CMPT 300 D200
Assignment 1

Notes:

1. You can do this assignment individually or in a team of two. If you are doing it in a group, then only one submission per group is required.
2. You may submit multiple times until the deadline. Grade penalties will be imposed for late submissions (see the course outline for the details).
3. Always plan before coding.
4. All the codes in this lab must be done using C language only. No other languages should not be used.
5. Use function-level and inline comments throughout your code. We will not be specifically grading documentation. However, remember that you will not be able to comment on your code unless sufficiently documented. Take the time to document your code as you develop it properly.
6. We will be carefully analyzing the code submitted to look for plagiarism signs, so please do not do it! If you are unsure about what is allowed, please talk to an instructor or a TA.

Aim: The assignment aims to refresh your knowledge of Linux, C, and Makefiles.

Makefile tutorial
https://www.cs.swarthmore.edu/~newhall/unixhelp/howto_makefiles.html#creating

Question 1 - [20 marks]
Write a program to display all the running processes on your system, along with their respective process ID.

A process is a running instance of a program. A process has many attributes like *Process ID, process State, program Counter, etc*. PID refers to process ids commonly used by operating system kernels, such as Linux, Unix, macOS and Windows. It is a unique ID that is automatically assigned to each process when it is created. For example, the 'init' process has a fixed PID of '1'.

Hint: To solve this, you have to find process id numbers from the '/proc' directory. Each sub-directory in the '/proc' directory with a numeric name is the process's id.

Steps:

1) In your Ubuntu system, create a new directory.

2) Using a text editor (such as gedit or vi), create a new C file called "`my_current_processes.c`". This program will contain the main function, which will display the currently running processes and their respective process ids.
    a. Read the process id from the "/proc" directory. Then read the process name from the "`/proc/<process_id>/cmdline`" file and display them.
    b. To read the directory name from the "/proc" directory, you need to include "dirent.h".

3) Using your text editor, create a Makefile (it must be called Makefile, with no extension). A Makefile defines how the code will be built (compiled). When creating a Makefile, be aware of the fact that it is very sensitive to tabs and spaces.

4) Run ***make***
   Open the terminal and go to the directory that contains `my_current_processes.c` file and type command 'make.' It should produce an executable file without any errors. If you have errors, fix them.

5) Run **'./my_current_processes'** to run the program

Test cases:

The **"ps aux"** command is used to list the currently running processes on your system. You can run this command on your terminal to view them.

- Run **"ps aux"** to check if the process ID and name that your program is generating matches the actual process.

Note: *You can look for a specific process name by using the grep command in your terminal. For example:* ***"ps aux | grep <process name>"***

- Kill a process and rerun your program to see that your program does not output the killed process.
Note: *You can use the* ***Kill*** *command to kill a process using its process ID and* ***pkill*** *command to kill a process using its process name*

Steps:

1. Open a browser, for example, Mozilla Firefox. Once you do this, you will see a process called "*firefox*" when you run **"`ps aux | grep firefox.`"**

2. Run **'./my_current_processes'** to see that your program also outputs the firefox process.
3. Run **"pkill firefox"** to kill all the firefox processes
4. Run **'./my_current_processes'** again to see that all firefox processes are actually killed.

Question 2. [20 marks]
Write a C program that will print the memory utilization of your system.

A program must be loaded into the main memory before it runs. CPU reads the instructions and data from the main memory, executes them, and stores their output back to the main memory. In general, modern computers have 1GB to 128GB of main memory (RAM). It is sufficient for running most of the processes. Modern operating systems load multiple processes in the main memory. To find out the memory utilization, you need to use the following formula:

$$Memoryutilization = \frac{(total - free - buffers - cached - slab)}{total} \times 100$$

Your C program will use the above equation to prints the memory utilization of your system.

Steps:

1) In your Ubuntu system, create a new directory.
2) Using a text editor (such as gedit or vi), create a new C file called 'my_memory_util.c'.
3) Read the memory-related information from file '/proc/meminfo.'
4) Use the above equation and display memory utilization.
5) Create a Makefile
6) Run *make*
7) Type './my_memory_util' to run the program

Test cases:

- Open multiple tabs in your browser and notice if the memory utilization increases. It should increase.
- Kill a few processes (for example, all the browser processes) using the same technique from the previous question and notice if the memory utilization decreases. It should decrease.

Question 3. [10 marks]
Write a C program to display the operating system's name and kernel version.

Linux is the family of operating systems that are based on the Unix operating system. For example, Ubuntu, Debian, RedHat, Fedora, etc., are Linux-based operating systems. Every operating system has its name and version. You need to write a C program to figure out and display the installed operating system's name and version along with the kernel version.

Hint: The name and version of your operating system are stored in '/etc/os-release' file, and the kernel version id is stored in '/proc/version' file.

Steps:

1) In your Ubuntu system, create a new directory.
2) Using a text editor (such as gedit or vi), create a new C file called 'my_version.c'.
3) Read and display the operating name and version from '/etc/os-release' and kernel version from '/proc/version.'
4) Create a Makefile
5) Run *make*
6) Type' ./version' to run the program

Question 4. [2 marks]

For each question, please specify how long you spend on it and what you learned. Was the question reasonable? (This question is worth marks, so please do it!)

Submission Instructions:

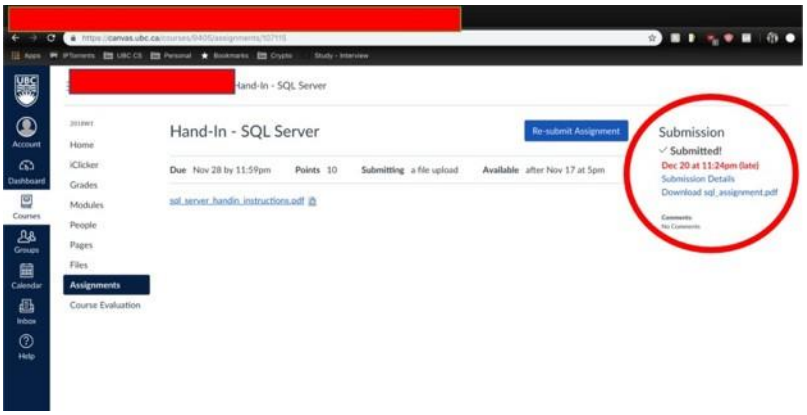
Submit a ZIP file of your assignment folder. Your submission is composed of the following files, named exactly as mentioned in the instructions:

- **Source code**
  - `my_current_processes.c`
  - `my_memory_util.c`
  - `my_version.c`

- **Makefile**: the Makefile provides the following functionality:
    - **all**: compiles your program (this is the default behaviour), producing an executable file named same as the C file.
    - **clean**: deletes the executable file and any intermediate files (.o, specifically)

- **README.txt**: documentation
    - Include your name (and team member name) and assignment number at the top of the file
    - Write down any resources that you consulted (URLs).
    - Write down the name of the class members with whom you have discussed your solution.
    - Mention the claimed late days for the assignment.
    - The answer to Question 4.

After submitting the file on canvas, you should be able to see the below message:



Grading Criteria

- Correctness (90%): passes all of the (hidden) unit tests
- Makefile (5%): satisfies the make file requirements
- README.txt (5%): contains the required information