Scheduling Algorithms

Session: 2021 – 2025

Submitted by:

Muhammad Yaqoob 2021-CS-118

Supervised by:

Ms. Abqa Javed

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Contents

FCFS

```
~$ ./fcfs
Enter the number of processes: 3
Enter the arrival time and burst time of each process:
Arrival time of process 1: 3
Burst time of process 1: 2
Arrival time of process 2: 4
Burst time of process 2: 1
Arrival time of process 3: 5
Burst time of process 3: 2
Process AT      BT      CT      TAT     WT
1       3       2       5       2       0
2       4       1       6       2       1
3       5       2       8       3       1
Average Waiting Time: 0.67
Average Turnaround Time: 2.33
~$
```

Figure 1: FCFS

```
#include <stdio.h>

int main() {
    int n, i;
    float avgWT = 0, avgTAT = 0;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    int AT[n], BT[n], CT[n], TAT[n], WT[n];
    printf("Enter the arrival time and burst time of each process:\n");
    for (i = 0; i < n; i++) {
        printf("Arrival time of process %d: ", i+1);
        scanf("%d", &AT[i]);
        printf("Burst time of process %d: ", i+1);
        scanf("%d", &BT[i]);
    }
    // Calculate completion time for each process
    CT[0] = AT[0] + BT[0];
    for (i = 1; i < n; i++) {
        if (CT[i-1] < AT[i]) {
            CT[i] = AT[i] + BT[i];
        } else {
            CT[i] = CT[i-1] + BT[i];
        }
    }
    // Calculate turnaround time and waiting time for each process
```

```
    for (i = 0; i < n; i++) {
        TAT[i] = CT[i] - AT[i];
        WT[i] = TAT[i] - BT[i];
        avgTAT += TAT[i];
        avgWT += WT[i];
    }
    // Calculate average turnaround time and average waiting time
    avgTAT /= n;
    avgWT /= n;
    // Display results
    printf("Process\tAT\tBT\tCT\tTAT\tWT\n");
    for (i = 0; i < n; i++) {
        printf("%d\t%d\t%d\t%d\t%d\t%d\n", i+1, AT[i], BT[i], CT[i], TAT[i], WT[i
    ]);
    }
    printf("Average Waiting Time: %.2f\n", avgWT);
    printf("Average Turnaround Time: %.2f\n", avgTAT);
    return 0;
}
```

Priority Scheduling Algorithm

```
~$ ./psaFinal22
Enter the number of processes: 4
Enter the arrival time, burst time and priority for process 1: 3 4 5
Enter the arrival time, burst time and priority for process 2: 3 6 2
Enter the arrival time, burst time and priority for process 3: 3 4 5
Enter the arrival time, burst time and priority for process 4: 6 7 3

Process Arrival Time    Burst Time      Priority        Completion Time Turnaround Time Waiting Time
2       3               6               2               9                       6                       0
4       6               7               3               16                      10                      3
1       3               4               5               20                      17                      13
3       3               4               5               24                      21                      17

Average Waiting Time: 8.25
Average Turnaround Time: 13.50~$
```

FIGURE 2: Priority Scheduling Algorithm

```
#include<stdio.h>

int main() {
    int n, i, j, temp, sum = 0, total_wait = 0, total_turnaround = 0;
    float avg_wait, avg_turnaround;
    printf("Enter the number of processes: ");
    scanf("%d", &n);

    int process[n], arrival_time[n], burst_time[n], priority[n], waiting_time[n],
     turnaround_time[n], completion_time[n];

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

```
    printf("Enter the arrival time, burst time and priority for process %d:
", i+1);
    scanf("%d %d %d", &arrival_time[i], &burst_time[i], &priority[i]);
    process[i] = i+1;
}

// Sorting based on priority
for(i=0; i<n-1; i++) {
    for(j=0; j<n-i-1; j++) {
        if(priority[j] > priority[j+1]) {
            temp = priority[j];
            priority[j] = priority[j+1];
            priority[j+1] = temp;

            temp = burst_time[j];
            burst_time[j] = burst_time[j+1];
            burst_time[j+1] = temp;

            temp = arrival_time[j];
            arrival_time[j] = arrival_time[j+1];
            arrival_time[j+1] = temp;

            temp = process[j];
            process[j] = process[j+1];
            process[j+1] = temp;
        }
    }
}

// Calculation of waiting time, turnaround time, completion time
for(i=0; i<n; i++) {
    if(i == 0) {
        completion_time[i] = arrival_time[i] + burst_time[i];
    } else {
        if(arrival_time[i] > completion_time[i-1]) {
            completion_time[i] = arrival_time[i] + burst_time[i];
        } else {
            completion_time[i] = completion_time[i-1] + burst_time[i];
        }
    }
    turnaround_time[i] = completion_time[i] - arrival_time[i];
    waiting_time[i] = turnaround_time[i] - burst_time[i];

    total_wait += waiting_time[i];
    total_turnaround += turnaround_time[i];
}

// Calculation of average waiting time and average turnaround time
avg_wait = (float)total_wait / n;
avg_turnaround = (float)total_turnaround / n;

// Displaying the results
printf("\nProcess\tArrival Time\tBurst Time\tPriority\tCompletion Time\
tTurnaround Time\tWaiting Time\n");
for(i=0; i<n; i++) {
```

```
        printf("%d\t%d\t\t%d\t\t%d\t\t%d\t\t\t%d\t\t\t%d\n", process[i],
    arrival_time[i], burst_time[i], priority[i], completion_time[i],
    turnaround_time[i], waiting_time[i]);
    }
    printf("\nAverage Waiting Time: %.2f\nAverage Turnaround Time: %.2f",
    avg_wait, avg_turnaround);
    }
```

Shortest job first

```
~$ ./sjf
Enter the number of processes: 4
Enter the burst time for each process:
Burst time for process 1: 4
Burst time for process 2: 3
Burst time for process 3: 5
Burst time for process 4: 4
Process Burst Time      Waiting Time    Turnaround Time Completion Time
1       3               0               3               3
2       4               3               7               10
3       4               7               11              18
4       5               11              16              27
Average waiting time: 5.25
Average turnaround time: 9.25
~$
```

FIGURE 3: Shortest job first

```
#include<stdio.h>

void sjf(int n, int bt[], int wt[], int tat[]) {
    int i, j, temp, completion_time[n], smallest;

    // initialize the waiting time and completion time of all processes to 0
    for(i = 0; i < n; i++) {
        wt[i] = 0;
        completion_time[i] = 0;
    }

    // find the completion time for each process
    for(i = 0; i < n; i++) {
        smallest = i;
        for(j = i+1; j < n; j++) {
            if(bt[j] < bt[smallest]) {
                smallest = j;
            }
        }
        temp = bt[i];
        bt[i] = bt[smallest];
```

```
        bt[smallest] = temp;

        temp = completion_time[i];
        completion_time[i] = completion_time[smallest];
        completion_time[smallest] = temp;

        completion_time[i] = (i == 0) ? bt[i] : (completion_time[i-1] + bt[i]);
    }

    // find the waiting time and turnaround time for each process
    for(i = 0; i < n; i++) {
        tat[i] = completion_time[i];
        wt[i] = tat[i] - bt[i];
    }
}

int main() {
    int n, i;
    float avg_wt = 0, avg_tat = 0;
    printf("Enter the number of processes: ");
    scanf("%d", &n);

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

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

    sjf(n, bt, wt, tat);

    printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\tCompletion Time\n
    ");
    for(i = 0; i < n; i++) {
        printf("%d\t%d\t\t%d\t\t%d\t\t%d\n", i+1, bt[i], wt[i], tat[i], tat[i]+wt
    [i]);
        avg_wt += wt[i];
        avg_tat += tat[i];
    }

    avg_wt /= n;
    avg_tat /= n;

    printf("Average waiting time: %.2f\n", avg_wt);
    printf("Average turnaround time: %.2f\n", avg_tat);

    return 0;
}
```

Round Robbin

```
~$ ./rr2
Enter the number of processes: 4
Enter the time quantum: 3
Enter arrival time and burst time for process 1: 4 5
Enter arrival time and burst time for process 2: 4 6
Enter arrival time and burst time for process 3: 4 5
Enter arrival time and burst time for process 4: 7 6

Process AT      BT      CT      TAT     WT
P1      4       5       14      10      5
P2      4       6       17      13      7
P3      4       5       19      15      10
P4      7       6       22      15      9

Average Turnaround Time: 13.25
Average Waiting Time: 7.75
~$
```

FIGURE 4: Round Robbin

```
#include<stdio.h>

struct process {
    int pid;
    int arrival_time;
    int burst_time;
    int remaining_time;
    int waiting_time;
    int turnaround_time;
    int completion_time;
};

int main() {
    int n, tq, i, j, time = 0, sum_bt = 0, sum_tat = 0, sum_wt = 0;
    float avg_tat, avg_wt;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    printf("Enter the time quantum: ");
    scanf("%d", &tq);
    struct process p[n];
    for(i=0; i<n; i++) {
        printf("Enter arrival time and burst time for process %d: ", i+1);
        scanf("%d %d", &p[i].arrival_time, &p[i].burst_time);
        p[i].pid = i+1;
        p[i].remaining_time = p[i].burst_time;
        sum_bt += p[i].burst_time;
```

```
    }
    while(sum_bt > 0) {
        for(i=0; i<n; i++) {
            if(p[i].remaining_time > 0) {
                if(p[i].remaining_time <= tq) {
                    time += p[i].remaining_time;
                    p[i].completion_time = time;
                    sum_bt -= p[i].burst_time;
                    p[i].turnaround_time = p[i].completion_time - p[i].
    arrival_time;
                    p[i].waiting_time = p[i].turnaround_time - p[i].burst_time;
                    p[i].remaining_time = 0;
                }
                else {
                    time += tq;
                    p[i].remaining_time -= tq;
                }
            }
        }
    }
    printf("\nProcess\tAT\tBT\tCT\tTAT\tWT\n");
    for(i=0; i<n; i++) {
        printf("P%d\t%d\t%d\t%d\t%d\t%d\n", p[i].pid, p[i].arrival_time, p[i].
    burst_time, p[i].completion_time, p[i].turnaround_time, p[i].waiting_time);
        sum_tat += p[i].turnaround_time;
        sum_wt += p[i].waiting_time;
    }
    avg_tat = (float)sum_tat/n;
    avg_wt = (float)sum_wt/n;
    printf("\nAverage Turnaround Time: %.2f\nAverage Waiting Time: %.2f\n",
    avg_tat, avg_wt);
    return 0;
}
```