

FACULTATEA CALCULATOARE, INFORMATICA SI MICROELECTRONICA

UNIVERSITATEA TEHNICA A MOLDOVEI

MEDII INTERACTIVE DE DEZVOLTARE A PRODUSELOR SOFT

LUCRAREA DE LABORATOR#2

Version Control Systems si modul de setare a unui server

Autor:

Chifa VLADISLAV

lector asistent:

Irina COJANU

lector superior:

Svetlana COJOCARU

Lucrare de laborator # 2

1 Scopul lucrării de laborator

Version Control Systems si modul de setare a unui server

2 Obiective

- Intelegerea si folosirea CLI (basic level)
- Administrarea remote a masinilor linux machine folosind SSH (remote code editing)
- Version Control Systems (git — mercurial — svn)
- Compileaza codul C/C++/Java/Python prin intermediul CLI, folosind compilatoarele gcc/g++/javac/python

3 Realizarea lucrarii de laborator

3.1 Tasks and Points

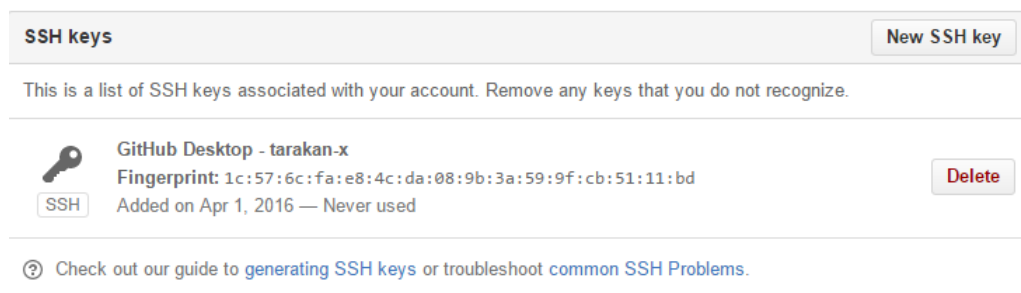
- Basic Level (nota 5 — 6) :
 - conecteaza-te la server folosind SSH
 - compileaza cel putin 2 sample programs din setul HelloWorldPrograms folosind CLI
 - executa primul commit folosind VCS
- Normal Level (nota 7 — 8):
 - initializeaza un nou repository
 - configureaza-ti VCS
 - crearea branch-urilor (creeaza cel putin 2 branches)
 - commit pe ambele branch-uri (cel putin 1 commit per branch)
- Advanced Level (nota 9 — 10):
 - seteaza un branch to track a remote origin pe care vei putea sa faci push (ex. Github, Bitbucket or custom server)
 - reseteaza un branch la commit-ul anterior
 - merge 2 branches
 - rezolvarea conflictelor a 2 branches

3.2 Analiza lucrarii de laborator

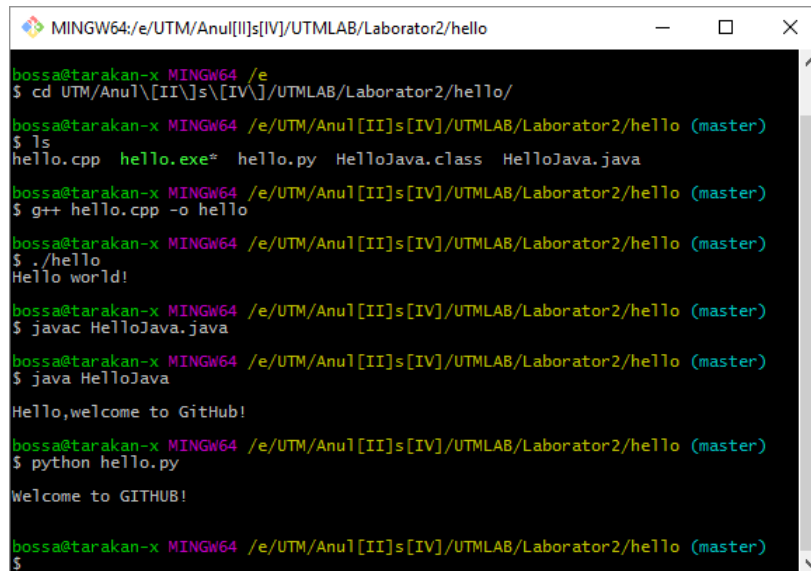
Linkul la repositoryul GITHUB:

<https://github.com/tarakan-x/MIDPS>

Pentru a realiza aceasta lucrare de laborator *m – am* inregistrat pe github.com si am instalat *git – bash*, am generat o cheie SSH si am adaugat aceasta cheie publica pe github pentru a identifica acest calculator.



Pentru a compila programe scrise in *C++*, *Java*, *Python* avem nevoie de a seta directiile spre *g++*, *javac* si *python* in fisierul *bash_profile* din directoriul unde este instalat *Git – Bash*. Pentru a compila programul scris in Java utilizam *javac* pentru compilare si *java* namefile pentru a rula programul nostru, in cazul programuli *C++* utilizam comanda *g++ hello.cpp -o namefile*, si *./namefile* pentru a rula programul si in cazul unui program scris in Python utilizam sintaxa *python namefile.py* pentru a rula programul.



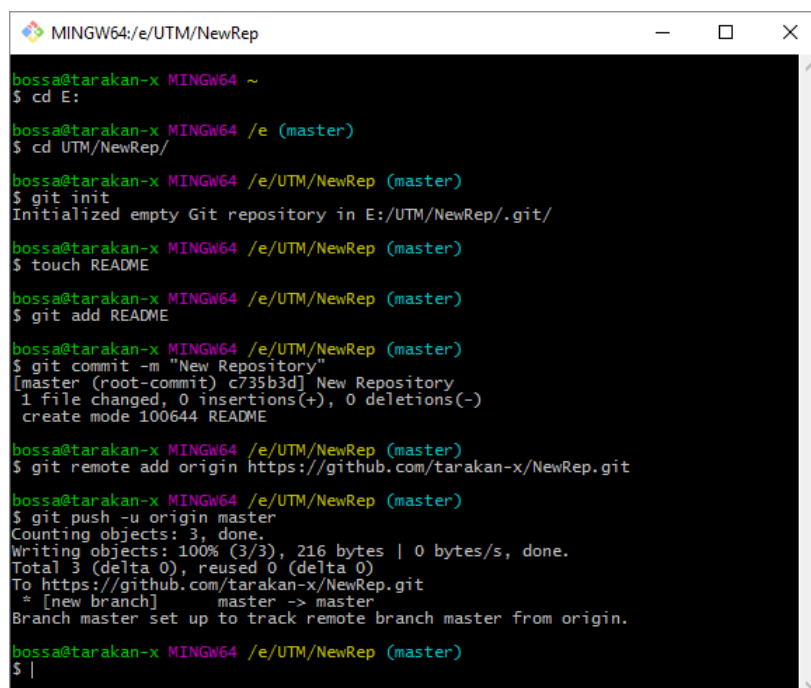
```

MINGW64:/e/UTM/Anul[II]s[IV]/UTMLAB/Laborator2/hello
bossa@tarakan-x MINGW64 /e
$ cd UTM/Anul[II]s[IV]/UTMLAB/Laborator2/hello/
bossa@tarakan-x MINGW64 /e/UTM/Anul[II]s[IV]/UTMLAB/Laborator2/hello (master)
$ ls
hello.cpp  hello.exe*  hello.py  HelloJava.class  HelloJava.java
bossa@tarakan-x MINGW64 /e/UTM/Anul[II]s[IV]/UTMLAB/Laborator2/hello (master)
$ g++ hello.cpp -o hello
bossa@tarakan-x MINGW64 /e/UTM/Anul[II]s[IV]/UTMLAB/Laborator2/hello (master)
$ ./hello
Hello world!
bossa@tarakan-x MINGW64 /e/UTM/Anul[II]s[IV]/UTMLAB/Laborator2/hello (master)
$ javac HelloJava.java
bossa@tarakan-x MINGW64 /e/UTM/Anul[II]s[IV]/UTMLAB/Laborator2/hello (master)
$ java HelloJava
Hello, welcome to GitHub!
bossa@tarakan-x MINGW64 /e/UTM/Anul[II]s[IV]/UTMLAB/Laborator2/hello (master)
$ python hello.py
Welcome to GITHUB!
bossa@tarakan-x MINGW64 /e/UTM/Anul[II]s[IV]/UTMLAB/Laborator2/hello (master)
$

```

Pentru fiecare schimbare pe care o facem pe repozitoriu putem lasa un mesaj folosind comanda *git commit -m "mesaj"* astfel organizam mai bine repozitoriul si putem vedea ce schimbari au avut loc.

Am initializat un nou repozitoriu cu numele *NewRep* cu *git init*, si am configurat acest repozitoriu cu *git config -global user.name* si *user.email*.



```

MINGW64:/e/UTM/NewRep
bossa@tarakan-x MINGW64 ~
$ cd E:
bossa@tarakan-x MINGW64 /e (master)
$ cd UTM/NewRep/
bossa@tarakan-x MINGW64 /e/UTM/NewRep (master)
$ git init
Initialized empty Git repository in E:/UTM/NewRep/.git/
bossa@tarakan-x MINGW64 /e/UTM/NewRep (master)
$ touch README
bossa@tarakan-x MINGW64 /e/UTM/NewRep (master)
$ git add README
bossa@tarakan-x MINGW64 /e/UTM/NewRep (master)
$ git commit -m "New Repository"
[master (root-commit) c735b3d] New Repository
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README
bossa@tarakan-x MINGW64 /e/UTM/NewRep (master)
$ git remote add origin https://github.com/tarakan-x/NewRep.git
bossa@tarakan-x MINGW64 /e/UTM/NewRep (master)
$ git push -u origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 216 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/tarakan-x/NewRep.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
bossa@tarakan-x MINGW64 /e/UTM/NewRep (master)
$ |

```

Am creat doua branch-uri cu numele *FIRST* si *SECOND* folosind comanda *git branch "numele"*.

```
MINGW64:/e/UTM/NewRep
Writing objects: 100% (3/3), 210 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/tarakan-x/NewRep.git
 * [new branch] master -> master
Branch master set up to track remote branch master from origin.

bossa@tarakan-x MINGW64 /e/UTM/NewRep (master)
$ git branch
* master

bossa@tarakan-x MINGW64 /e/UTM/NewRep (master)
$ git branch FIRST

bossa@tarakan-x MINGW64 /e/UTM/NewRep (master)
$ git branch SECOND

bossa@tarakan-x MINGW64 /e/UTM/NewRep (master)
$ git branch
  FIRST
  SECOND
* master

bossa@tarakan-x MINGW64 /e/UTM/NewRep (master)
$ |
```

Am adaugat un fisier pe branch-ul FIRST.

```
MINGW64:/e/UTM/Anul[II]s[IV]/NewRep
bossa@tarakan-x MINGW64 /e/UTM/Anul[II]s[IV]/NewRep (master)
$ git checkout FIRST
Switched to branch 'FIRST'

bossa@tarakan-x MINGW64 /e/UTM/Anul[II]s[IV]/NewRep (FIRST)
$ touch file.txt

bossa@tarakan-x MINGW64 /e/UTM/Anul[II]s[IV]/NewRep (FIRST)
$ notepad file.txt

bossa@tarakan-x MINGW64 /e/UTM/Anul[II]s[IV]/NewRep (FIRST)
$ git add file.txt

bossa@tarakan-x MINGW64 /e/UTM/Anul[II]s[IV]/NewRep (FIRST)
$ git commit -m"add new file to branch"
[FIRST 3bac3fc] add new file to branch
1 file changed, 1 insertion(+)
 create mode 100644 file.txt

bossa@tarakan-x MINGW64 /e/UTM/Anul[II]s[IV]/NewRep (FIRST)
$ git push origin FIRST
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 289 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/tarakan-x/NewRep.git
 * [new branch] FIRST -> FIRST

bossa@tarakan-x MINGW64 /e/UTM/Anul[II]s[IV]/NewRep (FIRST)
$ ls
file.txt  README.md

bossa@tarakan-x MINGW64 /e/UTM/Anul[II]s[IV]/NewRep (FIRST)
$ |
```

Am adaugat si un fisier pe branch-ul SECOND.

```
MINGW64:/e/UTM/Anu1[II]s[IV]/NewRep
To https://github.com/tarakan-x/NewRep.git
* [new branch] FIRST -> FIRST

bossa@tarakan-x MINGW64 /e/UTM/Anu1[II]s[IV]/NewRep (FIRST)
$ ls
file.txt  README.md

bossa@tarakan-x MINGW64 /e/UTM/Anu1[II]s[IV]/NewRep (FIRST)
$ git checkout SECOND
Switched to branch 'SECOND'

bossa@tarakan-x MINGW64 /e/UTM/Anu1[II]s[IV]/NewRep (SECOND)
$ ls
README.md

bossa@tarakan-x MINGW64 /e/UTM/Anu1[II]s[IV]/NewRep (SECOND)
$ touch file.txt

bossa@tarakan-x MINGW64 /e/UTM/Anu1[II]s[IV]/NewRep (SECOND)
$ notepad file.txt

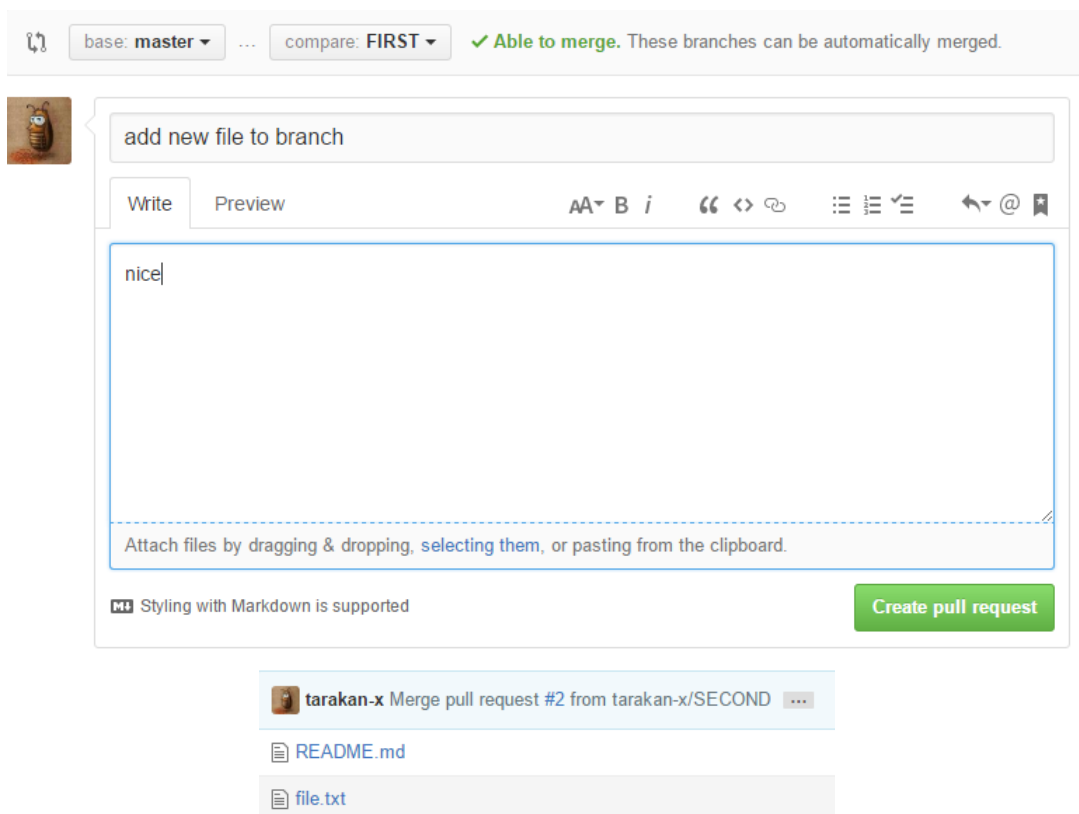
bossa@tarakan-x MINGW64 /e/UTM/Anu1[II]s[IV]/NewRep (SECOND)
$ git add file.txt

bossa@tarakan-x MINGW64 /e/UTM/Anu1[II]s[IV]/NewRep (SECOND)
$ git commit -m "add new file "
[SECOND 8d137a7] add new file
1 file changed, 1 insertion(+)
create mode 100644 file.txt

bossa@tarakan-x MINGW64 /e/UTM/Anu1[II]s[IV]/NewRep (SECOND)
$ git push origin SECOND
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 286 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/tarakan-x/NewRep.git
* [new branch] SECOND -> SECOND

bossa@tarakan-x MINGW64 /e/UTM/Anu1[II]s[IV]/NewRep (SECOND)
$ |
```

Cind accesam github.com ca master putem accepta schimbarile de pe celelalte branch-uri astfel fisierele vor fi adaugate pe master.La fel putem lasa si un comentariu pentru acel commit.



Am facut merge la branch-ul FIRST cu SECOND .

```
MINGW64:/e/UTM/Anu[II]s[IV]/NewRep
$ git checkout FIRST
Switched to branch 'FIRST'

bossa@tarakan-x MINGW64 /e/UTM/Anu[II]s[IV]/NewRep (FIRST)
$ git merge SECOND
Auto-merging file.txt
CONFLICT (add/add): Merge conflict in file.txt
Automatic merge failed; fix conflicts and then commit the result.

bossa@tarakan-x MINGW64 /e/UTM/Anu[II]s[IV]/NewRep (FIRST|MERGING)
$
```

În cazul cînd pe un branch avem un fișier cu un conținut oarecare și pe al branch același fișier dar cu conținut diferit atunci cînd încercăm să facem merge a acestor două branch-uri atunci primim un mesaj de conflict. Dacă deschidem fișierul acolo vor fi afișate problemele care trebuie înlăturate.

```
file - Notepad
File Edit Format View Help
|<<<<<< HEAD
hello
=====
not hello
>>>>>> SECOND
```

Pentru a rezolva această problemă putem modifica conținutul fișierului și după care facem din nou git add și commit astfel rezolvăm acest conflict.

Concluzie

În această lucrare de laborator am studiat Version Control System numit github.com. Github-ul oferă posibilitate de a ține proiectul online, care poate

de tip public și privat. Am efectuat task-urile propuse precum ar fi compilare unor mici programe C++, Java, Python de tipul HELLO WORLD, efectuare commiturilor, initializarea unui repository nou și altele. Pentru a efectua aceste operații am utilizat Git-Bash care este un terminal cu comenzi asemănătoare cu cel din linux. Comenzile sunt simple și e

facile pentru a găzdui un proiect.

References