Shortest Job First Algorithm

Aishi De

```
aishi@Aishi:~$ vi sjf.c
aishi@Aishi:~$ cat sjf.c

#include <stdio.h>

struct Process {
    int pid;
    int arrival_time;
    int burst_time;
    int start_time;
    int completion_time;
    int turnaround_time;
    int waiting_time;
    int is_completed;
};

int main() {
    int n, completed = 0, current_time = 0, i, min_index;
    struct Process p[10];

    printf("Enter number of processes: ");
    scanf("%d", &n);

    for (i = 0; i < n; i++) {
        p[i].pid = i + 1;
        printf("Enter arrival time of process P%d: ", p[i].pid);
        scanf("%d", &p[i].arrival_time);
        printf("Enter burst time of process P%d: ", p[i].pid);
        scanf("%d", &p[i].burst_time);
        p[i].is_completed = 0;
    }

    printf("\n");

    while (completed != n) {
        int min_bt = 9999;
        min_index = -1;
```

```
            if (p[i].burst_time < min_bt) {
                min_bt = p[i].burst_time;
                min_index = i;
            }
        }
    }

    if (min_index == -1) {
        current_time++;
    } else {
        p[min_index].start_time = current_time;
        p[min_index].completion_time = current_time + p[min_index].burst_time;
        p[min_index].turnaround_time = p[min_index].completion_time - p[min_index].arrival_time;
        p[min_index].waiting_time = p[min_index].turnaround_time - p[min_index].burst_time;

        current_time = p[min_index].completion_time;
        p[min_index].is_completed = 1;
        completed++;
    }
}

// Print process table
printf("PID\tAT\tBT\tST\tCT\tTAT\tWT\n");
for (i = 0; i < n; i++) {
    printf("P%d\t%d\t%d\t%d\t%d\t%d\t%d\n", p[i].pid, p[i].arrival_time,
           p[i].burst_time, p[i].start_time, p[i].completion_time,
           p[i].turnaround_time, p[i].waiting_time);
}

// Gantt Chart
printf("\nGantt Chart:\n|");
for (i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        if (p[j].start_time == p[i].start_time) {
            printf("  P%d  |", p[j].pid);
            break;
        }
```

```
                printf("  P%d  |", p[j].pid);
                break;
            }
        }
    }

    printf("\n%d", p[0].start_time);
    for (i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (p[j].start_time == p[i].start_time) {
                printf("     %d", p[j].completion_time);
                break;
            }
        }
    }

    printf("\n");

    return 0;
}
```

```
aishi@Aishi:~$ touch output
aishi@Aishi:~$ gcc sjf.c -o output
aishi@Aishi:~$ ./output
Enter number of processes: 3
Enter arrival time of process P1: 0
Enter burst time of process P1: 6
Enter arrival time of process P2: 1
Enter burst time of process P2: 4
Enter arrival time of process P3: 2
Enter burst time of process P3: 2

PID     AT      BT      ST      CT      TAT     WT
P1      0       6       0       6       6       0
P2      1       4       8       12      11      7
P3      2       2       6       8       6       4

Gantt Chart:
|  P1  |  P2  |  P3  |
0     6     12     8
aishi@Aishi:~$
```