NAME : Jenil Kotadia

ROLL NO : 19

SE-DIV: 2

Vidyavardhini's College of Engineering & Technology
Department of Computer Engineering
Academic Year : 2023-24

Experiment no 6

Aim: To study and implement process scheduling algorithm FCFS and SJF using CPU -OS Simulator

Objective: Its Main Objective is to increase CPU utilization and hence the throughput of the system by keeping the CPU as busy as possible .

Theory:

FIRST COME FIRST SERVE:

Simplest CPU scheduling algorithm that schedules according to arrival times of processes. The first come first serve scheduling algorithm states that the process that requests the CPU first is allocated the CPU first. It is implemented by using the FIFO queue. When a process enters the ready queue, its PCB is linked to the tail of the queue. When the CPU is free, it is allocated to the process at the head of the queue. The running process is then removed from the queue. FCFS is a non-preemptive scheduling algorithm.

SHORTEST JOB FIRST:

The shortest job first (SJF) or shortest job next, is a scheduling policy that selects the waiting process with the smallest execution time to execute next. SJN, also known as Shortest Job Next (SJN), can be preemptive or non-preemptive.

Characteristics of SJF Scheduling:

Shortest Job first has the advantage of having a minimum average waiting time among all scheduling algorithms.

It is a Greedy Algorithm.

It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing.

CODE:

```
#include <stdio.h>

void findWaitingTime(int processes[], int n, int bt[], int wt[]) {
    wt[0] = 0;
    for (int i = 1; i < n; i++)
        wt[i] = bt[i - 1] + wt[i - 1];
}

void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[]) {
    for (int i = 0; i < n; i++)
        tat[i] = bt[i] + wt[i];
}

void findavgTime(int processes[], int n, int bt[]) {
    int wt[n], tat[n], total_wt = 0, total_tat = 0;

    findWaitingTime(processes, n, bt, wt);
    findTurnAroundTime(processes, n, bt, wt, tat);
```

```
    printf("Processes\tBurst time\tWaiting time\tTurnaround time\n");
    for (int i = 0; i < n; i++) {
        total_wt += wt[i];
        total_tat += tat[i];
        printf("%d\t\t%d\t\t%d\t\t%d\n", i + 1, bt[i], wt[i], tat[i]);
    }

    printf("Average waiting time = %.2f\n", (float)total_wt / n);
    printf("Average turnaround time = %.2f\n", (float)total_tat / n);
}

int main() {
    int processes[] = {1, 2, 3};
    int n = sizeof(processes) / sizeof(processes[0]);
    int burst_time[] = {11, 9, 7};

    findavgTime(processes, n, burst_time);
    return 0;
}
```

OUTPUT:

```
/tmp/8jHtB4Xp6E.o
Processes   Burst time  Waiting time    Turnaround time
1       11      0       11
2       9       11      20
3       7       20      27
Average waiting time = 10.33
Average turnaround time = 19.33


=== Code Execution Successful ===
```

CODE:

```
#include <stdio.h>
int main()
{
    int A[100][4];
    int i, j, n, total = 0, index, temp;
    float avg_wt, avg_tat;
    printf("Enter number of process: ");
    scanf("%d", &n);
    printf("Enter Burst Time:\n");

    for (i = 0; i < n; i++) {
        printf("P%d: ", i + 1);
        scanf("%d", &A[i][1]);
        A[i][0] = i + 1;
    }

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

```
        index = i;
        for (j = i + 1; j < n; j++)
                if (A[j][1] < A[index][1])
                        index = j;
        temp = A[i][1];
        A[i][1] = A[index][1];
        A[index][1] = temp;

        temp = A[i][0];
        A[i][0] = A[index][0];
        A[index][0] = temp;
}
A[0][2] = 0;

for (i = 1; i < n; i++) {
        A[i][2] = 0;
        for (j = 0; j < i; j++)
                A[i][2] += A[j][1];
        total += A[i][2];
}
avg_wt = (float)total / n;
total = 0;
printf("P       BT     WT    TAT\n");

for (i = 0; i < n; i++) {
        A[i][3] = A[i][1] + A[i][2];
        total += A[i][3];
```

```
            printf("P%d  %d     %d     %d\n", A[i][0],
                    A[i][1], A[i][2], A[i][3]);
        }
        avg_tat = (float)total / n;
        printf("Average Waiting Time= %f", avg_wt);
        printf("\nAverage Turnaround Time= %f", avg_tat);
}
```

OUTPUT:

```
/tmp/qyi8BYO4Ts.o
Enter number of process: 4
Enter Burst Time:
P1: 20
P2: 10
P3: 5
P4: 15
P    BT  WT  TAT
P3   5   0   5
P2   10  5   15
P4   15  15  30
P1   20  30  50
Average Waiting Time= 12.500000
Average Turnaround Time= 25.000000

=== Code Execution Successful ===
```

Conclusion :

FCFS (First-Come-First-Serve): Simple and intuitive. Executes processes in the order they arrive. May lead to longer average waiting times.

SJF (Shortest Job First): Prioritizes processes with the shortest execution time. Aims to minimize average waiting time. Can result in starvation for longer processes.