Laboratory 2

Multithreading – Deadlock.

In an operating system, a deadlock occurs when a process P1 or a thread T1 is waiting for a resource held by another process P2 or another thread T2, which in turn is waiting for the release of a resource held by process P1 or thread T1.

Please open the archive laborator2.zip, and inside you will find the "deadlock" folder. Go into it and compile the file using the command:

```
gcc -o deadlock deadlock.c -lpthread
```

Run the program. Does it get stuck? Is it a perfect deadlock?

Practice 1(2pt). Create a perfect deadlock by adding a barrier after acquiring the first resource.
Hint: You can use a `pthread_barrier_t` structure, see the function:
https://man7.org/linux/man-pages/man3/pthread_barrier_wait.3p.html

Practice 2(2pt). Let's return to the initial deadlock. Compile the executable using the command:

```
gcc -O0 -g -o deadlock deadlock.c
```

Run the command:

```
gdb ./deadlock
```

Once gdb has started, run the command "run."

Your program has likely become stuck at this point. To stop it and analyze what's happening, send a SIGINT signal to the program by pressing CTRL+C in the console where gdb is running.
The program is now stopped, allowing you to analyze it.

Type the command:
`info threads` - how many threads does it display? Why?

Repeat the same steps for the process you implemented in step 1. Does info threads now display 3 threads?

Type the commands:

```
thread 2
info stack
thread 3
info stack
```

After the above commands, you should have been able to observe where each thread is blocked. By analyzing an executable in this way, you can identify the resources each thread is waiting for and deduce the cause of the deadlock.

Library hijacking

Practice 3 (3pt).
Read and implement the shared library (shlib.so) as described in the tutorial at https://sumit-ghosh.com/posts/hijacking-library-functions-code-injection-ld-preload/
Exploit the “printf” function used in the main program to obtain shell access, similar to what’s done in the tutorial with the “_init”. s