**Bahria University, Lahore Campus**

Department of Computer Sciences

Lab Journal 9

**(Spring 2024)**

|  |  |  |
| --- | --- | --- |
| Course: | **Operating System Lab** | Date: 5/16/24 |
| 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 CPU scheduling algorithm for Round Robin.

## Lab Tasks :

### Task 01: Calculate the Average Time using Round Robin. Draw the GANTT Chart.

**Task 02:** Write the output for Round Robin Scheduling Algorithm.

**Lab Grading Sheet :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Max Marks** | **Obtained Marks** | **Comments(*if any*)** |
| 1. | 10 |  |  |
| 2. | 10 |  |  |
| **Total** | **20** |  | **Signature** |

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

# Lab 9: Round Robin Scheduling

**Objective(s):**

To write a C program to implement CPU scheduling algorithm for Round Robin.

**Tool(s) used:**

Ubuntu, VIM Editor

CPU scheduler will decide which process should be given the CPU for its execution. For this its use different algorithm to choose among the process. One algorithm among that is Round robin algorithm. In this algorithm we are assigning some time slice. The process is allocated according to the time slice, if the process service time is less than the time slice then process itself will release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue. If the CPU burst of the currently running process is longer than time quantum, the timer will go off and will cause an interrupt to the operating system. A context switch will be executed and the process will be put at the tail of the ready queue.

### Task 01: Calculate the Average Time using Round Robin. Draw the GANTT Chart. TQ=2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Process No | Arrival Time (AT) | Burst Time (BT) | Complete Time (CT) | Turnaround Time (CT-AT) | Waiting Time (TAT-BT) | Response Time |
| P1 | 0 | 10 | 22 | 22 | 12 | 0 |
| P2 | 1 | 4 | 12 | 11 | 7 | 2 |
| P3 | 2 | 5 | 18 | 16 | 11 | 4 |
| P4 | 3 | 3 | 17 | 14 | 11 | 6 |

Ready Queue:

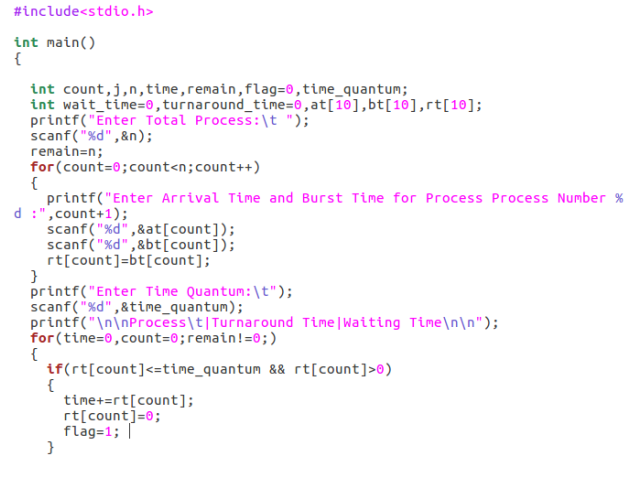
Gantt Chart:

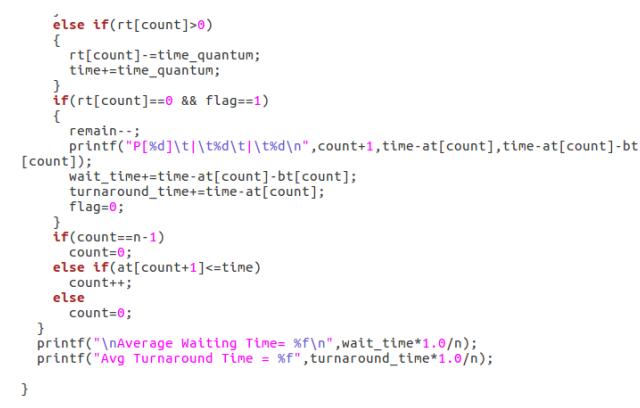
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P4 | P1 | P2 | P3 | P4 | P1 | P3 | P1 |  |

0 2 4 6 8 10 12 14 15 17 18 22

### Task 02: Write the output of Round Robin Scheduling Algorithm. Attach the screenshot of dry run of code.

**Code**





**#include <iostream>**

**#include <climits>**

**using namespace std;**

**struct Process {**

**int AT, BT, ST[20], WT, FT, TAT, pos;**

**};**

**int quant;**

**int main() {**

**int n, i, j;**

**cout << "Enter the no. of processes: ";**

**cin >> n;**

**Process p[n];**

**cout << "Enter the quantum: " << endl;**

**cin >> quant;**

**cout << "Enter the process numbers: " << endl;**

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

**cin >> p[i].pos;**

**cout << "Enter the Arrival time of processes: " << endl;**

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

**cin >> p[i].AT;**

**cout << "Enter the Burst time of processes: " << endl;**

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

**cin >> p[i].BT;**

**int c = n, s[n][20];**

**float time = 0, mini = INT\_MAX, b[n], a[n];**

**int index = -1;**

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

**b[i] = p[i].BT;**

**a[i] = p[i].AT;**

**for (j = 0; j < 20; j++) {**

**s[i][j] = -1;**

**}**

**}**

**int tot\_wt, tot\_tat;**

**tot\_wt = 0;**

**tot\_tat = 0;**

**bool flag = false;**

**while (c != 0) {**

**mini = INT\_MAX;**

**flag = false;**

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

**float p = time + 0.1;**

**if (a[i] <= p && mini > a[i] && b[i] > 0) {**

**index = i;**

**mini = a[i];**

**flag = true;**

**}**

**}**

**if (!flag) {**

**time++;**

**continue;**

**}**

**j = 0;**

**while (s[index][j] != -1) {**

**j++;**

**}**

**if (s[index][j] == -1) {**

**s[index][j] = time;**

**p[index].ST[j] = time;**

**}**

**if (b[index] <= quant) {**

**time += b[index];**

**b[index] = 0;**

**} else {**

**time += quant;**

**b[index] -= quant;**

**}**

**if (b[index] > 0) {**

**a[index] = time + 0.1;**

**}**

**if (b[index] == 0) {**

**c--;**

**p[index].FT = time;**

**p[index].WT = p[index].FT - p[index].AT - p[index].BT;**

**tot\_wt += p[index].WT;**

**p[index].TAT = p[index].BT + p[index].WT;**

**tot\_tat += p[index].TAT;**

**}**

**}**

**cout << "Process number ";**

**cout << "Arrival time ";**

**cout << "Burst time ";**

**cout << "\tStart time";**

**j = 0;**

**while (j != 10) {**

**j += 1;**

**cout << " ";**

**}**

**cout << "\t\tFinal time";**

**cout << "\tWait Time ";**

**cout << "\tTurnAround Time" << endl;**

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

**cout << p[i].pos << "\t\t";**

**cout << p[i].AT << "\t\t";**

**cout << p[i].BT << "\t";**

**j = 0;**

**int v = 0;**

**while (s[i][j] != -1) {**

**cout << p[i].ST[j] << " ";**

**j++;**

**v += 3;**

**}**

**while (v != 40) {**

**cout << " ";**

**v += 1;**

**}**

**cout << p[i].FT << "\t\t";**

**cout << p[i].WT << "\t\t";**

**cout << p[i].TAT << endl;**

**}**

**double avg\_wt, avg\_tat;**

**avg\_wt = tot\_wt / static\_cast<double>(n);**

**avg\_tat = tot\_tat / static\_cast<double>(n);**

**cout << "The average wait time is: " << avg\_wt << endl;**

**cout << "The average TurnAround time is: " << avg\_tat << endl;**

**return 0;**

**}**

**OUTPUT**

## 