Write program to implement SJF scheduling algorithm

**CODE**

*//SJF scheduling algorithm*

#include<iostream>

using *namespace* std;

*int* main()

{

*int* n;

    cout<<"Please enter the number of processes: ";

    cin>>n;

*int* burst[n],process[n];

    for(*int* i=0; i<n; i++)

    {

        cout<<"Please enter the CPU Burst Time for process P"<<i+1<<" :";

        cin>>burst[i];

        process[i]=i+1;

    }

*int* j,k;

*int* temp1,temp2;

    for(j=1; j<n; j++) *//sorting burst time and swapping elements of process[] along with burst[]*

    {

        temp1=burst[j];

        temp2=process[j];

        for(k=j; k>0 && temp1<burst[k-1]; k--)

        {

            burst[k]=burst[k-1];

            process[k]=process[k-1];

        }

        burst[k]=temp1;

        process[k]=temp2;

    }

*int* wait\_time[n],turnaround\_time[n];

    wait\_time[0]=0;

    for(*int* i=1; i<n; i++)

        wait\_time[i]=wait\_time[i-1]+burst[i-1]; *//calculating wait time*

    for(*int* i=0; i<n; i++)

        turnaround\_time[i]=wait\_time[i]+burst[i]; *//calculating turnaround time*

*float* avg\_wt=0,avg\_tt=0;

    cout<<"Processes \t  Burst Time \t   Waiting Time  \t  Turnaround Time"<<endl; *//printing wait time & turnaround time*

    for(*int* i=0; i<n; i++)

    {

        cout<<"P"<<process[i]<<"  \t\t   "<<burst[i]<<"   \t\t      "<<wait\_time[i]<<"   \t\t\t   "<<turnaround\_time[i]<<endl;

        avg\_wt+=wait\_time[i]; *//calculating total sum of wait time*

        avg\_tt+=turnaround\_time[i]; *//calculating total sum of turnaround time*

    }

    avg\_wt=avg\_wt/n; *//calculating avg wait time*

    avg\_tt=avg\_tt/n; *//calculating avg turnaround time*

    cout<<"\nAverage Waiting Time = "<<avg\_wt<<endl;

    cout<<"\nAverage Turnaround Time = "<<avg\_tt<<endl;

    return 0;

}

**OUTPUT**

**A picture containing graphical user interface

Description automatically generated**

Write program to implement SRTF scheduling algorithm

**CODE**

#include <iostream>

using *namespace* std;

*void* waiting\_time(*struct* process *a*[], *int* *n*);

*struct* process

{

*int* process\_id;

*int* burst\_time;

*int* waiting\_time;

*int* arrival\_time;

*int* remain\_time;

} arr[100];

*int* process\_finish[100];

*int* main()

{

    arr[99].remain\_time = 9999;

*int* n; *//No of process in variable n*

    cout << "\nPlease enter the number of Processes : ";

    cin >> n;

    cout << endl;

    for (*int* i = 0; i < n; i++) *//Take the Burst time for each process by using loop*

    {

        arr[i].process\_id = i + 1; *//increment the process\_id by 1 after each burst\_time*

        cout << "Please enter the CPU Burst Time of P" << i + 1 << " : ";

        cin >> arr[i].burst\_time;

        arr[i].remain\_time = arr[i].burst\_time; *//copy each process burst\_time to another array remain\_time[]*

        cout << "Please enter the Arrival Time : ";

        cin >> arr[i].arrival\_time;

        cout << endl;

    }

    waiting\_time(arr, n);

    return 0;

}

*void* waiting\_time(*struct* process *a*[], *int* *n*)

{

*int* remain = 0, sum\_wait = 0, sum\_turnaround = 0, endTime, smallest;

    cout << "\n\nProcess  Turnaround Time  Waiting Time\n\n";

*int* process\_f = 0; *// handle the INDEX of array process\_finish.*

    for (*int* time = 0; remain != *n*; time++)

    {

        smallest = 99;

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

        {

            if ((*a*[i].arrival\_time <= time) && (*a*[i].remain\_time < *a*[smallest].remain\_time) && (*a*[i].remain\_time > 0))

            {

                smallest = i;

            }

        }

*a*[smallest].remain\_time--;

        if (*a*[smallest].remain\_time == 0)

        {

            process\_finish[process\_f] = smallest + 1; *//to assign a process # which finish the total job*

            process\_f++;

*a*[smallest].process\_id = smallest + 1; *//to ssign a process\_id*

*int* tt;

            remain++; *//One process complete the total job*

            endTime = time + 1; *//Total competional time of process*

            tt = endTime - *a*[smallest].arrival\_time; *//Calculate the TURNaround TIME (competionalTime - TT )*

*a*[smallest].waiting\_time = tt - *a*[smallest].burst\_time; *//Calculate the Waiting Time*

            cout << "\nP[" << smallest + 1 << "]\t\t" << tt << "\t\t" << *a*[smallest].waiting\_time;

            sum\_wait += tt - *a*[smallest].burst\_time; *//For find Average Waiting Time*

        }

    }

    cout << "\n\nAverage Waiting Time = " << sum\_wait \* 1.0 / *n*;

}

**OUTPUT**

**Text

Description automatically generated**