Lab2 -sjf

#include <stdio.h>

#include <limits.h>

struct Process {

int pid;

int arrival\_time;

int burst\_time;

int remaining\_time;

int waiting\_time;

int turnaround\_time;

int completed;

};

void sortByBurstTime(struct Process p[], int n) {

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (p[j].burst\_time > p[j + 1].burst\_time) {

struct Process temp = p[j];

p[j] = p[j + 1];

p[j + 1] = temp;

}

}

}

}

void sjf\_non\_preemptive(struct Process p[], int n) {

sortByBurstTime(p, n);

p[0].waiting\_time = 0;

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

p[i].waiting\_time = p[i - 1].waiting\_time + p[i - 1].burst\_time;

}

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

p[i].turnaround\_time = p[i].waiting\_time + p[i].burst\_time;

}

}

void sjf\_preemptive(struct Process p[], int n) {

int completed = 0, time = 0, min\_index;

while (completed != n) {

min\_index = -1;

int min\_time = INT\_MAX;

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

if (p[i].arrival\_time <= time && p[i].remaining\_time > 0 && p[i].remaining\_time < min\_time) {

min\_time = p[i].remaining\_time;

min\_index = i;

}

}

if (min\_index == -1) {

time++;

continue;

}

p[min\_index].remaining\_time--;

time++;

if (p[min\_index].remaining\_time == 0) {

p[min\_index].completed = 1;

completed++;

p[min\_index].turnaround\_time = time - p[min\_index].arrival\_time;

p[min\_index].waiting\_time = p[min\_index].turnaround\_time - p[min\_index].burst\_time;

}

}

}

void displayProcesses(struct Process p[], int n) {

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

printf("\nProcess\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\n");

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

printf("P%d\t%d\t\t%d\t\t%d\t\t%d\n", p[i].pid, p[i].arrival\_time, p[i].burst\_time, p[i].waiting\_time, p[i].turnaround\_time);

total\_wt += p[i].waiting\_time;

total\_tat += p[i].turnaround\_time;

}

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

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

}

int main() {

int n, choice;

printf("Enter number of processes: ");

scanf("%d", &n);

struct Process p[n];

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

p[i].pid = i + 1;

printf("Enter Arrival Time and Burst Time for Process %d: ", i + 1);

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

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

p[i].completed = 0;

}

printf("Choose Scheduling Algorithm:\n1. Non-Preemptive SJF\n2. Preemptive SJF (SRTF)\nEnter choice: ");

scanf("%d", &choice);

if (choice == 1) {

sjf\_non\_preemptive(p, n);

} else if (choice == 2) {

sjf\_preemptive(p, n);

} else {

printf("Invalid choice!\n");

return 1;

}

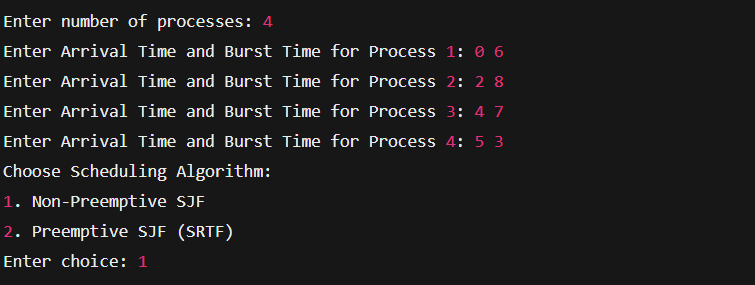
displayProcesses(p, n);

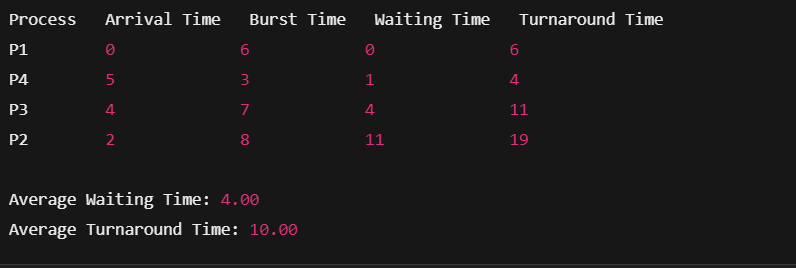
return 0;

}

Output:

Non preemtive:





Preemptive:

