Lab 2 - Description

(System Calls in C)

Lab Overview:

For this lab, we will be learning how to execute some system calls in C. In Linux systems programming, system calls are the main way our program interacts with the OS kernel. System calls are how a program enters the kernel to perform some task. Programs use system calls to perform a variety of operations such as: creating processes, doing network and file IO, and much more. In this lab, we will learn how to use system calls to implement a novel command (**lfcat**) for our pseudo-shell. lfcat() is a novel command that lists all files and their contents. **(Note: The method used to implement the command in this lab is the same as expected for other commands in proj. 1)**

Make sure you don't mix the lab source files and directories with the project (files such as command.h, command.c, and output.txt could create confusion). Keep the lab and project work in separate folders.

Core Tasks:

1. Implement the program main with a stub function for lfcat.
2. List all files in the current dir using a loop.
3. Redirect STDOUT to a file with freopen().
4. Write the contents of each file to STDOUT.
5. Test your code with Valgrind for memory leaks.

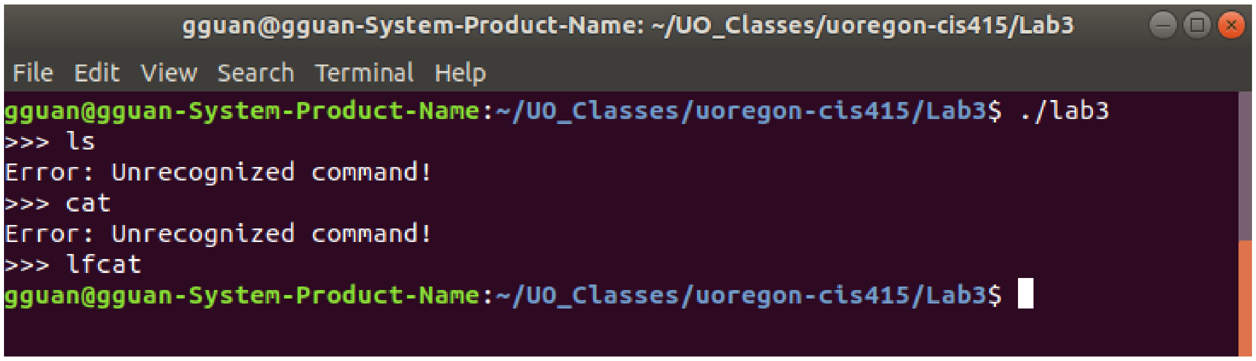
Goal:

List content of all readable files in the current directory to an output file, using **write** and **freopen**. (excluding source file and output file)

Task Details:

1. Implement the program main with a stub function for lfcat**.**

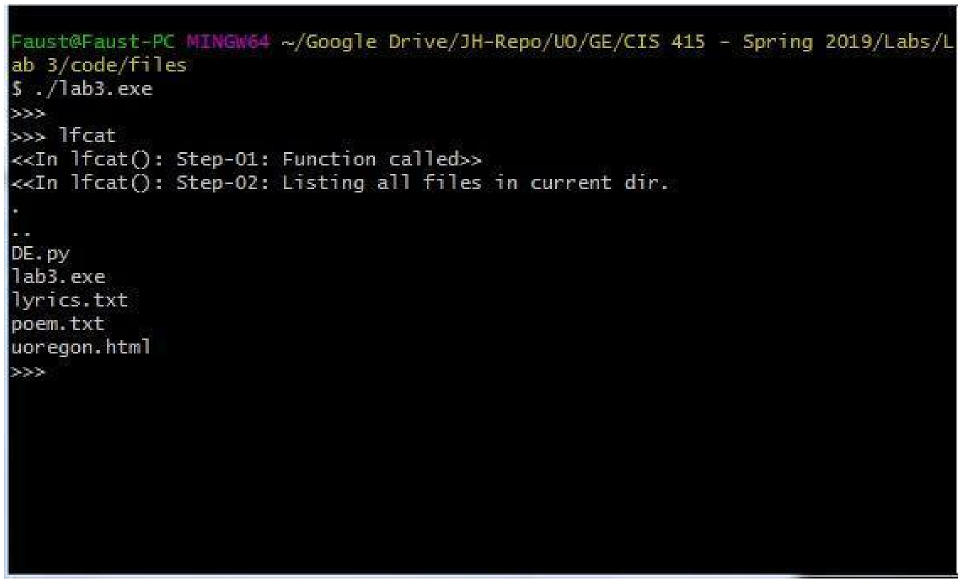
a. For this task, you can edit your code from lab 1. Implement the behavior shown in Fig. 1. First, the program will display a prompt followed by collecting user input as discussed in project 1. The program will also contain the ability to write all stdout to a file named “output.txt”. For this step, it is sufficient to use a stub for the lfcat function that prints out a message such as the one shown in fig. 1. **You do not need to do all of the error checkings but it is strongly advised that you do to give you the best footing for project 1.**


gguan@gguan-System-Product-Name: ~/UO_Classes/uoregon-cis415/Lab3
File Edit View Search Terminal Help

```
gguan@gguan-System-Product-Name:~/UO_Classes/uoregon-cis415/Lab3$ ./lab3
>>> ls
Error: Unrecognized command!
>>> cat
Error: Unrecognized command!
>>> lfcat
gguan@gguan-System-Product-Name:~/UO_Classes/uoregon-cis415/Lab3$ 
```

Fig. 1: Function stub and calling the function (with error checking)
(You **don't have to** worry about executing multiple commands with ";")

2. In lfcat: List all files in the current dir using a loop.
This step can be hidden in your submission, the instruction for you is for testing purposes.
    a. In your implementation: you must utilize the following system calls:
        i. **getcwd(2)** : http://man7.org/linux/man-pages/man2/getcwd.2.html
        ii. **opendir(3)**: http://man7.org/linux/man-pages/man3/opendir.3.html
        iii. **readdir(3)**: http://man7.org/linux/man-pages/man3/readdir.3.html
        iv. **closedir(3)**: http://man7.org/linux/man-pages/man3/closedir.3.html
    b. See Fig. 2 below for how the listing would look in your code at this step (format for listing is recommended for the project). (you can use print statements at this intermediate step.)



```
Faust@Faust-PC MINGW64 ~/Google Drive/JH-Repo/UO/GE/CIS 415 - Spring 2019/Labs/L
ab 3/code/files
$ ./lab3.exe
>>>
>>> lfcat
<<In lfcat(): Step-01: Function called>>
<<In lfcat(): Step-02: Listing all files in current dir.
.
..
DE.py
lab3.exe
lyrics.txt
poem.txt
uoregon.html
>>>
```

Fig.2: Listing all entries

3. Redirect STDOUT to output.txt
    a. You can use the function below to redirect STDOUT to a file. (all output to STDOUT will now be saved in the output.txt file)
        i. **freopen(3p)**: https://man7.org/linux/man-pages/man3/freopen.3p.html
4. Write the contents of each file to STDOUT.
    a. Now that we have the file names, edit your function to write these filenames to STDOUT using the write system call.
        i. **write(2):** http://man7.org/linux/man-pages/man2/write.2.html
    b. In the loop from step 2:
        i. Open the file for reading. (Hint: the name is specified by **`d->d_name`**)
        ii. Read in the current file. You must use **getline(3)**
        iii. Write the filename and contents to STDOUT. (See the attached output.txt for format and sample output)
        iv. Close the file using **fclose()** you may also need to null assign your buffer/file descriptors.
        v. Write 80 "-" characters then repeat the loop.
    c. Clean your code to get rid of any print statements in the lfcat function. See the provided output.txt for the format
    d. Your program can quit right after lfcat() is executed.
5. Test your code with Valgrind for memory leaks.

Submission Requirements:

Note: This lab should be a trivial amount of work if you have already made good progress on your project. The main should work exactly the same as in the project so feel free to copy-paste your project main. In order to receive points for this lab the student must do the following:

1. Attend the labs and make sure your attendance is recorded.
2. Submit your **output.txt**. (It **must** be the same as the attached output.txt, order of the files does not matter)
3. Valgrind output with leak-check and mem-check showing no memory leaks.
4. Submit your lab folder to Canvas within the deadline.
5. Lab folder must contain:
    - i. main.c
    - ii. command.c
    - iii. output.txt
    - iv. makefile: the command to build your code should be:
        1. gcc main.c command.c (make sure to use command.h)
    - v. screenshot of building, running, and running with Valgrind, and system information (screenfetch)
    - vi. Other supporting files

The naming convention of the zip/tar file while uploading to canvas

- ◦ UoID_duckID_LAB/ProjectX (an example is given below)
    - ▪ UOID : alex
    - ▪ DuckID: 951505xxx
    - ▪ Submission for: Lab2
    - ▪ So the name of the zip/tar file should be:

      alex_951505xxx_Lab2.tar **or** alex_951505xxx_Lab2.zip