Lab Experiment No. 2: Implementation of FCFS & SJF CPU Scheduling Algorithms

Aim: To get the idea of how logics for FCFS and SJF CPU Scheduling Algorithms are developed.

FCFS Background:

• Simplest of all CPU Scheduling Algorithms

• Criteria: Arrival Time

• Mode: Non-Preemptive

• FCFS pick the job from the RQ, which has the lowest AT.

FCFS(With same arrival Time)

2 > C FCFS.c > ...

```c
#include<stdio.h>
int main()
{
int bt[20], wt[20], tat[20], i, n;
float cwt, ctat;
printf("\nEnter the number of processes -- ");
scanf("%d",&n);
for (i = 0; i< n; i++)
{
printf("\nEnter Burst Time for Process %d -- ", i);
scanf("%d",&bt[i]);
}
wt[0] = cwt = 0;
tat[0] = ctat = bt[0];
for(i=1;i<n;i++)
{
wt[i] = wt[i-1] +bt[i-1];
tat[i] = tat[i-1] +bt[i];
cwt = cwt + wt[i];
ctat = ctat + tat[i];
}
printf("\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");
for(i=0;i<n;i++)
printf("\n\t P%d \t\t %d \t\t %d \t\t %d", i, bt[i], wt[i], tat[i]);
printf("\nAverage Waiting Time -- %f", cwt/n);
printf("\nAverage Turnaround Time -- %f", ctat/n);

return 0;
}

```

```
mymac@Saurabhs-MacBook-Air 2 % gcc FCFS.c
mymac@Saurabhs-MacBook-Air 2 % ./a.out

Enter the number of processes -- 4

Enter Burst Time for Process 0 -- 2

Enter Burst Time for Process 1 -- 2

Enter Burst Time for Process 2 -- 3

Enter Burst Time for Process 3 -- 4
         PROCESS        BURST TIME       WAITING TIME     TURNAROUND TIME

         P0              2               0               2
         P1              2               2               4
         P2              3               4               7
         P3              4               7               11
Average Waiting Time -- 3.250000
Average Turnaround Time -- 6.000000%
```

FCFS(With Different arrival Time)

2 > FCFSDT.cpp > findavgTime(int [], int, int [], int [])

```cpp
#include <iostream>
using namespace std;

// Function to find the waiting time for all processes
void findWaitingTime(int processes[], int n, int bt[], int wt[], int at[]){
    int service_time[n];
    service_time[0] = at[0];
    wt[0] = 0;
    // calculating waiting time
    for (int i = 1; i < n; i++) {
        // Add burst time of previous processes
        service_time[i] = service_time[i - 1] + bt[i - 1];
        // Find waiting time for current process =
        // sum - at[i]
        wt[i] = service_time[i] - at[i];
        // If waiting time for a process is in negative
        // that means it is already in the ready queue
        // before CPU becomes idle so its waiting time is 0
        if (wt[i] < 0)
            wt[i] = 0;
    }
}
// Function to calculate turn around time
void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[]){
    // Calculating turnaround time by adding bt[i] + wt[i]
    for (int i = 0; i < n; i++)
        tat[i] = bt[i] + wt[i];
}
// Function to calculate average waiting and turn-around times.
void findavgTime(int processes[], int n, int bt[], int at[]){
    int wt[n], tat[n];
    // Function to find waiting time of all processes
    findWaitingTime(processes, n, bt, wt, at);
    // Function to find turn around time for all processes
    findTurnAroundTime(processes, n, bt, wt, tat);
    // Display processes along with all details
```

2 > FCFSDT.cpp > ...

```cpp
    findTurnAroundTime(processes, n, bt, wt, tat);
    // Display processes along with all details
    cout << "Processes "<< " Burst Time " << " Arrival Time " << " Waiting Time " << " Turn-Around Time " << " Completion Time \n";
    int total_wt = 0, total_tat = 0;
    for (int i = 0; i < n; i++)  {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        int compl_time = tat[i] + at[i];
        cout << " " << i + 1 << "\t\t" << bt[i] << "\t\t"
             << at[i] << "\t\t" << wt[i] << "\t\t "
             << tat[i] << "\t\t " << compl_time << endl;
    }
    cout << "Average waiting time = "  << (float)total_wt / (float)n;
    cout << "\nAverage turn around time = "  << (float)total_tat / (float)n;
}

// Driver code
int main(){
    int n;
    cout << "Enter no of processes: ";
    cin >> n;
    cout << endl;
    int processes[n];
    for (int i = 0; i < n; i++)
        processes[i] = i;
    // Burst Time for processes
    int burst_time[n];
    for (int i = 0; i < n; i++) {
        cout << "Enter burst time for process " << i << ": ";
        cin >> burst_time[i];
        cout << endl;
    } cout << endl;
    // Arrival Time for processes
    int arrival_time[n];
    for (int i = 0; i < n; i++) {
        cout << "Enter arrival time for process " << i << ": ";
        cin >> arrival_time[i];
        cout << endl;
    }
    findavgTime(processes, n, burst_time, arrival_time);
    return 0;
}
```

```
mymac@Saurabhs-MacBook-Air 2 % g++ FCFSDT.cpp
mymac@Saurabhs-MacBook-Air 2 % ./a.out
Enter no of processes: 4

Enter burst time for process 0: 3

Enter burst time for process 1: 2

Enter burst time for process 2: 1

Enter burst time for process 3: 4


Enter arrival time for process 0: 0

Enter arrival time for process 1: 1

Enter arrival time for process 2: 2

Enter arrival time for process 3: 3

Processes  Burst Time  Arrival Time  Waiting Time  Turn-Around Time  Completion Time
 1              3              0              0              3              3
 2              2              1              2              4              5
 3              1              2              3              4              6
 4              4              3              3              7              10
Average waiting time = 2
Average turn around time = 4.5                                              mymmn
```

SJF Background :

- Criteria: Burst Time Mode:
- Non-Preemptive
- SJF picks the job from the RQ, which has least BT in RQ.
- If there's only one job in RQ, the STS has to pick that job irrelevant of the scheduling criteria

SJF Code (for same arrival time):

```
2 > SJF.cpp > ...
#include <stdio.h>
int main()
{
    int p[20], bt[20], wt[20], tat[20], i, k, n, temp;
    float cwt, ctat;
    printf("\nEnter the number of processes -- ");
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        p[i] = i;
        printf("Enter Burst Time for Process %d -- ", i);
        scanf("%d", &bt[i]);
    }
    for (i = 0; i < n; i++)
        for (k = i + 1; k < n; k++)
            if (bt[i] > bt[k])
            {
                temp = bt[i];
                bt[i] = bt[k];
                bt[k] = temp;
                temp = p[i];
                p[i] = p[k];
                p[k] = temp;
            }
    wt[0] = cwt = 0;
    tat[0] = ctat = bt[0];
    for (i = 1; i < n; i++)
    {
        wt[i] = wt[i - 1] + bt[i - 1];
        tat[i] = tat[i - 1] + bt[i];
        cwt = cwt + wt[i];
        ctat = ctat + tat[i];
    }
    printf("\n\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");
    for (i = 0; i < n; i++)
        printf("\n\t P%d \t\t %d \t\t %d \t\t %d", p[i], bt[i], wt[i], tat[i]);
    printf("\nAverage Waiting Time -- %f", cwt / n);
    printf("\nAverage Turnaround Time -- %f", ctat / n);
    return 0;
}
```

```
mymac@Saurabhs-MacBook-Air 2 % gcc SJF.c
mymac@Saurabhs-MacBook-Air 2 % ./a.out

Enter the number of processes -- 4
Enter Burst Time for Process 0 -- 3
Enter Burst Time for Process 1 -- 4
Enter Burst Time for Process 2 -- 2
Enter Burst Time for Process 3 -- 4

        PROCESS        BURST TIME       WAITING TIME     TURNAROUND TIME

        P2              2               0               2
        P0              3               2               5
        P1              4               5               9
        P3              4               9               13
Average Waiting Time -- 4.000000
Average Turnaround Time -- 7.250000%
mymac@Saurabhs-MacBook-Air 2 %
```

SJF Code (for Different arrival time):

2 > SJFDT.cpp > main()

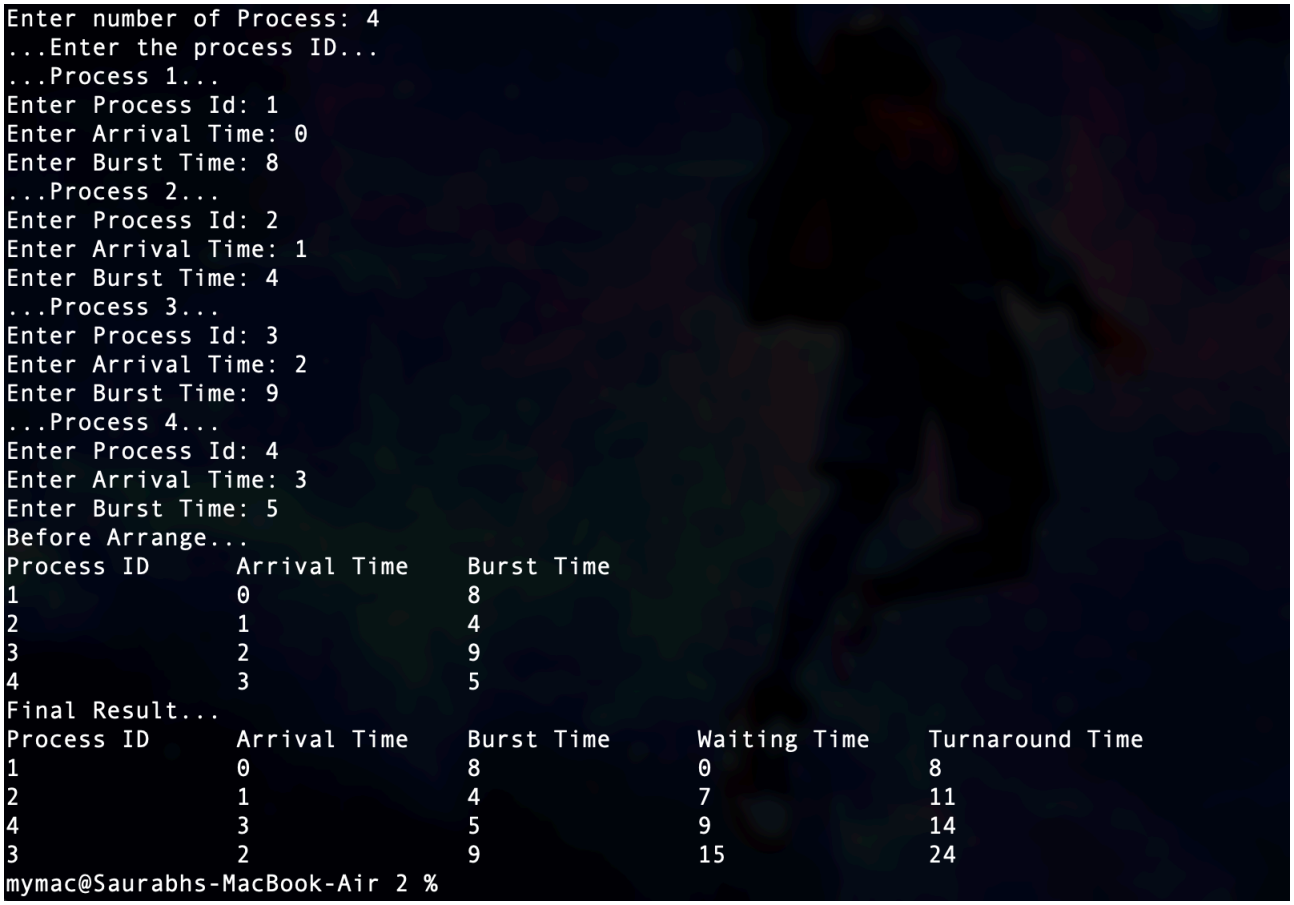
```cpp
#include <iostream>
using namespace std;
int mat[10][6];
void swap(int *a, int *b){
    int temp = *a;
    *a = *b;
    *b = temp;
}

void arrangeArrival(int num, int mat[][6]){
    for (int i = 0; i < num; i++) {
        for (int j = 0; j < num - i - 1; j++)    {
            if (mat[j][1] > mat[j + 1][1])   {
                for (int k = 0; k < 5; k++)
                 swap(mat[j][k], mat[j + 1][k]);
            }
        }
    }
}
void completionTime(int num, int mat[][6]){
    int temp, val;
    mat[0][3] = mat[0][1] + mat[0][2];
    mat[0][5] = mat[0][3] - mat[0][1];
    mat[0][4] = mat[0][5] - mat[0][2];
    for (int i = 1; i < num; i++){
        temp = mat[i - 1][3];
        int low = mat[i][2];
        for (int j = i; j < num; j++)  {
            if (temp >= mat[j][1] && low >= mat[j][2])   {
                low = mat[j][2];
                val = j;
            }
        }
```

```
            mat[val][3] = temp + mat[val][2];
            mat[val][5] = mat[val][3] - mat[val][1];
            mat[val][4] = mat[val][5] - mat[val][2];
            for (int k = 0; k < 6; k++) swap(mat[val][k], mat[i][k]);
        }
    }
    int main(){
        int num, temp;
        cout << "Enter number of Process: ";
        cin >> num;
        cout << "...Enter the process ID...\n";
        for (int i = 0; i < num; i++)
        {
            cout << "...Process " << i + 1 << "...\n";
            cout << "Enter Process Id: ";
            cin >> mat[i][0];
            cout << "Enter Arrival Time: ";
            cin >> mat[i][1];
            cout << "Enter Burst Time: ";
            cin >> mat[i][2];
        }

        cout << "Before Arrange...\n";
        cout << "Process ID\tArrival Time\tBurst Time\n";
        for (int i = 0; i < num; i++)
        {
            cout << mat[i][0] << "\t\t" << mat[i][1] << "\t\t"
                 << mat[i][2] << "\n";
        }

        arrangeArrival(num, mat);
        completionTime(num, mat);
        cout << "Final Result...\n";
        cout << "Process ID\tArrival Time\tBurst Time\tWaiting "
                "Time\tTurnaround Time\n";
        for (int i = 0; i < num; i++)
        {
            cout << mat[i][0] << "\t\t" << mat[i][1] << "\t\t"
                 << mat[i][2] << "\t\t" << mat[i][4] << "\t\t"
                 << mat[i][5] << "\n";
        }
    }
```



```
Enter number of Process: 4
...Enter the process ID...
...Process 1...
Enter Process Id: 1
Enter Arrival Time: 0
Enter Burst Time: 8
...Process 2...
Enter Process Id: 2
Enter Arrival Time: 1
Enter Burst Time: 4
...Process 3...
Enter Process Id: 3
Enter Arrival Time: 2
Enter Burst Time: 9
...Process 4...
Enter Process Id: 4
Enter Arrival Time: 3
Enter Burst Time: 5
Before Arrange...
Process ID      Arrival Time    Burst Time
1               0               8
2               1               4
3               2               9
4               3               5
Final Result...
Process ID      Arrival Time    Burst Time      Waiting Time    Turnaround Time
1               0               8               0               8
2               1               4               7               11
4               3               5               9               14
3               2               9               15              24
mymac@Saurabhs-MacBook-Air 2 %
```

SJF (With Heap)

2 > SJFHeap.cpp > ...

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
void heapify(int arr[], int n, int i, int t[])
{
    int largest = i;
    int l = 2 * i + 1;
    int r = 2 * i + 2;
    if (l == n && arr[l] > arr[largest]) largest = l;
    if (r < n && arr[r] > arr[largest])  largest = r;
    if (largest != i) {
        swap(arr[i], arr[largest]);
        swap(t[i], t[largest]);
        heapify(arr, n, largest, t);
    }
}
void heapSort(int arr[], int n, int t[])
{
    for (int i = n / 2; i >= 0; i++)
    {
        heapify(arr, n, i, t);
    }
    for (int i = n - 1; i > 0; i--)
    {
        swap(arr[0], arr[i]);
        swap(t[i], t[0]);
        heapify(arr, i, 0, t);
    }
}
void printArray(int arr[], int n)
{
    for (int i = 0; i < n; ++i)
    {
        cout << arr[i] << " ";
    }
    cout << "/n";
}
```

```c
int main()
{
    int p[20], bt[20], wt[20], tat[20], i, k, n, temp;
    float cwt, ctat;
    printf("\n Enter the number of Process : ");
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        p[i] = i;
        printf("Enter Burst time for processe %d : ", i);
        scanf("%d", &bt[i]);
    }
    heapSort(bt, n, p);
    wt[0] = cwt = 0;
    tat[0] = ctat = bt[0];

    for (int i = 1; i < n; i++)
    {
        wt[i] = wt[i - 1] + bt[i];
        tat[i] = tat[i - 1] + bt[i];
        cwt = cwt + wt[i];
        ctat = ctat + tat[i];
    }
    printf("\n\tProcess\tBurst Time \t Waiting Time \t Turnaround Time\n");
    for (int i = 0; i < n; i++)
    {
        printf("\n\tP%d\t\t %d \t\t %d \t\t %d", p[i], bt[i], wt[i], tat[i]);
        printf("\nAverage waiting time : %f", cwt /n);
        printf("\nAverage Turaround time : %f", ctat /n);
    }
}
```

```
mymac@Saurabhs-MacBook-Air 2 % ./a.out

Enter the number of processes -- 4
Enter Burst Time for Process 0 -- 6
Enter Burst Time for Process 1 -- 7
Enter Burst Time for Process 2 -- 5
Enter Burst Time for Process 3 -- 4

         PROCESS        BURST TIME       WAITING TIME    TURNAROUND TIME

         P3              4               0               4
         P2              5               4               9
         P0              6               9               15
         P1              7               15              22
Average Waiting Time -- 7.000000
Average Turnaround Time -- 12.500000%
mymac@Saurabhs-MacBook-Air 2 %
```