Using Git for ECE-486 Labs
Spring 2017

An ECE Git server has been set up to facilitate collaboration within lab groups and to allow TA/Instructor access to lab work. The server is located at

https://gitlab.eece.maine.edu

Lab group "projects" have been set up with lab team members given read/write access to the projects. *All code submitted for grading must be uploaded to the Git repository for the lab group by the due-date for the assignment.* TAs will only grade the copy of the assignment which has been pushed to the repository before the deadline.

A great introduction to the use of Git can be found in Scott Chacon's "Pro Git" book, which is published by Apress and is available online at

http://git-scm.com/book

Look here (and especially in Chapter 2) for the real details on how to install and use Git.

The remainder of this document gives some essential commands for using Git to support your labs for ECE-486.

Initial Setup

1. Log into the https://gitlab.eece.maine.edu server.

   If you have not already, log into the Git repository server using your MaineStreet credentials. You'll need to log in once in order to register your name on the system so that TAs can add you to your appropriate lab group team.

   Once you've logged in, check to see if you've been added to the team for the software project for your lab group. The project name will be something like "`ece486_2017_group_04`". If you don't see the project, see a TA and ask them to add you to the team. Then re-check.

2. Set up your "Git identity" that Git uses when you access the system:

   ```
   git config --global user.name "John Doe"
   git config --global user.email johndoe@example.com
   ```

3. Set up an SSH Key to allow connection between your computer and gitlab:

   Log into the server and check the "SSH Keys" tab. Follow the directions.

Using gitlab

1. Clone the repository for your lab group:

   "cloning" a repository creates your own local copy of your group's work. You can make any changes you want to your copy, but your group (and the TAs) won't know anything about it until you register all your changes back into the repository (so read on to the next steps!). Assuming you've been added to "group 4", the clone command is similar to:

   ```
   git clone git@gitlab.eece.maine.edu:donald.hummels/ece486_2017_group_04.git
   ```

   You should now have a directory named `ece486_2017_group_04`, with sub-directories for each lab, and (if your lab partners have been at work) the most recent content created by your lab group.

2. Add files to the "tracked" list of files.

   Git will track changes on all files in the repository. If you've created a new file in the directory structure that should also be tracked (and ultimately included in the repository), you'll need add the file to the list of files being tracked. Here are a few example commands:

   ```
   git add newfile.c
   git add *.c *.h
   ```

3. If you like the changes and/or additions, "commit".

   Once you have a collection of changes (or file additions) that you're happy with, you should "commit" the changes to your local copy of the repository. The changes and additions are all logged, and you will be able to recover the committed version at any later date. The commit command will require you to specify a message so that you can describe what is being modified. To use your default editor to enter the message, type

   ```
   git commit -a
   ```

   Alternatively, you can specify a short message on the command line, as in:

   ```
   git commit -a -m "I fixed everything and lab 2 works perfectly now!"
   ```

4. "push" your changes back to the repository.

   So far, only your local copy has been modified. To let your lab partners (or the graders) see your work, you need to send your changes back to the repository.

   ```
   git push
   ```

There are a few other essential tasks that you'll need to be familiar with:

Updating your local copy: Let's assume your lab partner has been at work and has managed to push some changes back to the repository. Your local copy should be brought up to date so that you can see her work, and make additional changes. The "pull" command will grab the most recent version of the repository, and try to merge them with any changes that you have made. (Hopefully, you and your partner have not been editing the same lines of the same file... if so, you'll have to resolve the differences.)

```
git pull
```

We are using the "Centralized Workflow" model described in Chapter 5 of the Pro Git book (http://git-scm.com/book/en/Distributed-Git-Distributed-Workflows). If you and your lab partner are both working on cloned versions of the repository at the same time, the first person to "push" their changes to the repository will have no trouble. The second person will not be able to push their changes without some extra work: They'll be required to "pull" the new version from the repository and merge it with their own work (and commit) before being allowed to push a new version back to the repository.

Taking inventory of any changes you've made: To see changes made in your copy since the last "commit", use: "`git status`".

Changing file names and locations: You can remove a file: "`git rm filename`". To move a file to a new location: "`git mv oldfilename newfilename`".

Displaying the Commit history: Here's where the commit messages come in handy. To see who has changed what and when, use: "`git log`".

Checking the repository

Especially if you're new to Git, it's worth taking the time to make sure that your changes have been uploaded to the repository, and are available to your other lab partners (and to the graders). Log into the server (https://gitlab.eece.maine.edu) and open your lab group project. You should be able to view the code, and look over any changes that have been made by any of your lab partners.