

Lab 3 - Description

(System Calls in C)

Lab Overview:

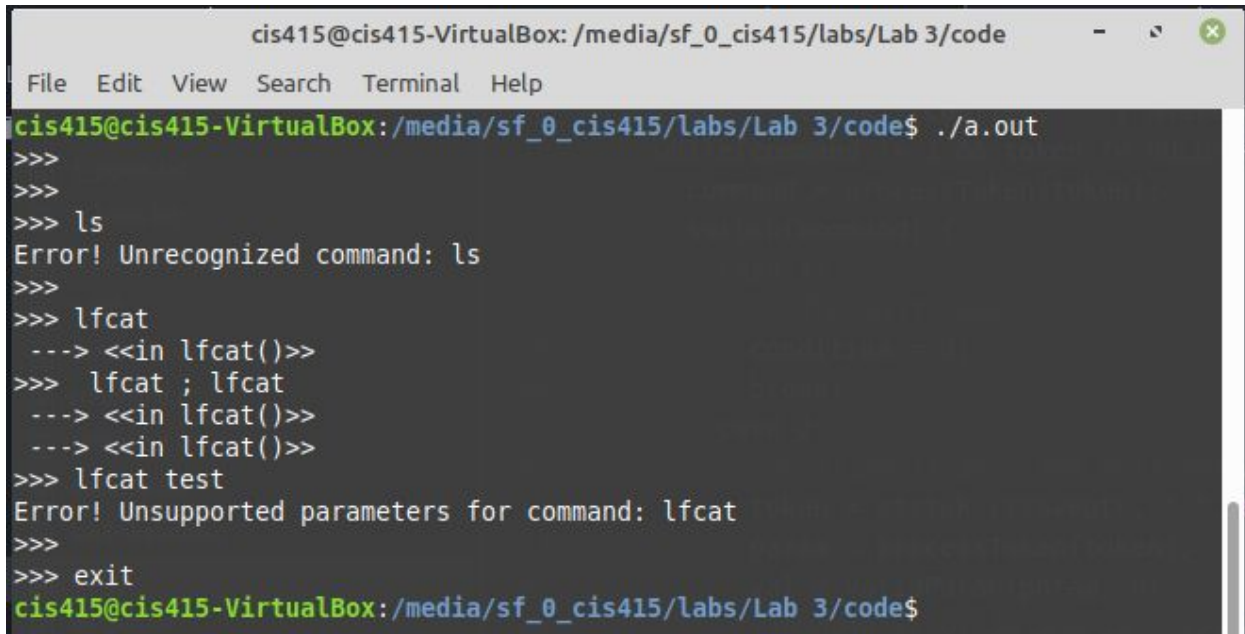
For this lab, we will be learning how to execute system calls in C. In Linux systems programming, system calls are the main way our program interacts with the OS kernel. System calls are how a program enters the kernel to perform some task. Programs use system calls to perform a variety of operations such as: creating processes, doing network and file IO, and much more. In this lab, we will learn how to use system calls to implement a novel command (**lfc**) for our pseudo-shell. **lfc()** 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)**

Core Tasks:

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

Task Details:

1. Implement the program main with a stub function for **lfc** .
 - a. For this task you can either edit your code from lab 2 or start with the attached skeleton files. See the attached skeleton files for a general idea on what to do for both the main and **lfc** in either case. From here, 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" should the user provide the -f flag along with an input filename. For this step, it is sufficient to use a stub for the **lfc** function that prints out a message such as the one shown in fig. 1. **Note: Don't forget to do for the two cases shown. 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.**



```
cis415@cis415-VirtualBox: /media/sf_0_cis415/labs/Lab 3/code
File Edit View Search Terminal Help
cis415@cis415-VirtualBox:/media/sf_0_cis415/labs/Lab 3/code$ ./a.out
>>>
>>>
>>> ls
Error! Unrecognized command: ls
>>>
>>> lfcats
---> <<in lfcats()>>
>>> lfcats ; lfcats
---> <<in lfcats()>>
---> <<in lfcats()>>
>>> lfcats test
Error! Unsupported parameters for command: lfcats
>>>
>>> exit
cis415@cis415-VirtualBox:/media/sf_0_cis415/labs/Lab 3/code$
```

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

2. In lfcats: 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. (you can use print statements at this intermediate step.)

```

Faust@Faust-PC MINGW64 ~/Google Drive/JH-Repo/U0/GE/CIS 415 - Spring 2019/Labs/L
ab 3/code/files
$ ./lab3.exe
>>>
>>> lfcats
<<In lfcats(): Step-01: Function called>>
<<In lfcats(): Step-02: Listing all files in current dir.
.
..
DE.py
lab3.exe
lyrics.txt
poem.txt
uoregon.html
>>>

```

Fig.2: Listing all entries

3. 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 lfcats function. See the provided output.txt for the format
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 and the file **must** be submitted onto Canvas before the end of the lab period. The

files lab3-skeleton.c and command-skeleton.c have been provided for reference in order to give you a solid idea on how to begin. **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. Compile and run your code in file mode (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 lab folder to Canvas. (the deadline for submission is 2 pm on Friday. The assignment must be submitted before then to receive credit.)
 - a. Lab folder must contain:
 - i. main.c
 - ii. command.c
 - iii. log.txt
 - iv. makefile: the command to build your code should be:
 1. gcc main.c command.c (make sure to use command.h)