Lab 3 - Description

(System Calls in C)

Lab Overview:

For this lab, we will be learning how to execute 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 and glibc to implement a novel command (**lfcat**) for our pseudo-shell. lfcat() is a novel command that lists all files and their contents to an output file. **(Note: The method used to implement the command in this lab is the same as expected for other commands in proj. 1)**

Core Tasks:

1. Create a stub function lfcat() .
2. List all files in the current dir using a loop.
3. Write the contents of each file into the output file.
4. Test your code with Valgrind for memory leaks.

Task Details:

1. **Create a stub function lfcat() .**
    a. Using the code from lab-2 (see lab3-skeleton.c, it's better if you use your own) delete the code for displaying the tokenized string and put in a function call to lfcat. (**Note: be sure to check for errors!**) See Fig. 1 below for the functionalities of the command call and error checking.

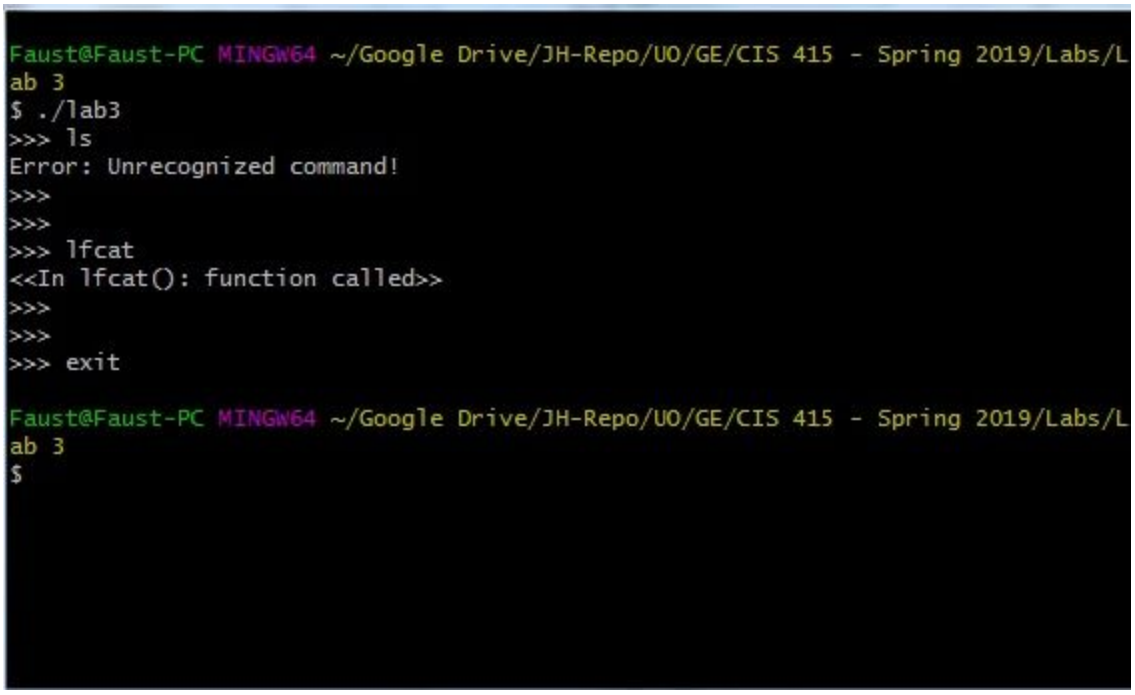

Fig. 1: Function stub and calling the function (with error checking)

2. List all files in the current dir using a loop.
    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 listing should look in your code at this step.

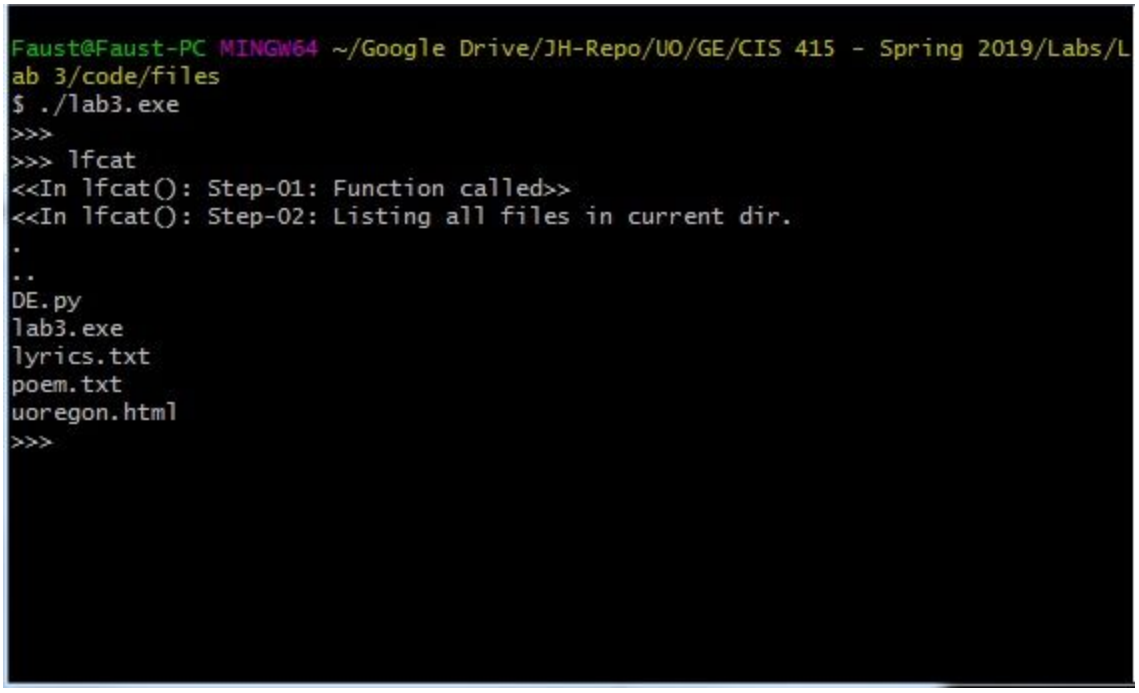

Fig.2: Listing all entries

3. Write the contents of each file into the output file.
    a. Now that we have the file names, edit your function to open a file named "output.txt".
    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 "output.txt". (See the attached output.txt for format and sample output)
    c. After all files are looped through, close output.txt and the dir.
    d. Clean your code to get rid of any print statements in the lfcat function. See Fig. 3 for a complete sample run of the finished lab.

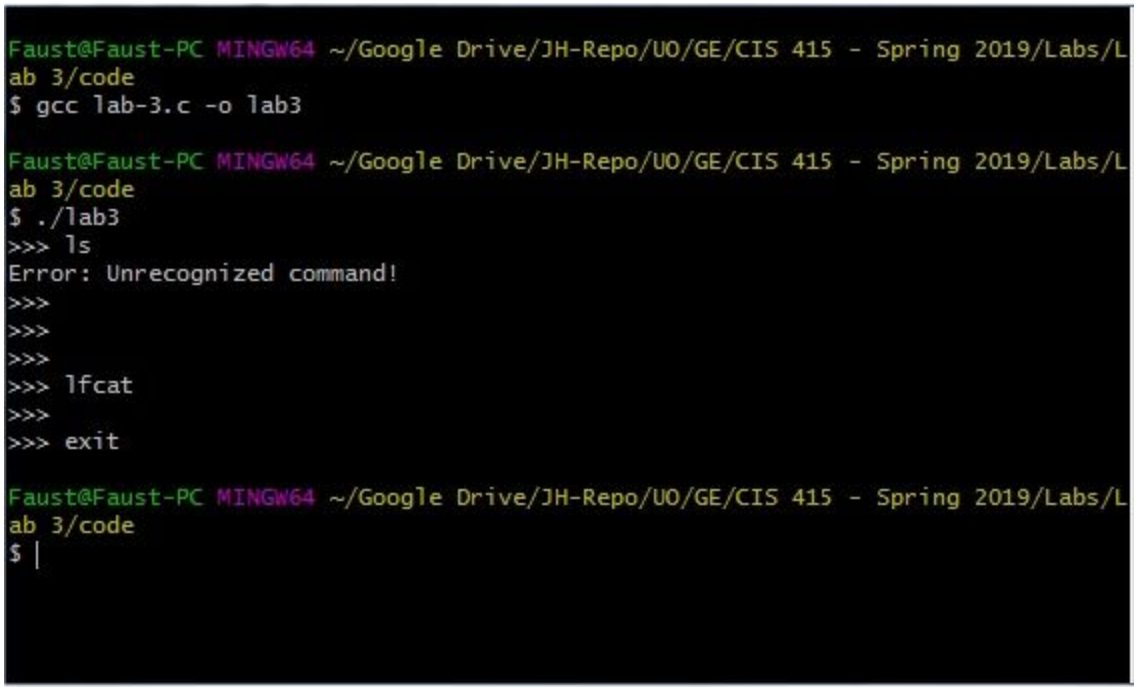

Fig. 3: A run of the completed lab-3

4. Test your code with Valgrind for memory leaks.

Submission Requirements:

In order to receive any credit for a lab, completion of the labs' core tasks must be demonstrated to the TA's. A file lab3-skeleton.c has been provided for reference in case you could not complete lab2 (however it's best if you use your own code from lab2). In order to receive points for this lab the student must do the following:

1. Compile and run your code (we must see it compile and execute successfully).
2. Show us your **output.txt**. (It **must** be the same as the attached output.txt)
3. Valgrind output with leak-check and mem-check showing no memory leaks.
4. Submit your file to Canvas.