4ITRC2 Operating System Lab

Kanchan Padme

24I4192

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.

#include <stdio.h>

// 1. First Come First Serve (FCFS) Scheduling Algorithm

void fcfs(int processes[], int n, int burst\_time[]) {

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\_time[i - 1];

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

turnaround\_time[i] = wait\_time[i] + burst\_time[i];

printf("\nFCFS Scheduling:\nProcess \t Burst Time \t Waiting Time \t Turnaround Time\n");

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

printf("P%d \t %d \t\t %d \t\t %d\n", processes[i], burst\_time[i], wait\_time[i], turnaround\_time[i]);

}

// 2. Shortest Job First (SJF) Scheduling Algorithm

Kanchan Padme

24I4192

void sjf(int processes[], int n, int burst\_time[]) {

int temp, i, j, wait\_time[n], turnaround\_time[n], pos;

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

pos = i;

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

if (burst\_time[j] < burst\_time[pos])

pos = j;

temp = burst\_time[i];

burst\_time[i] = burst\_time[pos];

burst\_time[pos] = temp;

temp = processes[i];

processes[i] = processes[pos];

processes[pos] = temp;

}

wait\_time[0] = 0;

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

wait\_time[i] = wait\_time[i - 1] + burst\_time[i - 1];

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

turnaround\_time[i] = wait\_time[i] + burst\_time[i];

printf("\nSJF Scheduling:\nProcess \t Burst Time \t Waiting Time \t Turnaround Time\n");

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

printf("P%d \t %d \t\t %d \t\t %d\n", processes[i], burst\_time[i], wait\_time[i], turnaround\_time[i]);

}

// 3. Round Robin Scheduling Algorithm

void round\_robin(int processes[], int n, int burst\_time[], int quantum) {

Kanchan Padme

24I4192

int remaining\_time[n], wait\_time[n], turnaround\_time[n], t = 0;

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

remaining\_time[i] = burst\_time[i];

int done;

do {

done = 1;

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

if (remaining\_time[i] > 0) {

done = 0;

if (remaining\_time[i] > quantum) {

t += quantum;

remaining\_time[i] -= quantum;

} else {

t += remaining\_time[i];

wait\_time[i] = t - burst\_time[i];

remaining\_time[i] = 0;

}

}

}

} while (!done);

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

turnaround\_time[i] = burst\_time[i] + wait\_time[i];

printf("\nRound Robin Scheduling:\nProcess \t Burst Time \t Waiting Time \t Turnaround Time\n");

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

printf("P%d \t %d \t\t %d \t\t %d\n", processes[i], burst\_time[i], wait\_time[i], turnaround\_time[i]);

}

Kanchan Padme

24I4192

int main() {

int n;

printf("Enter number of processes: ");

scanf("%d", &n);

int processes[n], burst\_time[n];

printf("Enter burst times:\n");

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

processes[i] = i + 1;

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

scanf("%d", &burst\_time[i]);

}

fcfs(processes, n, burst\_time);

sjf(processes, n, burst\_time);

int quantum;

printf("Enter time quantum for Round Robin: ");

scanf("%d", &quantum);

round\_robin(processes, n, burst\_time, quantum);

return 0;

}