Open | fcfs.c | ~/OS1/Assignment 3 | Save

queue.h | fcfs.c | ex.c | C2.c | B1.c

```c
#include<stdio.h>
#include<stdlib.h>
#define MAX 10

typedef struct process{
    int id;
    int at, bt, wt, tat;
}process;


int currentTime=0;
int processCount;
float avgTat;
float avgWt;

process p[MAX];



void sort() {
    process temp;
    int i,j;
    //printf("N : %d\n", processCount);
    for(i=0; i<processCount; i++){
        for(j=0; j<processCount-i-1; j++){
            //printf("i and j : %d %d\n", i, j);
            if(p[j].at > p[j+1].at ) {
                temp = p[j];
                p[j] = p[j+1];
                p[j+1] = temp;
            }
        }
    }

}
```

C | Tab Width: 4 | Ln 1, Col 1 | INS

```
34
35 }
36
37
38 void schedule(){
39     int i=0;
40     for(i=0; i<processCount; i++){
41     printf("| %d : P%d ", currentTime, p[i].id);
42         if(i==0){
43             p[i].wt = p[i].at- currentTime >=0 ? p[i].at-currentTime : currentTime-p[i].at;
44             currentTime += p[i].at + p[i].bt;
45             p[i].tat= currentTime - p[i].at;
46             avgTat += p[i].tat;
47             avgWt += p[i].wt;
48         }
49         else {
50             p[i].wt = currentTime - p[i].at;
51             currentTime += p[i].bt;
52             p[i].tat = currentTime - p[i].at;
53             avgTat += p[i].tat;
54             avgWt += p[i].wt;
55         }
56     }
57
58     printf(" | %d : idle |\n\n ",currentTime);
59     avgTat = avgTat/processCount;
60     avgWt = avgWt/processCount;
61 }
62
63
64
65 void read_process(){
66     int i;
67     printf("Enter the no of process : ");
```

```
62
63
64
65 void read_process(){
66     int i;
67     printf("Enter the no of process : ");
68     scanf("%d", &processCount);
69
70     for(i=0; i<processCount; i++){
71         printf("Enter the arrival time and burst time of %dth process resp. : ", i+1);
72         scanf("%d %d", &p[i].at, &p[i].bt);
73         p[i].id = i+1;
74         //printf("at & bt : %d %d", p[i].at, p[i].bt);
75         printf("\n");
76     }
77
78 }
79
80
81 void printData(){
82     int i;
83     printf("Process\t Arrival time\t Burst time\t Waiting time\t Turn around time\n");
84     for(i=0; i<processCount; i++){
85         printf("P%d\t\t%d\t\t%d\t\t%d\t\t%d\n", p[i].id, p[i].at, p[i].bt, p[i].wt, p[i].tat);
86     }
87     printf("\n");
88     printf("Average turn around time : %.2f\n", avgTat);
89     printf("Average waiting time : %.2f\n", avgWt);
90     printf("All process ended at time : %d", currentTime);
91     printf("\n\n");
92
93
94 }
95
96 void randomBurst() {
```

```c
void printData(){
    int i;
    printf("Process\t Arrival time\t Burst time\t Waiting time\t Turn around time\n");
    for(i=0; i<processCount; i++){
        printf("P%d\t\t%d\t\t%d\t\t%d\t\t%d\n", p[i].id, p[i].at, p[i].bt, p[i].wt, p[i].tat);
    }
    printf("\n");
    printf("Average turn around time : %.2f\n", avgTat);
    printf("Average waiting time : %.2f\n", avgWt);
    printf("All process ended at time : %d", currentTime);
    printf("\n\n");


}

void randomBurst() {
    int i=0;
    for(i=0; i<processCount; i++){
        p[i].bt = rand()/100000000;
        p[i].at += currentTime ;
    }
}

void main(){
    read_process();
    sort();
    printf("---------------------- Gantt Chart ----------------------\n\n");
    schedule();
    printData();
    currentTime += 2;
    randomBurst();
    printf("---------------------- Gantt Chart ----------------------\n\n");
    schedule();
    printData();
}
```

Open | B1.c | ~/OS1/Assignment 3 | Save

queue.h | fcfs.c | C2.c | B1.c

```c
#include<stdio.h>
#define MAX 10

typedef struct process {
    int id, at, bt, rt, tat, wt;
} Process;

Process p[MAX] ;
int currentTime =0 ;

void schedule(int n){
    int completed = 0;
    int execIndex = -1;

    while(completed < n){
        int minBurst = 999;
        int minIndex = -1;
        int i;

        for(i=0; i<n; i++){
            if(p[i].at <= currentTime && p[i].rt < minBurst  &&p[i].rt > 0){
                minBurst = p[i].rt;
                minIndex = i;
            }
        }

        if(minIndex != -1){
            if(minIndex != execIndex)
            printf("| %d : P%d ", currentTime, p[minIndex].id);

            p[minIndex].rt--;
            currentTime++;

            execIndex = minIndex;

```

C | Tab Width: 4 | Ln 1, Col 1 | INS

```
30
31            p[minIndex].rt--;
32            currentTime++;
33
34            execIndex = minIndex;
35
36            if(p[minIndex].rt == 0){
37                completed++;
38                p[minIndex].tat = currentTime - p[minIndex].at;
39                p[minIndex].wt = p[minIndex].tat - p[minIndex].bt ;
40            }
41        }
42        else{
43            printf("| %d : idle ", currentTime);
44            currentTime++;
45        }
46
47    }
48    printf("| %d : idle |\n", currentTime);
49}
50
51
52void randomBurst(int n) {
53    int i=0;
54    for(i=0; i<n; i++){
55        p[i].bt = rand()/100000000;
56        p[i].rt = p[i].bt;
57        p[i].at += currentTime ;
58    }
59}
60
61void printData (int n){
62    float totalTat = 0;
63    float totalWt = 0;
```

```
58      }
59  }
60
61  void printData (int n){
62      float totalTat = 0;
63      float totalWt = 0;
64      int i;
65
66      printf("\n\nProcess\tWaiting time\tTurn Around time\n");
67      for(i=0; i<n ; i++){
68          printf("P%d\t\t%d\t\t%d\n",p[i].id, p[i].wt,p[i].tat);
69          totalTat += p[i].tat;
70          totalWt += p[i].wt;
71      }
72
73      totalTat = totalTat/n;
74      totalWt = totalWt/n;
75      printf("\nTotal Average time : %.2f\n", totalTat);
76      printf("Total Average time : %.2f\n", totalWt);
77  }
78
79
80  void main() {
81      int n, i;
82      printf("Enter the no of processes : ");
83      scanf("%d", &n);
84
85      for(i=0; i<n; i++){
86          printf("Enter the at & bt of process %d : ", i+1);
87          scanf("%d%d", &p[i].at, &p[i].bt);
88          p[i].rt = p[i].bt;
89          p[i].id = i+1;
90      }
91
92      printf("\n********** Gantt chart **********\n");
```

```
69          totalTat += p[i].tat;
70          totalWt += p[i].wt;
71      }
72
73      totalTat = totalTat/n;
74      totalWt = totalWt/n;
75      printf("\nTotal Average time : %.2f\n", totalTat);
76      printf("Total Average time : %.2f\n", totalWt);
77  }
78
79
80  void main() {
81      int n, i;
82      printf("Enter the no of processes : ");
83      scanf("%d", &n);
84
85      for(i=0; i<n; i++){
86          printf("Enter the at & bt of process %d : ", i+1);
87          scanf("%d%d", &p[i].at, &p[i].bt);
88          p[i].rt = p[i].bt;
89          p[i].id = i+1;
90      }
91
92      printf("\n********************* Gantt chart *********************\n");
93
94      schedule(n);
95      printData(n);
96
97      currentTime+=2;
98      randomBurst(n);
99
100     printf("\n********************* Gantt chart *********************\n");
101     schedule(n);
102     printData(n);
103 }
```

Open | C2.c ~/OS1/Assignment 3 | Save

queue.h | fcfs.c | C2.c | B1.c

```c
#include<stdio.h>
#define MAX 10

typedef struct process
{
    int id, at, bt, rt, tat, wt;
} Process;

Process p[MAX] ;
int currentTime =0 ;


void schedule(int n, int timeQuantum){
    int completed = 0;
    int execTime = 0;

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

            if(p[i].at <= currentTime && p[i].rt>0){
                execTime = p[i].rt < timeQuantum ? p[i].rt : timeQuantum;

                printf("| %d : P%d ", currentTime, p[i].id);

                p[i].rt -= execTime;
                currentTime += execTime;
                executed = 1;
                if(p[i].rt == 0){
                    completed++;
                    p[i].tat = currentTime - p[i].at;
                    p[i].wt = p[i].tat - p[i].bt ;
                }

```

C | Tab Width: 4 | Ln 1, Col 1 | INS

```
            }

        }
    }
    if(!executed){
        printf("| %d : idle ", currentTime);
        currentTime++;
    }



}
printf("| %d : idle |\n", currentTime);
}


void randomBurst(int n) {
    int i=0;
    for(i=0; i<n; i++){
        p[i].bt = rand()/100000000;
        p[i].rt = p[i].bt;
        p[i].at += currentTime ;
    }
}


void printData (int n){
    float totalTat = 0;
    float totalWt = 0;
    int i;

    printf("\n\nProcess\tWaiting time\tTurn Around time\n");
    for(i=0; i<n ; i++){
        printf("P%d\t\t%d\t\t%d\n",p[i].id, p[i].wt,p[i].tat);
```

```
    totalTat = totalTat/n;
    totalWt = totalWt/n;
    printf("\nTotal Average time : %.2f\n", totalTat);
    printf("Total Average time : %.2f\n", totalWt);
}



void main() {
    int n, i, timeQuantum;
    printf("Enter the no of processes : ");
    scanf("%d", &n);
    printf("Enter the time quantum : ");
    scanf("%d", &timeQuantum);

    for(i=0; i<n; i++){
        printf("Enter the at & bt of process %d : ", i+1);
        scanf("%d%d", &p[i].at, &p[i].bt);
        p[i].rt = p[i].bt;
        p[i].id = i+1;
    }

    printf("\n********** Gantt chart **********\n");
    schedule(n, timeQuantum);
    printData(n);

    currentTime += 2;
    randomBurst(n);

    printf("\n********** Gantt chart **********\n");
    schedule(n, timeQuantum);
    printData(n);

}
```