Research Computing: Practical session 3

1 Using Makefiles

Download `https://www-internal.lsc.phy.cam.ac.uk/pmb39/Compiling_Example.tar.bz2` and un-tar it.

You can compile two executables manually using:

```
g++ library.C -c -o library.o
g++ example1.C -c -o example1.o
g++ example1.o library.o -o example1 -lcblas
g++ example2.C -c -o example2.o
g++ example2.o library.o -o example2 -lcblas
```

and then run them as `./example1` and `./example2`.

Your job is to write a `Makefile` that will do this for you.

In order to check that your `Makefile` has the correct dependencies, try the following:

1. `make example1` or `make example2` should compile each executable separately.
2. `make all` should compile both executables (if they do not already exist)
3. `make clean` should delete object files and executables.
4. Run `touch library.C` then `make all`. This should *only* recompile `library.o` and then relink the two executables.
5. Run `touch example1.C` then `make example1`. This should *only* recompile `example1.o` and then link `example1`.
6. Run `touch library.H` then `make all`. This should recompile all three object files and relink two executables.

You may need a few attempts to get the above to work consistently with all dependencies.

Note: `touch file` simply changes the last-modified time of `file` to be now. This has the same effect as editing the file and saving changes.

2 Using git

Try the following:

1. Initialize a new `git` repository in the `Compiling_Example` directory.
2. Add the original `.C` and `.H` files and commit them to the repository.
3. Add your `Makefile` and commit this separately to the repository.
4. Write a Bash script `test_makefile.sh` that checks whether the 6 test conditions for the `Makefile` are satisfied. (Hint: Use `make all` and `wc -l` to check how many commands have been run.)

5. Commit this script to the repository.

6. Create a file `.git/hooks/pre-commit` that calls `test_makefile.sh`. Make the `pre-commit` file executable.

7. Make any simple change to one of the `.C` files and commit the change. You should see that `git` now checks your `Makefile` before performing the commit.

8. Change the `Makefile` so that it does not pass the above tests.

9. Try commiting the changed file. You should be unable to do so as the test fails.

Note that the client-side hooks described above are not committed to the repository. Ideally, this sort of thing should be implemented as a server-side test instead. However, this is dependent on the remote system (e.g. GitHub/GitLab/etc.).