OS LAB 2

NAME: Aditya Anand

ROLL NO.: 20124009

BRANCH: IT

|  |  |  |  |
| --- | --- | --- | --- |
| S No. | Title | Date of Implementation | Remarks |
| 1 | Program to implement First Come First Serve Process of CPU Scheduling | 24-01-2022 |  |

FIRST COME FIRST SERVE CPU SCHEDULING

CRITERIA: Arrival Time

MODE: Non pre-emptive

GIVEN: List of processes with their arrival and burst time.

CODE:

#include<bits/stdc++.h>

using namespace :: std;

class process{

    public:

        int id;

        int arrivalTime;

        int burstTime;

        int completionTime;

        int TAT;

        int WT;

        int RT;

};

void FCFS(vector<process> &v){

    int cur\_time = 0;

    int id = 0;

    for(int i=0; i<v.size(); i++){

        if(cur\_time<v[i].arrivalTime){

            cout<<"CPU idle from "<<cur\_time<<" to "<<v[i].arrivalTime<<endl;

            cur\_time = v[i].arrivalTime;

        }

        v[i].completionTime = cur\_time+v[i].burstTime;

        v[i].RT = cur\_time-v[i].arrivalTime;

        cout<<"Process P"<<v[i].id+1<<": start time = "<<cur\_time<<" completion time = "<<v[i].completionTime<<endl;

        cur\_time+=v[i].burstTime;

    }

}

int main(){

    cout<<"FIRST COME FIRST SERVE CPU SCHEDULING ALGORITHM C++ IMPLEMENTATION\n";

    cout<<"Name: Aditya Anand\tRoll No.:20124009\t Branch: IT\n\n\n";

    int n=0;

    cout<<"Enter the number of processes: ";

    cin>>n;

    cout<<"Enter the arrival times and burst times of "<<n<<" processes: \n";

    vector<process> v(n);

    for(int i=0; i<n; i++){

        cin>>v[i].arrivalTime>>v[i].burstTime;

        v[i].id = i;

    }

    cout<<"-------------------------------------------------------------------------------\n";

    cout<<"\n";

    FCFS(v);

    cout<<"\n";

    cout<<"-------------------------------------------------------------------------------\n";

    cout<<"\n\n";

    int t\_TAT=0;

    int t\_CT=0;

    for(int i=0; i<n; i++){

        v[i].TAT = v[i].completionTime-v[i].arrivalTime;

        v[i].WT = v[i].TAT-v[i].burstTime;

        t\_TAT+=v[i].TAT;

        t\_CT+=v[i].completionTime;

    }

    for(auto p:v){

        cout<<"Process: "<<p.id<<"\tArrival Time:"<<p.arrivalTime<<"\tBurst Time:"<<p.burstTime<<"\tCompletion Time:"<<p.completionTime;

        cout<<"\tTurn Around Time:"<<p.TAT<<"\tWaiting Time:"<<p.WT<<"\tResponse Time:"<<p.RT<<"\n";

    }

    cout<<"\nAverage Turn Around Time: "<<(float)((1.0\*t\_TAT)/(1.0\*n))<<"\n";

    cout<<"\nAverage Completion Time: "<<(float)((1.0\*t\_CT)/(1.0\*n))<<"\n";

    return 0;

}

RESULT:

