

Ministerul Educației al Republicii Moldova

Universitatea Tehnică a Moldovei

Catedra: Automatica și Tehnologii Informaționale

# RAPORT

Lucrare de laborator Nr.2

*la disciplina:*

*Medii Interactive de Dezvoltare a Produselor Soft*

*tema:*

*Version Control Systems si modul de setare a unui server*

A efectuat: st. gr. TI-144

A verificat: lect.univ.

Plotnicu R.

Cojanu I.

Chișinău 2016

## **Laboratory work #2**

### **1. Scopul lucrării**

Studierea bazelor lucrului cu VCS

### **2. Obiective**

1. Înțelegerea și folosirea CLI (basic level)
2. Administrarea remote a mașinilor linux 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/python

### **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

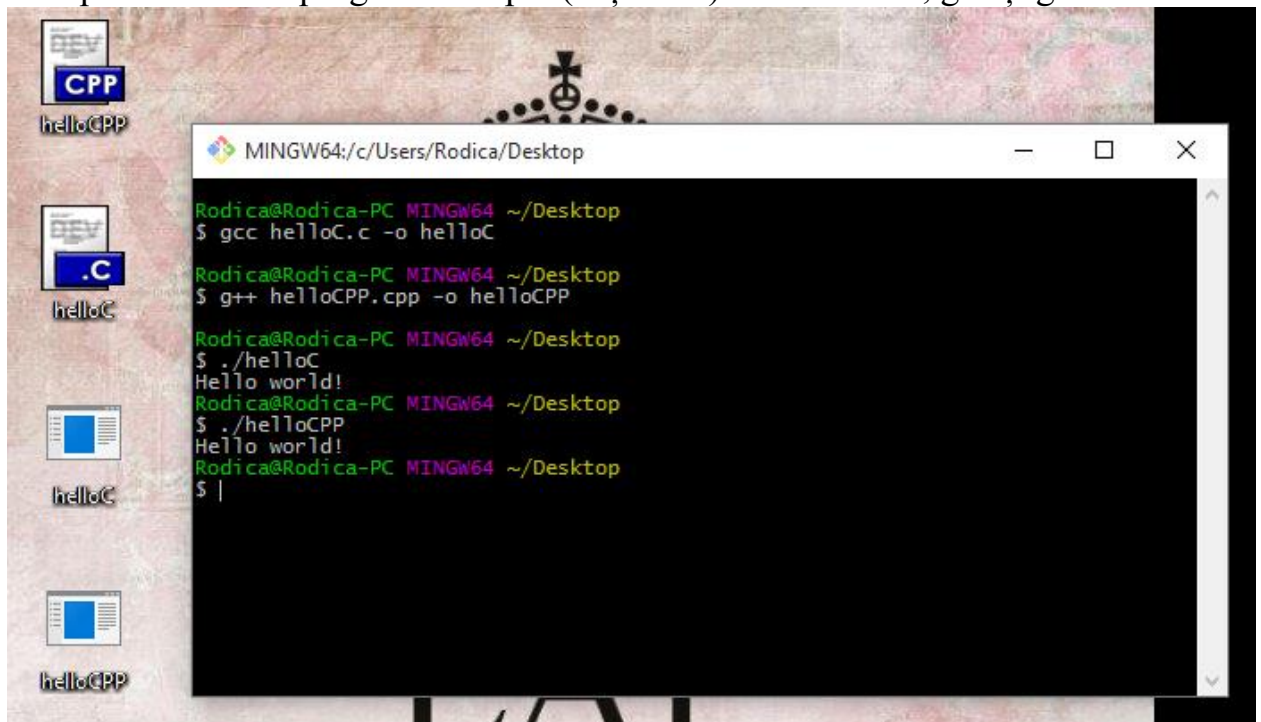
#### **3.2 Analiza lucrării de laborator**

Link spre repozitoriu: <https://github.com/PlotnicuRodica/MIDPS>

1. Inițial, am creat un repozitoriu pe github. com
2. Mai apoi, am generat cheia SSH publică folosind comanda ssh-keygen. Am adăgat-o în setările repozitorului pentru a avea acces spre remote de la calculatorul personal.
3. Am testat conexiunea făcând un commit, apoi push cu un simplu fișier .txt. Mai apoi am șters acest fișier.
4. Am adăugat fișierele pentru lucrarea de laborator nr.1 utilizând comanda git add.
5. Am adăugat fișierele readme și .gitignore. Fișierul readme l-am modificat după necesitate.
6. Am creat două programe simple HelloWorld în C și C++ și le-am compilat utilizând CLI.
7. Am creat un branch nou, în care am încărcat fișierele programelor în folderul Lab#2 și screenshotul compilării acestora.
7. Am făcut merge la branchul, care conține fișierele compilate în CLI cu branch-ul principal master.
8. Am creat un sample program pentru crearea unei situații de conflict între cele 2 branch-uri.
9. Am trecut la cel de-al doilea branch, am editat fișierul adăugat anterior după care am făcut commit. Apoi am făcut aceeași operație cu brachul master.
10. Apoi am utilizat comanda merge și s-a afișat conflictul apărut.
11. Am observat în sample program conflictul apărut, am efectuat schimbările necesare și am salvat versiunea finală.
12. Conflictul s-a rezolvat după executarea unui commit.

### 3.3 Imagini

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

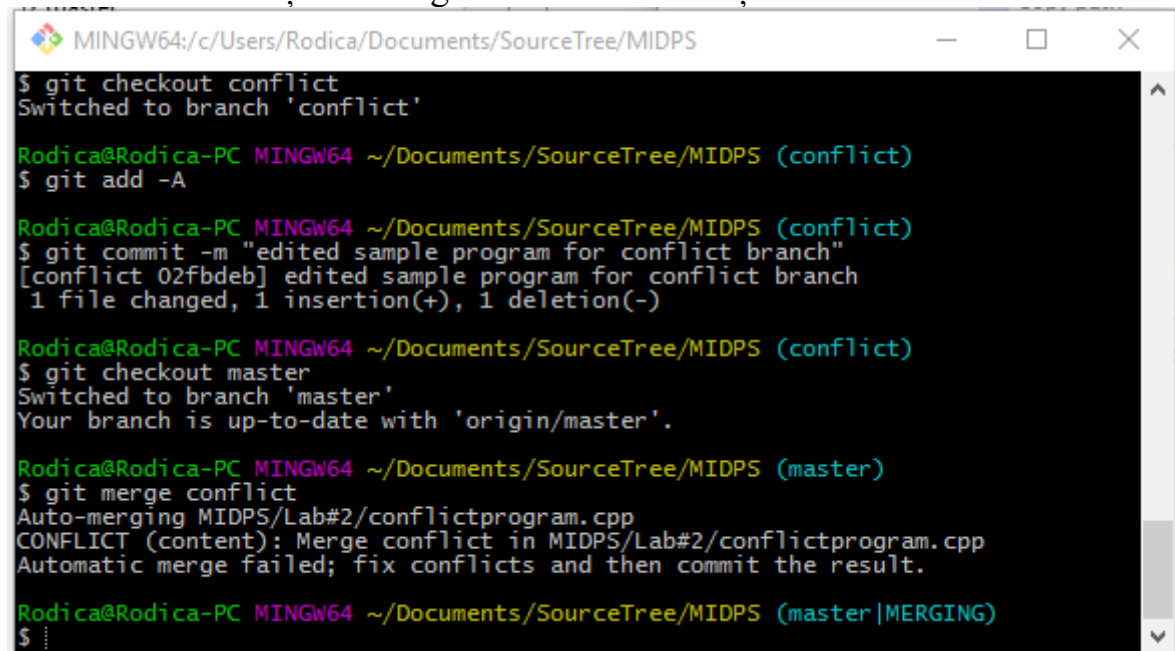


The screenshot shows a Windows desktop with a MINGW64 terminal window open. The terminal displays the following commands and output:

```
MINGW64:/c/Users/Rodica/Desktop
Rodica@Rodica-PC MINGW64 ~/Desktop
$ gcc helloC.c -o helloC
Rodica@Rodica-PC MINGW64 ~/Desktop
$ g++ helloCPP.cpp -o helloCPP
Rodica@Rodica-PC MINGW64 ~/Desktop
$ ./helloC
Hello world!
Rodica@Rodica-PC MINGW64 ~/Desktop
$ ./helloCPP
Hello world!
Rodica@Rodica-PC MINGW64 ~/Desktop
$ |
```

On the desktop, there are four icons: 'helloCPP' (with a CPP icon), 'helloC' (with a .C icon), and two other icons labeled 'helloC' and 'helloCPP'.

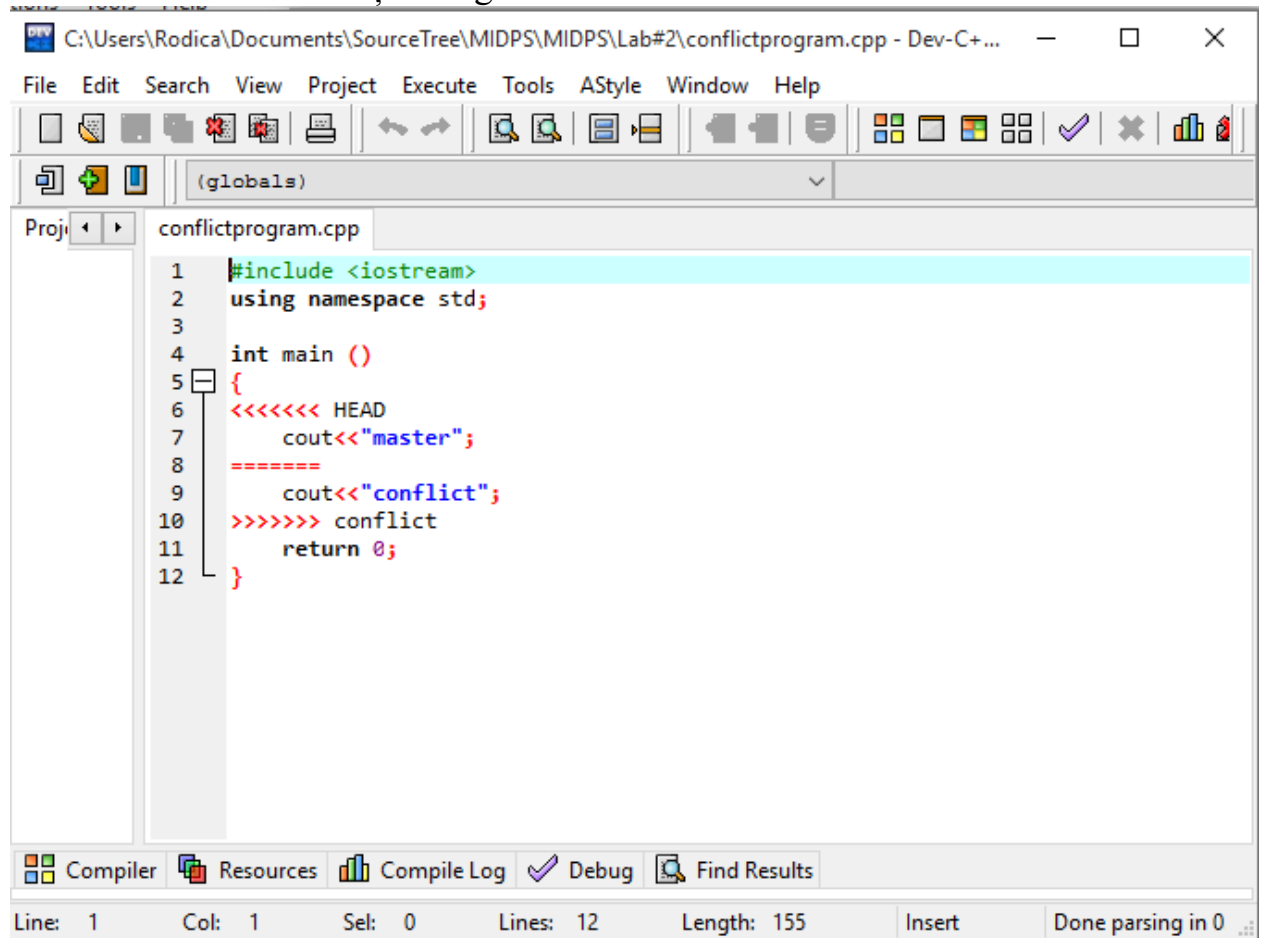
Executarea instrucțiunii merge ce creează o situație de conflict



The screenshot shows a MINGW64 terminal window with the following commands and output:

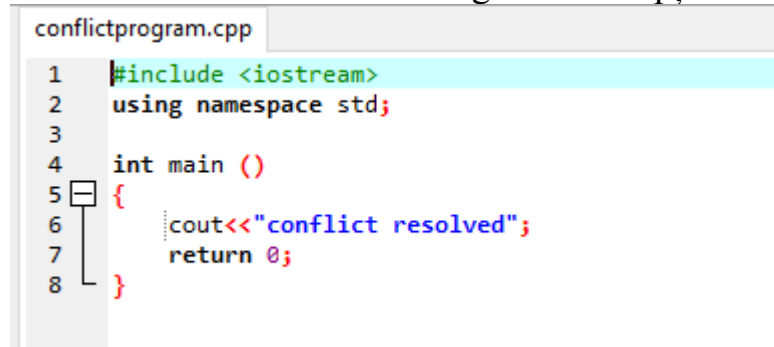
```
MINGW64:/c/Users/Rodica/Documents/SourceTree/MIDPS
$ git checkout conflict
Switched to branch 'conflict'
Rodica@Rodica-PC MINGW64 ~/Documents/SourceTree/MIDPS (conflict)
$ git add -A
Rodica@Rodica-PC MINGW64 ~/Documents/SourceTree/MIDPS (conflict)
$ git commit -m "edited sample program for conflict branch"
[conflict 02fbdeb] edited sample program for conflict branch
1 file changed, 1 insertion(+), 1 deletion(-)
Rodica@Rodica-PC MINGW64 ~/Documents/SourceTree/MIDPS (conflict)
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
Rodica@Rodica-PC MINGW64 ~/Documents/SourceTree/MIDPS (master)
$ git merge conflict
Auto-merging MIDPS/Lab#2/conflictprogram.cpp
CONFLICT (content): Merge conflict in MIDPS/Lab#2/conflictprogram.cpp
Automatic merge failed; fix conflicts and then commit the result.
Rodica@Rodica-PC MINGW64 ~/Documents/SourceTree/MIDPS (master|MERGING)
$ |
```

## Conflictul în cod evidențiat de git



```
1 #include <iostream>
2 using namespace std;
3
4 int main ()
5 {
6 <<<<<< HEAD
7     cout<<"master";
8     =====
9     cout<<"conflict";
10 >>>>>> conflict
11     return 0;
12 }
```

## Rezolvarea conflictului sau alegerea unei opțiuni alternative



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

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

```
MINGW64:/c/Users/Rodica/Documents/SourceTree/MIDPS
$ git merge conflict
Auto-merging MIDPS/Lab#2/conflictprogram.cpp
CONFLICT (content): Merge conflict in MIDPS/Lab#2/conflictprogram.cpp
Automatic merge failed; fix conflicts and then commit the result.

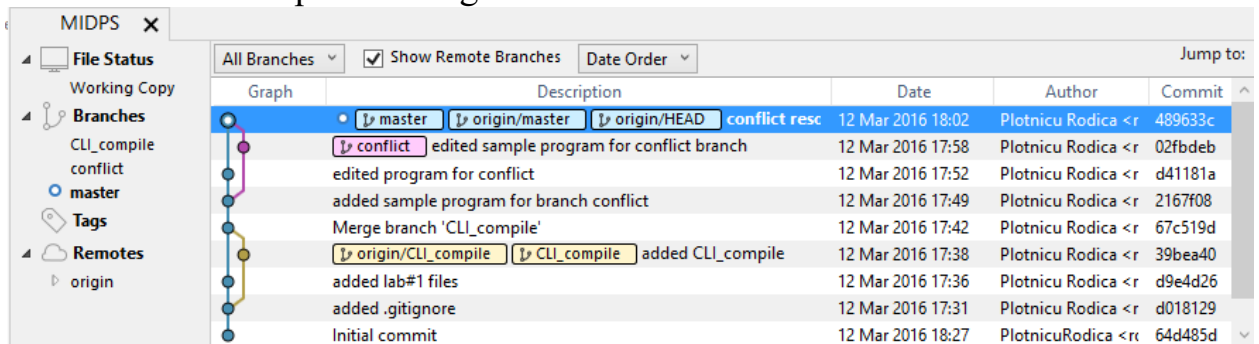
Rodica@Rodica-PC MINGW64 ~/Documents/SourceTree/MIDPS (master|MERGING)
$ git add -A

Rodica@Rodica-PC MINGW64 ~/Documents/SourceTree/MIDPS (master|MERGING)
$ git commit -m "conflict resolved"
[master 489633c] conflict resolved

Rodica@Rodica-PC MINGW64 ~/Documents/SourceTree/MIDPS (master)
$ git push origin master
Counting objects: 10, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 1.04 KiB | 0 bytes/s, done.
Total 10 (delta 2), reused 0 (delta 0)
To https://github.com/PlotnicuRodica/MIDPS.git
d41181a..489633c master -> master

Rodica@Rodica-PC MINGW64 ~/Documents/SourceTree/MIDPS (master)
$ |
```

Procesul final cu reprezentare grafică folosind soft-ul SourceTree



The screenshot shows the SourceTree application interface. On the left, a sidebar contains 'File Status' (Working Copy), 'Branches' (CLI\_compile, conflict, master), 'Tags', and 'Remotes' (origin). The main area displays a commit history table with columns: Graph, Description, Date, Author, and Commit. The table lists several commits, including 'Initial commit', 'added .gitignore', 'added lab#1 files', 'added CLI\_compile', 'Merge branch 'CLI\_compile'', and 'added sample program for branch conflict'. The top commit is 'conflict resolved' (489633c) by Plotnicu Rodica on 12 Mar 2016.

Graph	Description	Date	Author	Commit
	conflict resolved	12 Mar 2016 18:02	Plotnicu Rodica <r	489633c
	edited sample program for conflict branch	12 Mar 2016 17:58	Plotnicu Rodica <r	02fbdeb
	edited program for conflict	12 Mar 2016 17:52	Plotnicu Rodica <r	d41181a
	added sample program for branch conflict	12 Mar 2016 17:49	Plotnicu Rodica <r	2167f08
	Merge branch 'CLI_compile'	12 Mar 2016 17:42	Plotnicu Rodica <r	67c519d
	added CLI_compile	12 Mar 2016 17:38	Plotnicu Rodica <r	39bea40
	added lab#1 files	12 Mar 2016 17:36	Plotnicu Rodica <r	d9e4d26
	added .gitignore	12 Mar 2016 17:31	Plotnicu Rodica <r	d018129
	Initial commit	12 Mar 2016 18:27	PlotnicuRodica <r	64d485d

## Concluzie

În urma realizării acestei lucrări de laborator, am aflat cum putem realiza crearea unui repozitoriu și cum îl putem utiliza. De asemenea, am studiat metodele de lucru cu Version Control Systems, am utilizat CLI. Am utilizat repozitoriul pentru a salva fișiere pe un server remote. VCS poate fi utilizat pentru mentinerea unui proiect mare, pentru lucrul asupra acestui proiect a mai multor persoane, ce este foarte eficient. Putem să împărțim proiectul în mai multe sub-proiecte utilizând branch-uri; să ne întoarcem la versiuni precedente a proiectului utilizând tracking-ul versiunilor sale. Astfel, VCS este foarte util în lucrul cu proiecte mari.

## Referinte

1. Google //google it