CSE 303 – Fundamentals of Operating Systems

HW#2– FCFS CPU Scheduling with IO

Due Date: Wednesday, 28 December 2022, @23:59 pm (midnight) No extension!!!

In this assignment, you will implement a program to show the performance of FCFS scheduling algorithm with I/O burst. Your program should get a file (e.g., "jobs.txt") as the command-line input, and read the contents of the file. This file contains a set of processes. For example, consider the file with the following content:

```
346:(45,15);(16,20);(80,10);(40,-1)
2547:(15,10);(60,15);(90,10);(85,20);(20,-1)
49:(30,15);(40,20);(5,15);(10,15);(15,-1)
```

In this example we have 3 processes, each process is represented in a separate line. The general format of a line is as follows:

<process-id>:(< cpu-burst$_1$, io-burst$_1$>);(< cpu-burst$_2$, io-burst$_2$>);...(< cpu-burst$_i$, io-burst$_i$>)

The first token is the unique process id. After process-id you have a colon (:) delimiter. Then you will see a list of tuples separated by semicolons (;). Each tuple in parentheses indicates the next cpu-burst and io-burst lengths of the process. The cpu and io burst length in terms of milliseconds. If the last io-burst is -1, then it means that the process terminates without making an I/O.

- Note that this input is just an example, I may use a different input file having a different content for testing your codes, but the format of the file will be same. (Be careful on duplicate last line issue when reading from the input file)
- You will assume that ;
  - all the jobs arrive at the same time (t=0), the order of arrival is the same as the order of process-ids (i.e., smaller ids arrive earlier).
  - the process never waits at the device queues and I/O starts immediately.
  - The IDLE process is executed if no other processes are ready to run.

First Come First Served (FCFS) Algorithm

Implement FCFS scheduling policy. You should print the following

a. **Average turnaround time**: The average of the turnaround times of **all process**
b. **Average waiting time**: The average of the total waiting time for **all processes**.
c. **The number of times that the IDLE process executed.**
d. **Print a HALT message in the end of processing.**

Submission Guidelines: You must implement this project in either C (120 points) or Java (over 100 points). In both cases you must provide a Makefile file for compiling your homework (if not provided then -10 points, if it does not work -20 points).

Make sure that compilation is **error-free (if not -40)** and **warning-free (if not -10)**.
In the beginning of codes write the name and student id of the author as a comment

Upload your files to classroom. For C: You should rename your file as "yourstudentid.c", Your codes will be compiled using gcc (version 9.4.0) on an Ubuntu 20.04.2 as follows:

```
#gcc --std=c99 -Wall -O2 -o yourstudentid.exe yourstudentid.c
```

Make the necessary changes in your Makefile to support the above parameters.

and executed as follows:

```
#./yourstudentid.exe samplejobs.txt
```

CSE 303 – Fundamentals of Operating Systems

HW#2– FCFS CPU Scheduling with IO

For Java, javac (11.0.17) will be used on Ubuntu 20.04.2. File naming conventions will be similar, however you should add a letter “c” in front of your student id.

```
#javac cyourstudentid.java   (e.g. c20150808123.java)
#java cyourstudentid samplejobs.txt
```

- Do not assume the filename parameter(-5 points)
- Do not provide unnecessary output e.g. debug info in your submission (-5 points)
- If your application freeze during the operations (-40 points)
- Source code without comments (-10 points)
- For java users, be careful when you use package. It may create issues during the compile steps (in case -5 points)