# CSCI 360 - Lab 1 Report

Sami Al-Qusus
Sept 17, 2018

Assignment Goals:

- Learning about the fork and exec commands.

Assignment Instructions (Dr. LoPinto):

- Create a simple shell program that reads in a command from the keyboard, forks a new process, and uses exec to run the command.
- The command can run a simple test programs like 'ls' or ones that you write (can be very simple.)
- Your shell should be ready to execute other commands until you exit (can be CTL-C. Since you are reading commands and doing fork and exec in a loop there is the possibility of a fork bomb.
- DO NOT DO THIS ON OTTER. USE THE LAB MACHINES.
- Note that 'exec' will replace the executable image. IT SHOULD NOT RETURN to the calling program. You must abort the program if exec returns.

What I did:

- Wrote a simple shell program in c++ that promps the user to enter a command.
- Then the program forks a new process.
- Then in the child the program uses execvp to run the command provided by the user.

What I learnt:

- I learnt that I can create a child process with fork() with this
  ```
  pid_t pid= fork();
  ```
    - That creates a duplicate process.
    - After fork each process behaves in a different way.
    - Change in one doesn't affect the other.
    - An example of system calls for process management.
- Used `waitpid(-1, NULL, 0);` in the parent process to wait until the child processes to exit, for the parent to continue.

- We use exec system call to run a given command on the terminal, exec causes the core image to replaced by the file name in its first parameter.
  - Most of the time the first arg is the file to be executed
  - The second points to the argument array
  - The third is a pointer to the environment array.
- Last thing I learnt is that we need to use abort() at the end of the child process to not run into a fork bomb ie not get out of the child.

Credits:

**MODERN OPERATING SYSTEMS**

- Modern Operating Systems 3rd ed.
- Dr. LoPinto lecture notes.
- youtube: Sandie Xie using fork() and exec() in C.