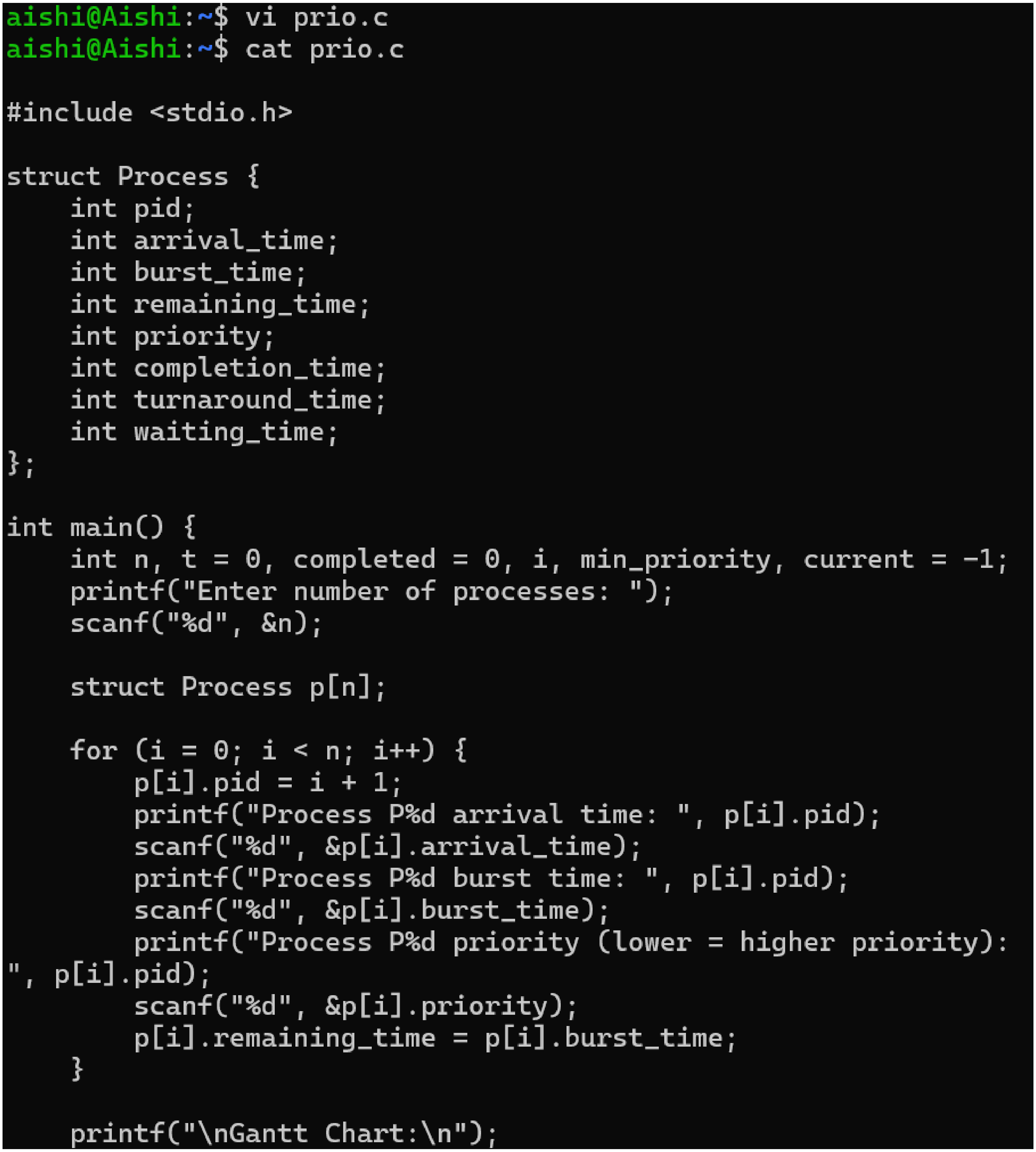
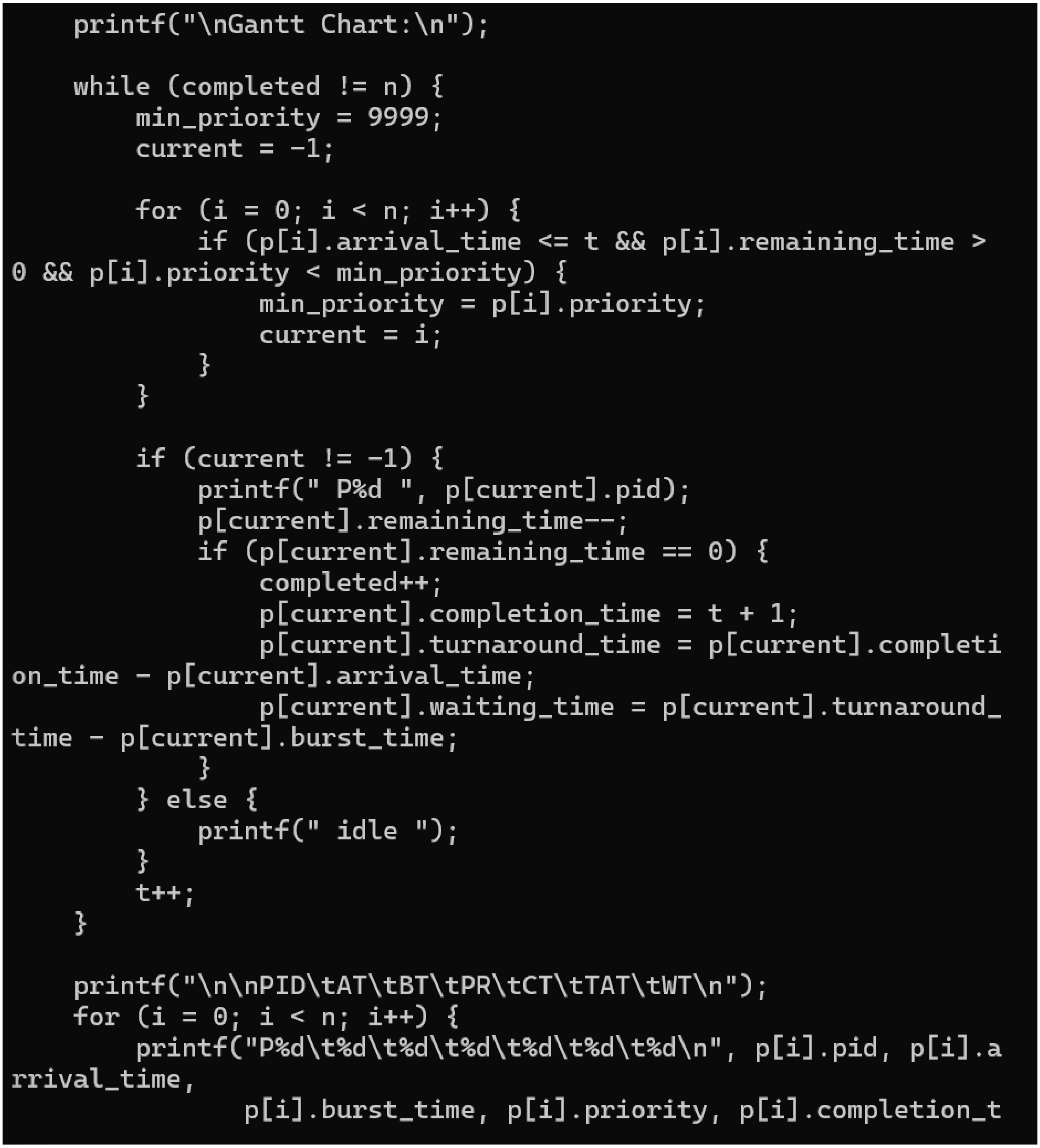
Preemptive Priority Scheduling

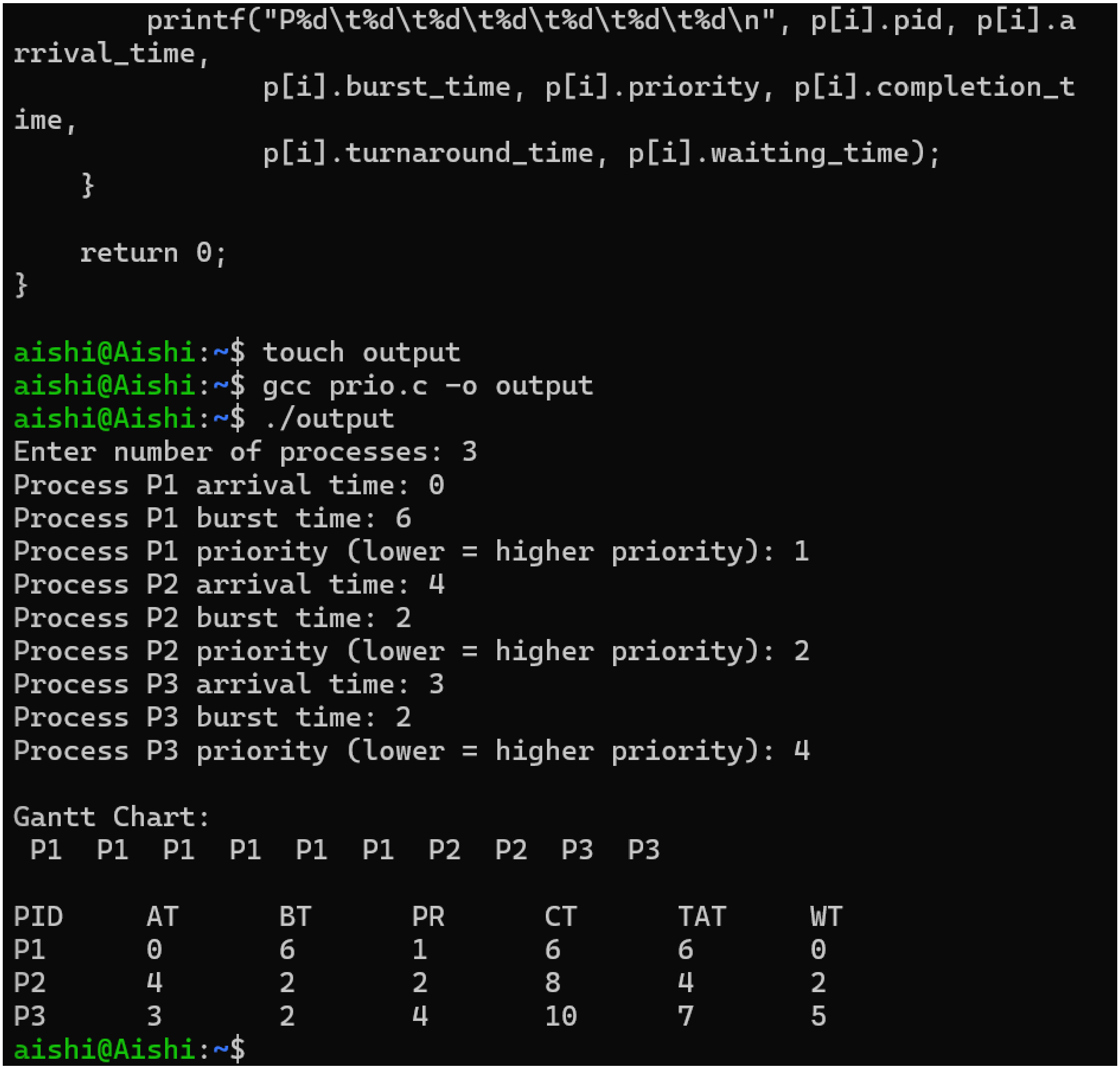
Aishi De



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

#include <stdio.h>

struct Process {
    int pid;
    int arrival_time;
    int burst_time;
    int remaining_time;
    int priority;
    int completion_time;
    int turnaround_time;
    int waiting_time;
};

int main() {
    int n, t = 0, completed = 0, i, min_priority, current = -1;
    printf("Enter number of processes: ");
    scanf("%d", &n);

    struct Process p[n];

    for (i = 0; i < n; i++) {
        p[i].pid = i + 1;
        printf("Process P%d arrival time: ", p[i].pid);
        scanf("%d", &p[i].arrival_time);
        printf("Process P%d burst time: ", p[i].pid);
        scanf("%d", &p[i].burst_time);
        printf("Process P%d priority (lower = higher priority):
", p[i].pid);
        scanf("%d", &p[i].priority);
        p[i].remaining_time = p[i].burst_time;
    }

    printf("\nGantt Chart:\n");
```



```
    printf("\nGantt Chart:\n");

    while (completed != n) {
        min_priority = 9999;
        current = -1;

        for (i = 0; i < n; i++) {
            if (p[i].arrival_time <= t && p[i].remaining_time >
0 && p[i].priority < min_priority) {
                min_priority = p[i].priority;
                current = i;
            }
        }

        if (current != -1) {
            printf(" P%d ", p[current].pid);
            p[current].remaining_time--;
            if (p[current].remaining_time == 0) {
                completed++;
                p[current].completion_time = t + 1;
                p[current].turnaround_time = p[current].completi
on_time - p[current].arrival_time;
                p[current].waiting_time = p[current].turnaround_
time - p[current].burst_time;
            }
        } else {
            printf(" idle ");
        }
        t++;
    }

    printf("\n\nPID\tAT\tBT\tPR\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].a
rrival_time,
               p[i].burst_time, p[i].priority, p[i].completion_t
```



```
        printf("P%d\t%d\t%d\t%d\t%d\t%d\t%d\n", p[i].pid, p[i].a
rrival_time,
               p[i].burst_time, p[i].priority, p[i].completion_t
ime,
               p[i].turnaround_time, p[i].waiting_time);
    }

    return 0;
}

aishi@Aishi:~$ touch output
aishi@Aishi:~$ gcc prio.c -o output
aishi@Aishi:~$ ./output
Enter number of processes: 3
Process P1 arrival time: 0
Process P1 burst time: 6
Process P1 priority (lower = higher priority): 1
Process P2 arrival time: 4
Process P2 burst time: 2
Process P2 priority (lower = higher priority): 2
Process P3 arrival time: 3
Process P3 burst time: 2
Process P3 priority (lower = higher priority): 4

Gantt Chart:
 P1  P1  P1  P1  P1  P1  P2  P2  P3  P3

PID     AT      BT      PR      CT      TAT     WT
P1      0       6       1       6       6       0
P2      4       2       2       8       4       2
P3      3       2       4       10      7       5
aishi@Aishi:~$
```