Advanced Homework 10

Due: Wednesday, November 21st, 11:59PM (Hard Deadline)

Submission Instructions

To receive credit for this assignment you will need to stop by someone's office hours, demo your running code, and answer some questions. **Make sure to check the office hour schedule as the real due date is at the last office hours before the date listed above.** This applies to assignments that need to be gone over with a TA only. **Extra credit is given for early turn-ins of advanced exercises. These details can be found on the website under the advanced homework grading policy.**

Docker

Docker is an open source tool designed to make it easy to build and run applications using containers. Containers allow developers to package an application with all of its dependencies and ship it as one image. As a result, developers can be assured that their program will run exactly how they expect it to on any machine that has Docker installed.

In some ways Docker is quite similar to virtual machines like the one you've been using throughout the course as both give you a contained environment mostly isolated from the rest of your system. Some of the key differences, however, are that Docker containers are both faster and more lightweight than virtual machines. You can read more info on Docker and how its containers compare to VM's here.

To get started, install docker from here. To verify that your installation completed successfully, run `docker run hello-world` in your terminal, you should see the following:

```
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
d1725b59e92d: Pull complete
Digest: sha256:0add3ace90ecb4adbf7777e9aacf18357296e799f81cabc9fde470971e499788
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
...
```

The Assignment

Set up a Docker environment with GDB installed and push it to the Docker Hub.

For many of your C++ projects (i.e. EECS 281 projects), you might want to use the latest version of GDB which isn't available on MacOS or Windows. One way to get around this is to build a docker container to run GDB in. This allows you to compile, run, and debug the program locally, eliminating the need to sync to CAEN every time you want to test something out.

For this assignment, set up a linux based Docker container with GDB and push it to the Docker Hub (you will need an account for this). You can use any distro you want (Alpine is a nice 5 MB linux distro used a lot with docker) but it'll be simplest to build from an Ubuntu image.

Some links you might find useful are this Docker tutorial and this Dockerfile tutorial

Submission checkoff

- ☐ Show off your Dockerfile
- ☐ Use GDB on a project without copying the project fiels over to the container
- ☐ Show that you've got your image on the Docker Hub

`Expect` Respect

This sections touches on a handy scripting language called `expect` that can be used to automate interactions with remote servers. What we'll be doing is partially automating the login procedure for Michigan servers. We say partially because of the two-factor step during the login procedure.

First you'll need to install `expect` for your system (Ubuntu and MacOS both have packages available for download).

Some guiding steps this script should accomplish are:

1. Prompt the user for their uniqname password and store it in a variable. This output should **not** be displayed on the screen when the user types in their password!
2. Initiate a connection to CAEN and `send` the password previously provided by the user.
3. Select one of the Duo options to fire off the push/call/text automatically.

Although the creating of this script doesn't save too much time during the login process for CAEN, this tool can be useful for a lot of other tasks where you are using key-based access to servers and aren't prompted for passwords or Duo options.

As always, there are multiple ways to accomplish some of the tasks above. A potential solution requires no more than about 20 lines of `expect` scripting.

Submission checkoff

- ☐ Explain why it's a bad idea to hardcode the password for the user into the expect script
- ☐ You'll almost certainly have `\r` in your script. Explain what this means/does
- ☐ Show off your script working
  - ☐ Prompting the user for their password (and not displaying it on the screen)
  - ☐ Automatically selecting one of the Duo options
  - ☐ Successfully logging the user into CAEN