_tat = total\_tat + tat[i];

## cout << " " << i+1 << "\t\t" << bt[i] <<"\t "

## << wt[i] <<"\t\t " << tat[i] <<endl;

## }

## 

## cout << "Average waiting time = "

## << (float)total\_wt / (float)n;

## cout << "\nAverage turn around time = "

## << (float)total\_tat / (float)n;

## }

## 

## int main()

## {

## 

## int processes[] = { 1, 2, 3};

## int n = sizeof processes / sizeof processes[0];

## 

## 

## int burst\_time[] = {10, 5, 8};

## 

## findavgTime(processes, n, burst\_time);

## return 0;

## }

**OUTPUT**

## 

**Shortest Job First**

CPU scheduler will decide which process should be given to the CPU for its execution. For this it uses different algorithms to choose among the processes. One among that algorithm is Shortest Job First. In this algorithm the process which has less service time given the CPU after finishing its request only it will allow CPU to execute next other process.

Task 3 : Calculate the Average Time using SJF Algorithm.

|  |  |  |  |
| --- | --- | --- | --- |
| **Process** | **Duration** | **Order** | **Arrival Time** |
| P1 | 6 | 1 | 0 |
| P2 | 8 | 2 | 0 |
| P3 | 7 | 3 | 0 |
| P4 | 3 | 4 | 0 |

## Sorted Process Is :

|  |  |  |  |
| --- | --- | --- | --- |
| **Process** | **Duration** | **Order** | **Arrival Time** |
| P4 | 3 | 4 | 0 |
| P1 | 6 | 1 | 0 |
| P3 | 7 | 3 | 0 |
| P2 | 8 | 2 | 0 |

## Process Order: P4 -> P1 -> P3 -> P2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | Duration | Order | Arrival Time | Completion Time | Waiting Time |
| P4 | 3 | 4 | 0 | 3 | 0 |
| P1 | 6 | 1 | 0 | 9 | 3 |
| P3 | 7 | 3 | 0 | 16 | 9 |
| P2 | 8 | 2 | 0 | 24 | 16 |

## Average Waiting Time = (0 + 3 + 9 + 16) / 4 = 28 / 4 = 7

**Task 4**  Write the program for Shortest Job First scheduling algorithm.

**Step 1 :** Get the number of process.

**Step 2 :** Get the id and service time for each process.

**Step 3 :** Initially the waiting time of first short process as 0 and total time of first short is process the service time of that process.

**Step 4 :** Calculate the total time and waiting time of remaining process.

**Step 5 :** Waiting time of one process is the total time of the previous process.

**Step 6 :** Total time of process is calculated by adding the waiting time and service time of each process.

**Step 7 :** Total waiting time calculated by adding the waiting time of each process.

**Step 8 :** Total turnaround time calculated by adding all total time of each process.

**Step 9 :** Calculate average waiting time by dividing the total waiting time by total number of process.

**Step 10 :** Calculate average turnaround time by dividing the total waiting time bytotal number of process.

## **Step 11 :** Display the result.

**Program**

#include <bits/stdc++.h>

using namespace std;

struct Process {

int pid;

int bt;

int art;

};

void findWaitingTime(Process proc[], int n,

int wt[])

{

int rt[n];

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

rt[i] = proc[i].bt;

int complete = 0, t = 0, minm = INT\_MAX;

int shortest = 0, finish\_time;

bool check = false;

while (complete != n) {

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

if ((proc[j].art <= t) &&

(rt[j] < minm) && rt[j] > 0) {

minm = rt[j];

shortest = j;

check = true;

}

}

if (check == false) {

t++;

continue;

}

rt[shortest]--;

minm = rt[shortest];

if (minm == 0)

minm = INT\_MAX;

if (rt[shortest] == 0) {

complete++;

check = false;

finish\_time = t + 1;

wt[shortest] = finish\_time -

proc[shortest].bt -

proc[shortest].art;

if (wt[shortest] < 0)

wt[shortest] = 0;

}

t++;

}

}

void findTurnAroundTime(Process proc[], int n,

int wt[], int tat[])

{

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

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

}

void findavgTime(Process proc[], int n)

{

int wt[n], tat[n], total\_wt = 0,

total\_tat = 0;

findWaitingTime(proc, n, wt);

findTurnAroundTime(proc, n, wt, tat);

cout << " P\t\t"

<< "BT\t\t"

<< "WT\t\t"

<< "TAT\t\t\n";

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

total\_wt = total\_wt + wt[i];

total\_tat = total\_tat + tat[i];

cout << " " << proc[i].pid << "\t\t"

<< proc[i].bt << "\t\t " << wt[i]

<< "\t\t " << tat[i] << endl;

}

cout << "\nAverage waiting time = "

<< (float)total\_wt / (float)n;

cout << "\nAverage turn around time = "

<< (float)total\_tat / (float)n;

}

int main()

{

Process proc[] = { { 1, 6, 2 }, { 2, 2, 5 },

{ 3, 8, 1 }, { 4, 3, 0}, {5, 4, 4} };

int n = sizeof(proc) / sizeof(proc[0]);

findavgTime(proc, n);

return 0;

}

**OUTPUT**

## 