

FACULTATEA CALCULATOARE, INFORMATICĂ ȘI MICROELECTRONICĂ

UNIVERSITATEA TEHNICĂ A MOLDOVEI

MEDII INTERACTIVE DE DEZVOLTARE A PRODUSELOR SOFT

LUCRARE DE LABORATOR #2

VERSION CONTROL SYSTEMS ȘI MODUL DE SETARE A UNUI SERVER

Autor:

st. gr. TI-141

Valeria BALINSCHI

lector asistent:

Irina COJANU

lector superior:

Svetlana COJOCARU

1. Scopul lucrării

Însușirea noțiunii de Version Control Systems și a modului de setare a unui server.

2. Obiectivele lucrării

- a) Înțelegerea și folosirea CLI (basic level)
- b) Administrarea remote a mașinilor linux machine folosind SSH (remote code editing)
- c) Version Control Systems (git || mercurial || svn)
- d) Compilează codul C/C++/Java/Python prin intermediul CLI, folosind compilatoarele gcc/g++/javac/python

3. Efectuarea lucrării de laborator

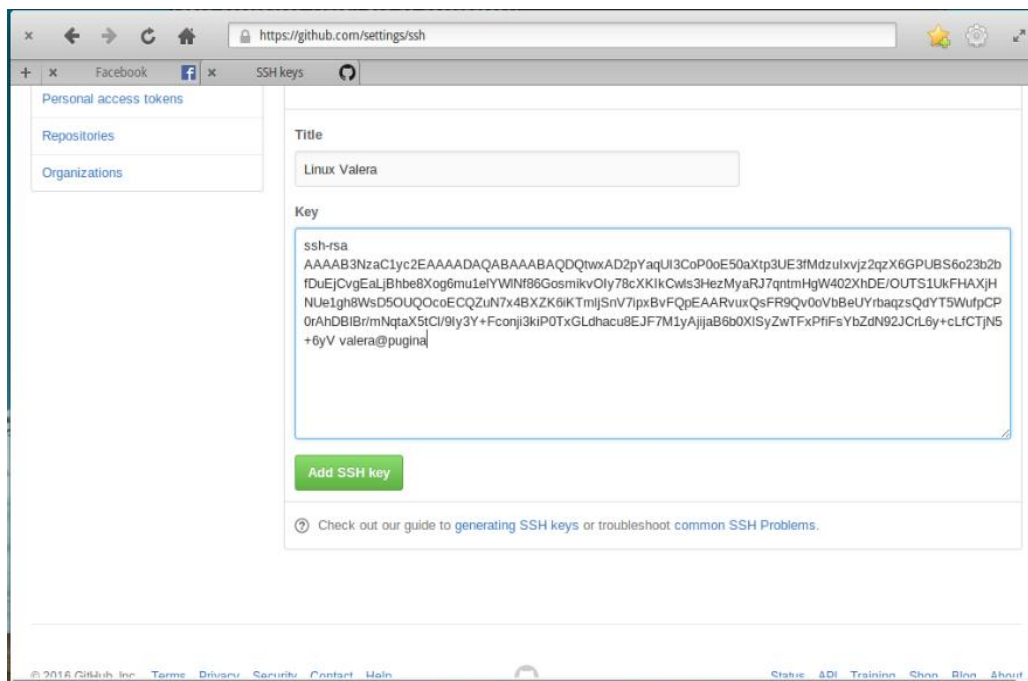
3.1. Task-uri implementate

- *Basic Level* (nota 5 || 6):
 - conectează-te la server folosind SSH
 - compilează cel puțin 2 sample programs din setul HelloWorldPrograms folosind CLI
 - execută primul commit folosind VCS
- *Normal Level* (nota 7 || 8):
 - inițializează un nou repository
 - configurează-ți VCS
 - crearea branch-urilor (creează cel puțin 2 branches)
 - commit pe ambele branch-uri (cel puțin 1 commit per branch)
- *Advanced Level* (nota 9 || 10):
 - setează un branch to track a remote origin pe care vei putea sa faci push (ex. Github, Bitbucket or custom server)
 - resetează un branch la commit-ul anterior
 - merge 2 branches
 - conflict solving between 2 branches
- *Bonus Point*:
 - Scrie un script care va compila HelloWorldPrograms projects: c, cpp, java, python, ruby.

3.2. Realizarea lucrării de laborator

- *Basic Level* (nota 5 || 6) + *Bonus Point*:
 - conectează-te la server folosind SSH

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/valera/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/valera/.ssh/id_rsa.
Your public key has been saved in /home/valera/.ssh/id_rsa.pub.
The key fingerprint is:
95:53:bb:c1:67:43:f7:95:aa:68:04:4b:15:c9:5c:1d valera@pugina
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      +o+.oEo + |
|      o + + + oo|
|      . o + + = .|
|      . o . * . |
|      S . o     |
|      o .       |
|      .         |
+-----+
valera@pugina:~/git$ cat /home/valera/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDQtwxAD2pYaqUI3CoP0oE50aXtp3UE3fMdzulxvjz2qzX6GPUBS6o23b
2bfDuEjCvgEaLjBhbe8Xog6mu1eYwlnf86Gosmikv0Iy78cXKIkCwls3HezMyaRJ7qntmHgW402XhDE/OUTS1UkFHAXjH
NUe1gh8WsD5OUQOcoECQZuN7x4BXZK6iKtm1jSnV7ipxBvFQpEAARvuxQsFR9Qv0oVbBeUYrbaqzsQdYT5WufpCP0rAhDB
lBr/mNqtaX5tCl/9Iy3Y+Fconji3kiP0TxGLdhacu8EJF7M1yAjijaB6b0XlSyZwTFxPfiFsYbZdN92JCrL6y+cLlCTjN5
+6yV valera@pugina
valera@pugina:~/git$
```



- compilează cel puțin 2 sample programs din setul HelloWorldPrograms folosind CLI

```
#include <stdio.h>

int main()
{
    printf("Hello World!\n");
    return 0;
}
```

hello.c

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello World!" << endl;
    return 0;
}
```

hello.cpp

```
GNU nano 2.2.6 File: hello.py
print "Hello World!"
```

hello.py

```
GNU nano 2.2.6 File: hello.rb
puts "Hello World!"
```

hello.rb

```
GNU nano 2.2.6 File: hello.java
class HelloWorld {
    public static void main(String args[]) {
        System.out.println("Hello World!");
    }
}
```

hello.java

```
GNU nano 2.2.6 File: helloscript.sh
#!/bin/bash

gcc -o helloc.out hello.c && ./helloc
g++ -o hellocpp.out hello.cpp && ./hellocpp
javac hello.java && java -cp /home/valera/git/lab2 HelloWorld
python hello.py
ruby hello.rb
```

helloscript.sh

```
root@pugina:/home/valera/git/lab2# ./helloscript.sh
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

Rezultatele rulării programelor

- execută primul commit folosind VCS

```

x newrep: cat
+ x newrep: cat
valera@pugina:~/git/newrep$ nano new.txt
valera@pugina:~/git/newrep$ git status
On branch doi
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   new.txt

no changes added to commit (use "git add" and/or "git commit -a")
valera@pugina:~/git/newrep$ git add .
valera@pugina:~/git/newrep$ git commit -m "mod new"
[doi a17ad78] mod new
1 file changed, 1 insertion(+), 1 deletion(-)

```

- *Normal Level* (nota 7 || 8):

- inițializează un nou repository

```

x newrep: git
+ x newrep: git
valera@pugina:~/git$ mkdir newrep
valera@pugina:~/git$ cd newrep
valera@pugina:~/git/newrep$ git init
Initialized empty Git repository in /home/valera/git/newrep/.git/
valera@pugina:~/git/newrep$

```

- configurează-ți VCS

```

Initialized empty Git repository in /home/valera/git/newrep/.git/
valera@pugina:~/git/newrep$ git config --list
user.email=3vbalinschi@gmail.com
user.name=vbalinschi
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
valera@pugina:~/git/newrep$ git config --global alias.ck "git checkout"
valera@pugina:~/git/newrep$ git config --global alias.adcm "git add . && git commit -m"
valera@pugina:~/git/newrep$ git config --list
user.email=3vbalinschi@gmail.com
user.name=vbalinschi
alias.ck=git checkout
alias.adcm=git add . && git commit -m
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
valera@pugina:~/git/newrep$

```

- crearea branch-urilor (creează cel puțin 2 branches)
- commit pe ambele branch-uri (cel puțin 1 commit per branch)

```

x newrep: git
+ x newrep: git
valera@pugina:~/git/newrep$ git checkout -b unu
Switched to a new branch 'unu'
valera@pugina:~/git/newrep$ echo "unu" >> unu.txt
valera@pugina:~/git/newrep$ git add .
valera@pugina:~/git/newrep$ git commit -m "unu"
[unu d9fdcbc] unu
1 file changed, 1 insertion(+)
create mode 100644 unu.txt
valera@pugina:~/git/newrep$ git checkout -b doi
Switched to a new branch 'doi'
valera@pugina:~/git/newrep$ echo "doi" >> doi.txt
valera@pugina:~/git/newrep$ git add .
valera@pugina:~/git/newrep$ git commit -m "doi"
[doi f96c565] doi
1 file changed, 1 insertion(+)
create mode 100644 doi.txt
valera@pugina:~/git/newrep$ git branch
* doi
  master
  unu
valera@pugina:~/git/newrep$

```

- *Advanced Level* (nota 9 || 10):

- setează un branch to track a remote origin pe care vei putea sa faci push (ex. Github, Bitbucket or custom server)

```

x newrep: git
+ x newrep: git
valera@pugina:~/git/newrep$ git checkout --track -b overlord origin/master
Branch overlord set up to track remote branch master from origin.
Switched to a new branch 'overlord'
valera@pugina:~/git/newrep$ git add .
valera@pugina:~/git/newrep$ git commit -m "add to ner branch"
On branch overlord
Your branch is up-to-date with 'origin/master'.

nothing to commit, working directory clean
valera@pugina:~/git/newrep$ git push origin overlord
Total 0 (delta 0), reused 0 (delta 0)
To git@github.com:vbalinschi/MIDPS.git
 * [new branch]      overlord -> overlord
valera@pugina:~/git/newrep$

```

- resetează un branch la commit-ul anterior

```

x newrep: git
+ x newrep: git
valera@pugina:~/git/newrep$ git log --graph --all --oneline
* 5a523a3 dell
* 4e198be Delete over.t
* 6462e11 hell
* 5946b3e add123
* 2b4b7fd add lab2
* 98cfdaa add readme for lab1
* e657700 delete lololo
* ff94392 add gitignore
* 9f2bf80 Raport1
* 7fb703d Add Lab#1
* 78afe47 first commit
* 455a093 resolve conf
|\
| * 406a5f2 conf
| * 40af614 conf
| * a17ad78 mod new
| * f96c565 doi
|/
* d9fdcbc unu
* 3002c2b add
valera@pugina:~/git/newrep$ git reset --hard
HEAD is now at 455a093 resolve conf
valera@pugina:~/git/newrep$

```

- merge 2 branches

```

valera@pugina:~/git/newrep$ echo "conf = false" >> c.txt
valera@pugina:~/git/newrep$ git add . && git commit -m "conf"
[master 40af614] conf
1 file changed, 1 insertion(+)
create mode 100644 c.txt
valera@pugina:~/git/newrep$ git chekout unu
git: 'chekout' is not a git command. See 'git --help'.

Did you mean this?
    checkout
valera@pugina:~/git/newrep$ git checkout unu
Switched to branch 'unu'
valera@pugina:~/git/newrep$ ls
new.txt unu.txt
valera@pugina:~/git/newrep$ nano new.txt
valera@pugina:~/git/newrep$ echo "conf = true" >>c.txt
valera@pugina:~/git/newrep$ git add . && git commit -m "conf"
[unu 406a5f2] conf
1 file changed, 1 insertion(+)
create mode 100644 c.txt
valera@pugina:~/git/newrep$ git checkout master
Switched to branch 'master'
valera@pugina:~/git/newrep$ git merge unu
Auto-merging c.txt
CONFLICT (add/add): Merge conflict in c.txt
Automatic merge failed; fix conflicts and then commit the result.
valera@pugina:~/git/newrep$

```

- rezolvarea conflictelor a 2 branches

```
GNU nano 2.2.6 File: c.txt
<<<<<< HEAD
conf = false
=====
conf = true
>>>>>> unu
```

```
valera@pugina:~/git/newrep$ git merge unu
Auto-merging c.txt
CONFLICT (add/add): Merge conflict in c.txt
Automatic merge failed; fix conflicts and then commit the result.
valera@pugina:~/git/newrep$ nano c.txt
valera@pugina:~/git/newrep$ git add .
valera@pugina:~/git/newrep$ git commit -m "resolve conf"
[master 455a093] resolve conf
valera@pugina:~/git/newrep$ git status
On branch master
nothing to commit, working directory clean
valera@pugina:~/git/newrep$
```


Concluzie

În urma realizării laboratorului nr.2 la tema: *"Version Control Systems si modul de setare a unui server"*, am însușit modul de utilizare a CLI, de administrarea remote a mașinilor linux machine folosind SSH.

Am efectuat conexiunea la un remote server folosind SSH (drept server remote, am folosit o mașină virtuală). În continuare compilând programele din lista dată și efectuând primul commit, folosind VCS.

Am creat 2 branch-uri asupra cărora am efectuat un commit din nou.

La fel am scris un script *"helloscript.sh"*, prin intermediul căruia am compilat HelloWorldPrograms projects din lista dată: c, cpp, java, python, ruby.

De asemenea am însușit instalarea corectă a Ubuntu pe o mașină virtuală, cât și bazele utilizării comenzilor din Linux.

În timpul efectuării laboratorului am lucrat cu comenzi ca :

- `git init` – crearea unui repository dintr-un fișier existent
- `git remote add origin` – pentru interconectarea repositoryului local cu cel de pe github.
- `git commit -m` – pentru înregistrarea unei schimbări(snapshot), ca `git add`.
- `git config --global` – operațiuni cu fișierul de configurație GIT Bash
- `git checkout` – pentru selectarea ramurii curente de lucru.
- `git status` – informații despre starea fișierelor
- `git mergetool` – ustensilă pentru soluționarea conflictelor ce le poate crea `git merge`
- `git merge` – actualizarea schimbărilor de pe două sau mai multe ramuri.

În concluzie, am pus în practică VCS-ul GIT Bash creând un repository și inițializându-l, clonând repository și efectuând diverse comenzi în el.

Bibliografie

1. <http://www.vogella.com/tutorials/Git/article.html>
2. <http://www.psychocats.net/ubuntu/virtualbox>
3. http://www.manniwood.com/starting_a_project_with_git.html