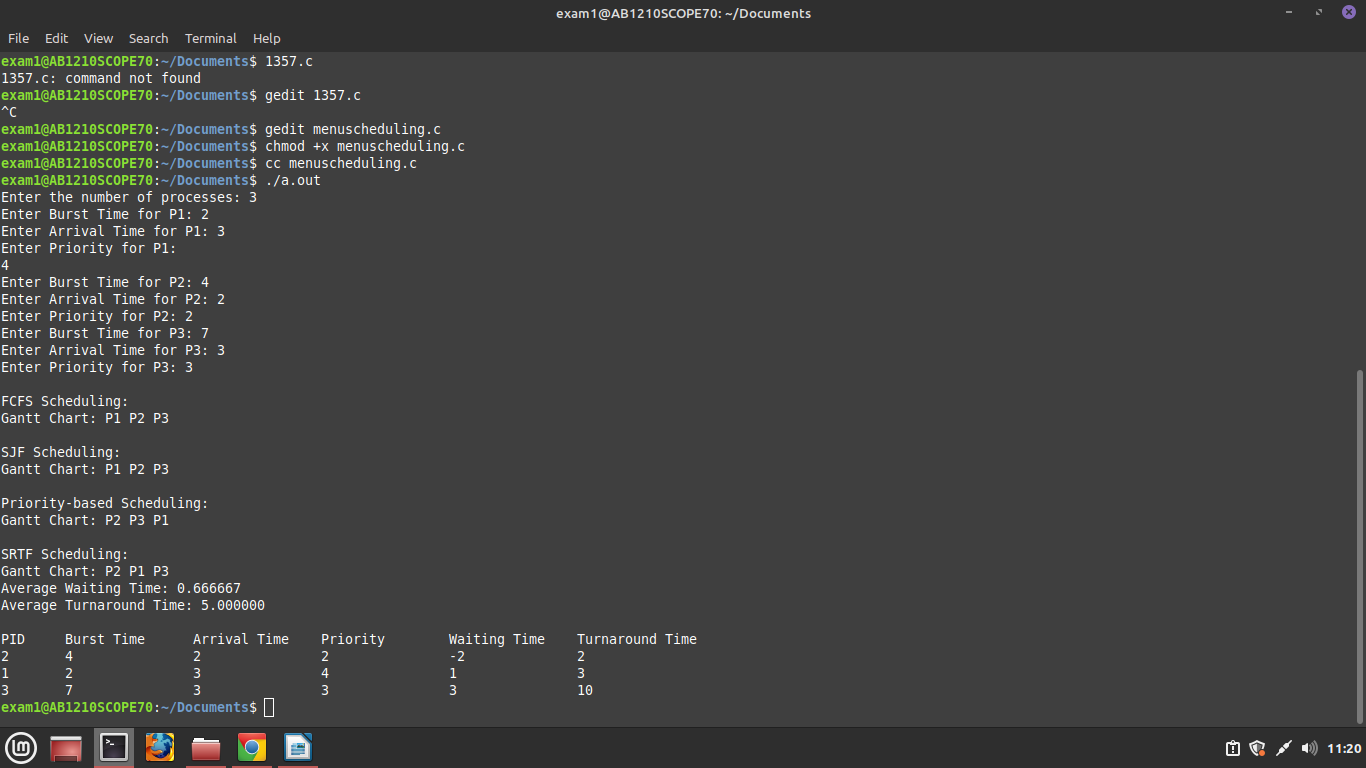
22BCE1357

Anantabh Sahu

#include <stdio.h>



struct Process {

int pid;

int burstTime;

int arrivalTime;

int priority;

int waitingTime;

int turnaroundTime;

};

void fcfs(struct Process\* processes, int n) {

int currentTime = 0;

printf("\nFCFS Scheduling:\n");

printf("Gantt Chart: ");

// Calculating waiting time and turnaround time for each process

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

if (processes[i].arrivalTime > currentTime) {

currentTime = processes[i].arrivalTime;

}

processes[i].waitingTime = currentTime - processes[i].arrivalTime;

processes[i].turnaroundTime = processes[i].waitingTime + processes[i].burstTime;

currentTime += processes[i].burstTime;

printf("P%d ", processes[i].pid);

}

printf("\n");

}

void sjf(struct Process\* processes, int n) {

// Sorting processes based on burst time

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

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

if (processes[j].burstTime > processes[j+1].burstTime) {

struct Process temp = processes[j];

processes[j] = processes[j+1];

processes[j+1] = temp;

}

}

}

printf("\nSJF Scheduling:\n");

printf("Gantt Chart: ");

// Calculating waiting time and turnaround time for each process

int currentTime = 0;

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

if (processes[i].arrivalTime > currentTime) {

currentTime = processes[i].arrivalTime;

}

processes[i].waitingTime = currentTime - processes[i].arrivalTime;

processes[i].turnaroundTime = processes[i].waitingTime + processes[i].burstTime;

currentTime += processes[i].burstTime;

printf("P%d ", processes[i].pid);

}

printf("\n");

}

void priority(struct Process\* processes, int n) {

// Sorting processes based on priority

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

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

if (processes[j].priority > processes[j+1].priority) {

struct Process temp = processes[j];

processes[j] = processes[j+1];

processes[j+1] = temp;

}

}

}

printf("\nPriority-based Scheduling:\n");

printf("Gantt Chart: ");

// Calculating waiting time and turnaround time for each process

int currentTime = 0;

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

if (processes[i].arrivalTime > currentTime) {

currentTime = processes[i].arrivalTime;

}

processes[i].waitingTime = currentTime - processes[i].arrivalTime;

processes[i].turnaroundTime = processes[i].waitingTime + processes[i].burstTime;

currentTime += processes[i].burstTime;

printf("P%d ", processes[i].pid);

}

printf("\n");

}

void srtf(struct Process\* processes, int n) {

// Sorting processes based on arrival time

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

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

if (processes[j].arrivalTime > processes[j+1].arrivalTime) {

struct Process temp = processes[j];

processes[j] = processes[j+1];

processes[j+1] = temp;

}

}

}

printf("\nSRTF Scheduling:\n");

printf("Gantt Chart: ");

// Calculating waiting time and turnaround time for each process

int currentTime = 0;

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

if (i != 0 && processes[i].arrivalTime > currentTime) {

currentTime = processes[i].arrivalTime;

}

int shortestProcess = i;

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

if (processes[j].burstTime < processes[shortestProcess].burstTime &&

processes[j].arrivalTime <= currentTime) {

shortestProcess = j;

}

}

struct Process temp = processes[i];

processes[i] = processes[shortestProcess];

processes[shortestProcess] = temp;

processes[i].waitingTime = currentTime - processes[i].arrivalTime;

processes[i].turnaroundTime = processes[i].waitingTime + processes[i].burstTime;

currentTime += processes[i].burstTime;

printf("P%d ", processes[i].pid);

}

printf("\n");

}

void calculateAvgWaitingTime(struct Process\* processes, int n) {

int totalWaitingTime = 0;

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

totalWaitingTime += processes[i].waitingTime;

}

float avgWaitingTime = (float) totalWaitingTime / n;

printf("Average Waiting Time: %f\n", avgWaitingTime);

}

void calculateAvgTurnaroundTime(struct Process\* processes, int n) {

int totalTurnaroundTime = 0;

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

totalTurnaroundTime += processes[i].turnaroundTime;

}

float avgTurnaroundTime = (float) totalTurnaroundTime / n;

printf("Average Turnaround Time: %f\n", avgTurnaroundTime);

}

void printProcessDetails(struct Process\* processes, int n) {

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

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

printf("%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", processes[i].pid, processes[i].burstTime,

processes[i].arrivalTime, processes[i].priority, processes[i].waitingTime,

processes[i].turnaroundTime);

}

}

int main() {

int n;

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

scanf("%d", &n);

struct Process processes[n];

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

processes[i].pid = i + 1;

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

scanf("%d", &processes[i].burstTime);

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

scanf("%d", &processes[i].arrivalTime);

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

scanf("%d", &processes[i].priority);

}

fcfs(processes, n);

sjf(processes, n);

priority(processes, n);

srtf(processes, n);

calculateAvgWaitingTime(processes, n);

calculateAvgTurnaroundTime(processes, n);

printProcessDetails(processes, n);

return 0;

}