|  |  |  |  |
| --- | --- | --- | --- |
| **Fr. Conceicao Rodrigues College of Engineering**  **Department of Computer Engineering** | | | |
| **Student’s Roll No** |  | **Students Name** |  |
| **Date of Performance** |  | **SE Computer – Div** | **A / B** |

**Aim:** Study Process Scheduling

**Lab Outcome:**

**CSL403.2:** Implement various Process scheduling algorithm and evaluate their performance.

**Problem Statements:**

Batch (A): First Come First Serve (FCFS) ,Non Preemptive Shortest Job First (SJF)

Batch (B): Non Preemptive Shortest Job First (SJF) ,Shortest Remaining Time First (SRTF)

Batch (C ): Round Robin Algorithm (RR), Non Preemptive Priority (NPP)

Batch (D): Non Preemptive Priority (NPP), Premptive Priority (PP)

1. Calculate WT, AWT, TAT, ATAT.

2. Compare the result of algorithms for a problem and find which algorithm is performing better.

**References:**

[**https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/?ref=lbp**](https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/?ref=lbp)

|  |  |  |  |
| --- | --- | --- | --- |
| **On time Submission(2)** | **Knowledge of Topic(4)** | **Implementation and Demonstraion(4)** | **Total (10)** |
|  |  |  |  |
| **Signature of Faculty** |  | **Date of Submission** |  |

#include <stdio.h>

int main()

{

    //Input

    int process\_table[100][4];

    int i, j, n, total = 0, index, temp;

    float avg\_wt, avg\_tat;

    printf("Enter number of process: ");

    scanf("%d", &n);

    printf("Enter Burst Time:\n");

    for (i = 0; i < n; i++) {

        printf("P%d: ", i + 1);

        scanf("%d", &process\_table[i][1]);

        process\_table[i][0] = i + 1;

    }

    //Sorting

    for (i = 0; i < n; i++) {

        index = i;

        for (j = i + 1; j < n; j++)

            if (process\_table[j][1] < process\_table[index][1])

                index = j;

        temp = process\_table[i][1];

        process\_table[i][1] = process\_table[index][1];

        process\_table[index][1] = temp;

        temp = process\_table[i][0];

        process\_table[i][0] = process\_table[index][0];

        process\_table[index][0] = temp;

    }

    //Waiting Time Calculation

    process\_table[0][2] = 0;

    for (i = 1; i < n; i++)

    {

        process\_table[i][2] = 0;

        for (j = 0; j < i; j++)

            process\_table[i][2] += process\_table[j][1];

        total += process\_table[i][2];

    }

    avg\_wt = (float)total / n;

    total = 0;

    printf("P    BT  WT  TAT\n");

    //Turnaround time

    for (i = 0; i < n; i++) {

        process\_table[i][3] = process\_table[i][1] + process\_table[i][2];

        total += process\_table[i][3];

        //printing the output

        printf("P%d  %d  %d  %d\n", process\_table[i][0],

            process\_table[i][1], process\_table[i][2], process\_table[i][3]);

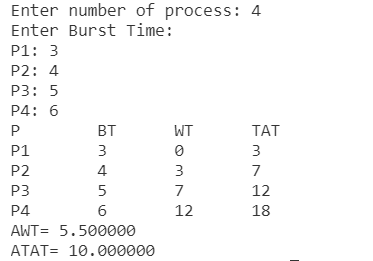
    }

    avg\_tat = (float)total / n;

    printf("AWT= %f", avg\_wt);

    printf("\nATAT= %f\n", avg\_tat);

}



#include <stdio.h>

int main()

{

    int n, ari[10], bur[10], total = 0, i, j, small, temp, procs[100], k, waiting[10], finish[10];

    float tavg = 0.0, wavg = 0.0;

    printf("ENTER THE NUMBER OF PROCESSES:");

    scanf("%d", &n);

    for (i = 0; i < n; i++)

    {

        printf("ENTER THE ARRIVAL TIME OF PROCESS %d:\t", i);

        scanf("%d", &ari[i]);

        printf("ENTER THE BURST TIME OF PROCESS %d:\t", i);

        scanf("%d", &bur[i]);

        waiting[i] = 0;

        total += bur[i];

    }

    for (i = 0; i < n; i++)

    {

        for (j = i + 1; j < n; j++)

        {

            if (ari[i] > ari[j])

            {

                temp = ari[i];

                ari[i] = ari[j];

                ari[j] = temp;

                temp = bur[i];

                bur[i] = bur[j];

                bur[j] = temp;

            }

        }

    }

    for (i = 0; i < total; i++)

    {

        small = 3200;

        for (j = 0; j < n; j++)

        {

            if ((bur[j] != 0) && (ari[j] <= i) && (bur[j] < small))

            {

                small = bur[j];

                 k = j;

            }

        }

        bur[k]--;

        procs[i] = k;

    }

    k = 0;

    for (i = 0; i < total; i++)

    {

        for (j = 0; j < n; j++)

        {

            if (procs[i] == j)

            {

                finish[j] = i;

                waiting[j]++;

            }

        }

    }

    for (i = 0; i < n; i++)

    {

        printf("\n PROCESS %d:-FINISH TIME==> %d TURNAROUND TIME==>%d WAITING TIME==>%d\n", i + 1, finish[i] + 1, (finish[i] - ari[i]) + 1, (((finish[i] + 1) - waiting[i]) - ari[i]));

        wavg = wavg + (((finish[i] + 1) - waiting[i]) - ari[i]);

        tavg = tavg + ((finish[i] - ari[i]) + 1);

    }

    printf("\n WAvG==>\t%f\n TAVG==>\t%f\n", (wavg / n), (tavg / n));

    return 0;

}

