Ministerul Educaţiei 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

Chişinău 2016

Scopul lucrării:

- 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

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

Sarcinile lucrării:

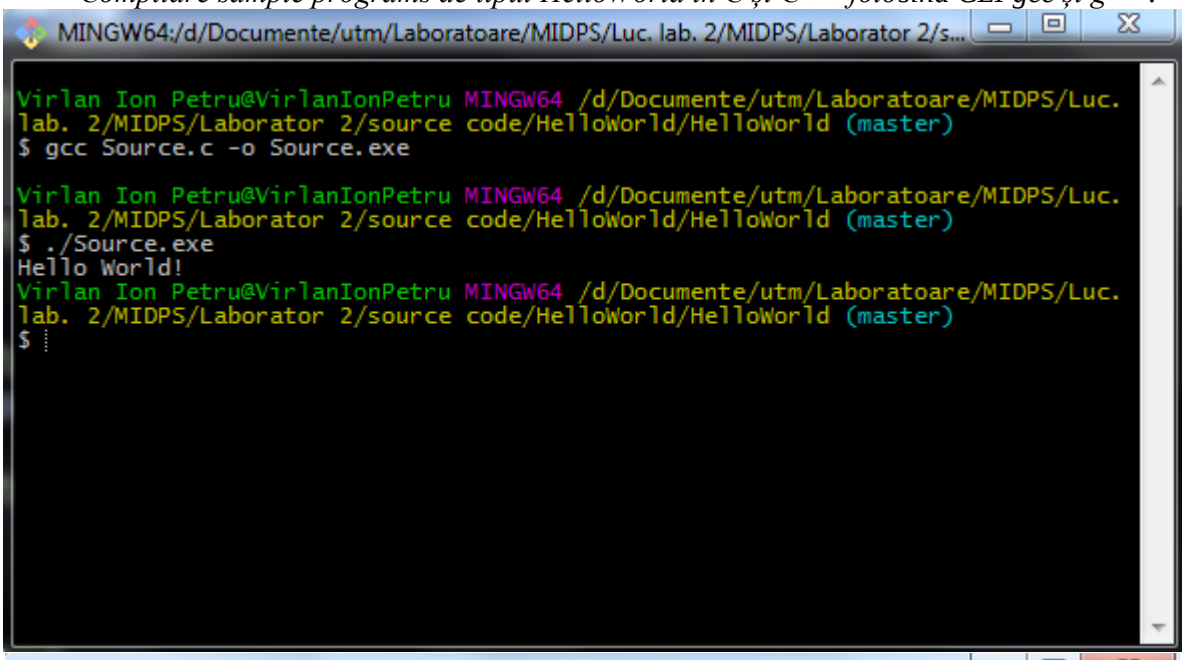
- De conectarea unui server folosind SSH
- De compilat sample programs utilizînd CLI
- De inițializat un repozitoriu
- De configurat VCS
- De lucrat cu git utilizînd: comit, push, branch, merge
- De rezolvat un conflict între 2 branch-uri

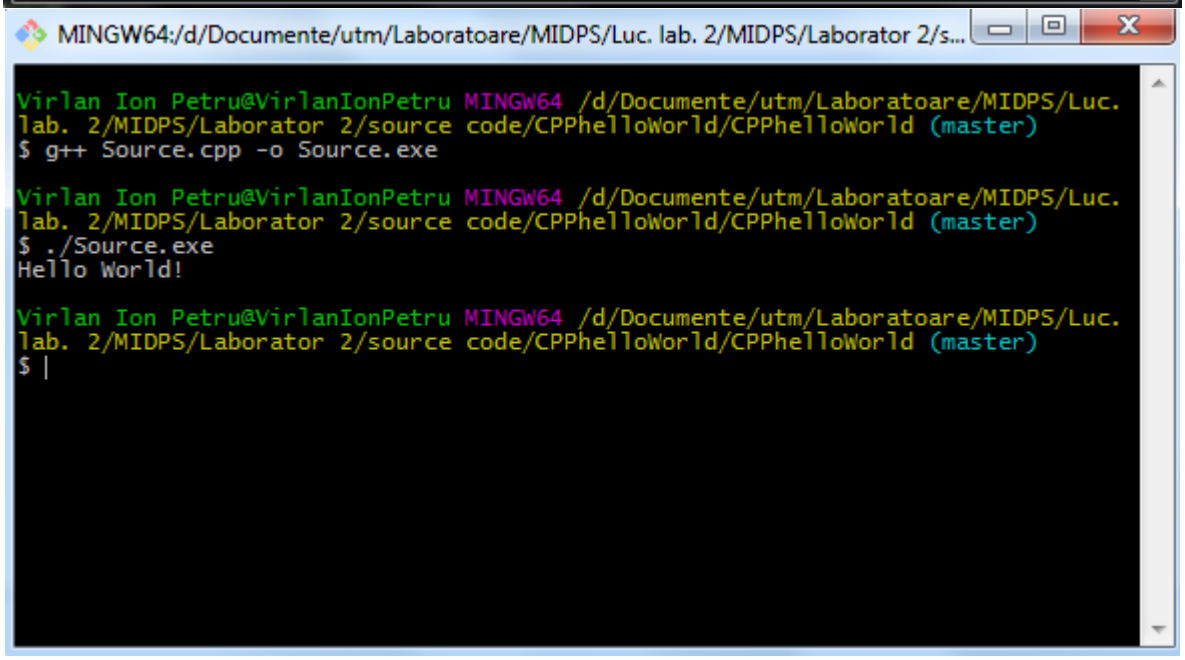
Realizarea lucrării:

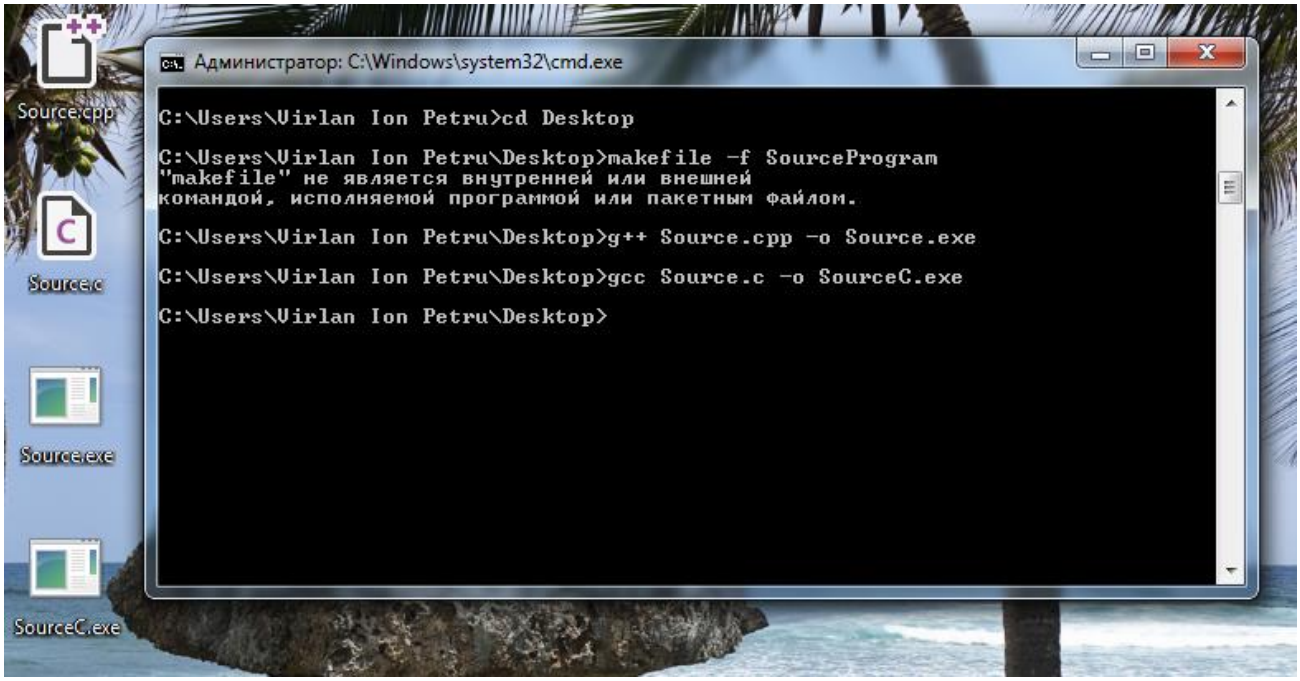
1. Inițial am creat un nou repozitoriu pe *github.com*
2. Apoi am generat SSH key utilizînd comanda ssh-keygen adăugîndul în setările repozitoriului
3. Testarea am făcut-o prin comanda commit și push cu un fișier text
4. Am postat fișierile de laboratorul nr. 1 utilizind *git add* . apoi *git commit* și apoi *git push*
5. Am creat două sample programs de tipul HelloWorld în C și C++, compilîndu-le folosind CLI, am creat un branch cu numele *source code* unde am încărcat screenshot-urile și firșierele sample programs, realizînd și pasul lucrarii de lalborator 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 adaugat fișierele *redname.md* și *.gitignore* în care am scris numele, grupa, facultatea și universitatea și respectiv extensiile firierilor de sistem care trebuiesc ignorate
8. Am trecut de la branch-ul principal *master* la branch-ul secunndar *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,* făcînd același lucru și cu branchul principal master
11. Cu comanda merge introdusă în terminal am depistat ”eroarea” conflictul apărut
12. Versiunea finață a fost salvată și pesă în branch-ul principal master
13. Cu ajutorul comenzei comit a fost anulat conflictul si cu comenzile delete branch postate mai jos ca screenshot
14. Am spucificat procesul de lucru la prima si a doua lucrare de laborator menționînd sarcinile care au fost realizate
15. În total au fost facute 3 branch-uri dintrea care unul a fost cel principal, de bază *master* și au fost făcute anumite acțiuni între ele după cun se cerea în sarcina lucrării de laborator

Rezultate screenshot:

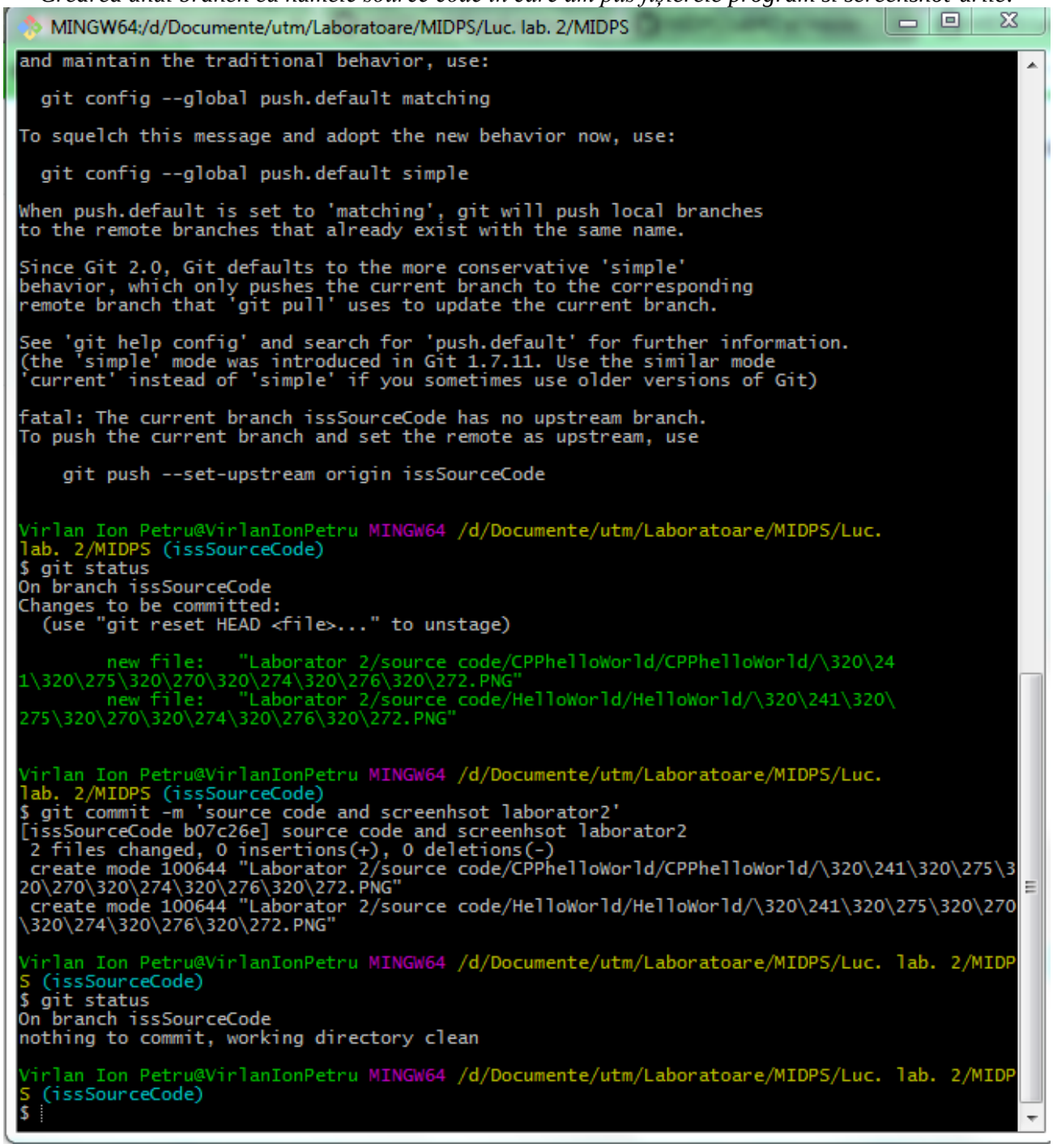
Compilare sample programs de tipul HelloWorld în C și C++ folosind CLI gcc și g++:



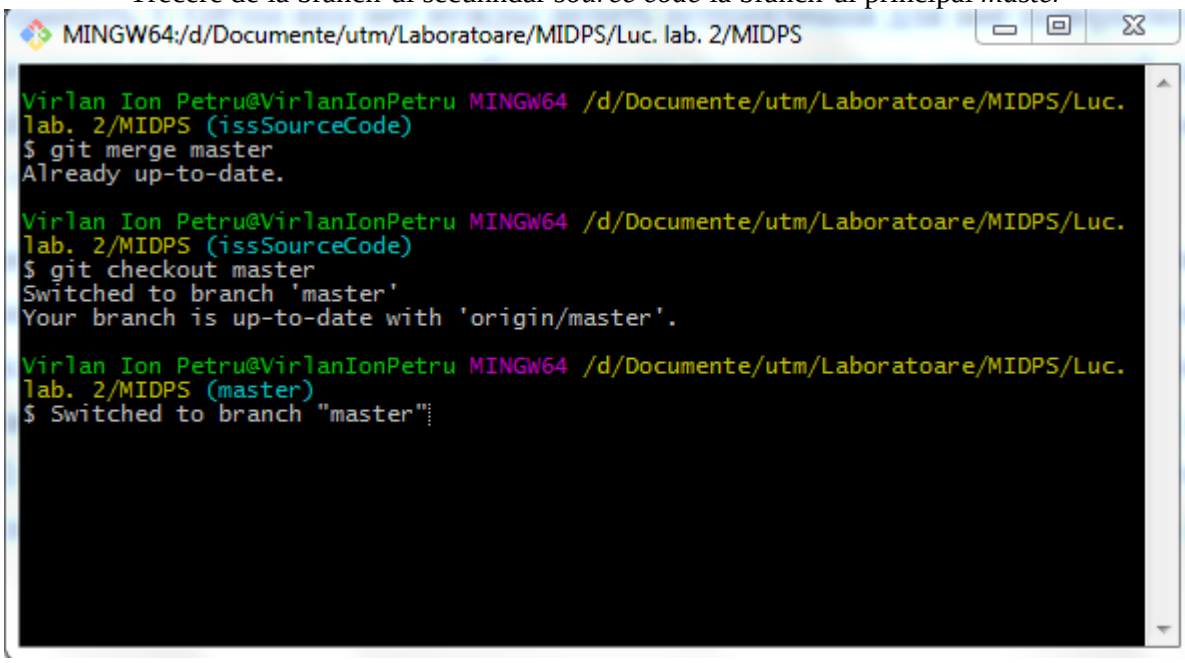




Crearea unui branch cu numele source code in care am pus fișierele program si screenshot-urile:



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



Crearea unui conflict:

```
Virlan Ion Petru@VirlanIonPetru MINGW64 /d/Documente/utm/Laboratoare/MIDPS/Luc. lab. 2/MIDPS (master)
$ git merge mastrer 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*

```
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

```
Virlan Ion Petru@VirlanIonPetru 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*

```
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 ”fară frică” de a pierde ceva sau frica de a face o careva eroare fară 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 cînd 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 gare ni le oferă git-ul cu VCS prin crearea si monitorizarea proiectului pe un repozitoriu 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