**Expr 6 c: Priority Scheduling**

**Code:**

#include <stdio.h>

#include <string.h>

#define MAX 10

// Process structure

struct Process {

    char name[10];

    int burstTime, priority;

    int waitingTime, turnaroundTime;

};

int main() {

    struct Process p[MAX], temp;

    int n;

    int totalWT = 0, totalTAT = 0;

    // Step 1: Input number of processes

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

    scanf("%d", &n);

    // Step 2: Input process details

    for (int i = 0; i < n; i++) {

        printf("\nEnter name for process %d: ", i + 1);

        scanf("%s", p[i].name);

        printf("Enter burst time for %s: ", p[i].name);

        scanf("%d", &p[i].burstTime);

        printf("Enter priority for %s: ", p[i].name);

        scanf("%d", &p[i].priority);

        // Step 4: Initialize waiting time and turnaround time

        p[i].waitingTime = 0;

        p[i].turnaroundTime = 0;

    }

    // Step 3: Sort processes based on priority (lowest number is highest priority)

    for (int i = 0; i < n - 1; i++) {

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

            if (p[j].priority < p[i].priority) {

                temp = p[i];

                p[i] = p[j];

                p[j] = temp;

            }

        }

    }

    // Step 4: Calculate waiting time and turnaround time

    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 5: Calculate total waiting time and total turnaround time

    for (int i = 0; i < n; i++) {

        totalWT += p[i].waitingTime;

        totalTAT += p[i].turnaroundTime;

    }

    // Step 6: Display results

    printf("\nProcess\tBurst\tPriority\tWaiting\tTurnaround\n");

    for (int i = 0; i < n; i++) {

        printf("%s\t%d\t%d\t\t%d\t%d\n", p[i].name, p[i].burstTime, p[i].priority, p[i].waitingTime, p[i].turnaroundTime);

    }

    // Display total and average times

    float avgWT = (float)totalWT / n;

    float avgTAT = (float)totalTAT / n;

    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 burst time for P1: 6

Enter priority for P1: 2

Enter name for process 2: P2

Enter burst time for P2: 8

Enter priority for P2: 1

Enter name for process 3: P3

Enter burst time for P3: 7

Enter priority for P3: 3

Process Burst Priority Waiting Turnaround

P2 8 1 0 8

P1 6 2 8 14

P3 7 3 14 21

Total Waiting Time: 22

Average Waiting Time: 7.33

Total Turnaround Time: 43

Average Turnaround Time: 14.33

**Result:**

Thus the Priority Scheduling Code is implemented in fedora using the C language