Luscious Locks

HW0 due **2019-01-23 23:59**
Graded files:

- hw0.md

Lab Assignment due **2019-01-23 23:59**
Graded files:

- submission.txt

Content

Learning Objectives

The learning objectives for Luscious Locks are:

- Getting familiar with GDB and system programming.

HW0

You have already been assigned a HW0 for the class. Add your answers to `hw0.md` in your course repository. Remember to commit and push your answers. We'll spend the first part of class going over the various questions from HW0. Your grade for this lab is partly HW0 and partly the assignment below.

Background Guide

Read this (/tutorials/background) to set up your course!

Development

Read the Development Guide (/tutorials/development)

Git

(Read the first section of the development site!) You will be using git (https://illinois.edu/assignments/13/git) for this course. First go to the repository creator (https://edu.cs.illinois.edu/create-ghe-repo/cs241-sp19/). **DO NOT DO THE README TUTORIAL!**

Once you are in your VM, you'll need to set up some global defaults

```
git config --global user.name "FIRST_NAME LAST_NAME"
git config --global user.email NETID@example.com
```

Make sure to replace `FIRST_NAME` and `LAST_NAME` and `NETID@example.com` with your information. For example:

```
git config --global user.name "Lawrence Angrave"
git config --global user.email "angrave@illinois.edu"
```

Your code will not be graded if you git config is not correctly set. **No exceptions!** Then checkout your repository as follows:

```
git clone https://github-dev.cs.illinois.edu/cs241-sp19/NETID.git
```

which will check out your entire git repo into a folder called 'NETID' into your current directory. Now change your directory into that folder:

```
cd NETID
```

You've probably noticed the repository is empty! In order to grab the latest version of our assignment, complete the following steps. This adds the `_release` repository as an extra remote, and this step must be completed every time you want to initialize your repository on a new machine.

```
git remote add release https://github-dev.cs.illinois.edu/cs241-sp19/_release.git
git pull release master
git push origin master
```

Graded Files

One section we will have on the top of every assignment is a section called `graded_files`. These are the files that we use to grade the assignment. git (https://illinois.edu/assignments/13/git) does not allow us to set the files to readonly to prevent changing them (header files, Makefile). You have to be careful to avoid changing these files. We are working on a system that prevents you from accidentally committing the changes, but there is no easy way to set permissions on your local copy. **No exceptions**: please make sure you're only submitting modifications to the graded files listed.

Background and Assignment

You are working for ShadyCorp Inc. Your boss discover's that your competitor, ShadierCorp Inc., has uploaded a safe to your company's production server. Your boss is curious and wants to know what's in the safe. Luckily you boss has a lockpick to help you.

Initial analysis of the safe indicates that it has two modes: interactive mode and file mode.

Interactive mode use:

```
$ ./safe <netid>
```

File mode use:

```
$ ./safe <netid> <input_file>
```

In interactive mode the safe allows you directly provide input to the safe and attempt to unlock it. In file mode the safe reads input line by line from specified file. In both of these modes the safe will reset if given the wrong input. Luckily you have a lockpick which will help you find the correct input to unlock the safe.

The lockpick can be used as follows:

```
$ ./lockpick
```

This will launch GDB and load the safe debug symbols and source code. Use your debugging skills to figure out how to unlock the safe. Keep in mind that to run a program in GDB with arguments you use `r arg1 arg2 arg3 etc.`, so to run the safe with your netid the command would be `r <netid>`.

You should begin by running the lockpick and then starting the safe with `r <netid>`. Read the debugging guide section of the docs for more information about how to use GDB.

Answer Submission

There is a file in your repository called `submission.txt`. You should provide the answers to each phase of the safe in this file. Provide one answer per line. For example if you think the safe has three phases and the answers to the phases are `answer1`, `answer2`, `answer3` then your `submission.txt` should look like:

```
answer1
answer2
answer3
```

Testing

You can use your `submission.txt` file with the safe's file mode to test your answers.

```
$ ./safe <netid> submission.txt
```

If you have the correct answers then `Sucessfully unlocked the safe!` will be printed.

Make sure you use your own netid for everything in this assignment. We will be grading your answers using your netid.

The Outline

- Log into your VM
- Clone or Update your git repository on your VM
- Use the lockpick to figure out how to unlock the safe
- Add your answers to `submission.txt`
- `git commit -a -m "My Submission"` (commit your work to git).

CS 241 Makefile

There is no Makefile for this assignment, but all future assignment in this class will use a Makefile similar to the one which follows. It's important to have a basic understanding of how it works.

Note: This is not a class about makefile programming, so you will not need to know the advanced parts of makefiles (pattern matching, expansion phases, etc). Still, it is important that you know a little bit about how they work for a future assignment.

```
# directory to store object files
OBJS_DIR = .objs

# define the EXES
EXE_SHELL=shell
EXES_STUDENT=$(EXE_SHELL)

OBJS_SHELL=$(EXE_SHELL).o format.o

# set up compiler
CC = clang
INCLUDES=-I./includes/
WARNINGS = -Wall -Wextra -Werror -Wno-error=unused-parameter
CFLAGS_DEBUG   = -O0 $(INCLUDES) $(WARNINGS) -g -std=c99 -c -MMD -MP -D_GNU_SOURCE -DDEBU
G
CFLAGS_RELEASE = -O2 $(INCLUDES) $(WARNINGS) -g -std=c99 -c -MMD -MP -D_GNU_SOURCE

# set up linker
LD = clang
LDFLAGS = -Llibs/ -lprovided

.PHONY: all
all: release

# build types
# run clean before building debug so that all of the release executables
# disappear
.PHONY: release
.PHONY: debug

release: $(EXES_STUDENT)
debug: clean $(EXES_STUDENT:%=%-debug)

# include dependencies
-include $(OBJS_DIR)/*.d

$(OBJS_DIR):
@mkdir -p $(OBJS_DIR)

# patterns to create objects
# keep the debug and release postfix for object files so that we can always
# separate them correctly
$(OBJS_DIR)/%-debug.o: %.c | $(OBJS_DIR)
$(CC) $(CFLAGS_DEBUG) $< -o $@

$(OBJS_DIR)/%-release.o: %.c | $(OBJS_DIR)
$(CC) $(CFLAGS_RELEASE) $< -o $@

# exes
$(EXE_SHELL)-debug: $(OBJS_SHELL:%.o=$(OBJS_DIR)/%-debug.o)
$(LD) $^ $(LDFLAGS) -o $@

$(EXE_SHELL): $(OBJS_SHELL:%.o=$(OBJS_DIR)/%-release.o)
$(LD) $^ $(LDFLAGS) -o $@

.PHONY: clean
clean:
```

```
rm -rf .objs $(EXES_STUDENT) $(EXES_STUDENT:%=%-debug)
```

This looks scary, but if you Google some makefile basics and carefully read the comments it should start to make sense. These are the things you will need to know, at the minimum:

- Compile the assignment:

```
make
```

- Clean up the assignment directory:

```
make clean
```

- Compile a debug-able version of your code that you can use gdb on:

```
make debug
```

- Compile a release version of your assignment that you test with:

```
make release
```

Lab Attendance

Part of your grade in this class relies on you attending labs. Towards the end of every lab, we will ask you to swipe out (swipe your i-card). You may only leave early if you show that you have finished the lab to your lab attendant or if the lab attendant calls time. If you are more than ten minutes late to class, then your lab attendant reserves the right to not swipe you out for the day. You may never swipe yourself out without your lab attendant's consent. Any violation will result in a zero in lab attendance for the semester.

Due to seating limitations, you are required to go to the lab section you signed up for. If you wish to go to any other lab section, you may get permission from the TA of another section to go to their section, provided that either:

- You will be working on your laptop in the room
- There is seating available where registered students get priority.

You must email the TA of that lab section beforehand.

You can still get credit for attending a different section due to special or occasional circumstances by making arrangements with the graduate assistant (GA) at cs241admin@illinois.edu (mailto:cs241admin@illinois.edu) if you need to change your registered lab if you start regularly going to a different lab. Please contact Holly Bagwell in the academic office (SC 1210) to change your section without having to drop your enrollment.

We will never grant exemptions for lab attendance. If you have an interview, then you are just going to have to use your lab attendance drop. You also can *not* make up lab attendance. Be warned that forgetting to swipe out is not a valid excuse—your lab attendant is not allowed to vouch for your attendance.