1 Introduction

To get started, use this invitation link
`https://classroom.github.com/a/n-w2vnVl` to create your turn-in repository. `git clone` the repo to your the machine you will be working on, and run `make pick-C` or `make pick-rust` in the resulting folder.

2 Watch your language

Texas school district 666 needs your help in protecting their minors from harmful content. Specifically, if a program tries to write “evolution” to any file descriptor, they want it to instead write ”GOD”.

Your solution needs to work on unmodified system binaries, such as cat and echo, so we will implement a shared library to intercept writes and replace all occurrences of the offensive term. School district administration will ensure that the `LD_PRELOAD` environment variable is appropriately set when installing your fix. For now, intercepting only calls to `write` will be sufficient.

2.1 Preparation (50%)

Changing the behavior of the `write()` system call wrapper has implications beyond saving children's innocent minds. Because many functions that write to a file descriptor make use of this wrapper function, it can be difficult to debug problems in its implementation. Specifically, if it doesn't work, you can't print. And probably neither can `gdb`. Makes it hard to debug things.

As an intermediate step, create a shared library `libsafeprep.so`, which provides a new function `safewrite()` that takes arguments exactly like the `write()` system call wrapper.

The function `safewrite()` replaces every occurrence of the word `evolution` (lower-case only) in the passed in buffer, with the word `GOD`. It writes the new buffer using the standard `write()` function. It is not important whether it calls `write()` once or several times per call to `safewrite()`.

Test `libsafeprep.so` by linking it with the provided `safecat.c` file. `safecat` works like `cat`, but uses `safewrite()` instead of `write()` to write its output.

2.2 Turn-in Instructions

In the C folder, include a file `safe` which, when compiled with `gcc --shared safeprep.c -o libsafeprep.so` produces a shared library. When linked with safecat.c it produces a program that replaces every instance of the word ”evolution” with the word ”GOD” for every line of input before it is printed back to `stdout`.

2.2.1 Rust version

In the Rust folder, include a crate `safeprep` which, when built with `cargo build` produces a shared library `safeprep.so`. When linked with safecat.o (compile this from safecat.c with the command `gcc -c safecat.c` it produces a program that replaces every instance of the word "evolution" with the word "GOD" for every line of input before it is printed back to `stdout`.

2.3 Real Implementation (50%)

Having successfully implemented `safewrite()` in `libsafeprep.so`, create a shared library `libsafe.so` that implements a function `write` that works just like `safewrite()`. However, while `safewrite()` can call `write()` to create output, this won't work for `write()`: you'd end up with infinite recursion. Instead, use use dlsym() get a pointer to the original write function. You may have to fight the syntax for function pointers to make this pointer useful.

To test `libsafe.so`, run the target program with an `LD_PRELOAD prefix` like this:

```
LD_PRELOAD=libsafe.so cat darwin.txt
```

2.4 Turn-in Instructions

In the C folder, include a file `safe.c` which, when compiled with `gcc --shared safe.c -ldl -o libsafe.so` produces a shared library. When used with `LD_PRELOAD=./libsafe.so cat`, the library modifies the operation of cat so that every instance of the word "evolution" is changed to the word "GOD" for every line of input before it is printed back to `stdout`.

2.4.1 Rust version

In the Rust folder, include a crate `safe` which, when built with `cargo build` produces a shared library `libsafe.so`. When used with `LD_PRELOAD=./libsafe.so cat`, the library modifies the operation of cat so that every instance of the word "evolution" is changed to the word "GOD" for every line of input before it is printed back to `stdout`.