Working with remotes

In this section we will work through some of the commands required to use a remote Git repository. To start with you need to go to

https://github.com/

and sign up for a free account

You should now be presented with a webpage with the github bootcamp menu prominent.

Choose option 2 – Create repositories and the follow the on-screen instructions

Well done, you now have a remote repository that you can store you university work in if you want! Make sure that you keep github open in a web browser.

To see the remote connections in a given repository use the command

$ git remote

As we only have the one remote connection this should return

$ git remote

origin

If you want to see the URL Git has stored for that short name use the verbose method

$ git remote -v

Lets try it out using our remote repository. In your working directory create a new file called helloWorld.c and paste in the code

```
#include<stdio.h>
main()
{
   printf("Hello World\n");
}
```

Save, add and commit helloWorld.c with an appropriate message.

Now modify the program so it is reads

```
#include<stdio.h>
main()
{
   printf("Hello World\n");
   printf(�Hello You!\n�);
}
```

Again save, add and commit.

Now you should have two local commits but the remote repository still only contains the original commit, lets rectify this now.

Do

$ git push origin master

It will likely ask for your github username and password, enter them if prompted to do so. Now go back to your web browser and into you Hello-World repository. You should see that in addition to the README.md you originally created you can also see helloWorld.c. If you click on the filename you should see the latest version (with both printf�s). Just above the button you should see the number of commits in the current repository, it should say 3. Note that even though you have only pushed to github twice it has actually uploaded all of your local commits. If you click on the 3 commits link it will take you to a page showing your commit messages with the latest first (same as doing log on your machine). From here you can explore the data from previous commits by using the links on the right.

Find the second commit you did and click on the browse code link. You will now be taken to the main repository page again but instead of showing 3 commits it now only shows 2 and when you click on helloWorld.c it only shows the one printf statement.

The easiest way to return to the current version is to click on the <>Code option in the right hand side menu.

Create a new github directory for your maths_function.c and the upload your maths_functions.c to it.

Create a new local folder called maths_branching and cd into it and run

git clone https://github.com/<username>/foldername

Branching

We have touched on branching a briefly in the lectures and we will be covering it in more depth in the lecture next week but lets try some basic branching now.

cd into maths_branching/foldername

$ git branch testing

You have now created a new branch to allow you to work to your hearts content on experimental functions whilst not impacting the integrity of the master copy.

$ git branch should return

* master

  testing

The * indicates the current branch you are working on, you must be careful as simply creating a branch does not switch to that branch automatically. To do that we use

$ git checkout testing

Now modify your maths_function.c by adding in new functionality to take two numbers and work out the remainder of the first from the second. Save, add and commit your new change.

Do $ git log and you will see all of the previous commits. Take this opportunity to look at the log graph as in lab 2. You should see that it is still just a straight line. Now do

$ git checkout master

Look at the logs again and you will not see the commit from the testing branch, the master branch knows nothing about it at this point.

Now make a trivial change in maths_function.c save, add and commit it. We are now starting to diverge our branches but actually we want to bring them back together again. To do this we merge them

$ git merge testing

You will then be prompted to include a merging message, you can enter one or use the default message.

If we do $ git log we can now see all of the commits from both branches and the commit merging the two together.

Now do $git log --graph and you will be able to see the branches diverging and rejoinging after the merge. It is important to note that when the branches are separate the graph will merely be a straight line showing all of the commits known to the current branch only.