Software Methods and Tools

Fall 2015
Instructor: Yongjie Zheng

Lab #9: GIT

Description:

In this lab, we are going to work on the GIT version control system. The specific tasks are similar to the tasks of the Subversion lab. In short, we will create a remote repository and two local repositories. Based on them, we will try a number of basic operations of GIT, including staging, committing, branching, merging, pushing, pulling, and tracking branches. During this process, some questions are also given for you to further understand GIT.

Tasks:

1. Running GIT and first-time setup.

The GIT application can be started from Start > Specialized Programs > GIT > GIT Bash on lab machines.

Once the command line window pops up, type in

git config --global user.name "YOUR NAME"

git config --global core.editor emacs

2. Setting up a remote repository and doing an initial "push".

Go to the Desktop folder in the Q drive, and create a folder there named myGitRemote.git (mkdir myGitRemote.git).

Enter the myGitRemote.git folder (cd myGitRemote.git).

Initialize the folder as a GIT repository (git init --bare)

Check the content of the myGitRemote folder to see what a GIT repository looks like (ls).

Next, we are going to create a local repository and push its content to the remote repository, which is myGitRemote.git.

Go back to the Desktop folder (cd ..).

Create another folder named myGitLocalA (mkdir myGitLocalA).

Go to the new folder (cd myGitLocalA), and initialize a GIT repository there (git init).

Open the Windows file browser, copy and paste the Lunar Lander source code (three .java files) into the myGitLocalA folder.

Now tracking these files with GIT (git add *), and do an initial commit (git commit -m “Initial Import”).

All the files have been saved into your local repository.

Next, add the remote repository into the list of your remote servers (git remote add origin /q/Desktop/myGitRemote.git).

Push your local repository to the remote repository (git push origin master).

Done!

3. Cloning the remote repository and creating a new branch.

Now let’s create another local repository, assuming that it will be used by a different developer.

Go back to the Desktop folder (cd ..).

Clone the remote repository we created earlier (git clone file:///q/Desktop/myGitRemote.git myGitLocalB).

Go to the myGitLocalB directory (cd myGitLocalB), and check its content (ls). All three Java files should be there.

Create another branch (e.g. named development) and switch to it (git checkout -b development).

Open the file SpaceCraft.java, make some changes, save and close the file.

Check you current status (git status).

Stage the file SpaceCraft.java to be committed (git add SpaceCraft.java).

Commit your changes (git commit -m “Developing in progress.”).

Push this branch to the server (git push origin development).

Now switch back to the master branch (git checkout master).

Open the file SpaceCraft.java again. Are your changes still there? Why?

4. Remote Branching and Merging.

Go back to your first local working directory (cd /q/Desktop/myGitLocalA).

Check the information of the remote repository first (git remote show origin). Do you understand the returned information?

Let’s fetch the latest data from the repository first (git fetch origin).

Now you should have two remote branches: origin/master and origin/development.

Next, we are going to create a tracking branch from the remote branch origin/development (git checkout -b development origin/development).

Open the file SpaceCraft.java. Do you see the changes you just made? Again, why?

Now we are going to merge the branch development to the master branch.

(git checkout master)

(git merge development)

5. Resolving Conflicts.

In this last task of the lab, you need to design a scenario where merging conflicts actually happen. Resolve your conflicts. You scenario must involve remote branches.

END.