

Ministerul Educației al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică și Microelectronică

Catedra Tehnologii Informaționale

RAPORT

Lucrare de laborator Nr.2

Tema: Version Control Systems si modul
de setare a unui server

A efectuat:

St. gr. TI-142
Vîrlan Ion

A verificat:

I. Cojan

Chișinău 2016

Scopul lucrării:

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

Obiective:

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

Sarcinile lucrării:

- De conectarea unui server folosind SSH
- De compilat sample programs utilizând CLI
- De inițializat un repository
- De configurat VCS
- De lucrat cu git utilizând: comit, push, branch, merge
- De rezolvat un conflict între 2 branch-uri

Realizarea lucrării:

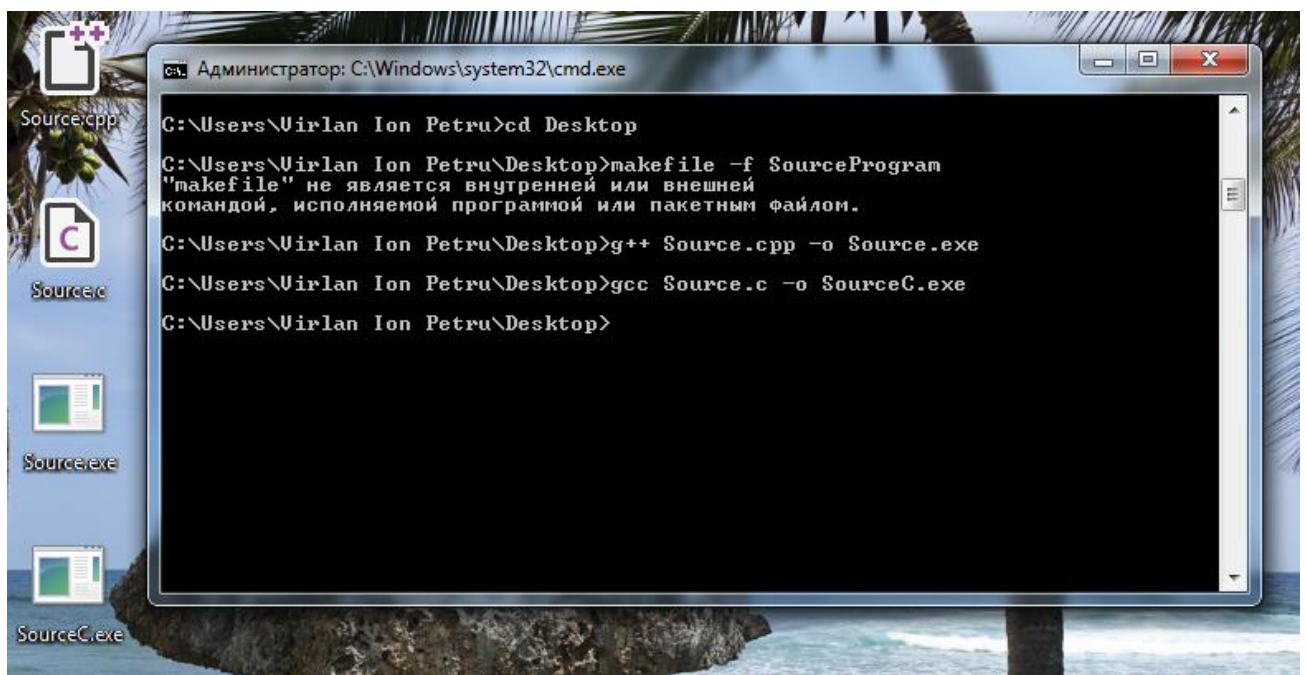
1. Inițial am creat un nou repository pe *github.com*
2. Apoi am generat SSH key utilizând comanda *ssh-keygen* adăugându-l în setările repositoryului
3. Testarea am făcut-o prin comanda *commit* și *push* cu un fișier text
4. Am postat fișierile de laborator nr. 1 utilizând *git add* . apoi *git commit* și apoi *git push*
5. Am creat două sample programs de tipul HelloWorld în C și C++, compilându-le folosind CLI, am creat un branch cu numele *source code* unde am încărcat screenshot-urile și fișierele sample programs, realizând și pasul lucrării de laborator de a crea branch-uri
6. Am executat comanda *merge* la branch-ul care conține fișierele deja menționate de la branch-ul principal master.
7. Am creat și adăugat fișierele *redname.md* și *.gitignore* în care am scris numele, grupa, facultatea și universitatea și respectiv extensiile fișierelor de sistem care trebuie ignorate
8. Am trecut de la branch-ul principal *master* la branch-ul secundar *source code* și invers pentru a putea lucra și îndeplini toate sarcinile, nevoile și dorințele care au apărut sau vor apărea
9. Pentru a crea o situație de conflict între branch-uri am mai creat un sample program
10. Apoi am mai creat al doilea branch, am editat fișierul program anterior menționat și am realizat comanda *commit*, făcând același lucru și cu branchul principal master
11. Cu comanda *merge* introdusă în terminal am depistat ”eroarea” conflictul apărut
12. Versiunea finală a fost salvată și pusă în branch-ul principal master
13. Cu ajutorul comenzii *comit* a fost anulat conflictul și cu comenzile *delete branch* postate mai jos ca screenshot
14. Am specificat procesul de lucru la prima și a doua lucrare de laborator menționând sarcinile care au fost realizate
15. În total au fost făcute 3 branch-uri dintre care unul a fost cel principal, de bază *master* și au fost făcute anumite acțiuni între ele după cum se cerea în sarcina lucrării de laborator

Rezultate screenshot:

Compilare sample programs de tipul HelloWorld în C și C++ folosind CLI gcc și g++:

```
MINGW64:/d/Documente/utm/Laboratoare/MIDPS/Luc. lab. 2/MIDPS/Laborator 2/s...  
Virlian Ion Petru@VirlianIonPetru MINGW64 /d/Documente/utm/Laboratoare/MIDPS/Luc.  
lab. 2/MIDPS/Laborator 2/source code/HelloWorld/HelloWorld (master)  
$ gcc Source.c -o Source.exe  
  
Virlian Ion Petru@VirlianIonPetru MINGW64 /d/Documente/utm/Laboratoare/MIDPS/Luc.  
lab. 2/MIDPS/Laborator 2/source code/HelloWorld/HelloWorld (master)  
$ ./Source.exe  
Hello World!  
Virlian Ion Petru@VirlianIonPetru MINGW64 /d/Documente/utm/Laboratoare/MIDPS/Luc.  
lab. 2/MIDPS/Laborator 2/source code/HelloWorld/HelloWorld (master)  
$ .....
```

```
MINGW64:/d/Documente/utm/Laboratoare/MIDPS/Luc. lab. 2/MIDPS/Laborator 2/s...  
Virlian Ion Petru@VirlianIonPetru MINGW64 /d/Documente/utm/Laboratoare/MIDPS/Luc.  
lab. 2/MIDPS/Laborator 2/source code/CPPHelloWorld/CPPHelloWorld (master)  
$ g++ Source.cpp -o Source.exe  
  
Virlian Ion Petru@VirlianIonPetru MINGW64 /d/Documente/utm/Laboratoare/MIDPS/Luc.  
lab. 2/MIDPS/Laborator 2/source code/CPPHelloWorld/CPPHelloWorld (master)  
$ ./Source.exe  
Hello World!  
Virlian Ion Petru@VirlianIonPetru MINGW64 /d/Documente/utm/Laboratoare/MIDPS/Luc.  
lab. 2/MIDPS/Laborator 2/source code/CPPHelloWorld/CPPHelloWorld (master)  
$ |
```



Crearea unui branch cu numele source code in care am pus fişierele program si screenshot-urile:

```
MINGW64:/d/Documente/utm/Laboratoare/MIDPS/Luc. lab. 2/MIDPS
and maintain the traditional behavior, use:
    git config --global push.default matching
To squelch this message and adopt the new behavior now, use:
    git config --global push.default simple
When push.default is set to 'matching', git will push local branches
to the remote branches that already exist with the same name.
Since Git 2.0, Git defaults to the more conservative 'simple'
behavior, which only pushes the current branch to the corresponding
remote branch that 'git pull' uses to update the current branch.
See 'git help config' and search for 'push.default' for further information.
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of Git)
fatal: The current branch issSourceCode has no upstream branch.
To push the current branch and set the remote as upstream, use
    git push --set-upstream origin issSourceCode

Virlian Ion Petru@VirlianIonPetru MINGW64 /d/Documente/utm/Laboratoare/MIDPS/Luc.
lab. 2/MIDPS (issSourceCode)
$ git status
On branch issSourceCode
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   "Laborator 2/source code/CPPhelloWorld/CPPhelloWorld/\320\24
1\320\275\320\270\320\274\320\276\320\272.PNG"
    new file:   "Laborator 2/source code/HelloWorld/HelloWorld/\320\241\320\
275\320\270\320\274\320\276\320\272.PNG"

Virlian Ion Petru@VirlianIonPetru MINGW64 /d/Documente/utm/Laboratoare/MIDPS/Luc.
lab. 2/MIDPS (issSourceCode)
$ git commit -m 'source code and screenhsot laborator2'
[issSourceCode b07c26e] source code and screenhsot laborator2
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 "Laborator 2/source code/CPPhelloWorld/CPPhelloWorld/\320\241\320\275\3
20\270\320\274\320\276\320\272.PNG"
 create mode 100644 "Laborator 2/source code/HelloWorld/HelloWorld/\320\241\320\275\320\270
\320\274\320\276\320\272.PNG"

Virlian Ion Petru@VirlianIonPetru MINGW64 /d/Documente/utm/Laboratoare/MIDPS/Luc. lab. 2/MIDP
$ (issSourceCode)
$ git status
On branch issSourceCode
nothing to commit, working directory clean

Virlian Ion Petru@VirlianIonPetru MINGW64 /d/Documente/utm/Laboratoare/MIDPS/Luc. lab. 2/MIDP
$ (issSourceCode)
$
```

Trecere de la branch-ul secundar *source code* la branch-ul principal *master*

```
MINGW64:/d/Documente/utm/Laboratoare/MIDPS/Luc. lab. 2/MIDPS
Virlian Ion Petru@VirlianIonPetru MINGW64 /d/Documente/utm/Laboratoare/MIDPS/Luc.
lab. 2/MIDPS (issSourceCode)
$ git merge master
Already up-to-date.

Virlian Ion Petru@VirlianIonPetru MINGW64 /d/Documente/utm/Laboratoare/MIDPS/Luc.
lab. 2/MIDPS (issSourceCode)
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.

Virlian Ion Petru@VirlianIonPetru MINGW64 /d/Documente/utm/Laboratoare/MIDPS/Luc.
lab. 2/MIDPS (master)
$ Switched to branch "master"
```

Crearea unui conflict:

```
Virlian Ion Petru@VirlianIonPetru MINGW64 /d/Documente/utm/Laboratoare/MIDPS/Luc.
lab. 2/MIDPS (master)
$ git merge master conflict
CONFLICT (content): Merge conflict in D:\Documente\utm\Laboratoare\MIDPS\Luc. lab.
2\MIDPS\Laborator 2\source code\branchConflict\branchConflict\ SourceConflict
.cpp
Automatic merge failed; fix conflicts and then commit the result.
```

Conflictul evidențiat de git

```
SourceConflict.cpp*  X
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6  <<<<<<< HEAD
7      cout << "Branch-ul master" << endl;
8  =====
9      cout << "Crearea unui conflict intre branch" << endl;
10 >>>>>>> refs/heads/conflict
11      return 0;
12  }
13
```

Rezolvarea conflictului de git

```
Virlian Ion Petru@VirlianIonPetru MINGW64 /d/Documente/utm/Laboratoare/MIDPS/Luc.
lab. 2/MIDPS (master)
$ git branch -d 2branch
Deleted branch 2branch (was 4de0bbd).
```

Rezolvarea conflictului în cod

```
SourceConflict.cpp*  X
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      cout << "Conflictul a fost rezolvat"<<endl;
7
8      return 0;
9  }
```

Concluzie

În procesul efectuării lucrării de laborator am făcut cunoștință cu metodele de lucru cu VCS (Version Control Systems) și anume cu posibilitatea de a lucra asupra unui proiect ”fără frică” de a pierde ceva sau frica de a face o careva eroare fără ca aceasta să mai poată fi corectată. Am lucrat folosind CLI cu github-ul și anume am făcut diferite acțiuni pentru a cunoaște mai bine și de a poseda această posibilitate de a pastra, înnoi, modifica, compila și de a accesa toate modificările făcute în orice timp și loc. Atunci când proiectul este mai voluminos și complex și apare necesitatea de a se crea pe subproiecte și de a lucra mai mulți (echipă) indiferent din ce loc geografic programistul va munci asupra proiectului, cu probabilitatea riscului la minim.

Pe parcursul lucrării am înțeles importanța, utilitatea, și eficiența acestor facilități pe care ni le oferă git-ul cu VCS prin crearea și monitorizarea proiectului pe un repository format fiindcă este ușor, rapid pentru a elabora lucruri frumoase și utile.

Bibliografie:

Îndrumar de laborator Lucrarea nr.2

<https://www.youtube.com/watch?v=mYjZtU1-u9Y>

<https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository>

https://www.youtube.com/results?search_query=learn+git+in+20+minutes

<https://desktop.github.com/>

<https://git-scm.com/book/en/v2>