

Expr 6 b: Shortest Job First

Code:

```
#include <stdio.h>
#include <string.h>

#define MAX 10

// Step 1: Declare structure
struct Process {
    char name[10];
    int arrivalTime, burstTime;
    int waitingTime, turnaroundTime;
};

int main() {
    struct Process p[MAX], temp;
    int n;
    int totalWT = 0, totalTAT = 0;

    // Step 2: Input number of processes
    printf("Enter the number of processes: ");
    scanf("%d", &n);

    // Step 3: Input process info
    for (int i = 0; i < n; i++) {
        printf("\nEnter name for process %d: ", i + 1);
        scanf("%s", p[i].name);
        printf("Enter arrival time for %s: ", p[i].name);
        scanf("%d", &p[i].arrivalTime);
        printf("Enter burst time for %s: ", p[i].name);
        scanf("%d", &p[i].burstTime);
        // Step 4: Initialize WT, TAT
        p[i].waitingTime = 0;
        p[i].turnaroundTime = 0;
    }

    // Step 5: Sort based on burst time (Simple selection sort)
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (p[j].burstTime < p[i].burstTime) {
                temp = p[i];
                p[i] = p[j];
                p[j] = temp;
            }
        }
    }

    // Step 6: Calculate WT and TAT
```

```

p[0].waitingTime = 0;
p[0].turnaroundTime = p[0].burstTime;

for (int i = 1; i < n; i++) {
    p[i].waitingTime = p[i - 1].waitingTime + p[i - 1].burstTime;
    p[i].turnaroundTime = p[i].waitingTime + p[i].burstTime;
}

// Step 7: Calculate total and average
for (int i = 0; i < n; i++) {
    totalWT += p[i].waitingTime;
    totalTAT += p[i].turnaroundTime;
}

float avgWT = (float)totalWT / n;
float avgTAT = (float)totalTAT / n;

// Step 8: Display results
printf("\nProcess\tArrival\tBurst\tWaiting\tTurnaround\n");
for (int i = 0; i < n; i++) {
    printf("%s\t%d\t%d\t%d\t%d\n",
        p[i].name, p[i].arrivalTime, p[i].burstTime,
        p[i].waitingTime, p[i].turnaroundTime);
}

printf("\nTotal Waiting Time: %d", totalWT);
printf("\nAverage Waiting Time: %.2f", avgWT);
printf("\nTotal Turnaround Time: %d", totalTAT);
printf("\nAverage Turnaround Time: %.2f\n", avgTAT);

return 0;
}

```

Output:

```

Enter the number of processes: 3
Enter name for process 1: P1
Enter arrival time for P1: 0
Enter burst time for P1: 8
Enter name for process 2: P2
Enter arrival time for P2: 1
Enter burst time for P2: 4
Enter name for process 3: P3
Enter arrival time for P3: 2
Enter burst time for P3: 2

```

Process		Arrival	Burst	Waiting	Turnaround
P3	2	2	0	2	
P2	1	4	2	6	
P1	0	8	6	14	

Total Waiting Time: 8

Average Waiting Time: 2.67

Total Turnaround Time: 22

Average Turnaround Time: 7.33

Result:

Thus the Shortest Job First Code is implemented in fedora using the C language