COMP 3413 Fall 2021 Assignment 1

Due Date: Friday, Sep. 17, 11:59 PM

Notes and Style

- must be written in C. No C++.
- your assignment code must be handed in electronically using the D2L drop box. Try early and re-submit later to avoid last minute D2L confusion
- Your program should be compilable with: gcc -Wall -o a1 a1.c If it doesn't successfully make an executable, your assignment **will immediately lose 50%**
- Your program must run. Programs that do not run, or segfault before working at all, **will immediately lose 50%**. Save early and often, and use version control. Submit often, only submit working versions.
- Your program must compile with –Wall (see above), **with no warnings showing**. You will lose marks for warnings
- Include a "listing.txt" file, a single .txt file with all your source files concatenated into it (e.g., in unix: cat *.h *.c > listing.txt). Good header text (at the start of each file) is imperative here.
- Use common-sense commenting. Provide function headers, comment regions, etc. No particular format or style is required. If some code's function is not immediately obvious from variable or function names, add comments.
- Error checking is extremely important in real world OS work, and you must do it rigorously. Error handing, however, is hard in C. Try to handle gracefully when you can, but for hard errors (out of memory, etc.), hard fail (exit) is OK (with good error messages).
- **Your assignment will be marked on FCS machines**

In the early days of computers, operating systems serviced users by scheduling jobs in the order in which they arrived (started). This is a First-Come First-Serve (FCFS) algorithm.

The downside of First-Come First-Serve (FCFS) is that you are at the mercy of the order in which jobs arrive. A large job at the beginning has everyone waiting and becoming irritated. A solution to this is to try and help people with small jobs get in and out quickly. This is called Shortest-Job-First (SJF). In this algorithm as jobs arrive they are added to the run list based on the duration of the job – shortest duration first! When the duration of multiple jobs are the same then the arrival time decides who goes first.

To properly implement SJF you also need "pre-emption" – the concept of periodically stopping a job to take away the CPU and decide who is the next eligible process to run (we will cover this in detail in class at a later date). This is needed to prevent large jobs that acquire the CPU and start computing to only hold the CPU while new short jobs arrive and have to wait. To implement pre-emption, you only execute a job for 1 period and then you add it back into the run list.

For example if we have 1 CPU:

User	Process	Arrival	Duration
Jim	A	2	5
Mary	B	2	3
Sue	D	5	5
Mary	C	6	2

This would result in:

Time	Job
2	B
3	B
4	B
5	A
6	C
7	C
8	A
9	A
10	A
11	A
12	D
13	D
14	D
15	D
16	D
17	IDLE

Summary

Jim	12
Mary	8
Sue	17

Write a C program that will read in job requests and print out the corresponding job schedule according to a SJF algorithm as above. The input format is each line entry contains a job separated by a tab. The first line of

input is ignored as the header. The output format is two tables. First table is the running time and the job currently executing (tab separated). The second table is a summary with the user name (in the order in which jobs arrive) and the time when their last job is completed.

Note: the input and output of your program is specified such that the marker can automatically test your submissions. Please follow the specifications exactly. Input will be read via stdin (e.g., keyboard). Output should be printed to stdout (i.e., printf, not to a file).

Your program will be tested with command line commands like this:
cat input1.txt | a1

Or

./a1 < input1.txt

You can find sample input and output files with the assignment on D2L. Be sure your program reads and prints the proper input and output formatting.