# CSCI 3453: Operating System Concepts
# Lab Assignment # 3

**Due Date:** May 7, 2020 @ 11:55 PM

## Goal of this programming assignment

The primary goal of this lab assignment is to understand various CPU scheduling algorithms, gain some experience building a simulator for CPU scheduling algorithm and to evaluate their performances.

## Requirements

1) Programming language: You must use either C or C++ to develop your program.
2) Running Environment: Your program should be compiled at CSEGRID and be able to be tested without errors.

## The implementation details

Write a program to simulate a CPU scheduler which selects a process from a ready queue and executes the process by using the given scheduling algorithm, display its actives and evaluate its performance based on measurements such as average turn-around time, average waiting time, and total number of context switching. When a process is scheduled, the simulator will simply print what the process is running at what time, collecting data and producing Gantt Chart-like outputs.

## Specifications

You have to implement FCFS (First Come First Serve), SRTF (Shortest Remaining Task First – preemptive), and RR (Round Robin) scheduling algorithms.

## Assumptions

Use the following assumptions when you design and implement your CPU simulator.

- There is only one CPU
- All processes perform only CPU operations.
- All processes have the same priority.
- The newly arrived process is directly added to the back of the ready queue.
- We use only ready queue for this simulation.
- Context switching is done in 0 ms.
- Context switch is performed only when a current process is moved to the back of ready queue.
- All time are given milliseconds.
- Use FCFS policy for breaking the tie.

**Measurements and Evaluation**

You program should collect the following information about each process:

- Time of completion
- Waiting time
- Turn around time
- No. of Context Switching

You program should calculate the following information using the collected data:

- Average CPU burst time
- Average waiting time
- Average turn around time
- Average response time
- Total number of Context Switching performed

**Simulator Input**

Process information (assume a maximum of 100 processes) will be read from a text file. The information for each process will include the following fields:

- pid: a unique numeric process ID.
- arrival time: arrival time for a process in ms.
- CPU burst time: the CPU time requested by a processes

An example of input file:

```
1  0  10
2  1   2
3  2   9
4  3   5
```

You can assume that all time values are integers and pids provided in the input file are unique.

Name of the input file, scheduling algorithm and quantumsize (for RR algorithm) will be provided as command line argument as follows:

```
myscheduler input_file FCFS
myscheduler input_file SRTF
myschedular input_file RR 4
```

**Sample Output**

The simulator will display both the history of process, like like Gantt Chart, and collected data. The expected output format is as follows:

```
************************************************************
************ Scheduling algorithm : FCFS *******************
************************************************************

P1                  P2  P3                P4
* * * * * * * * * * * * * * * * * * * * * * * * * *
(each star represents one ms)
```

| pid | arrival | CPU-burst | finish | waiting time | turn around | No. of Context |
|---|---|---|---|---|---|---|
| 1 | 0 | 10 | 10 | 0 | 10 | 0 |
| 2 | 1 | 2 | 12 | 9 | 11 | 0 |
| 3 | 2 | 9 | 21 | 10 | 19 | 0 |
| 4 | 3 | 5 | 26 | 18 | 23 | 0 |

```
Average CPU burst time = 6.50 ms
Average waiting time = 9.25 ms
Average turn around time = 15.75 ms,
Total No. of Context Switching Performed = 0
```

**Deliverables**

1. Program source codes includes all source program files, Makefile, and Readme
2. Sample Output (Screen shot).
3. Please include your name and due date in comments at the top of each of your C/ C++ code file.

**How to turn in my work**

Please do the followings when you submit your programming assignment.

- Create a tar or zip file that contains your written screen shot, source code, makefile and readme. DO NOT INCLUDE EXECUTABLES AND OBJECT FILES.
- Please use the following convention when you create a tar or zip file FirstName_LastName_Lab3.tar or FirstName_LastName_Lab3.zip
- Upload your tar file into Canvas.

**Rubric for Grading:**

| **Description** | **Points** |
|---|---|
| Implementation of scheduling algorithms (correctness) - FCFC | 50 |
| Implementation of scheduling algorithms (correctness) - SRTF | 50 |
| Implementation of scheduling algorithms (correctness) - RR | 50 |
| Formatting of output | 50 |
| Code is well formatted | 10 |
| Code is well documented. | 10 |
| **Total Points** | **220** |