Ministerul Educaţiei Republicii Moldova
Universitatea Tehnică a Moldovei

RAPORT

La disciplina: Medii interactive de dezvoltare a produselor soft
Tema: Version Control Systems şi modul de setare a unui server

A efectuat: Petrenco Mihai
st.gr.TI-144

A verificat: Cojocaru Svetlana
lect.univ.

Chişinău, 2016

Lucrarea de laborator Nr.2

Tema: Version Control Systems şi modul de setare a unui server.
Scopul lucrării: Studierea bazelor lucrului cu VCS.
Obiectivele lucrării:

- Înţelegerea şi folosirea CLI (basic level)
- Administrarea remote a maşinilor 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

Modul de lucru:

Paşii lucrării:

- conectează-te la server folosind SSH
- compilează cel puţin 2 sample programs din setul HelloWolrdPrograms folosind CLI
- execută primul commit folosind VCS
- iniţializează un nou repositoriu
- configurează-ţi VCS
- crearea branch-urilor (creează cel puţin 2 branches)
- commit pe ambele branch-uri (cel puţin 1 commit per branch)
- reseteaza un branch la commit-ul anterior
- merge 2 branches
- conflict solving between 2 branches
- Scrie un script care va compila HelloWolrdPrograms projects.

Efectuarea lucrării:

1. A fost creat un repositoriu pe github.com.
2. A fost stabilită conexiunea cu serverul prin generarea keygen-ului SSH prin instrucţiunea **ssh-keygen**, ea fiind adăugată în setări.
3. A fost creat un fişier şi iniţializat git-ul prin instrucţiunea **git init.**
4. A fost introdus .gitignore şi README.md prin instrucţiunea **git add**, **git commit** şi **git push.**
5. Au fost create două programe, una în C şi alta în C++, cu extensia respectivă **.c** şi **.cpp**
6. Cu ajutorul la Command Prompt, ele au fost compilate cu instrucţiunea:

gcc Hello1.c –o Hello1 şi **g++ Hello2.cpp –o Hello2**

7. Respectiv, au fost executate fişierele noi cu extensia **.exe** în Command Prompt, apelînd la ele prin: **/Hello1** şi **/Hello2.**
8. A fost creat un branch nou, unde au fost încărcate fişierele compilate în CLI, împreună cu imaginile respective şi fişierele originale .c şi .cpp.
9. Am făcut merge dintre branch-ul **master** şi **CLIcompiling**, unde se aflau fişierele compilate.
10. Am creat o situaţie de conflict, creînd un fişier **Conflict.txt** şi încărcîndu-l pe branch **master**.
11. Am creat un branch nou **ConflictBranch,** în care am modificat conţinutul fişierului **Conflict.txt**, astfel făcînd opţiunea merge imposibilă.
12. Am rezolvat conflictul şi am făcut merge între branch-uri.
13. A fost creat scriptul de compilare a programelor **run.bat**.

Imagini:

Iniţializarea git-ului şi adăugarea .gitignore şi README.md

```
home@home-?? MINGW64 ~/Desktop
$ cd MIDPS

home@home-?? MINGW64 ~/Desktop/MIDPS
$ git init
Initialized empty Git repository in C:/Users/home/Desktop/MIDPS/.git/

home@home-?? MINGW64 ~/Desktop/MIDPS (master)
$ git status
On branch master

Initial commit

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

        .gitignore
        README.md

nothing added to commit but untracked files present (use "git add" to track)
```

Efectuarea commit-ului

```
home@home-?? MINGW64 ~/Desktop/MIDPS (master)
$ git add .

home@home-?? MINGW64 ~/Desktop/MIDPS (master)
$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   .gitignore
        new file:   README.md


home@home-?? MINGW64 ~/Desktop/MIDPS (master)
$ git commit -m "Adding .gitignore and README.md files"
[master (root-commit) 529fecc] Adding .gitignore and README.md files
 2 files changed, 41 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 README.md
```

Crearea unui nou branch

```
home@home-?? MINGW64 ~/Desktop/MIDPS (master)
$ git branch CLIcompiling
```

Transferul la branch-ul nou

```
home@home-?? MINGW64 ~/Desktop/MIDPS (master)
$ git checkout CLIcompiling
Switched to branch 'CLIcompiling'
```

Adaugarea unui fisier .txt, care v-a fi resetat.

```
home@home-?? MINGW64 ~/Desktop/MIDPS (CLIcompiling)
$ git status
On branch CLIcompiling
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Laborator 2/GCC Compiler/Hello, is it me you're looking for.txt

nothing added to commit but untracked files present (use "git add" to track)

home@home-?? MINGW64 ~/Desktop/MIDPS (CLIcompiling)
$ git add .

home@home-?? MINGW64 ~/Desktop/MIDPS (CLIcompiling)
$ git commit -m "This is not the commit you're looking for."
[CLIcompiling cb997e0] This is not the commit you're looking for.
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Laborator 2/GCC Compiler/Hello, is it me you're looking for.txt
```

Resetarea la commit-ul precedent

```
home@home-?? MINGW64 ~/Desktop/MIDPS (CLIcompiling)
$ git reset --hard d746a7f
HEAD is now at d746a7f Adding the CLI files to the first branch
```

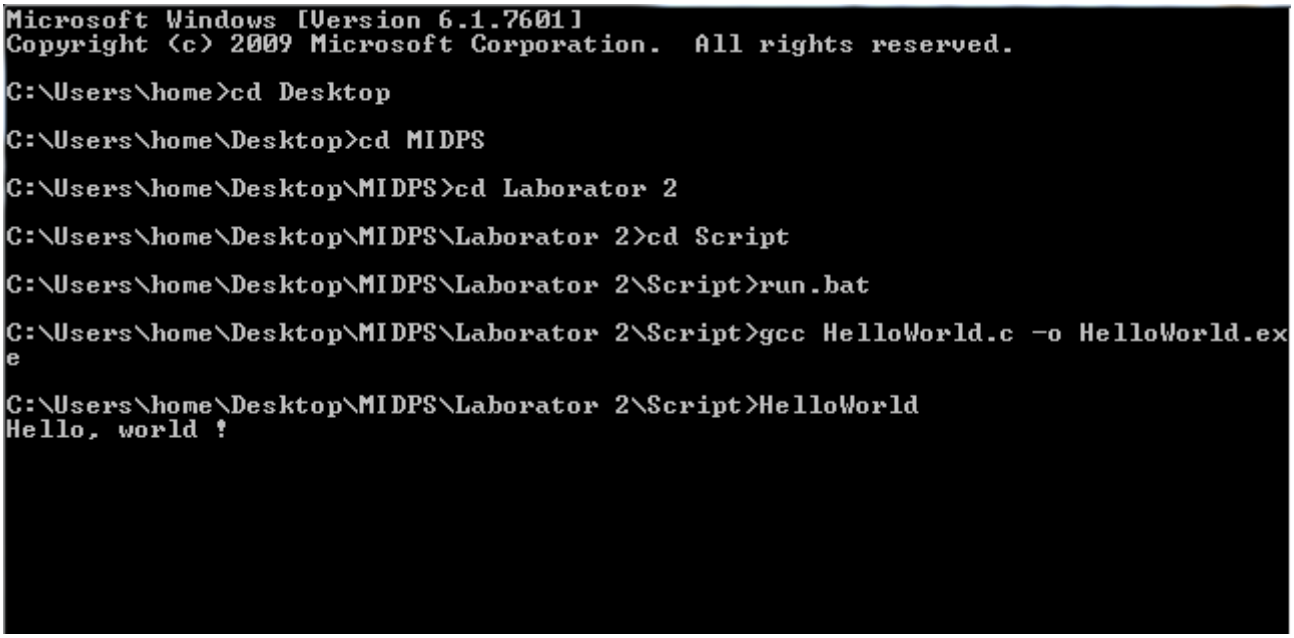
Merge între două branch-uri

```
home@home-?? MINGW64 ~/Desktop/MIDPS (master)
$ git merge master CLIcompiling
Updating 919d8cf..d746a7f
Fast-forward
 Laborator 2/GCC Compiler/Compiler.PNG | Bin 0 -> 114219 bytes
 Laborator 2/GCC Compiler/Hello1.c     |   9 +++++++++
 Laborator 2/GCC Compiler/Hello2.cpp   |   9 +++++++++
 Laborator 2/GCC Compiler/ReadMe.md    |  24 ++++++++++++++++++++++++
 4 files changed, 42 insertions(+)
 create mode 100644 Laborator 2/GCC Compiler/Compiler.PNG
 create mode 100644 Laborator 2/GCC Compiler/Hello1.c
 create mode 100644 Laborator 2/GCC Compiler/Hello2.cpp
 create mode 100644 Laborator 2/GCC Compiler/ReadMe.md
```

Afişarea situaţiei de conflict.

```
home@home-?? MINGW64 ~/Desktop/MIDPS (master)
$ git merge ConflictBranch
Auto-merging Laborator 2/GCC Compiler/Conflict.txt
CONFLICT (add/add): Merge conflict in Laborator 2/GCC Compiler/Conflict.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Executarea script-ului:



Concluzii:

În lucrarea dată de laborator, ne-am familiarizat cu VCS (Version Control Systems) şi am lucrat cu CLI. Am înţeles cum să creez un repositoriu şi să lucrez asupra lui în Git Bash de pe un remote server, cît şi crearea branch-urilor pentru lucrul în subproiecte pentru proiectele mari. VCS este un instrument ce verifică versiunea proiectului, permitîndu-ne să evităm conflicte. El este util pentru lucrul cu proiecte mari unde participă un număr mare de oameni, fiecare lucrînd asupra segmentului său.