# CptS 370 Program 1: System Calls in C/++ Report

In this report, I’ll explain the algorithm of my processes.cpp.

The purpose of this program is to simulate the operating system, receive the specified parameters and output the same results as the operating system instructions.

First, I specify the value returned by “pid” as the fork() function, which creates a process through the system call that proceeds the same way it did before. When it is less than zero, the error statement is printed. When it equals zero, start the child process. Only one process is executing this code before “pid=fork()” executes.

However, after this statement, it becomes multiple processes executing if(pipe(FD)<0) and so on... This determines the return value of the fork after the call: in the parent process, fork() returns the process ID of the new child process, and in the child process, fork() returns 0. Fork () returns a negative value if there is an error. The dup2 function is called in the child/grandchild process with arguments fd[WR] or fd\_grandChild[WR], and 1 or 0, which causes the file descriptor represented by 1 or 0 (stdout or stdin) to be replaced with a newly opened file descriptor. Redirect to the pipe while closing the unused end.

Finally, the execlp function is used to replace the image of the child process with the process image of the commands PS-A, Grep Argv [1], and WC-L. Once this command is executed, any output from it will be sent to the input end of the pipeline, thus the simulation effect is achieved.