

Name: Archit Benipal

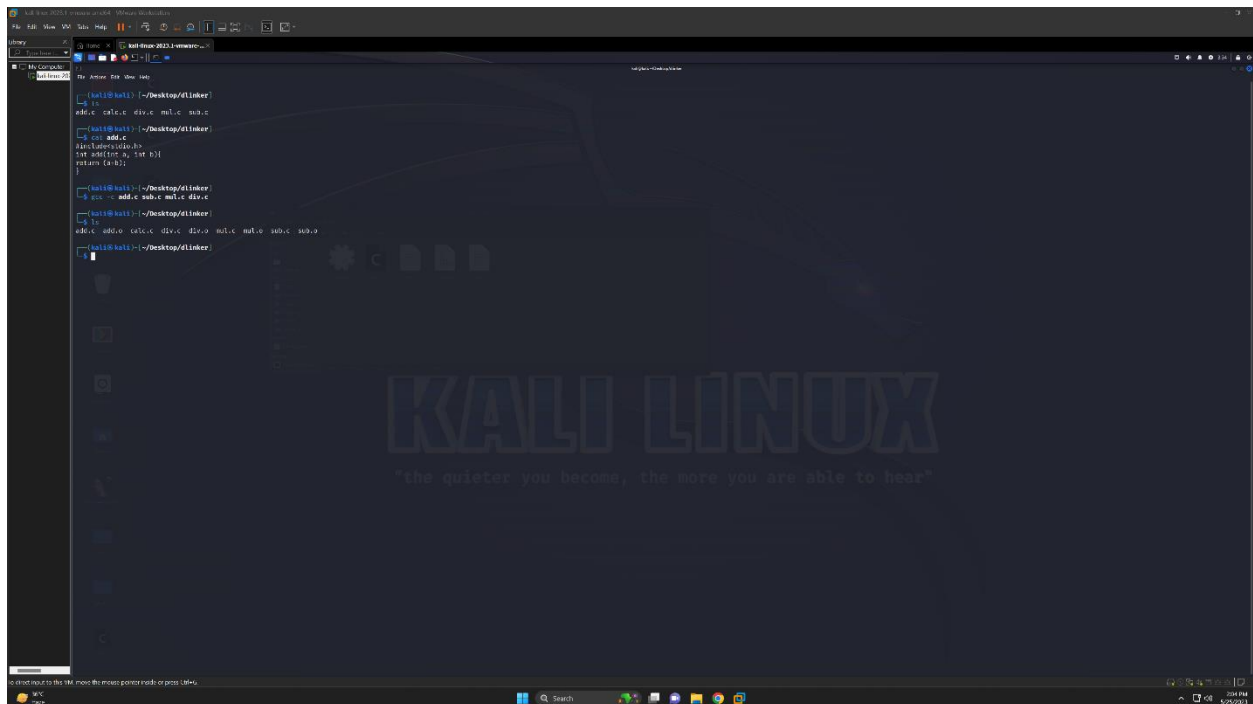
Subject: Software Supply Chain Security (CY 653)

**Assignment 4: Generate a shared lib (dynamically linked lib)
and integrate it to your software**

1. Study the online material, then repeat the exercise. Upload screenshots of your exercise. Use 'objdump -p <executable>' rather than 'ldd <executable>'.

Detailed breakdown: Method 1: Copying the shared library

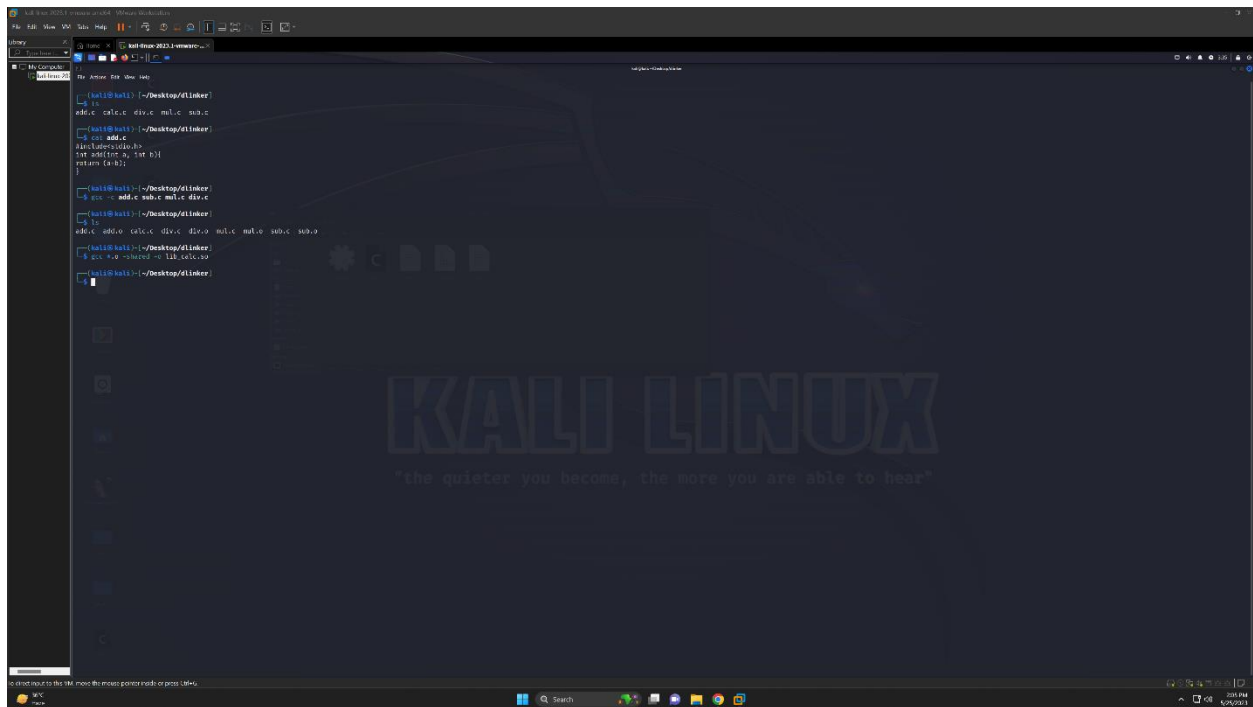
Step1: Created the c files for mathematical operations and also the main c file "calc." And also created the object files for the mathematical operator programs



```
kali@kali: ~/Desktop/4Linker
ls
add.c  calc.c  div.c  mul.c  sub.c
gcc -c add.c
gcc -c calc.c
gcc -c div.c
gcc -c mul.c
gcc -c sub.c
```

The -c switch tells gcc to skip the linking step and create the object file

Step 2: After that I created a shared library named “lib_calc.so”



The screenshot shows a Kali Linux desktop environment with a terminal window open. The terminal displays the following commands and their outputs:

```
kali@kali: ~/Desktop/Linker
$ gcc -c calc.c -o calc.o
$ gcc -shared -o lib_calc.so *.o
```

The terminal background features the Kali Linux logo and the slogan "the quieter you become, the more you are able to hear". The desktop taskbar at the bottom shows various application icons and the system clock indicating 10:51 PM on 6/26/2021.

Breakdown of the command use:

The command is used for compiling a shared library, specifically **lib_calc.so**, from all object files (*.o) in the current directory. Here's what each part of the command does:

- **gcc:** This is the GNU Compiler Collection. This command invokes the compiler. Although GCC supports several languages, it will interpret these files as C or C++ due to the .o extension.
- ***.o:** This is a wildcard that matches all files in the current directory that end in .o. These files are object files, which are compiled but not linked code.
- **-shared:** This is an option that tells GCC to produce a shared library rather than an executable. Shared libraries can be used by multiple programs at the same time.
- **-o lib_calc.so:** The -o option tells GCC where to put the output of the compilation. In this case, it will create a shared library called **lib_calc.so**.

Step 3: Created the object for the “calc.c” program using command “gcc -c calc.c -o calc.o”

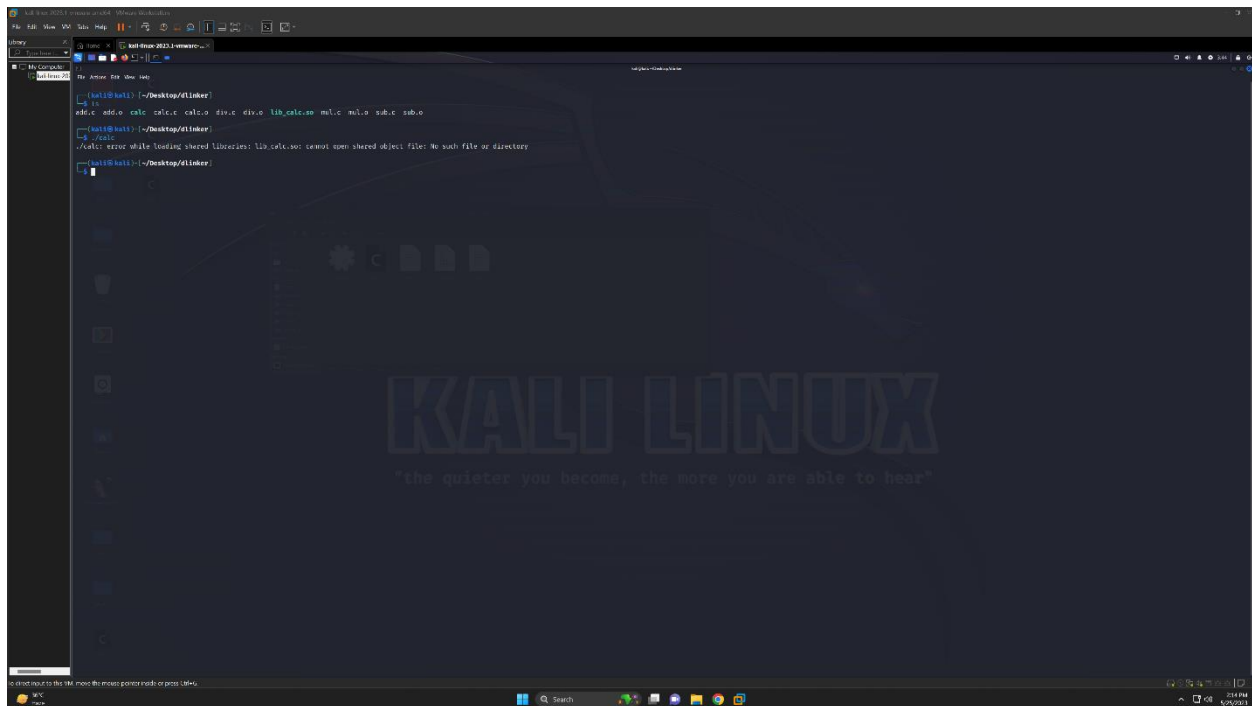
Step 4: For the getting the binary file for calc.c I used the command “gcc -o calc calc.o -L. -l_calc”

Break down of the command used

The command is used to compile and link an executable binary named **calc** using the object file **calc.o** and a shared library **lib_calc.so** located in the current directory:

- **gcc:** The GNU Compiler Collection command invokes the compiler.
- **-o calc:** The **-o** option specifies the output file name. In this case, it is **calc**.
- **calc.o:** This is the input object file to be used in the creation of the executable.
- **-L.:** This flag tells the linker to look in the current directory for library files.
- **-l_calc:** The **-l** option tells the linker to link against a library. In this case, the library is **lib_calc.so**. The linker automatically prepends **lib** and appends **.so** (or **.a** for static libraries) when searching for the library file.

Tried to run the calc binary to see the error message:



```
kali@kali: ~/Desktop/41Linker
└─$ ./calc
add: add: calc: calc: div: div: lib_calc.so: mul: mul: sub: sub:
kali@kali: ~/Desktop/41Linker
└─$ ./calc
./calc: error while loading shared libraries: lib_calc.so: cannot open shared object file: No such file or directory
kali@kali: ~/Desktop/41Linker
```

When I ran **objdump -p calc**, it will display a lot of information about the binary. One of the sections is called "Dynamic Section", and it lists the shared libraries that the binary links to at runtime. The entries I looked for start with **NEEDED**, followed by the name of the library.

Copied the shared library lib_calc.so to /usr/lib and ran the binary file

[illegible]

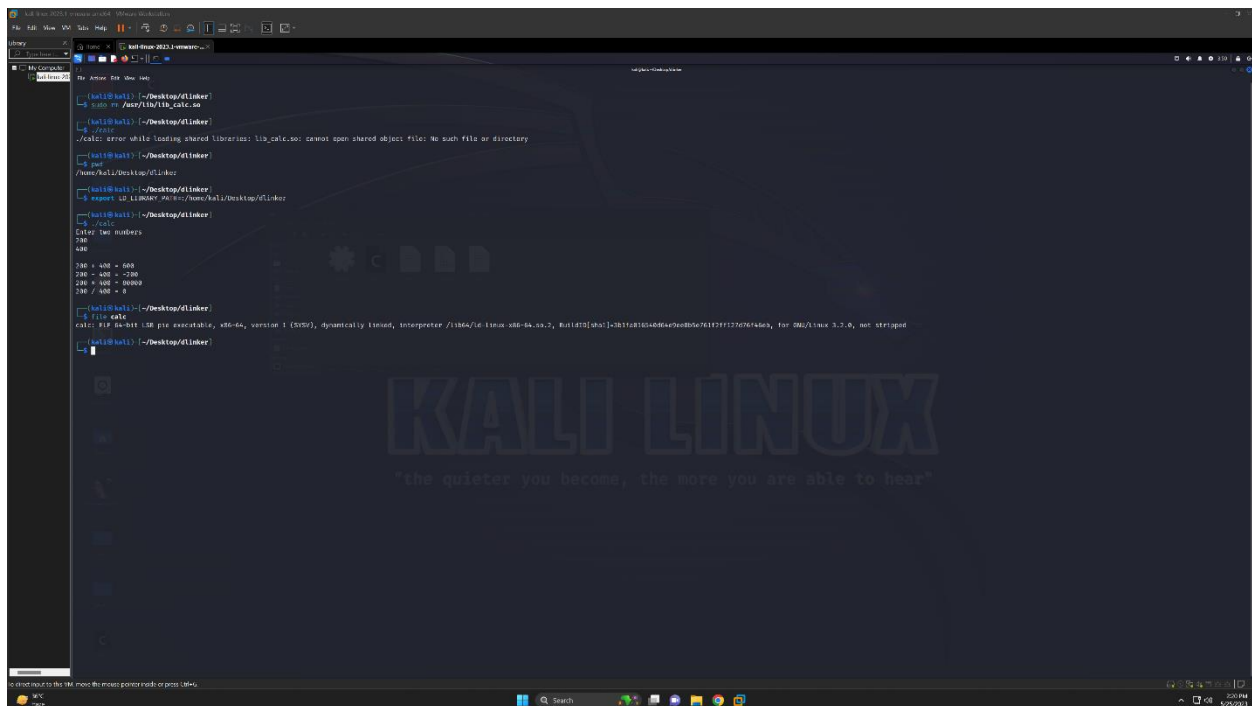
Method 2: Exporting the current working directory

Command used to use the current working directory “export LD_LIBRARY_PATH=./home/kali/Desktop/dlinker”

Breakdown of the command used:

The **LD_LIBRARY_PATH** environment variable is used by the dynamic linker/loader (**ld.so** or **ld-linux.so**) in Unix-like systems to find necessary shared libraries. When I run a binary that needs a shared library (.so file), the system looks in several directories to find the library, including the ones listed in **LD_LIBRARY_PATH**.

This command appends the directory **/home/kali/Desktop/dlinker** to the existing **LD_LIBRARY_PATH**. Any colon-separated directories listed in **LD_LIBRARY_PATH** will be searched for shared libraries. This is done before the standard set of directories are searched by the linker.



```
kali@kali:~/Desktop/dlinker$ export LD_LIBRARY_PATH=./home/kali/Desktop/dlinker
kali@kali:~/Desktop/dlinker$ ./calc
Enter two numbers
200
400
200 + 400 = 600
200 - 400 = -200
200 * 400 = 80000
200 / 400 = 0.5
kali@kali:~/Desktop/dlinker$
```

The screenshot shows a terminal window in Kali Linux. The user is in the directory `~/Desktop/dlinker`. They run the command `export LD_LIBRARY_PATH=./home/kali/Desktop/dlinker`. Then they run `./calc`, which prompts for two numbers. The user enters 200 and 400, and the program outputs the results of addition, subtraction, multiplication, and division. The terminal background features the Kali Linux logo and the slogan "the quieter you become, the more you are able to hear".

Note: For this method changes made with export are only valid for the current shell session. For persistent change across all sessions we need to modify the “`~/.bashrc`”