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)

#include <stdio.h>

int main() {

int n, i;

printf("Enter number of processes: ");

scanf("%d", &n);

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

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

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] = wt[i] + bt[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;

}

2. Shortest Job First (SJF)

#include <stdio.h>

int main() {

int n, i, j;

printf("Enter number of processes: ");

scanf("%d", &n);

int bt[n], p[n], wt[n], tat[n], temp;

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

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

p[i] = i+1;

printf("Enter burst time of P%d: ", i+1);

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

}

// Sorting burst time (SJF logic)

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

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

if(bt[i] > bt[j]) {

temp = bt[i]; bt[i] = bt[j]; bt[j] = temp;

temp = p[i]; p[i] = p[j]; p[j] = temp;

}

}

}

wt[0] = 0;

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

wt[i] = 0;

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

wt[i] += bt[j];

}

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", 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;

}

3. Round Robin Scheduling

#include <stdio.h>

int main() {

int i, n, time, remain, temps = 0, time\_quantum;

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

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

printf("Enter total 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", &time\_quantum);

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

for(time = 0, i = 0; remain != 0;) {

if(rt[i] > 0 && rt[i] <= time\_quantum) {

time += rt[i];

rt[i] = 0;

temps = 1;

} else if(rt[i] > 0) {

rt[i] -= time\_quantum;

time += time\_quantum;

}

if(rt[i] == 0 && temps == 1) {

remain--;

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

tat[i] = time;

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

avg\_wt += wt[i];

avg\_tat += tat[i];

temps = 0;

}

i = (i+1) % n;

}

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

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

return 0;

}