SCHEDULING ALGORITHM

AADITYA P

230701001

FIRST COME FIRST SERVE (FCFS) Scheduling

```
liveuser@localhost-live:~$ cat >fcfs.c
// File: fcfs.c
#include <stdio.h>

int main() {
    int n, i;
    float total_wt = 0, total_tat = 0;

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

    int bt[n], wt[n], tat[n];

    // Read the burst times
    printf("Enter the burst time of the processes: \n");
    for (i = 0; i < n; i++) {
        scanf("%d", &bt[i]);
    }

    // Calculate waiting time and turnaround time
    wt[0] = 0;  // First process has 0 waiting time
    tat[0] = bt[0];  // Turnaround time for the first process is its burst time

    for (i = 1; i < n; i++) {
        wt[i] = bt[i-1] + wt[i-1];  // Waiting time = sum of previous burst times
        tat[i] = wt[i] + bt[i];      // Turnaround time = waiting time + burst time
    }

    // Display the results
    printf("Process\tBurst Time\tWaiting Time\tTurn Around Time\n");
    for (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, bt[i], wt[i], tat[i]);
    }

    // Display averages
    printf("Average waiting time is: %.2f\n", total_wt/n);
    printf("Average Turn around Time is: %.2f\n", total_tat/n);

    return 0;
}
liveuser@localhost-live:~$ gcc fcfs.c -o fcfs
liveuser@localhost-live:~$ ./fcfs
Enter the number of processes: 3
Enter the burst time of the processes:
24 3 3
Process Burst Time      Waiting Time    Turn Around Time
0               24              0               24
1               3               24              27
2               3               27              30
Average waiting time is: 17.00
Average Turn around Time is: 27.00
liveuser@localhost-live:~$
```

SHORTEST JOB FIRST (SJF) Scheduling

liveuser

liveuser@localhost-live:~

liveuser

```
liveuser@localhost-live:~$ cat >sjf.c
// File: sjf.c
#include <stdio.h>

struct Process {
    int pid;
    int burst_time;
    int waiting_time;
    int turnaround_time;
};

int main() {
    int n, i, j;
    float total_wt = 0, total_tat = 0;

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

    struct Process proc[n];

    // Read the burst times
    printf("Enter the burst time of the processes: \n");
    for (i = 0; i < n; i++) {
        proc[i].pid = i;
        scanf("%d", &proc[i].burst_time);
    }

    // Sort processes based on burst time
    for (i = 0; i < n-1; i++) {
        for (j = i+1; j < n; j++) {
            if (proc[i].burst_time > proc[j].burst_time) {
                struct Process temp = proc[i];
                proc[i] = proc[j];
                proc[j] = temp;
            }
        }
    }

    // Calculate waiting time and turnaround time
    proc[0].waiting_time = 0;
    proc[0].turnaround_time = proc[0].burst_time;

    for (i = 1; i < n; i++) {
        proc[i].waiting_time = proc[i-1].waiting_time + proc[i-1].burst_time;
        proc[i].turnaround_time = proc[i].waiting_time + proc[i].burst_time;
    }

    // Display results
    printf("Process\tBurst Time\tWaiting Time\tTurn Around Time\n");
    for (i = 0; i < n; i++) {
        total_wt += proc[i].waiting_time;
        total_tat += proc[i].turnaround_time;
        printf("%d\t\t%d\t\t%d\t\t%d\n", proc[i].pid, proc[i].burst_time, proc[i].waiting_time, proc[i].turnaround_time);
    }

    // Display averages
    printf("Average waiting time is: %.2f\n", total_wt/n);
    printf("Average Turn around Time is: %.2f\n", total_tat/n);

    return 0;
}
```

```
liveuser@localhost-live:~$ gcc sjf.c -o sjf
liveuser@localhost-live:~$ ./sjf
Enter the number of processes: 4
Enter the burst time of the processes:
8 4 9 5
Process Burst Time      Waiting Time    Turn Around Time
1               4               0               4
3               5               4               9
0               8               9               17
2               9               17              26
Average waiting time is: 7.50
Average Turn around Time is: 14.00
```

PRIORITY Scheduling

```
liveuser@localhost-live:~$ cat >priority.c
// File: priority.c
#include <stdio.h>

struct Process {
    int pid;
    int burst_time;
    int waiting_time;
    int turnaround_time;
    int priority;
};

int main() {
    int n, i, j;
    float total_wt = 0, total_tat = 0;

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

    struct Process proc[n];

    // Read the process details
    printf("Enter the burst time and priority of the processes: \n");
    for (i = 0; i < n; i++) {
        proc[i].pid = i;
        printf("Process %d - Burst Time: ", i);
        scanf("%d", &proc[i].burst_time);
        printf("Process %d - Priority: ", i);
        scanf("%d", &proc[i].priority);
    }

    // Sort processes based on priority (lower value has higher priority)
    for (i = 0; i < n-1; i++) {
        for (j = i+1; j < n; j++) {
            if (proc[i].priority > proc[j].priority) {
                struct Process temp = proc[i];
                proc[i] = proc[j];
                proc[j] = temp;
            }
        }
    }

    // Calculate waiting time and turnaround time
    proc[0].waiting_time = 0;
    proc[0].turnaround_time = proc[0].burst_time;

    for (i = 1; i < n; i++) {
        proc[i].waiting_time = proc[i-1].waiting_time + proc[i-1].burst_time;
        proc[i].turnaround_time = proc[i].waiting_time + proc[i].burst_time;
    }

    // Display results
    printf("Process\tBurst Time\tPriority\tWaiting Time\tTurn Around Time\n");
    for (i = 0; i < n; i++) {
        total_wt += proc[i].waiting_time;
        total_tat += proc[i].turnaround_time;
        printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", proc[i].pid, proc[i].burst_time, proc[i].priority, proc[i].waiting_time, proc[i].turnaround_time);
    }
```

```
    }

    // Display averages
    printf("Average waiting time is: %.2f\n", total_wt/n);
    printf("Average Turn around Time is: %.2f\n", total_tat/n);

    return 0;
}
liveuser@localhost-live:~$ gcc priority.c -o priority
liveuser@localhost-live:~$ ./priority
Enter the number of processes: 3
Enter the burst time and priority of the processes:
Process 0 - Burst Time: 4
Process 0 - Priority: 2
Process 1 - Burst Time: 6
Process 1 - Priority: 1
Process 2 - Burst Time: 8
Process 2 - Priority: 3
Process Burst Time      Priority        Waiting Time    Turn Around Time
1               6               1               0               6
0               4               2               6               10
2               8               3               10              18
Average waiting time is: 5.33
Average Turn around Time is: 11.33
liveuser@localhost-live:~$
```

ROUND ROBIN (RR) Scheduling

```
liveuser@localhost-live:~$ cat > round_robin.c
// File: round_robin.c
#include <stdio.h>

struct Process {
    int pid;
    int burst_time;
    int remaining_time;
    int waiting_time;
    int turnaround_time;
};

int main() {
    int n, quantum, i, t = 0;
    float total_wt = 0, total_tat = 0;

    // Get the number of processes and time quantum
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    printf("Enter the time quantum: ");
    scanf("%d", &quantum);

    struct Process proc[n];

    // Read the burst times and process IDs
    for (i = 0; i < n; i++) {
        proc[i].pid = i;
        printf("Enter burst time for process %d: ", i);
        scanf("%d", &proc[i].burst_time);
        proc[i].remaining_time = proc[i].burst_time;
    }

    // Process Round Robin
    while (1) {
        int done = 1;
        for (i = 0; i < n; i++) {
            if (proc[i].remaining_time > 0) {
                done = 0;
                if (proc[i].remaining_time > quantum) {
                    proc[i].remaining_time -= quantum;
                    t += quantum;
                } else {
                    t += proc[i].remaining_time;
                    proc[i].waiting_time = t - proc[i].burst_time;
                    proc[i].turnaround_time = proc[i].waiting_time + proc[i].burst_time;
                    proc[i].remaining_time = 0;
                }
            }
        }
        if (done == 1) break;
    }

    // Display results
    printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (i = 0; i < n; i++) {
        total_wt += proc[i].waiting_time;
        total_tat += proc[i].turnaround_time;
        printf("%d\t\t%d\t\t%d\t\t%d\n", proc[i].pid, proc[i].burst_time, proc[i].waiting_time, proc[i].turnaround_time);
    }

    // Display averages
    printf("Average waiting time is: %.2f\n", total_wt/n);
    printf("Average Turnaround Time is: %.2f\n", total_tat/n);

    return 0;
}
```

```
liveuser@localhost-live:~$ gcc round_robin.c -o round_robin
liveuser@localhost-live:~$ ./round_robin
Enter the number of processes: 4
Enter the time quantum: 4
Enter burst time for process 0: 5
Enter burst time for process 1: 7
Enter burst time for process 2: 3
Enter burst time for process 3: 6
Process Burst Time      Waiting Time    Turnaround Time
0               5               11              16
1               7               12              19
2               3               8               11
3               6               15              21
Average waiting time is: 11.50
Average Turnaround Time is: 16.75
```