

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:

Capră MIHAI

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

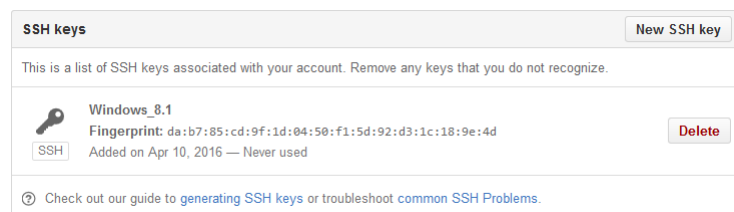
- Basic Level (nota 5 — 6) :
 - conecteaza-te la server folosind SSH
 - compileaza cel putin 2 sample programs din setul HelloWorldPrograms folosind CLI
 - executa primul commit folosind VCS
- Normal Level (nota 7 — 8):
 - initializeaza un nou repository
 - configureaza-ti VCS
 - crearea branch-urilor (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 repozitoriul GITHUB:

<https://github.com/MihaiCapra/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 HelloGITHUB* pentru

a rula programul nostru, in cazul programuli $C++$ utilizam comanda $g++$ hello.cpp -o hello, si ./hello pentru a rula programul si in cazul unui program scris in Python utilizam sintaxa python GITHUB.py pentru a rula programul.

The image shows two terminal windows from a MINGW64 environment. The left window shows the user listing files, compiling 'HelloGITHUB.java' with 'javac', and running 'java HelloGITHUB', which outputs 'Welcome to GITHUB!'. The right window shows the user navigating to the source directory, listing files, compiling 'hello.cpp' with 'g++' to create 'hello.exe', and then running './hello', which also outputs 'Welcome to GITHUB!'.

The image shows a terminal window where the user runs 'python GITHUB.py'. The output is a large, colorful ASCII art graphic. The graphic features the text 'GITHUB' in a large, stylized font, with 'MIDPS_2' and 'SOURCE' integrated into the design. The background is black, and the text is composed of various symbols like '@', '#', '+', and ':', colored in shades of blue, green, and yellow.

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.

```
MINGW64:/d/_MIDPS

Mike@armedcat MINGW64 /d/_MIDPS (master)
$ git commit -m "some source files(Python,Java,C++)"
[master 6af5790] some source files(Python,Java,C++)
5 files changed, 20 insertions(+)
create mode 100644 MIDPS_2/SOURCE/HelloGITHUB.class
create mode 100644 MIDPS_2/SOURCE/HelloGITHUB.java
create mode 100644 MIDPS_2/SOURCE/hello.cpp
create mode 100644 MIDPS_2/SOURCE/hello.exe
create mode 100644 MIDPS_2/SOURCE/hello.py

Mike@armedcat MINGW64 /d/_MIDPS (master)
$
```

Am initializat un nou repository cu numele NEWREPO cu git init,si am configurat acest repository cu git config –global user.name si user.email.

```
MINGW64:/d/_NEWREPO

$ git init
Initialized empty Git repository in D:/_NEWREPO/.git/

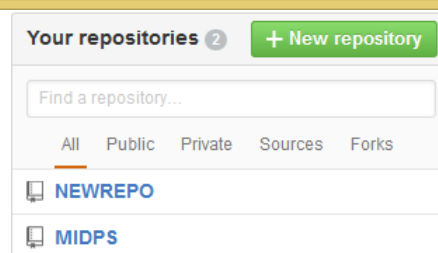
Mike@armedcat MINGW64 /d/_NEWREPO (master)
$ git add README.md
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory.

Mike@armedcat MINGW64 /d/_NEWREPO (master)
$ git commit -m "README.md"
[master (root-commit) 9b994a7] README.md
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory.
1 file changed, 1 insertion(+)
create mode 100644 README.md

Mike@armedcat MINGW64 /d/_NEWREPO (master)
$ git remote add origin https://github.com/MihaiCapra/NEWREPO.git

Mike@armedcat MINGW64 /d/_NEWREPO (master)
$ git push -u origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 231 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/MihaiCapra/NEWREPO.git
 * [new branch] master -> master
Branch master set up to track remote branch master from origin.

Mike@armedcat MINGW64 /d/_NEWREPO (master)
$
```



Am creat doua branch-uri cu numele A si B folosind comanda git branch "numele".

```
MINGW64:/d/_NEWREPO

Mike@armedcat MINGW64 /d/_NEWREPO (master)
$ git branch
* master

Mike@armedcat MINGW64 /d/_NEWREPO (master)
$ git branch A

Mike@armedcat MINGW64 /d/_NEWREPO (master)
$ git branch B

Mike@armedcat MINGW64 /d/_NEWREPO (master)
$ git branch
  A
  B
* master

Mike@armedcat MINGW64 /d/_NEWREPO (master)
$
```

Am adaugat un fisier pe branch-ul A.

```
MINGW64:/d/_NEWREPO

Mike@armedcat MINGW64 /d/_NEWREPO (A)
$ git status
On branch A
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   hello.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    GITHUB.py

Mike@armedcat MINGW64 /d/_NEWREPO (A)
$ git commit -m "hello.py"
[A 7e5b438] hello.py
1 file changed, 6 insertions(+)
create mode 100644 hello.py

Mike@armedcat MINGW64 /d/_NEWREPO (A)
$ git push -u origin A
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 349 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/MihaiCapra/NEWREPO.git
 * [new branch]    A -> A
Branch A set up to track remote branch A from origin.
```

Am adaugat si un fisier pe branch-ul B.

```
MINGW64:/d/_NEWREPO

$ git add GITHUB.py
Mike@armedcat MINGW64 /d/_NEWREPO (B)
$ git status
On branch B
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

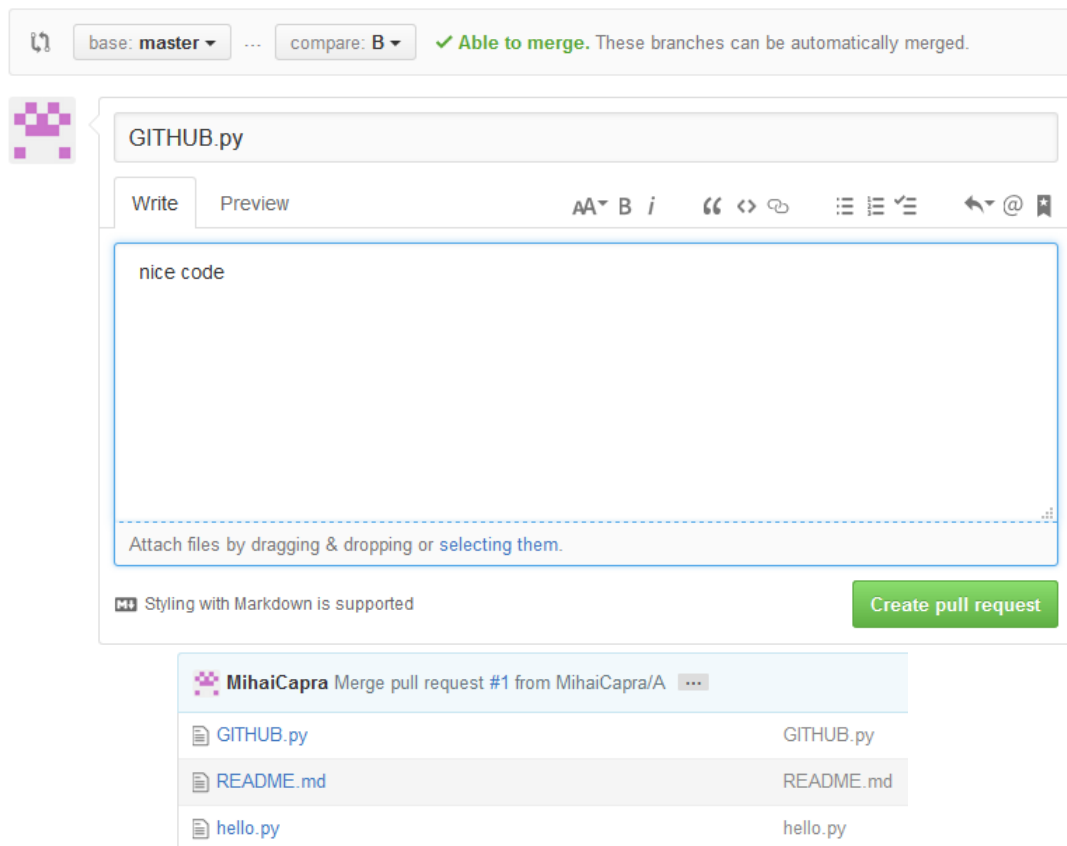
    new file:   GITHUB.py

Mike@armedcat MINGW64 /d/_NEWREPO (B)
$ git commit -m "GITHUB.py"
[B 375a49b] GITHUB.py
1 file changed, 32 insertions(+)
create mode 100644 GITHUB.py

Mike@armedcat MINGW64 /d/_NEWREPO (B)
$ git push -u origin B
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 636 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/MihaiCapra/NEWREPO.git
 * [new branch]    B -> B
Branch B set up to track remote branch B from origin.

Mike@armedcat MINGW64 /d/_NEWREPO (B)
$
```

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 setat branch-ul B track a remote.

```
MINGW64:/d/_NEWREPO
Mike@armedcat MINGW64 /d/_NEWREPO (B)
$ git status
On branch B
Your branch is up-to-date with 'origin/B'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   EMPTY.txt

Mike@armedcat MINGW64 /d/_NEWREPO (B)
$ git commit -m "empty.txt"
[B 33df252] empty.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 EMPTY.txt

Mike@armedcat MINGW64 /d/_NEWREPO (B)
$ git push --set-upstream origin B
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 311 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/MihaiCapra/NEWREPO.git
375a49b..33df252 B -> B
Branch B set up to track remote branch B from origin.

Mike@armedcat MINGW64 /d/_NEWREPO (B)
$
```

Am resetat branch-ul B la un commit anterior.

```
MINGW64:/d/_NEWREPO

Mike@armedcat MINGW64 /d/_NEWREPO (B)
$ git log --oneline
1171ec0 fix error non fast-forward
adb7f54 removed EMPTY.txt
04dc5e8 empty.txt
33df252 empty.txt
375a49b GITHUB.py
9b994a7 README.md

Mike@armedcat MINGW64 /d/_NEWREPO (B)
$ git reset --hard HEAD~4
HEAD is now at 9b994a7 README.md

Mike@armedcat MINGW64 /d/_NEWREPO (B)
$ |
```

Am facut merge la branch-ul B cu master.

```
MINGW64:/d/_NEWREPO

Mike@armedcat MINGW64 /d/_NEWREPO (master)
$ git merge B
Updating 9b994a7..cd7e189
Fast-forward
 EMPTY.txt | 0
 GITHUB.py | 32 ++++++
 hello.py  | 6 +++++
 3 files changed, 38 insertions(+)
 create mode 100644 EMPTY.txt
 create mode 100644 GITHUB.py
 create mode 100644 hello.py

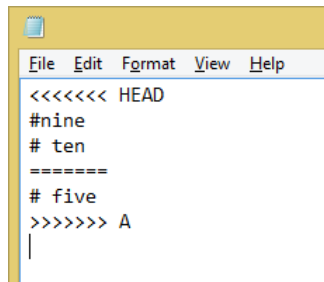
Mike@armedcat MINGW64 /d/_NEWREPO (master)
$
```

In cazul cind pe un branch avem un fisier cu un continut oarecare si pe al branch acelasi fisier dar cu continut diferit atunci cind incerca sa facem merge a acestor doua branch-uri atunci primim un mesaj de conflict. Daca deschidem fisierul acolo vor fi afisate problemele care trebuie inlaturate.

```
MINGW64:/d/_NEWREPO

Mike@armedcat MINGW64 /d/_NEWREPO (B)
$ git merge A
Auto-merging smth.md
CONFLICT (add/add): Merge conflict in smth.md
Automatic merge failed; fix conflicts and then commit the result.

Mike@armedcat MINGW64 /d/_NEWREPO (B|MERGING)
$ |
```

```
File Edit Format View Help
<<<<<< HEAD
#nine
# ten
=====
# five
>>>>>> A
|
```

Pentru a rezolva aceasta problema putem modifica continutul fisierului si dupa care faceem din nou git add si commit astfel rezolvam 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 fi 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 eficiente pentru a găzdui un proiect.

References

- 1 <https://www.youtube.com/playlist?list=PLg7s6cbtAD15G8lNyoaYDuKZSKyJrgwB->
- 2 <https://help.github.com/>