

Name: REVI THIMMA REDDY

Reg-No: 192325025

6. Construct a C program to implement preemptive priority scheduling algorithm

Aim:

To implement a preemptive priority scheduling algorithm in C to schedule processes based on their priority and calculate metrics like waiting time and turnaround time.

Algorithm:

1. Input the number of processes, their burst times, and priorities.
2. Initialize time to 0 and process data structures.
3. Continuously:
 - o Select the highest-priority process that is ready to execute.
 - o Execute it for one unit of time.
 - o Update the remaining burst time for the process.
4. Stop when all processes are complete.
5. Calculate waiting time and turnaround time for each process.

Procedure:

1. Read input data for processes (arrival time, burst time, priority).
2. Use a loop to simulate the scheduling clock:
 - o Find the process with the highest priority at the current time.
 - o Update burst times and track completed processes.
3. Calculate the waiting time and turnaround time for each process.
4. Display the schedule and computed metrics.

Code:

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

```
#include <limits.h>
```

```
struct Process {
```

```
    int pid, at, bt, pri, rt, wt, tat, completed;
```

```
};
```

```
int main() {
```

```

int n, time = 0, completed = 0;

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

scanf("%d", &n);

struct Process p[n];

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

    printf("Enter arrival time, burst time, priority for process %d: ", i + 1);

    scanf("%d %d %d", &p[i].at, &p[i].bt, &p[i].pri);

    p[i].pid = i + 1;

    p[i].rt = p[i].bt;

    p[i].completed = 0;

}

while (completed < n) {

    int idx = -1, min_pri = INT_MAX;

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

        if (p[i].at <= time && p[i].completed == 0 && p[i].pri < min_pri) {

            min_pri = p[i].pri;

            idx = i;

        }

    }

    if (idx != -1) {

        p[idx].rt--;
    }
}

```

```

    time++;

    if (p[idx].rt == 0) {

        p[idx].completed = 1;

        completed++;

        p[idx].tat = time - p[idx].at;

        p[idx].wt = p[idx].tat - p[idx].bt;

    }

} else {

    time++;

}

}

printf("\nPID\tAT\tBT\tPRI\tWT\tTAT\n");

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

    printf("%d\t%d\t%d\t%d\t%d\t%d\n", p[i].pid, p[i].at, p[i].bt, p[i].pri, p[i].wt, p[i].tat);

}


return 0;

}

```

Result:

Input: Number of processes, their arrival times, burst times, and priorities.

Output:

```
main.c
1 #include <stdio.h>
2 #include <limits.h>
3 struct Process {
4     int pid, at, bt, pri, rt, wt, tat, completed;
5 };
6 int main() {
7     int n, time = 0, completed = 0;
8     printf("Enter the number of processes: ");
9     scanf("%d", &n);
10    struct Process p[n];
11    for (int i = 0; i < n; i++) {
12        printf("Enter arrival time, burst time, priority for process %d: ", i + 1);
13        scanf("%d %d %d", &p[i].at, &p[i].bt, &p[i].pri);
14        p[i].pid = i + 1;
15        p[i].rt = p[i].bt;
16        p[i].completed = 0;
17    }
18    while (completed < n) {
19        int idx = -1, min_pri = INT_MAX;
20        for (int i = 0; i < n; i++) {
21            if (p[i].at <= time && p[i].completed == 0 && p[i].pri < min_pri) {
22                min_pri = p[i].pri;
23                idx = i;
24            }
25            if (idx != -1) {
26                p[idx].rt--;
27                time++;
28                if (p[idx].rt == 0) {
29                    p[idx].completed = 1;
30                    completed++;
31                    p[idx].tat = time - p[idx].at;
32                    p[idx].wt = p[idx].tat - p[idx].bt;
33                }
34            } else {
35                time++;
36            }
37        }
38        printf("\nPID\tAT\tBT\tPRI\tWT\tTAT\n");
39        for (int i = 0; i < n; i++) {
40            printf("%d\t%d\t%d\t%d\t%d\t%d\n", p[i].pid, p[i].at, p[i].bt, p[i].pri, p[i].wt, p[i].tat);
41        }
42    }
43    return 0;
44 }
```

Enter the number of processes: 3
Enter arrival time, burst time, priority for process 1: 2 3 4
Enter arrival time, burst time, priority for process 2: 1 2 3
Enter arrival time, burst time, priority for process 3: 1 2 4

PID	AT	BT	PRI	WT	TAT
1	2	3	4	1	4
2	1	2	3	0	2
3	1	2	4	5	7