Homework 7
Make

EECS 201 Winter 2020

Submission Instructions

This homework will be submitted as a repository on the UMich GitLab server. The repository you submit will be a **private** project called `eecs201-hw7`.

Preface

In this homework you'll be provided yet another zipped archive containing some starter files.

```
curl -O https://www.eecs.umich.edu/courses/eecs201/files/assignments/hw7-files.tar.gz
tar xzf hw7-files.tar.gz
cd hw7-files
```

Initialize a Git repository in the extracted `hw7-files` directory. Create a file called `REPORT` in this directory. Add all of the present files and commit them.

Create a **private** project named `eecs201-hw7` on the UMich GitLab (`gitlab.umich.edu`) and add the instructor `brng` as a **Reporter**. Set this UMich GitLab project as your remote: you'll be pushing to it in order to submit.

In this assignment you will be incrementally building up more complex Makefiles. First, we'll start at the beginning.

1 A fresh start

1. `cd` into the `1` directory.
2. Create a file named `Makefile`.
3. Create a rule with a target called `all` that has this for a recipe:
   `gcc -o nocat nocat.c`
4. Create a rule with target called `clean` that has this for a recipe:
   `rm -f nocat`
5. Move the `all` rule so that it'll run by default when `make` is run without a target specified.
6. Make sure that your Makefile works correctly.
7. Add and commit `Makefile`.

2 Phonies

1. `cd` into the `2` directory.
2. Take a look at the Makefile and note the existing rules.
3. Try running `make all`, `make clean`, and `make test`.
4. Note that their recipes do not run.
5. Fix the Makefile so that each of the rules can run their recipes.

3 Dependencies

1. `cd` into the `3` directory.
2. Take a look at the Makefile. Note what each target in the Makefile requires which file.
3. Edit the Makefile so that the `all` target will build the `nekosay` application. The best way to test this is to run `$ make clean` then `$ make all`. `$ make all` as a clean build should successfully build the application.

4 Not repeating yourself

1. `cd` into the `4` directory.
2. Create a file named `Makefile`.
3. Edit the `Makefile` so that:
   - It has a `CC` variable that is set to `gcc`
     This variable represents which C compiler to use.
   - It has a `BIN` variable that is set to `sum30`
     This variable represents what the output executable binary is named.
   - It has a `SRCS` variable that contains all the `.c` files under the `src` directory.
     (You shouldn't have to list each file individually...)
   - It has a rule to build the output executable binary. Don't worry about object code. Note that the `-o` flag for `gcc` sets the output name. This rule should have `SRCS` as a prerequisite. In addition, the recipe should make use of existing variables and **automatic variables** that refer to the target and prerequisites to avoid repeating yourself (e.g. you shouldn't be referring to `SRCS` or `BIN` in the compilation command).
   - There is a phony `all` target that builds the output executable binary.
   - There is a phony `clean` target that removes the output executable binary.
   - The `all` target should run when `$ make` is run (without a target specified).

5 Conclusion

1. Add and commit any changes you intend to submit.
2. Fill out the `REPORT` file in the following steps:
3. On the first line provide an integer time in minutes of how long it took for you to complete this assignment. It should just be an integer: no letters or words.
4. On the second line and beyond, write down what you learned while doing this assignment. If you already knew how to do all of this, put down "N/A".
5. Commit your `REPORT` file and push your commits to your remote.