CS 392: Homework Assignment 6
Due: April 25, 11:55pm

Philippos Mordohai
Department of Computer Science
Stevens Institute of Technology
Philippos.Mordohai@stevens.edu

Collaboration Policy. Homeworks will be done individually: each student must hand in their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems. Use of the Internet is allowed, but should not include searching for previous solutions or answers to the specific questions of the assignment.

Late Policy. No late submissions will be allowed without consent from the instructor. If urgent or unusual circumstances prevent you from submitting a homework assignment in time, please e-mail me explaining the situation.

Objective

The goal of this assignment is to combine system utilities using `fork()`, `pipe()` and `exec()`. (You can pick any member of the `exec` family of functions.)

Find all Subdirectories using `ls` and `grep` (100 points)

The object of this assignment is to implement the following pipe in C, and in addition **count the number of lines in the output**:

```
ls -l <argv1> | grep ^d
```

`ls -l` displays the contents of the directory provided as the only argument to the program using the long listing format (option `-l`).

`grep ^g` finds all lines that begin with `d`. This effectively returns all directories displayed by `ls -l`, since they look like:

```
drwx------ 2 root root   4096 Feb  9 01:14 hw2
```

while regular files begin with a -.

You will create two child processes, one for `ls`, and one for `grep`. You will connect the standard output of `ls` to the standard input of `grep` using one pipe, then you will connect

a second pipe to the standard output of `grep` and to the parent process to read from. The parent process will read from the read-end of the `grep` pipe until it reaches end-of-file (read() returns 0), print out all the text received, AND report how many lines were returned.

Requirements

(1) The program should accept one argument, which will be assumed to be a directory or file. (See below for error handling.)

(2) `popen(), fread()/fwrite()`, or any other functions that use FILE streams must not be used.

(3) Use `dup2()` to duplicate file descriptors, making sure you close all unused file descriptors.

(4) Check the return code of all the system and function calls you use in your program. If any of them fails, print an error message such as: `Error: grep failed.` and return EXIT_FAILURE.

Hints

(1) Make sure you have closed all file descriptors that you do not need. If the write end of a pipe is still unclosed, the process on the read end of it will think it is still open and your program may hang.

(2) Check the return value of the call to `dup2()`, and make sure it is successful.

(3) Add the line counting functionality and see Lab 8 for an example on creating child processes with a pipe connecting them.

(4) Test your program in a directory with subdirectories like the ones we have already made, or a system directory.

Deliverables

(1) A single C file with sufficient comments for us to understand the approach.

(2) Header files and a makefile are optional, but must be submitted in a zip file with the source file(s) if used.