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:
Victor Talpa

lector asistent:
Irina Cojanu
lector superior:
Svetlana Cojocaru

Laboratory work #2

1 Scopul lucrarii de laborator

Studierea bazelor lucrului cu VCS.

2 Obiective

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

3 Laboratory work implementation

3.1 Tasks and Points

1. Conectarea la server folosind SSH
2. Compilarea programelor folosind CLI
3. Inițializarea unui repozitoriu
4. Configurarea VCS
5. Lucrul cu git (commit, push, branch, merge)
6. Rezolvarea unui conflict între două branch-uri

3.2 Analiza lucrarii de laborator

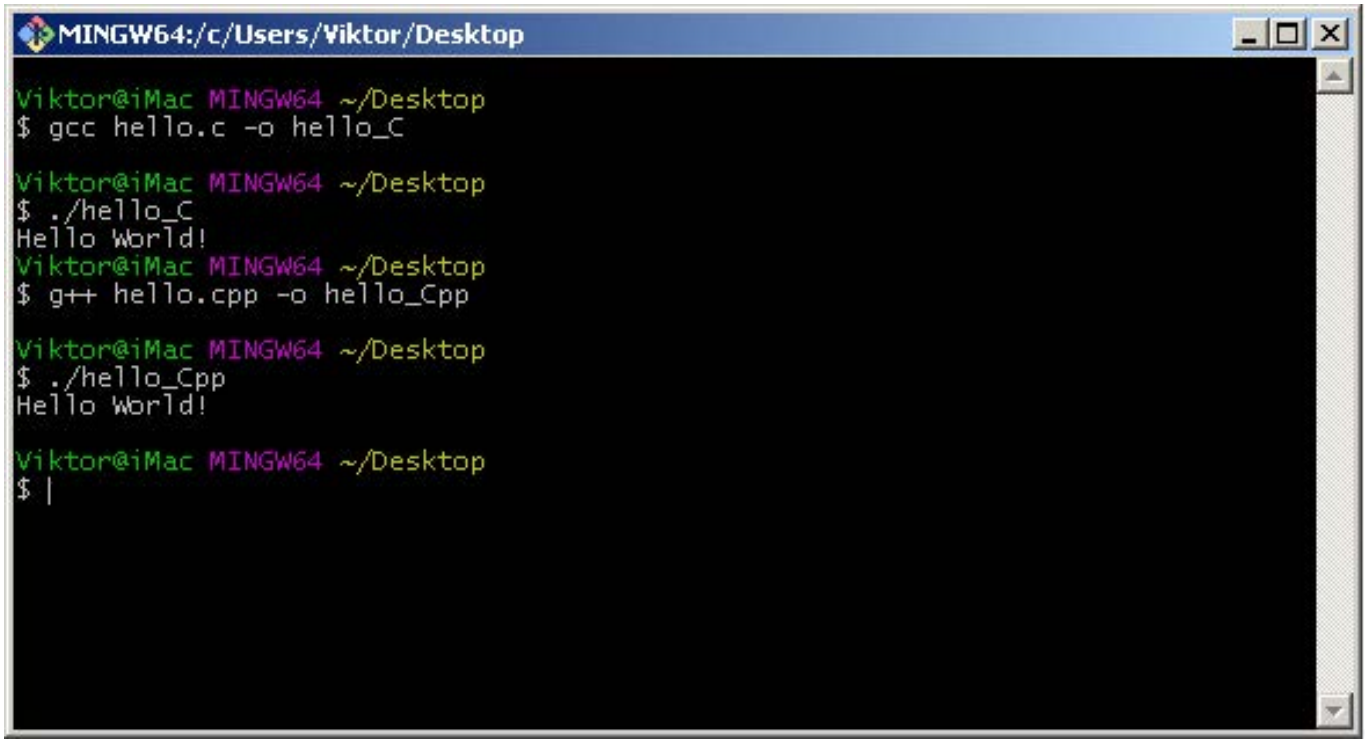
Link spre repozitoriu: https://github.com/VictorTalpa/MIDPS

1. Primul lucru a fost crearea unui repozitoriu pe github.com.
2. Dupa care, am generat SSH key folosind comanda ssh-keygen, care l-am adaugat in setarile repozitoriului pentru a avea acces spre remote de la calculatorul personal.
3. Am testat conexiunea facînd un commit apoi push cu un simplu fisier text.
4. Am adăugat fișierele efectuate pentru lucrarea de laborator nr.1 după care din nou a urmat un commit/push.
5. Am creat două programe simple în C și C++ și le-am compilat folosind CLI, dupa care am creat un branch și am încărcat atît fișierele programelor, cît și screenshot-uri a compilării, astfel efectuînd și pasul lucrării de a crea branch-uri.
6. Am adăugat fișierele readme și .gitignore care l-am modificat după propriile preferințe (pentru aceasta am folosit comentariul ";").
7. Am făcut merge la branch-ul ce conține fișierele legate de compilarea folosind CLI cu branch-ul principal master.
8. Am creat un program simplu pentru a crea o situație de conflict între branch-uri.
9. Am facut cel de-al doilea branch, am editat fișierul adăugat anterior după care a urmat un commit.
10. Același lucru am făcut pentru branch-ul master.
11. După care a urmat instrucțiunea merge și terminalul mi-a arătat conflictul apărut.

12. Am deschis fișierul cu program, unde am văzut conflictul apărut, și am scris versiunea finală care urmează să fie salvată.
13. După un commit a fost rezolvat conflictul.
14. Am adăugat fișireul readme în directoriul primei lucrări de laborator în care am specificat procesul de lucru la efectuarea lucrării de laborator nr.1.

3.3 Imagini

Compilarea a două programe simple (C și C++) folosind CLI, gcc și g++



Executarea instrucțiunii merge ce creează o situație de conflict

```
Viktor@IMAC ~/Google Drive/_University/SourceTree/MIDPS/MIDPS (master)
$ git merge master conflict
Auto-merging MIDPS/sample.cpp
CONFLICT (content): Merge conflict in MIDPS/sample.cpp
Automatic merge failed; fix conflicts and then commit the result.
```

Conflictul în cod evidențiat de git

[*] sample.cpp

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6
7   <<<<<<< HEAD
8       cout << "Master branch";
9   =======
10      cout << "Conflict branch";
11  >>>>>>> refs/heads/conflict
12
13      return 0;
14  }
15
```

Rezolvarea conflictului sau alegerea unei opțiuni alternative

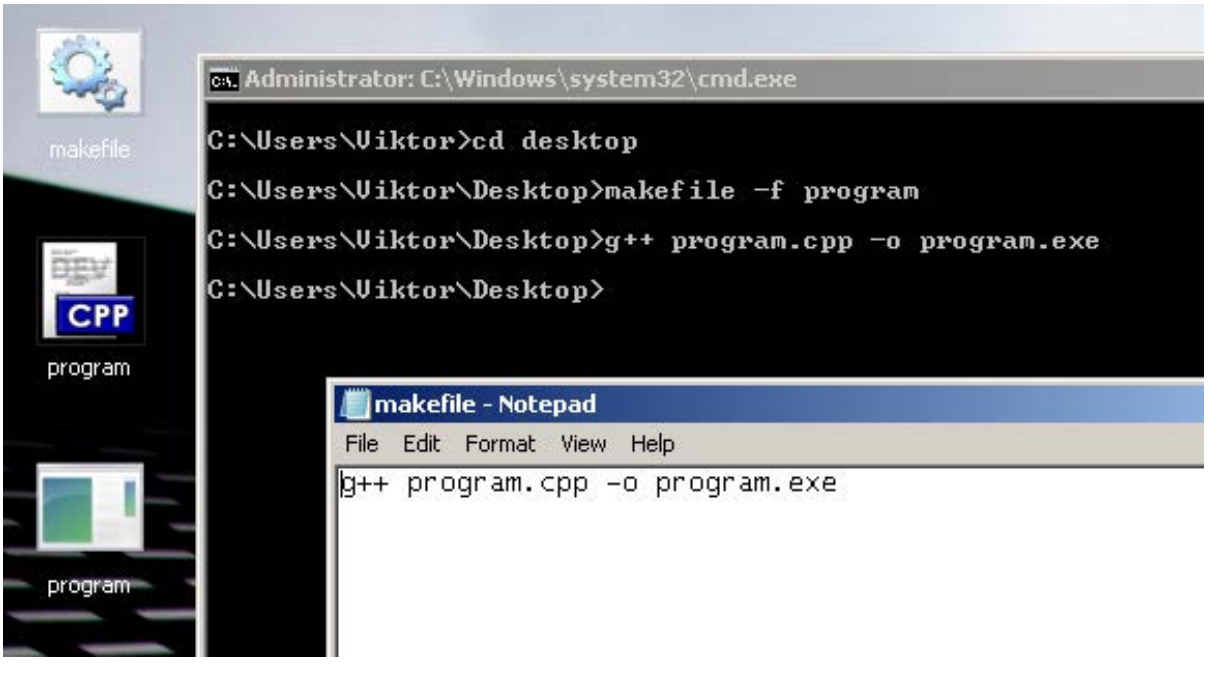
[*] sample.cpp

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6       cout << "Conflict Resolved !";
7
8       return 0;
9   }
```

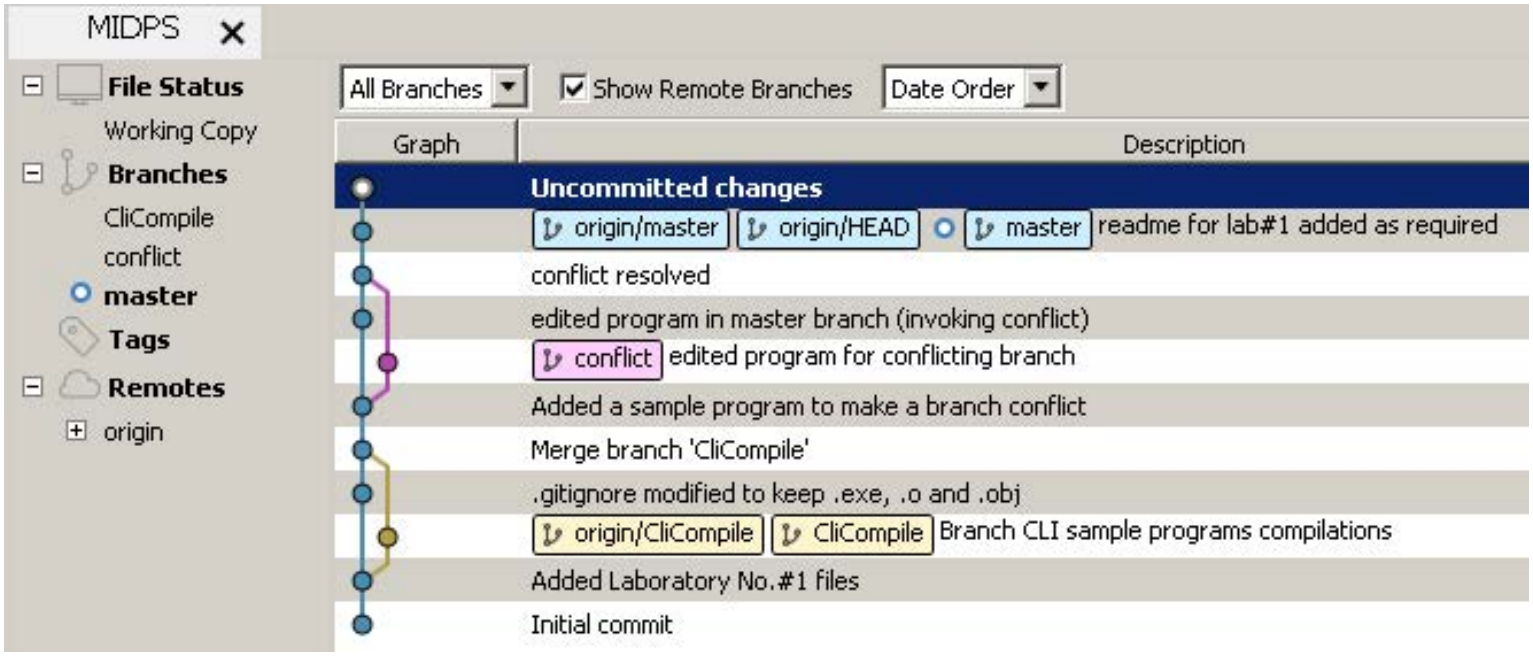
Commit-ul cu versiunea finală a fișierului ce a provocat conflictul

```
Viktor@IMAC ~/Google Drive/_University/SourceTree/MIDPS/MIDPS (master|MERGING)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

You have unmerged paths.
  (fix conflicts and run "git commit")

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   sample.cpp

no changes added to commit (use "git add" and/or "git commit -a")

Viktor@IMAC ~/Google Drive/_University/SourceTree/MIDPS/MIDPS (master|MERGING)
$ git add sample.cpp

Viktor@IMAC ~/Google Drive/_University/SourceTree/MIDPS/MIDPS (master|MERGING)
$ git commit -m "conflict resolved"
[master 27ba2bd] conflict resolved

Viktor@IMAC ~/Google Drive/_University/SourceTree/MIDPS/MIDPS (master)
$ git push origin master
Username for 'https://github.com': VictorTalpa
Password for 'https://VictorTalpa@github.com':
Counting objects: 17, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (12/12), done.
Writing objects: 100% (12/12), 1.30 KiB | 0 bytes/s, done.
Total 12 (delta 4), reused 0 (delta 0)
To https://github.com/VictorTalpa/VictorTalpa.git
   ec20186..27ba2bd  master -> master

Viktor@IMAC ~/Google Drive/_University/SourceTree/MIDPS/MIDPS (master)
$ _
```

Executarea scriptului makefile.bat care compilează programe C++



Procesul final cu reprezentare grafică folosind soft-ul SourceTree



Concluzie

În urma efectuării acestei lucrări de laborator, am studiat metodele de lucru cu VCS(Version Control Systems) cît și am făcut cunoștiință mai avansat la folosirea CLI. Am aflat cum putem crea un repozitoriu și să ne folosim de el pentru a menține un proiect în primul rînd pe un server remote, cît și impărțirea acestuia în sub-proiecte și lucrul la un proiect mare a mai multor persoane concomitent. VCS ne permite sa efectuăm procesul de tracking a versiunilor proiectelor noastre, ceea ce ne va permite să evităm situații cînd am dat greș în urma unor experimente cu proiectul nostru. La fel acest sistem permite lucrul concomitent a mai multor persoane la un proiect din diferite colțuri ale lumii. VCS permite crearea unor ramuri numite branch-uri pentru crearea unui proiect pe bucăți și cu ajutorul instrucțiunii merge putem uni toate aceste părți pentru a crea ceva mai mare. Majoritatea companiilor și programatorilor avansați folosesc VCS datorită beneficiilor acestora, ceea ce semnifică că e un lucru foarte important de studiat, merită utilizat și poate garanta un rezultat bun în performanța unui programator sau unui grup de programatori.

References

1 Getting Git Right, Atlassian Git Tutorials, *https://www.atlassian.com/git/*

2 Git Documentation, *https://git-scm.com/doc*

3 Google and Youtube