Homework 1 - Exploring Unix and OS Processes

You may work with a partner of your choice on this assignment or individually. Either way make sure your name(s) are clearly stated at the top of your submission file, as well as naming the file appropriately.

Deliverables: upload a file named `username.ext` or if you work with a partner `username1_username2.ext` where `username` is your Carleton email username and `ext` is whatever type of file you submit. It must be something that can be opened on the standard lab machines. For example my submission might be `sgoings.rtf`.

Note: This is an exploratory assignment, and I am more interested in seeing that you made an effort to reason about how the OS works than that you get all of the right answers. You will receive full credit for well thought-out answers even if they aren't exactly correct.

1. Mac OS X is built largely on top of Unix (we will ignore that Mach thing for now). Historically, Unix users have used command-line interfaces rather than GUIs, although current versions of Unix come with both. We will first become familiar with simple Unix commands in the Terminal application.

Review the following Unix commands. If you are unsure what a command does, you can type "man <command>", substituting in the name of the command for "<command>". (man is short for manual.)

i. cd
ii. cp
iii. mv
iv. rm
v. mkdir
vi. pwd
vii. chmod
viii. more
ix. tail
x. wc
xi. diff

Give the command(s) to complete the following operations:

a. change the permissions of a file named *test* so that anyone could write to it.
b. output the number of lines (and only the number of lines) in the file *test*
c. output just the last 3 lines in the file *test*

2. Next, we will explore the behavior of some programs and begin to see the difference between user programs and kernel programs. To do this, we will use the Activity Monitor application. Its icon looks like

You can find it in the Applications/Utilities folder, or use a handy mac trick - open spotlight with the keyboard shortcut *cmd-space* and then start typing *activity monitor*. Spotlight will most likely auto-complete correctly after just the first few letters. You can open anything this way on a mac, applications, files, folders, etc. Many other handy mac shortcuts can be found here: http://www.danrodney.com/mac/

Activity Monitor displays information about running processes. Below the list of processes you will see some summary information. Above the processes are several tabs that control what is shown in that summary and which info columns are displayed (by default the CPU tab is open). You can change the columns displayed no matter which tab you are in by going to the *View->Columns* menu and selecting or deselecting any given column.

For this question, I would like you to explore what your computer is spending its time and memory doing. Make sure you are seeing all processes by checking the setting in the *View* menu.

First open at least 3 typical applications you use
- e.g. a text editor, browser, chat program, etc.

a. Make sure the CPU tab is selected and answer the following questions
    a. How many processes are running in total?
    b. How many processes is just the user *root* running (*root* is the user name for the kernel (OS) processes).
    c. What % CPU is used by User processes vs. System processes?

    Sort the processes by % CPU (with highest at the top)

    d. What are the first 3 processes and what % CPU is each using?
    e. Do the order of the processes change much? Why or why not?
    f. Pick one process from the entire list that you don't recognize and find out what it does – give the name of the process and a 1-sentance description.

b. Now click on the Memory tab and make sure both the columns *Memory* and *Real Memory* are showing if they aren't already and that the processes are sorted by the *Memory* column.
    a. What are the top 3 processes on the list and how much memory is each using according to the *Memory* column?
    b. Do the order of the processes change much? Why or why not?
    c. You will see that *Real Mem* >= *Memory* for every process. Give at least 1 idea of what the difference could be in how these 2 columns are measuring memory use.
    d. How much memory does this computer have?
    e. How much of that memory is being used?
    f. Do a little research and briefly explain what is meant by app, wired, and compressed memory.

c. Look at the C program I have provided in the file *test.*c. Basic syntax is very similar in C and Java, so it should be clear that this program simply runs an infinite loop. Compile this program using the command

```
gcc –o testexe test.c
```

gcc is a C language compiler, and the –o option sets the name of the executable the compiler will create. If you forget the –o, an executable with the default name *a.out* will be created, and if a file named *a.out* already exists it will be overwritten. To run the created executable, enter `./testexe` on the command line. The ./ simply tells terminal that you want to run the file *testexe* in the current directory as a command as opposed to searching for an existing command named *testexe* in the normal command libraries. Answer the following questions using the program *testexe.* Make sure the CPU tab is selected in Activity Monitor.

   a. What does activity monitor report in the % CPU column for your program while running? Note you may have to wait several seconds for your program to show up in the activity monitor list.
   b. With your original program still running, run another instance of your program (*cmd-n* will open a new window of whatever application you're in, for example a new terminal window). What does the % CPU column show for each of the 2 instances of your program now? How is this possible?
   c. Start 3 more instances of your program so that you have 5 running total, now what is the % CPU of each? Try to do something else significant on your computer (e.g. start a new application, open and save a file, etc.), do you notice any decrease in responsiveness and if so how significant is it?

d. Look at the program I have provided in the file *test.py*. Dynamic memory allocation is one of the things that is very different in C as compared to Java or Python, so we will use Python for this for now. Select the Memory tab in Activity Monitor and answer the following questions using the *test.py* program.
   a. Run the program and immediately start watching its memory usage in activity monitor. Give a general description of what happens and why.
   b. Stop the program if you haven't yet and wait several seconds for memory to reset. Now quickly start 5 instances of the program and watch what happens in memory. Again give a general description of what happens.
   c. Wait until the "Swap Used" value starts increasing, then try to do something else significant on your computer (e.g. start a new application, open and save a file, etc.), do you notice any decrease in responsiveness and if so how significant is it?