

Advanced 7

Build systems

EECS 201 Fall 2020

Submission Instructions

This assignment will be submitted as a repository on the [UMich GitLab server](#). Create a Project on it with the name/path `eecs201-adv7` and add `brng` as a Reporter. The repository should have the following directory structure, starting from the repository's root:

```
/
|-- report.txt
|-- Makefile
|-- CMakeLists.txt
```

Preface

First, initialize a new Git repository and set up its remote appropriately. This repository will simply be used as a mechanism to submit a few files for this assignment.

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

```
$ wget https://www.eecs.umich.edu/courses/eecs201/files/assignments/adv7.tar.gz
```

These starter files are not to be submitted in your repository. They just serve as baseline data for you to work with.

1 A more featured Makefile (7)

In this part you'll be writing a more featured Makefile that extends some behavior of Makefiles written in the basic assignment, now with the addition of an object code and a linking step.

- This Makefile is meant to sit inside of `example-project`.
- All source code files will be in the `src/` directory
- Additional header files for the include path will be in the `inc/` directory. You will need to figure out how to add this directory to the include path for your compilation steps.
- `example-project` is only an example provided to you. When graded, there may be different source code files and subdirectories within the `inc/` and `src/` directories.
- This Makefile should produce object code under a `obj/` directory and mimic the source code's directory structure. For example, `src/sourcelib/coolthing.cpp` would have its corresponding object code file at `obj/sourcelib/coolthing.o`. This Makefile should handle creation of this directory. `mkdir -p` can help out with this (check out the manpage!).
- This Makefile should have and use a `CXX` variable set to `g++` for the compiler.
- This Makefile should have and use a `BIN` variable set to `app` for the output executable.
- This Makefile should have target dependencies set up so that individual object code targets can be run and the output executable target can be run to trigger the building of all dependencies.
- This Makefile should have a **phony** `all` target that builds the output executable. This target should have any necessary dependencies.

- This Makefile should have a **phony run** target that runs the output executable. This target should have any necessary dependencies i.e. **make run** from a clean state should not error out and should build the output executable before running it.
- This Makefile should have a **phony clean** target that deletes the **obj/** directory and the output executable.

When you're done, add and commit your Makefile to the submission repo.

2 Using another build system (3)

In lecture we alluded to the existence of other build systems. CMake is another tool that helps generate data needed for other build systems Make. Look up how to work with CMake and create a **CMakeLists.txt** file that can handle projects with the directory structure of **example-project**. The requirements will be much more lax for this part: this **CMakeLists.txt** file just needs to build an output application called **app**.

Add and commit your Makefile to the submission repo.

3 Conclusion

1. Commit your Makefile and/or your **CMakeLists.txt** file(s) to your submission repo.
2. Create a file called **report.txt**.
3. On the first line provide an integer time in minutes of how long it took for you to complete this assignment.
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. Add and commit this **report.txt** file. Push your repo.