**National University of Modern Languages**

Logo

Description automatically generated

**Lab Report#03**

**Roll # 2340**

**Class: BSCS 5B Morning**

**Subject: Operating System(Lab)**

**Submitted to: Mrs. Humaira Batool**

**Submitted by: Farhan Haider**

**Implement Pre-emptive SJF (Shortest Job First) CPU Scheduling Algorithm.**

// C++ program to implement Shortest Remaining Time First

// Shortest Remaining Time First (SRTF)

#include <bits/stdc++.h>

using namespace std;

struct Process {

int pid; // Process ID

int bt; // Burst Time

int art; // Arrival Time

};

// Function to find the waiting time for all

// processes

void **findWaitingTime**(Process proc[], int n,int wt[])

{

int rt[n];

// Copy the burst time into rt[]

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;

// Process until all processes gets

// completed

while (complete != n) {

// Find process with minimum

// remaining time among the

// processes that arrives till the

// current time`

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;

}

// Reduce remaining time by one

rt[shortest]--;

// Update minimum

minm = rt[shortest];

if (minm == 0)

minm = INT\_MAX;

// If a process gets completely

// executed

if (rt[shortest] == 0) {

// Increment complete

complete++;

check = false;

// Find finish time of current

// process

finish\_time = t + 1;

// Calculate waiting time

wt[shortest] = finish\_time -

proc[shortest].bt -

proc[shortest].art;

if (wt[shortest] < 0)

wt[shortest] = 0;

}

// Increment time

t++;

}

}

// Function to calculate turn around time

void **findTurnAroundTime**(Process proc[], int n,int wt[], int tat[])

{

// calculating turnaround 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

void findavgTime(Process proc[], int n)

{

int wt[n], tat[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 processes

findTurnAroundTime(proc, n, wt, tat);

// Display processes along with all

// details

cout << " P\t\t"

<<"Arrival Time \t\t"

<< "Burst Time\t\t"

<< "Wating Time\t\t"

<< "Turn Arround Time\t\t\n";

// Calculate total waiting time and

// total turnaround time

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

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

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

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

<<proc[i].art<<"\t\t\t"

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

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

}

cout << "\nAverage waiting time = "

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

cout << "\nAverage turn around time = "

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

}

// Driver code

int **main**()

{

Process proc[] = { { 1, 4, 3}, { 2, 2, 4 },

{ 3, 1, 5 }, { 4, 6, 2}, {5, 8, 1}, {6, 4, 2} };

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

findavgTime(proc, n);

return 0;

}

**OUTPUT:**

A picture containing graphical user interface

Description automatically generated