4ITRC2

Operating System Lab

Lab Assignment 5

Aim: To create C programs for the different scheduling algorithms.

To perform: Create and execute C programs for following CPU Scheduling Algorithms:

1. First Come First Serve (FCFS)

2. Shortest Job First (SJF)

3. Round Robin Scheduling

To Submit: C Codes for the above scheduling algorithms with their outputs.

SUBMITTED TO: SUBMITTED BY:

Prof. Jasneet Kaur Bhaskar Sharma 23I4123

1. **First Come First Serve (FCFS)**

#include <stdio.h>

int main() {

int n, i;

int bt[20], wt[20], tat[20];

float avg\_wt = 0, avg\_tat = 0;

printf("Enter total number of processes: ");

scanf("%d", &n);

printf("Enter Burst Time for each process:\n");

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

printf("P[%d]: ", i+1);

scanf("%d", &bt[i]);

}

wt[0] = 0;

for (i = 1; i < n; i++) {

wt[i] = wt[i-1] + bt[i-1];

}

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

tat[i] = bt[i] + wt[i];

avg\_wt += wt[i];

avg\_tat += tat[i];

}

printf("\nProcess\tBT\tWT\tTAT\n");

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

printf("P[%d]\t%d\t%d\t%d\n", i+1, bt[i], wt[i], tat[i]);

}

printf("\nAverage Waiting Time = %.2f", avg\_wt / n);

printf("\nAverage Turnaround Time = %.2f\n", avg\_tat / n);

return 0;

}

OUTPUT:-

Enter total number of processes: 3

Enter Burst Time for each process:

P[1]: 5

P[2]: 9

P[3]: 6

Process BT WT TAT

P[1] 5 0 5

P[2] 9 5 14

P[3] 6 14 20

Average Waiting Time = 6.33

Average Turnaround Time = 13.00

1. **Shortest Job First (Non-Preemptive)**

#include <stdio.h>

int main() {

int n, i, j, pos, temp;

int bt[20], p[20], wt[20], tat[20];

float avg\_wt = 0, avg\_tat = 0;

printf("Enter number of processes: ");

scanf("%d", &n);

printf("Enter Burst Time:\n");

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

printf("P[%d]: ", i+1);

scanf("%d", &bt[i]);

p[i] = i+1; // process number

}

// Sorting by burst time

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

pos = i;

for (j = i+1; j < n; j++) {

if (bt[j] < bt[pos]) pos = j;

}

// Swap

temp = bt[i];

bt[i] = bt[pos];

bt[pos] = temp;

temp = p[i];

p[i] = p[pos];

p[pos] = temp;

}

wt[0] = 0;

for (i = 1; i < n; i++) {

wt[i] = 0;

for (j = 0; j < i; j++)

wt[i] += bt[j];

avg\_wt += wt[i];

}

printf("\nProcess\tBT\tWT\tTAT\n");

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

tat[i] = bt[i] + wt[i];

avg\_tat += tat[i];

printf("P[%d]\t%d\t%d\t%d\n", p[i], bt[i], wt[i], tat[i]);

}

printf("\nAverage Waiting Time = %.2f", avg\_wt / n);

printf("\nAverage Turnaround Time = %.2f\n", avg\_tat / n);

return 0;

}

OUTPUT:-

Enter number of processes: 3

Enter Burst Time:

P[1]: 6

P[2]: 8

P[3]: 2

Process BT WT TAT

P[3] 2 0 2

P[1] 6 2 8

P[2] 8 8 16

Average Waiting Time = 3.33

Average Turnaround Time = 8.67

1. **Round Robin Scheduling**

#include <stdio.h>

int main() {

int i, n, time = 0, tq, remain, flag = 0;

int bt[10], rt[10], wt[10], tat[10];

float avg\_wt = 0, avg\_tat = 0;

printf("Enter number of processes: ");

scanf("%d", &n);

remain = n;

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

printf("Enter Burst Time for P[%d]: ", i+1);

scanf("%d", &bt[i]);

rt[i] = bt[i];

}

printf("Enter Time Quantum: ");

scanf("%d", &tq);

while (remain != 0) {

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

if (rt[i] > 0) {

if (rt[i] > tq) {

time += tq;

rt[i] -= tq;

} else {

time += rt[i];

wt[i] = time - bt[i];

rt[i] = 0;

tat[i] = time;

remain--;

}

}

}

}

printf("\nProcess\tBT\tWT\tTAT\n");

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

avg\_wt += wt[i];

avg\_tat += tat[i];

printf("P[%d]\t%d\t%d\t%d\n", i+1, bt[i], wt[i], tat[i]);

}

printf("\nAverage Waiting Time = %.2f", avg\_wt / n);

printf("\nAverage Turnaround Time = %.2f\n", avg\_tat / n);

return 0;

}

OUTPUT:-

Enter number of processes: 3

Enter Burst Time for P[1]: 10

Enter Burst Time for P[2]: 5

Enter Burst Time for P[3]: 8

Enter Time Quantum: 3

Process BT WT TAT

P[1] 10 13 23

P[2] 5 9 14

P[3] 8 14 22

Average Waiting Time = 12.00

Average Turnaround Time = 19.67