MIDPS LAB# 1

Cojocaru Andreea - Cristina

March 15, 2017

1 Tasks

1. Mandatory tasks:
   - connect to a server using SSH
   - run at least 2 sample programs from provided HelloWorldPrograms set
   - configure your VCS
   - initialize an empty repository
   - create branches (create at least 2 branches)
   - commit to different branches (at least 1 commit per branch)
2. Optional tasks:
   - set a branch to track a remote origin on which you are able to push
   - reset a branch to previous commit
   - merge 2 branches
   - create a meaningful pull request
   - GIT rebase
   - GIT hooks

1.1 Running programs from terminal.

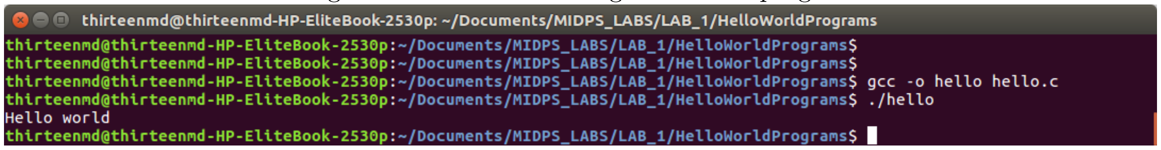
In order to run the problems, first we should open the terminal and go to our directory where the programs are located, in my case its in the folder "HelloWorldPrograms". To run the program written in the "C" language, we type the following commands:

```
gcc -o hello hello.c
./hello
```

For the program written in C++ the commands are the following:

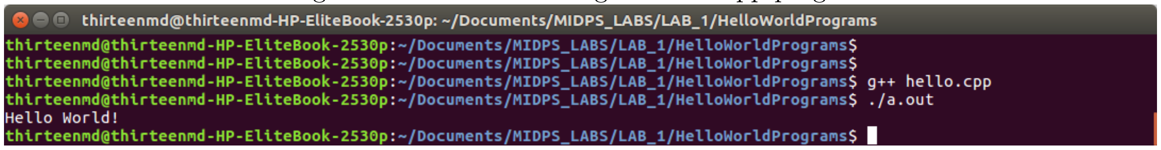
```
g++ hello.cpp
./a.out
```

Figure 1: Result of running the hello.c program



```
thirteenmd@thirteenmd-HP-EliteBook-2530p: ~/Documents/MIDPS_LABS/LAB_1/HelloWorldPrograms
thirteenmd@thirteenmd-HP-EliteBook-2530p:~/Documents/MIDPS_LABS/LAB_1/HelloWorldPrograms$
thirteenmd@thirteenmd-HP-EliteBook-2530p:~/Documents/MIDPS_LABS/LAB_1/HelloWorldPrograms$
thirteenmd@thirteenmd-HP-EliteBook-2530p:~/Documents/MIDPS_LABS/LAB_1/HelloWorldPrograms$ gcc -o hello hello.c
thirteenmd@thirteenmd-HP-EliteBook-2530p:~/Documents/MIDPS_LABS/LAB_1/HelloWorldPrograms$ ./hello
Hello world
thirteenmd@thirteenmd-HP-EliteBook-2530p:~/Documents/MIDPS_LABS/LAB_1/HelloWorldPrograms$
```

Figure 2: Result of running the hello.cpp program



```
thirteenmd@thirteenmd-HP-EliteBook-2530p: ~/Documents/MIDPS_LABS/LAB_1/HelloWorldPrograms
thirteenmd@thirteenmd-HP-EliteBook-2530p:~/Documents/MIDPS_LABS/LAB_1/HelloWorldPrograms$
thirteenmd@thirteenmd-HP-EliteBook-2530p:~/Documents/MIDPS_LABS/LAB_1/HelloWorldPrograms$
thirteenmd@thirteenmd-HP-EliteBook-2530p:~/Documents/MIDPS_LABS/LAB_1/HelloWorldPrograms$ g++ hello.cpp
thirteenmd@thirteenmd-HP-EliteBook-2530p:~/Documents/MIDPS_LABS/LAB_1/HelloWorldPrograms$ ./a.out
Hello World!
thirteenmd@thirteenmd-HP-EliteBook-2530p:~/Documents/MIDPS_LABS/LAB_1/HelloWorldPrograms$
```

1.2 Connect to a server using SSH. Configure your VCS. Initialize an empty repository.

Because I decided to use SSH to connect my laptop to git, I combined those tasks. First we need to install git, we open the terminal, then we type the following:

```
sudo apt install git
```

then we type,

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

thus we introduced ourselves to git. After that we type:

```
ssh-keygen -t rsa -b 4096 -C "my_email@example"
```

of course I used my actual email instead of the example above. This code generates a public/private rsa key pair. When we are prompted to ”Enter a file to save the key”, we press Enter, this accepts the default location. The we should enter a secure passphrase(be sure to remember it!).
After that, we type:

```
eval "$(ssh-agent -s)"
```

to start the ssh-agent in the background. Then we type:

```
ssh-add ~/.ssh/id_rsa
```

After that we copy the SSH key to the clipboard using the commands:

```
sudo apt-get install xclip
xclip -sel clip <~/.ssh/id_rsa.pub
```

then we open the github web-page, log in, click on the profile photo and select Settings. There in the user sidebar click SSH and GPG keys. Click New SSH key, in the title field we should add a descriptive for our key, for example if you yuse your personal MacBook, you might call it ”Personal MacBook”. Paste your key in the Key field and click Add SSH key. Then we type:

```
ssh -T git@github.com
```

if the result is:

```
Hi username! You've successfully authenticated, but GitHub does not provide shell access.
```

then everithing worked. To initiate an empty repository we use the command:

```
git init
```

To clone an already existing repository we use the command:

```
git git clone username@hostname:/path/to/repository
```

To set your remote's URL we use:

```
git remote set-url origin git@github.com:USERNAME/OTHERREPOSITORY.git
```

1.3 Branches

To create a branch in git we use the command:

```
git checkout -b [name_of_your_new_branch]
```

Push the branch on github:

```
git push origin [name_of_your_new_branch]
```

You can see all branches created by using:

```
git branch
```

To switch though branches we use:

```
git checkout [name_of_your_new_branch]
```

So, I created 2 branches: branchOne and btanchTwo, I added a file to each branch and pushed the changes to github. Then I added another file to branchTwo. Now to reset branchTwo to an older commit I will type:

```
git reset 6313b09

git reset --soft HEAD@{1}

git commit -m "Reversed to 6313b09"

git reset --hard

git push origin branchTwo
```

where 6313b09 is the commit code of the older commit. To set a branch to track a remote origin we should type:

```
#we set the remote if we didn't before
git remote add origin ssh://...

#now configure master to know to track
git config branch.master.remote origin
git config branch.master.merge refs/heads/master

#and push:
git push origin master
```

To merge two branches we type:

```
#we switch to the branch to be merged with(in my case master)
git checkout master

git merge branchTwo
git push origin master

#to delete branchTwo we type:
git push origin --delete branchTwo
git branch -d branchTwo
```

1.4 Create a meaningful pull request

So you made changes to a project a bugfix or maybe a new feature and you want to send it for inclusion in the official (upstream) sources. Perhaps you sent an email or opened an issue in the bugtracker, and the project maintainers asked you to send a Pull Request (PR) on GitHub.

1.4.1 Fork the repository

Forking a repository on GitHub means creating your own Git repository, which is a copy of the original. In order to Fork a repository we need to visit it, then in the upper-right theres a button named Fork. It also shows a number: how many times this repository was forked by other people). Press it, and it will create your own copy of the pull-request-tutorial repository, at `https://github.com/YOUR_USERNAME/pull-request-tutorial` (the real URL will, of course, contain your own username).

1.4.2 Download your fork and create a branch

Now, its time for you to make your changes in the source code (your bugfix or new feature). Start by downloading your repository to your computer. Go to the terminal, make sure git is installed in your computer and type:

```
git clone https://github.com/YOUR_USERNAME/pull-request-tutorial.git
```

This will download the files and create a directory called pull-request-tutorial that is linked to your fork (i.e. the copy of the repository under your control).

To avoid trouble later, lets create a new branch in our repository so that the work on our bugfix or feature is stored separately. Pick a meaningful name that represents the changes you plan to make in your code. In our example, Ill call it fix-typo:

```
git checkout -B fix-typo
```

1.4.3 Make your changes in your fork

You can make all the changes you deem needed to you fork, from implementing new features to deleting useless ones. after all the changes are made, you push them on github.

1.4.4 Make the Pull Request

In your repository page, the next time you open the page after pushing to a new branch, there is a big green button saying Compare & pull request. Click on it.
This will open a page in which you'll be able to further edit the description for your proposed changes. Write down a nice report explaining why these changes should be included in the official sources of your project, and then confirm.
The project authors will receive an email notification that you sent them a PR. Then its their turn to read it and comment. You will get notifications when they comment. If they suggest any changes to

your bugfix or feature, go back to Step 3, edit it and push again: your Pull Request will be automatically updated. If they are happy with the changes and want to integrate your contributions to the project, the maintainers will click Merge and your code will become part of the original repository!

1.5 GIT rebase

It's simple, with rebase you say to use another branch as the new base for your work.

If you have for example a branch master and you create a branch to implement a new feature, say you name it cool-feature, of course the master branch is the base for your new feature.

Now at a certain point you want to add the new feature you implemented in the master branch. You could just switch to master and merge the branchOne branch:

```
$git checkout master
$git merge branchOne
```

but this way a new dummy commit is added, if you want to avoid spaghetti-history you can rebase:

```
$git checkout branchOne
$git rebase master
```

and then merge it in master:

```
$git checkout master
$git merge branchOne
```

1.6 GIT hooks

Git hooks are scripts that Git executes before or after events such as: commit, push, and receive. Git hooks are a built-in feature - no need to download anything. Git hooks are run locally.

These hook scripts are only limited by a developer's imagination. Some example hook scripts include:

1. pre-commit: Check the commit message for spelling errors.
2. pre-receive: Enforce project coding standards.
3. post-commit: Email/SMS team members of a new commit.
4. post-receive: Push the code to production.

2 Conclusion

When we are working on a project I thing big problem is connection to your project from everywhere and by your collogue. So this laboratory work helped us to find a solution how we should keep our project files online and to the remote server. And the most beautiful advantage with GIT everyone work on the same project at the same moment but nobody lost and files or coding with this way. VCS is a wonderful tool which allow you to control your project, work with a team of developers and to track all the changes which was ever made. We've set a good background on VCS and Git in particular. So we have a lot of stuff to learn with this way and with this laboratory works.