Advanced Homework 5

Assigned: Friday, February 5, 11:00AM

Due: Before the first lecture on Friday, February 19

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.

1 Automated Background Testing with Git Hooks

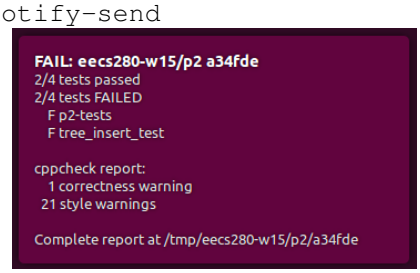
As projects grow, the number and complexity of test cases grows as well. While it's generally a good idea to run all of your test cases, it can be annoying to sit around and wait for a long test case that tests a part of your program that (you think) you didn't touch.

Hooks and scripting to the rescue! The goal is to write a post-commit hook that runs in the background. This script will make sure everything builds correctly, run all of your tests, and, as a bonus, run an external correctness checker that can help find mistakes.

First read over everything your script should do:

1. Create a directory in /tmp that is unique for this project if one does not already exist
   - *Hint: The project's current directory is already unique*
2. Assuming the directory you created above is at `$PROJ_TMP_DIR`, run
   - `git clone --local . $PROJ_TMP_DIR/$(git describe --always)`
3. Call a second script that runs in the backgound to execute the remaining tasks. This second script should do all of its work in the clone you just made.
   - *Hint: You will likely need to pass arguments to this secondary script*
4. Call `make` to build the project. You should save all of the regular output to a file named `build.log` and any errors to a file named `build.error`
   - If the build fails, your script should stop and notify that the build failed. See the notify section below.
5. Run every test case for this project
   - Your script should assume that any executable file with "test" in the name is a test case. It should **not** hard-code a list of tests to run
   - Your script should save the regular output and the error output of each test case to unique files.
   - Your script should assume that test cases return 0 when they pass and non-zero otherwise.
6. Run the `cppcheck` utility for this project
   - *You will need to install this utility first (`sudo apt-get install cppcheck`), your script may assume that cppcheck is already installed*
   - `cppcheck` is a style and correctness checker (it is a *linter* and a *static analysis* tool).
   - You should invoke `cppcheck` as "`cppcheck --enable=all .`" (try it out!)
   - You should again save the regular output and error output into separate files.

7. Generate a nice report when everything is done and alert the user using `notify-send`
   - Try running `notify-send "I am a title" "And I am a body"`
   - This is an example of my notification. You don't need to match my formatting, simply include in the notification what you think is useful. At a minimum it should include (a) how many test cases passed, and (b) if the style checker reported any warnings.



Notice that this hook is **generic**. It should work in any repository simply by copying both scripts into the `.git/hooks` directory.

To get started, use the EECS 280 W15 repository you created for Homework 4.

Create a directory to hold the hooks you're working on

- `mkdir ~/hooks`

This is going to be a kinda complicated script, so it's a good idea to put it under version control as well.

- `cd ~/hooks`
- `git init`

Inside this directory, create a file named `post-commit` with the following contents. Also be sure to make this script executable.

```
#!/usr/bin/env bash

echo "Hello, I am your commit hook running."

echo "Number of arguments I received: $#"
echo "Argument[0] (the program being run): $0"
echo "Argument[1]: $1" # This is empty if no arguments were passed
echo "Argument[2]: $2" # This is empty if only one argument was passed

# Notice what directory this script is executed from. This is important
# when you try to call your helper script
echo "$(pwd)"

sleep 10s && echo "It is annoying to wait for long commit hooks to finish"

sleep 10s & echo "We can put them in background them however so we don't have to wait"

notify-send "Test Message" "Blocking part of the commit hook finished"

echo "Notice that the second sleep is still running, we can tell by: $(pidof sleep)"
```

Don't forget to add and commit the starter code

- `git add post-commit`
- `git commit`

Now, go to the git repository you created in Homework 4 and install this hook

- `cd ~/eecs280-w15/p2/.git/hooks  # Or wherever you put this`
- `ln -s ~/hooks/post-commit .`

The command `ln -s` creates a symbolic link, or symlink. Recall symlinks are just like pointers. This design lets us install the same commit hook into multiple repositories just by pointing to it. If we ever update or make improvements to the hook, all of the repositories that use it will automatically get upgraded.

All right, we are finally set up. Make a new commit and test out the new hook!

Some final tips

- Build up your solution in parts. Get step 1 working then commit it. Then onto step 2.
- Steps 1 and 5 are probably the most challenging. Step 6 is almost an exact copy of step 4.
- You, of course, do not need to actually implement project 2 from EECS 280. I recommend something like this:

```
ppannuto@ppannuto-c4cs-vm:~/hw4> git log --oneline -n1 -p
dfaeb04 Force filter_test to always succeed
diff --git a/filter_test.cpp b/filter_test.cpp
index eebeb87..fa1bc1e 100644
--- a/filter_test.cpp
+++ b/filter_test.cpp
@@ -20,6 +20,7 @@ bool isPrime(int x)

 int main()
 {
+    return 0;
     int numbers[] = { 3, 20, 46, 43, 9, 17, 103, 102 };
     const int numSize = sizeof(numbers) / sizeof(int);
     int primes[] = { 3, 43, 17, 103 };
```

Submission checkoff:

- ☐ Explain what the clone command from step 2 does.
- ☐ Show off your hook working
  - ☐ With all passing test cases
  - ☐ With some failing test cases
  - ☐ Show how to find and examine the output from a failing test case
- ☐ Show that your hook is generic by installing it (symlinking it) into a repository for an EECS class you are currently taking
  - If you aren't in a programming EECS class, grab any other 280 project from W15.