DHA SUFFA UNIVERSITY
Department of Computer Science
CS-2004L
Operating Systems
Fall 2020

LAB 05 – Schedulers and FCFS Scheduling Algorithm

OBJECTIVE(S)

- Understanding the concept of schedulers
- Understanding FCFS Scheduling Algorithm

Schedulers

A process migrates among the various scheduling queues throughout its lifetime. The operating system must select, for scheduling purposes, processes from these queues in some fashion. The selection process is carried out by the appropriate scheduler.

Long-Term Scheduler

Long-term scheduling involves selecting the processes from the storage pool in the secondary memory and loading them into the ready queue in the main memory for execution. This is handled by the long-term scheduler or job scheduler.

The long-term scheduler controls the degree of multiprogramming. It must select a careful mixture of I/O bound and CPU bound processes to yield optimum system throughput. If it selects too many CPU bound processes then the I/O devices are idle and if it selects too many I/O bound processes then the processor has nothing to do. An I/O-bound process is one that spends more of its time doing I/O than it spends doing computations. A CPU-bound process, in contrast, generates I/O requests infrequently, using more of its time doing computations.

Short-Term Scheduler

Short-term scheduling involves selecting one of the processes from the ready queue and scheduling them for execution. This is done by the short-term scheduler. A scheduling algorithm is used to decide which process will be scheduled for execution next by the short-term scheduler.

The short-term scheduler executes much more frequently than the long-term scheduler as a process may execute only for a few milliseconds.

DHA SUFFA UNIVERSITY
Department of Computer Science
CS-2004L
Operating Systems
Fall 2020

First Come First Serve Scheduling Algorithm

In the "First come first serve" scheduling algorithm, as the name suggests, the process which arrives first, gets executed first, or we can say that the process which requests the CPU first, gets the CPU allocated first.

- First Come First Serve, is just like **FIFO**(First in First out) Queue data structure, where the data element which is added to the queue first, is the one who leaves the queue first.
- It's **easy to understand and implement** programmatically, using a Queue data structure, where a new process enters through the **tail** of the queue, and the scheduler selects the process from the **head** of the queue.
- A perfect real life example of FCFS scheduling is **buying tickets at the ticket counter**.

Algorithm:

Aim: Write a C++ Program to Input processes from users and run them on FCFS basis.

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Set the waiting time (WT) of the first process as '0' and its burst time as its turn-around time

Step 5: For each process in the Ready Q calculate

Turn-around time (TaT) of Process(n) = Completion Time – Arrival Time

Waiting Time for Process(n) = Turn-around time – Burst Time

Step 6: Calculate

Average waiting time = Total (sum) waiting Time / Number of processes

Average Turnaround time = Total (sum) Turnaround Time / Number of processes

Step 7: Stop the process

First Come First Serve Scheduling Example

Example: Consider the following table:

Process no.	Arrival Time	Burst Time
P1	0	6
P2	2	1
P3	5	4
P4	6	3

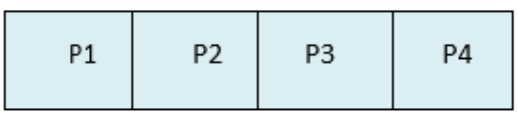
Find the average waiting time and average turnaround time using FCFS (First Come First Serve) algorithm.

Solution

Using first come first serve algorithm, ready queue and Gantt chart are:

Ready Queue: P1, P2, P3, P4

Gantt chart



0 6 7 11 14

Therefore,

Process No.	Arrival Time	Burst Time	Completion Time	Turnaround Time	Waiting Time
P1	0	6	6	6 - 0 = 6	**6 - 6 = 0**
P2	2	1	7	7 - 2 = 5	**5 - 1 = 4**
P3	5	4	11	11 - 5 = 6	**6 - 4 = 2**
P4	6	3	14	14 - 6 = 8	**8 - 3 = 5**

So,

Average Turnaround time = (6 + 5 + 6 + 8) / (4) = 6.25

Average Waiting time = (0 + 4 + 2 + 5) / (4) = 2.75

CODE:

```
// C++ program for implementation of FCFS
// scheduling
#include<iostream>
using namespace std;

// Function to find the completion time for all
// processes
void findCompletionTime(int process[], int n, int bt[], int ct[])
{
    //Calculating Completion time by adding
    //completion time of previous process with burst time of current process

    //Completion time of first process is same as burst time
    ct[0]=bt[0];

    for (int  i = 1; i < n ; i++ )
        ct[i] =  ct[i-1] + bt[i] ;

}
```

DHA SUFFA UNIVERSITY
Department of Computer Science
CS-2004L
Operating Systems
Fall 2020

```
// Function to find the Turn Around Time time for all
// processes
void findTurnAroundTime( int processes[], int n, int at[],
                         int tat[], int ct[])
{
    // calculating turn around time by subtracting
    // arrival time from completion time
    for (int  i = 0; i < n ; i++)
        tat[i] = ct[i] - at[i];
}

void findWaitingTime(int processes[], int n, int wt[],
                          int tat[], int bt[])
{
    // waiting time for first process is 0
    wt[0] = 0;

    // calculating waiting time by subtracting
    // burst time from turn around time
    for (int  i = 1; i < n ; i++ )
        wt[i] =  tat[i] - bt[i] ;
}

//Function to calculate average time and print table
void findavgTime( int processes[], int n, int bt[], int at[])
{
    int wt[n], tat[n], ct[n], total_wt = 0, total_tat = 0;

    findCompletionTime(processes, n, bt, ct );
```

DHA SUFFA UNIVERSITY
Department of Computer Science
CS-2004L
Operating Systems
Fall 2020

```
    //Function to find turn around time for all processes
    findTurnAroundTime(processes, n, at, tat, ct);
    //Function to find waiting time of all processes
    findWaitingTime(processes, n, wt, tat, bt);



    //Display processes along with all details
    cout << "Processes  "<< " Arrival Time  "<< " Burst time  "
        << " Completion time  "  << " Turn around time  " << " Waiting time\n";

    // Calculate total waiting time and total turn
    // around time
    for (int  i=0; i<n; i++)
    {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        cout << "   " << i+1 << "\t\t" << at[i] <<"\t    "
            << bt[i] <<"\t\t  " << ct[i] <<"\t\t  " << tat[i] <<"\t\t    "<< wt[i] <<endl;
    }

    cout << "Average waiting time = "
         << (float)total_wt / (float)n;
    cout << "\nAverage turn around time = "
         << (float)total_tat / (float)n;
}

// Driver code
int main()
{
    //process id's
    int processes[] = { 1, 2, 3,4};
    int n = 4;

    //Burst time of all processes
    int  burst_time[] = {6, 1, 4, 3};
    int  arrival_time[] = {0, 2, 5, 6};

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

Output:

```
Processes   Arrival Time   Burst time   Completion time   Turn around time   Wai
ting time
   1            0            6             6                 6                0
   2            2            1             7                 5                4
   3            5            4             11                6                2
   4            6            3             14                8                5
Average waiting time = 2.75
Average turn around time = 6.25
Process returned 0 (0x0)   execution time : 0.469 s
Press any key to continue.
```

DHA SUFFA UNIVERSITY
Department of Computer Science
CS-2004L
Operating Systems
Fall 2020

ASSIGNMENT # 05

1. Write a C++ program to implement the mechanism of a long term scheduler. Your long term scheduler should carefully select the I/O bound and CPU bound processes, keeping the ratio as 70 % CPU bound processes and 30% I/O bound processes.

SUBMISSION GUIDELINES

- Take a screenshot of each task (code and output).
- Place all the screenshots in a single word file labeled with Roll No and Lab No. e.g. **'cs172xxx_Lab01'.**
- Convert the file into PDF.
- Make a folder labeled with Roll No, place the pdf and .cpp/.java/.sh files in this folder
- Submit the file at LMS