OPERATING SYSTEM - Mock Exam

Nishant Kumar Giri

AC-1254

CODE

#include <iostream>

using *namespace* std;

*void* NonPreemptivePriority() {

*int* n, temp1, temp2, temp3;

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

    cin >> n; *// taking number of process*

*/\* storing burst time, process number and its priority in array \*/*

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

    for (*int* i = 0; i < n; i++) {

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

        cin >> burst[i]; *// taking burst time for each process*

        process[i] = i + 1;

    }

    cout << endl;

    for (*int* i = 0; i < n; i++) {

        cout << "Please enter priority of P" << i + 1 << ": ";

        cin >> priority[i]; *// taking priority of each process*

    }

*int* x, y;

    for (x = 1; x < n; x++) *// sorting array on the basis of priority array*

    {

        temp1 = priority[x];

        temp2 = process[x];

        temp3 = burst[x];

        for (y = x; y > 0 && temp1 < priority[y - 1]; y--) {

*// swapping elements of process[] along with priority*

            priority[y] = priority[y - 1];

            process[y] = process[y - 1];

*// swapping elements of burst[] along with priority*

            burst[y] = burst[y - 1];

        }

        priority[y] = temp1;

        process[y] = temp2;

        burst[y] = temp3;

    }

*int* WaitingTime[n], TurnaroundTime[n];

*float* AvgWaitingTime = 0, AvgTurnAroundTime = 0;

    WaitingTime[0] = 0;

    for (*int* i = 1; i < n; i++)

        WaitingTime[i] =

            WaitingTime[i - 1] + burst[i - 1]; *// calculating wait time*

    for (*int* i = 0; i < n; i++)

        TurnaroundTime[i] =

            WaitingTime[i] + burst[i]; *// calculating turnaround time*

    cout << "   Burst Time    Waiting Time    Turnaround Time"

         << endl; *// printing wait time & turnaround time*

    for (*int* i = 0; i < n; i++) {

        cout << "P" << process[i] << "   " << burst[i] << "     "

             << WaitingTime[i] << "     " << TurnaroundTime[i] << endl;

        AvgWaitingTime += WaitingTime[i]; *// calculating total sum of wait time*

        AvgTurnAroundTime +=

            TurnaroundTime[i]; *// calculating total sum of turnaround time*

    }

    AvgWaitingTime = AvgWaitingTime / n; *// calculating average wait time*

    AvgTurnAroundTime =

        AvgTurnAroundTime / n; *// calculating average turnaround time*

    cout << "\nAverage Waiting Time: " << AvgWaitingTime << endl;

    cout << "\nAverage Turnaround Time: " << AvgTurnAroundTime << endl;

}

*void* PreemptivePriority() {

*int* n;

    cout << "Please enter number of processes: "; *// taking number of process*

    cin >> n;

*/\* storing process, priortity, burst time and arrival time in array \*/*

*int* burst[20], arrival[20], process[20], priority[20];

    for (*int* i = 0; i < n; i++) {

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

             << ": "; *// taking burst time*

        cin >> burst[i];

        cout << "Please enter Arrival Time for process P" << i + 1

             << ": "; *// taking arrival time*

        cin >> arrival[i];

        cout << "Please enter Priority for process P" << i + 1

             << ": "; *// taking priority*

        cin >> priority[i];

        process[i] = i + 1; *// array to store process number*

        cout << endl;

    }

*int* temp\_burst[n];

    for (*int* i = 0; i < n; i++) *// copying burst array to a temp array*

        temp\_burst[i] = burst[i];

*int* count = 0; *// count stores number of processes completed*

*int* min, end; *// points process with minimum burst time*

*int* wait[n], turn[n], ct[n];

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

*int* i;

        min = 9;

        for (i = 0; i < n; i++) {

            if (arrival[i] <= time && priority[i] < priority[min] &&

                burst[i] > 0)

*/\* if process with priority greater than priority[min] found then*

*updating value of min with it \*/*

            {

                min = i;

            }

        }

        burst[min]--; *// decrementing burst[min] by 1 since it completed 1sec*

*// from its burst time*

        if (burst[min] == 0) *// when a process get executed completely*

        {

            count++;

            end = time + 1;

            ct[min] = end;

            wait[min] = end - arrival[min] -

                        temp\_burst[min]; *// calculating waiting time*

            turn[min] = end - arrival[min]; *// calculating turnaround time*

        }

    }

    cout << "\nProcess\tPriority\tArrival Time\tBurst Time\tCompletion "

            "Time\t\tWaiting Time\t\tTurnaround Time\n"; *// displaying details*

    for (*int* i = 0; i < n; i++) {

        cout << process[i] << "\t" << priority[i] << "\t\t" << arrival[i]

             << "\t\t" << temp\_burst[i] << "\t\t" << ct[i] << "\t\t\t"

             << wait[i] << "\t\t\t" << turn[i];

        cout << endl;

    }

*float* AvgWaitingTime = 0;

    for (*int* i = 0; i < n; i++) *// calculating avg wait time*

    {

        AvgWaitingTime += wait[i];

    }

*float* AvgTurnAround=0;

    for(*int* i =0;i<n;i++){

        AvgTurnAround += turn[i];

    }

    AvgWaitingTime /= n;

    AvgTurnAround /= n;

    cout<<"------------------------------------------------------"<<endl;

    cout << "\n\nAverage Waiting Time: " << AvgWaitingTime << endl;

    cout<<"------------------------------------------------------"<<endl;

    cout<<"\nAverage Turn Around Time:  "<<AvgTurnAround<<endl;

    cout<<"------------------------------------------------------"<<endl;

}

*int* main() {

*int* choice;

*char* answer;

    do {

        cout<<"------------------------------------------------------"<<endl;

        cout << "MENU:" << endl;

        cout<<"------------------------------------------------------"<<endl;

        cout << "1. Non Preemptive Priority" << endl;

        cout << "2. Preemptive Priority" << endl;

        cout<<"------------------------------------------------------"<<endl;

        cin >> choice;

        switch (choice) {

            case 1:

                NonPreemptivePriority();

                break;

            case 2:

                PreemptivePriority();

                break;

            default:

                cout << "ERROR!! Wrong option" << endl;

                break;

        }

        cout << "Do you want to continue (Y/N)?: ";

        cin >> answer;

    } while (answer == 'y' || answer == 'Y');

    return 0;

}

OUTPUT

Non-Preemptive

Text

Description automatically generated

Premptive

Text

Description automatically generated