Operating System Lab

(4ITRC2)

IT IV Semester

Submitted by

Himanshu Priyadarshi Ahirwar

23I4026

IT-A

Submitted to

Jasneet Kaur

Department of Information Technology

Institute of Engineering and Technology

Devi Ahilya Vishwavidhyalaya, Indore (M.P.) India

(www.iet.dauniv.ac.in**)**

Session Jan-May, 2025

CPU Scheduling Algorithms:

1. First Come First Serve (FCFS)

- FCFS is the simplest CPU scheduling algorithm where the process that arrives first gets executed first, following a strict queue-like order.

Algorithm (C):-

```
#include <stdio.h>

int main() {
    int n, i;
    int bt[20], wt[20], tat[20];
    float avg_wt = 0, avg_tat = 0;

    printf("Enter number of processes: ");
    scanf("%d", &n);

    printf("Enter burst time for each process:\n");
    for (i = 0; i < n; i++) {
        printf("P[%d]: ", i + 1);
        scanf("%d", &bt[i]);
    }

    wt[0] = 0;

    // Calculate waiting time
    for (i = 1; i < n; i++) {
        wt[i] = bt[i - 1] + wt[i - 1];
    }

    // Calculate turnaround time
    for (i = 0; i < n; i++) {
        tat[i] = bt[i] + wt[i];
        avg_wt += wt[i];
        avg_tat += tat[i];
    }

    printf("\nProcess\tBT\tWT\tTAT\n");
    for (i = 0; i < n; i++) {
        printf("P[%d]\t%d\t%d\t%d\n", i + 1, bt[i], wt[i], tat[i]);
    }
```

```
    printf("\nAverage Waiting Time = %.2f", avg_wt / n);
    printf("\nAverage Turnaround Time = %.2f\n", avg_tat / n);
    return 0;
}
```

Sample Input:-

Enter number of processes: 4

Enter burst time for each process:

P[1]: 5

P[2]: 8

P[3]: 6

P[4]: 3

Output:-

Process	Burst	Waiting	Turnaround
P[1]	5	0	5
P[2]	8	5	13
P[3]	6	13	19
P[4]	3	19	22

Average Waiting Time = 9.25

Average Turnaround Time = 14.75

2. Shortest Job First (SJF) – Non-preemptive

- SJF selects the process with the smallest burst time next, minimizing average waiting time but requiring knowledge of future process durations.

Algorithm (C):-

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

```
int main() {
    int n, i, j;
    int bt[20], p[20], wt[20], tat[20], temp;
    float avg_wt = 0, avg_tat = 0;

    printf("Enter total number of processes: ");
    scanf("%d", &n);

    printf("Enter burst time for each process:\n");
    for (i = 0; i < n; i++) {
        printf("P[%d]: ", i + 1);
        scanf("%d", &bt[i]);
        p[i] = i + 1;
    }

    // Sorting burst times (SJF)
    for (i = 0; i < n - 1; i++) {
        for (j = i + 1; j < n; j++) {
            if (bt[i] > bt[j]) {
                temp = bt[i];
                bt[i] = bt[j];
                bt[j] = temp;

                temp = p[i];
                p[i] = p[j];
                p[j] = temp;
            }
```

```
        }
    }

    wt[0] = 0;

    for (i = 1; i < n; i++) {
        wt[i] = 0;
        for (j = 0; j < i; j++)
            wt[i] += bt[j];
    }

    for (i = 0; i < n; i++) {
        tat[i] = bt[i] + wt[i];
        avg_wt += wt[i];
        avg_tat += tat[i];
    }

    printf("\nProcess\tBT\tWT\tTAT\n");
    for (i = 0; i < n; i++) {
        printf("P[%d]\t%d\t%d\t%d\n", p[i], bt[i], wt[i], tat[i]);
    }

    printf("\nAverage Waiting Time = %.2f", avg_wt / n);
    printf("\nAverage Turnaround Time = %.2f\n", avg_tat / n);
    return 0;
}
```

Sample Input:-

Enter total number of processes: 4

P[1]: 6

P[2]: 8

P[3]: 7

P[4]: 3

Output:-

Process	Burst	Waiting	Turnaround
P[4]	3	0	3
P[1]	6	3	9
P[3]	7	9	16
P[2]	8	16	24

Average Waiting Time = 7.00

Average Turnaround Time = 13.00

3. Round Robin Scheduling

- Round Robin assigns a fixed time quantum to each process in a cyclic order, ensuring fair CPU allocation and better response time for all processes.

Algorithm (C):-

```
#include <stdio.h>

int main() {
    int n, i, qt, count = 0, temp, sq = 0;
    int bt[10], rem_bt[10], wt[10], tat[10];
    float avg_wt = 0, avg_tat = 0;

    printf("Enter number of processes: ");
    scanf("%d", &n);
```

```
    printf("Enter burst time for each process:\n");
    for (i = 0; i < n; i++) {
        printf("P[%d]: ", i + 1);
        scanf("%d", &bt[i]);
        rem_bt[i] = bt[i];
    }

    printf("Enter time quantum: ");
    scanf("%d", &qt);

    while (1) {
        int done = 1;
        for (i = 0; i < n; i++) {
            if (rem_bt[i] > 0) {
                done = 0;
                if (rem_bt[i] > qt) {
                    sq += qt;
                    rem_bt[i] -= qt;
                } else {
                    sq += rem_bt[i];
                    wt[i] = sq - bt[i];
                    rem_bt[i] = 0;
                }
            }
        }
        if (done == 1)
            break;
    }

    printf("\nProcess\tBT\tWT\tTAT\n");
    for (i = 0; i < n; i++) {
        tat[i] = bt[i] + wt[i];
        avg_wt += wt[i];
        avg_tat += tat[i];
        printf("P[%d]\t%d\t%d\t%d\n", i + 1, bt[i], wt[i], tat[i]);
    }

    printf("\nAverage Waiting Time = %.2f", avg_wt / n);
    printf("\nAverage Turnaround Time = %.2f\n", avg_tat / n);
    return 0;
}
```

Sample Input:-

Enter number of processes: 4
P[1]: 24
P[2]: 3
P[3]: 3
P[4]: 12
Enter time quantum: 4

Output:-

Process	Burst	Waiting	Turnaround
P[1]	24	26	50
P[2]	3	4	7
P[3]	3	7	10
P[4]	12	18	30

Average Waiting Time = 13.75
Average Turnaround Time = 24.25