

Name: K.SaiKrishna

Reg-No: 192311106

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;

}

```

Output:

Welcome, *K Sai Krishna*

Create New Project

My Projects

Classroom new

Learn Programming

Programming Questions

Upgrade

Logout

```
65
66     return 0;
67 }
```

Enter the number of processes: 3
Enter arrival time, burst time, and priority of process 1: 1 2 3
Enter arrival time, burst time, and priority of process 2: 4 56
7
Enter arrival time, burst time, and priority of process 3: 8 9 11

| PID | AT | BT | Pri | CT | TAT | WT |
|-----|----|----|-----|----|-----|----|
| 1 | 1 | 2 | 3 | 3 | 2 | 0 |
| 2 | 4 | 56 | 7 | 60 | 56 | 0 |
| 3 | 8 | 9 | 11 | 69 | 61 | 52 |

...Program finished with exit code 0
Press ENTER to exit console.

Result:

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