CSCI 3751 Fundamentals of Unix
Programming Lab Assignment #2 (50 points)

This lab has two parts.

Part1: Extend Lab 1 assignment to handle file redirection (30 points)

Goal

The purpose of this lab is to allow you to become comfortable with the following fundamental Unix system calls:

- dup()/dup2()
- file redirection

Deliverable

Submit your source codes, test results, and *Makefile* to the Canvas. Please don't include any source codes of apue.3e.

Requirements:

Extend Lab 1 work to handle following four types of file redirections. At least the first four commands shown in the box below should work as expected.

```
1) > outfile
2) < infile
3) < infile > outfile
4) >& outfile
```

```
# Our shell interpreter still doesn't support
# expanding file extension, "*.*"
$ ls -lrt [file name 1] [file name2] [file name3] > outfile
$ cat < hello.c
$ sort < hello.c > outfile1
$ ls -l [an existing file] [a non-existing file] >& outfile
$ ls -l [an existing file] [a non-existing file] > outfile 2 >& 1

NOTE: >&output,&>output,>output 2>&1 (They are equivalent redirections)
```

NOTE 1: If you did not finish your lab1 assignment and you don't plan to finish it, you may write your own unit test driver program to demonstrate your redirections work.

NOTE 2: Don't put too much effort on the command-line input syntax, such as handling white spaces and erroneous input. You may assume that the users are perfect AI users who conform to your own arbitrary syntax restrictions. For example, assume that the user will use one space for each redirection metacharacter as shown in the example above.

Part2: Extend Fig 4.22 to implement "myFind [arg1] [arg2]" where arg1: <starting path>, arg2: <file name> search string. (20 points)

Goal

The purpose of this lab is to learn Unix file structure and traversing its files (ftw).

Deliverable

Upload your source code, test results, and **Makefile** in tar format, such as lab2-your-stdid.tar to the class Canvas.

Requirements:

- Carefully review the code in Fig 4.22. Make sure you fully understand entire code. It's a well-written **C** code that you might as well take some time to **appreciate** it.

- Copy and paste the code in **Fig 4.22** to your work area. Then modify the code to implement a `find(1)`-like command (***`myFind`***). The command interface is shown below and it takes two arguments. It starts its file traversal from the path specified by the first argument. During the traversal, it lists all the files whose file name contain the substring (pattern) specified by the second argument. No other original UNIX ***`'find'`*** command options need to be supported.

  - $myFind [STARTING_POINT] [PATTERN]

  NOTE: If done correctly, a straightforward implementation may take just a few lines of code modifications.

- See the next page for expected output.

Expected output: see the next page

```
[namsu@csci-gnode-02 ftw]$ myFind $APUE_HOME mycat 2>/dev/null
/export/homes/namsu/3751/apue.3e/fileio/mycat.c 262
/export/homes/namsu/3751/apue.3e/fileio/mycat 13472
/export/homes/namsu/3751/apue.3e/intro/mycat.c 262
/export/homes/namsu/3751/apue.3e/intro/mycat 13472
Total number of files37514
Total number of bytes: 27468

[namsu@csci-gnode-02 ftw]$ myFind /etc/ pas 2>/dev/null
/etc//openldap/certs/password 45
/etc//passwd 1447
/etc//passwd- 1390
/etc//pam.d/passwd 188
/etc//pam.d/password-auth-ac.pre_cdc 1094
/etc//pam.d/password-auth-ac.cdc 1451
/etc//security/opasswd 0
/etc//centrifydc/passwd.ovr.sample 2074
Total number of files: 8
Total number of bytes: 7689
```