Filesystem isolation #IF6 Pending

Complete previous stages to gain access to this stage.

Your Task Pending MEDIUM

In the previous stage, we executed a program that existed locally on our machine. This program had write access to the whole filesystem, which means that it could do **dangerous** things!

In this stage, you'll use **chroot** to ensure that the program you execute doesn't have access to any files on the host machine. Create an empty temporary directory and `chroot` into it when executing the command. You'll need to copy the binary being executed too.

At the time of writing this, the implementation of chroot in Rust's standard library (**std::os::unix::fs::chroot**) is still a nightly-only experimental API. We've included **libc** as a dependency instead.

When executing your program within the chroot directory, you might run into an error that says `no such file or directory` even if the binary exists within the chroot. This is because **Command::output()** expects `/dev/null` to be present. You can work around this by creating an empty `/dev/null` file inside the chroot directory. This cryptic error effects Go programs too, more details **here**.

Just like the previous stage, the tester will run your program like this:

```
mydocker run alpine:latest /usr/local/bin/docker-explorer ls /some_dir
```