

Versionarea codului sursă utilizând GIT

Scopul lucrării

Lucrarea de laborator are ca scop studiul și înțelegerea principiilor de funcționare și utilizare a sistemului distribuit de control al versiunilor numit GIT.

Obiectivele

Crearea unui repozitoriu distant, localizat de serviciul *github*, și sincronizarea tuturor modificărilor efectuate asupra repozitoriului local.

Ce este VCS ?

Sistemele de versionare (VCS, Version Control Systems - eng.) servesc la gestionarea versiunilor multiple ale fișierelor incluse într-un proiect colaborativ. Fiecare modificare efectuată asupra elementului de proiect se memorizează împreună cu autorului schimbării. Important de menționat că în orice moment de timp se poate reveni la o versiune anterioară a entității.

Motivatia principala consta in posibilitatea ca diferiti membri ai echipei, aflati eventual in spatii geografice indepartate, sa poata lucra simultan la proiect, urmand ca, la final, modificarile lor sa fie reunite in noi versiuni ale proiectului. De asemenea, exista si alte avantaje. Cand se observa un bug, se poate reveni la o versiune anterioara, in vederea determinarii momentului introducerii acestuia in program. In acelasi timp, se poate urma o dezvoltare pe ramuri (branches), in care se lucreaza, in paralel, la multiple versiuni ale proiectului - de exemplu, una in care se doreste inlaturarea bug-urilor, iar cealalta, in care se urmareste adaugarea de noi functionalitati, inaintea slefuirii celor existente.

Terminologie

repository - pe server, conține ierarhia de fișiere și informațiile de versiune;

working copy - varianta locală, obținută de la server, pe care se fac modificările;

revision - o versiune a unui document. (v1, v2, v3...).

checkout - aducerea pe masina locala a versiunii de pe server, sub forma unei working copy

update/pull: actualizarea repozitoriului local în funcție de modificările survenite, între timp, pe server. Se aduc doar fișierele modificate;

commit - înregistrează o nouă versiune a fișierului (fișierelor) modificat în repozitoriu.

commit message - un mesaj asociat unei acțiuni commit care descrie schimbările făcute în noua versiune.

changelog - o listă a versiunilor (commit-urilor) unui fișier/proiect de obicei însoțită de mesajele asociate fiecărui commit.

diff: Afișează diferențele dintre două versiuni a unui fișier sau dintre fișierul modificat local (pe working copy) și o versiune de pe repository.

revert - renunțarea la ultimele modificări (locale) făcute într-un fișier din working copy, și revenirea la ultima versiune aflată în repozitoriu sau la o versiune la alegere.

branch - creează o "copie" a unui fișier/proiect pentru modificări „în paralel” fără a afecta starea actuală a unui proiect.

merge - aplică ultimele modificări dintr-o versiune a unui fișier peste alt fișier;

conflict - situația în care un merge nu se poate executa automat și modificările locale sunt în conflict cu modificările din repozitoriu.

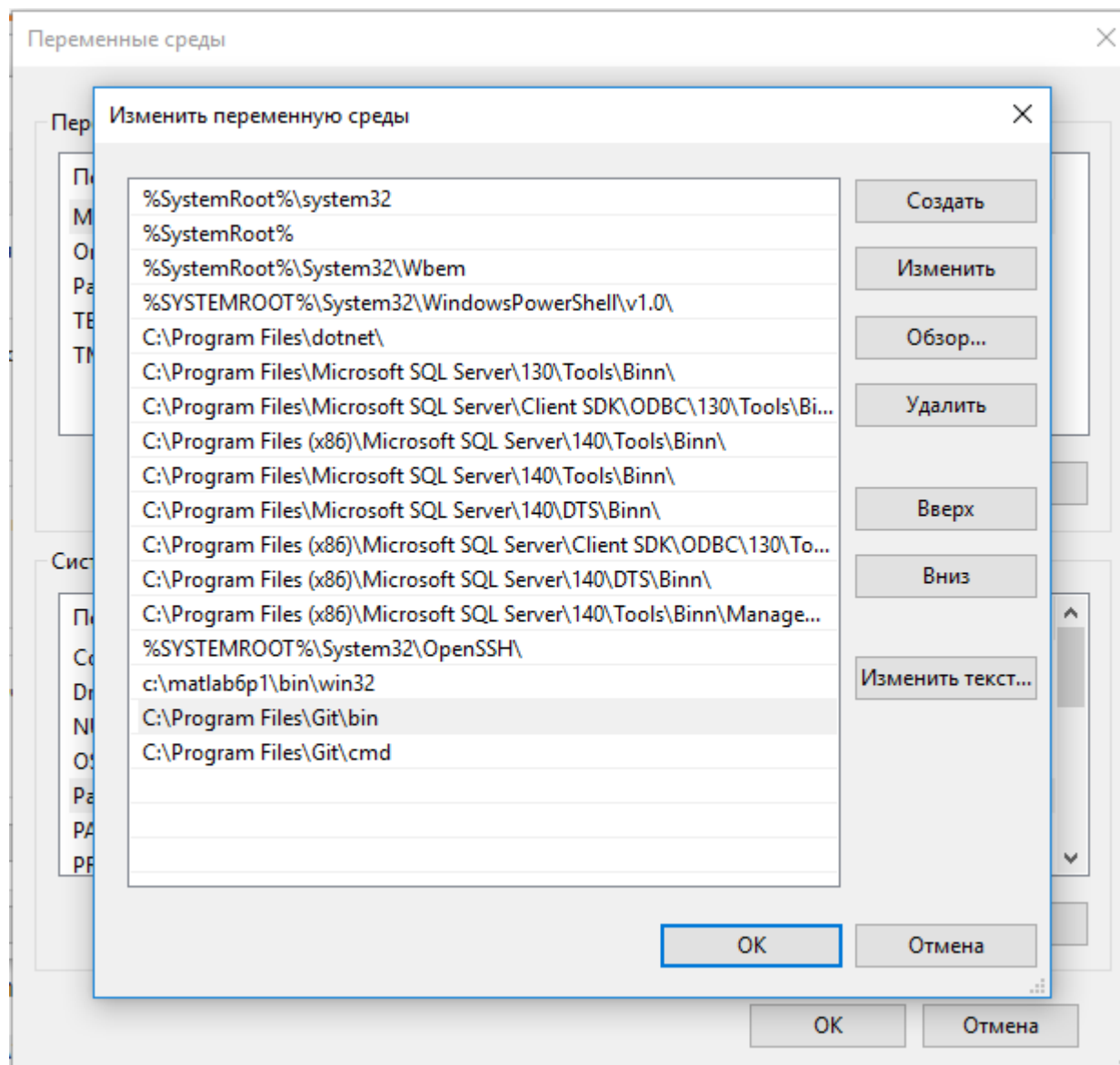
resolve: rezolvarea (de obicei manuală) a conflictelor apărute într-un fișier după un merge.

Proces de lucru

Initial trebuie de creat folder pentru a pastra copia repozitoriului de pe Github. Repozitoriu local va fi pastrat pe diskul D. Acolo creez folder Network_Programming (el fiind numele de repository). Toate manipulatiile fac cu *Command Line* (**cmd**) si cu ajutorul ei ajung direct in centrul folderului creat.

```
d:
mkdir Network_Programming
cd Network_Programming
```

Ca sa lucrez cu git comenzile, trebuie sa scriu PATHurile globale de git/bin si git/cmd in Windows.



Aici eu initializez gitul. In repository local va fi creat git folder in care va fi pastrata toata istoria modificarii continutului din folder respectiv (adaugarea, stergerea si modificarea continutului).

```
git init
```

```
Командная строка
status      Show the working tree status

grow, mark and tweak your common history
branch      List, create, or delete branches
checkout    Switch branches or restore working tree files
commit      Record changes to the repository
diff        Show changes between commits, commit and working tree, etc
merge       Join two or more development histories together
rebase      Reapply commits on top of another base tip
tag         Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
fetch       Download objects and refs from another repository
pull        Fetch from and integrate with another repository or a local branch
push        Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.

D:\Network_Programming>git init
Initialized empty Git repository in D:/Network_Programming/.git/

D:\Network_Programming>
```

E timp sa creez cateva filuri, pentru ca repositoryu dat inca nu contine nici un file. Cu ajutorul comenzii **echo** din **cmd** creez filuri cu careva continut intre ele.

```
echo "First file to be committed" > first.txt
echo "This is my second file..." > second.txt
```

Verific daca au fost adaugate filurile respective.

```
dir
```

```
Командная строка
push        Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.

D:\Network_Programming>git init
Initialized empty Git repository in D:/Network_Programming/.git/

D:\Network_Programming>echo "First file to be committed" first.txt
"First file to be committed" first.txt

D:\Network_Programming>echo "First file to be committed" > first.txt

D:\Network_Programming> echo "This is my second file..." > second.txt

D:\Network_Programming>dir
Том в устройстве D имеет метку Work
Серийный номер тома: E89F-322B

Содержимое папки D:\Network_Programming

06.02.2019  12:16    <DIR>          .
06.02.2019  12:16    <DIR>          ..
06.02.2019  12:15             30 first.txt
06.02.2019  12:16             30 second.txt
               2 файлов             60 байт
               2 папок   80 579 723 264 байт свободно

D:\Network_Programming>
```

Toata lista comenzilor de git poate fi vazuta, daca o sa introduc urmatoarea comanda:

```
git help -a
```

Ca sa continuu lucru trebuie sa apas una din urmatoarele comenzi:

```
CTRL+C
```

sau

```
Q
```

E timpul sa adaug modificarile pe care am facut in repository local pe cel global din Github. Initial pe Gitul meu trebuie sa fie creat un repo gol care are aceeasi nume ca si cel local: *Network_Programming*

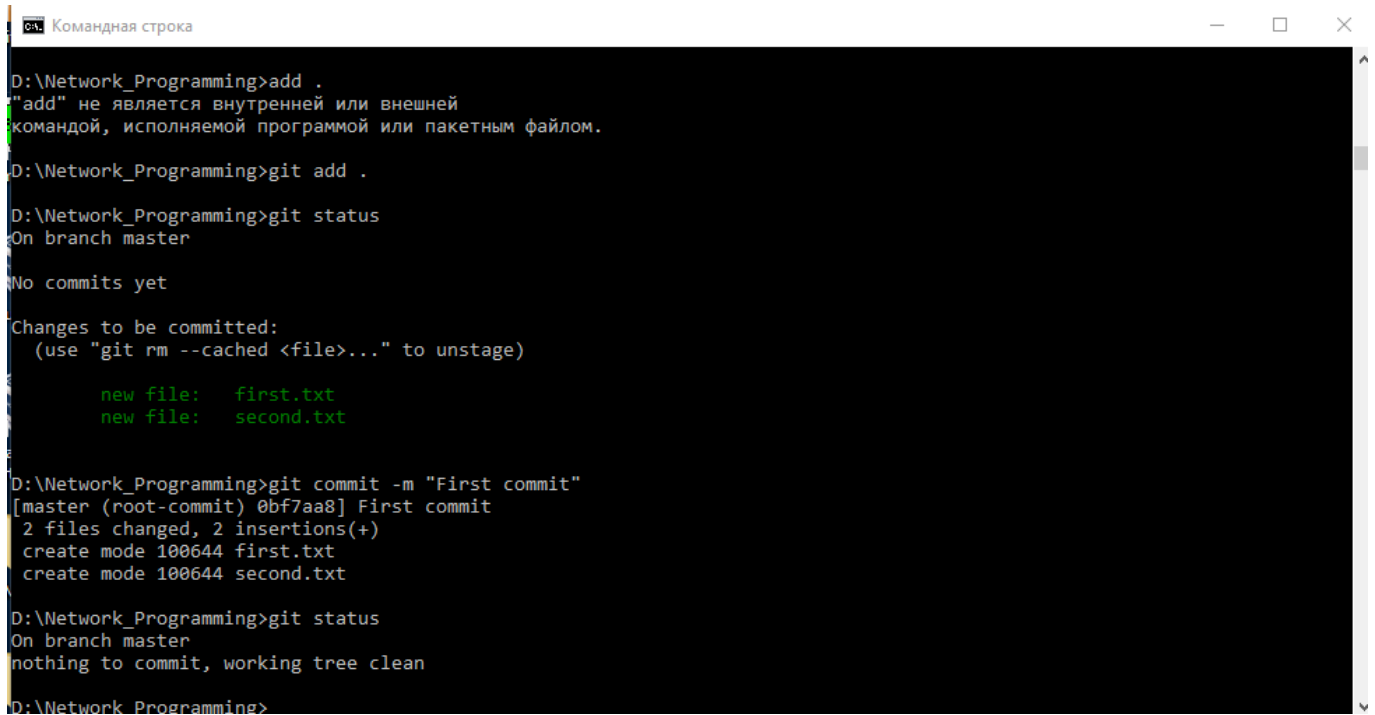
Cu ajutorul comenzii *git status* pot sa vad cum se petrec modificarile respective.

```
git add .  
git status
```

```
patch-id          Compute unique ID for a patch  
sh-i18n           Git's i18n setup code for shell scripts  
sh-setup          Common Git shell script setup code  
strip-space       Remove unnecessary whitespace  
  
External commands  
  flow  
  lfs  
  
D:\Network_Programming>  
  
D:\Network_Programming>add .  
"add" не является внутренней или внешней  
командой, исполняемой программой или пакетным файлом.  
  
D:\Network_Programming>git add .  
  
D:\Network_Programming>git status  
On branch master  
  
No commits yet  
  
Changes to be committed:  
  (use "git rm --cached <file>..." to unstage)  
  
    new file:   first.txt  
    new file:   second.txt  
  
D:\Network_Programming>
```

Pentru a face un commit trebuie de scris comanda de jos si sa adauge un comentariu ca mai apoi sa fie clar cu ce fel de modificari eu am treaba la acest commit. Commitul acesta este initial.

```
git commit -m "First commit"
git status
```



```
Командная строка
D:\Network_Programming>add .
"add" не является внутренней или внешней
командой, исполняемой программой или пакетным файлом.

D:\Network_Programming>git add .

D:\Network_Programming>git status
On branch master

No commits yet

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

        new file:   first.txt
        new file:   second.txt

D:\Network_Programming>git commit -m "First commit"
[master (root-commit) 0bf7aa8] First commit
 2 files changed, 2 insertions(+)
 create mode 100644 first.txt
 create mode 100644 second.txt

D:\Network_Programming>git status
On branch master
nothing to commit, working tree clean

D:\Network_Programming>
```

Aici trebuie sa conectez repozitoriu local cu cel de la git. La git initial a fos creat un repo gol. Daca acela repozitoriu ar fi continea careva fisiere care nu-s prezente in directoriu meu local, ar fi nevoie sa utilizez *pull* comanda.

```
git remote add origin https://github.com/VladGanuscheak/Network_Programming
```

Daca conexiune ar fi existat deja, sistemul ar da urmatorul mesaj: ***fatal: remote origin already exists.***

Acum trimet toate modificarile pe Github:

```
git push -u origin master
```

Scriind urmatoarea comanda

```
git status
```

verific ca operatia data s-a finisat cu succes!

Cind lucram cu git, deseori ne utilizam diferite ramuri (dev, releas, hotfix etc.). Aici creez unul dintre ele:

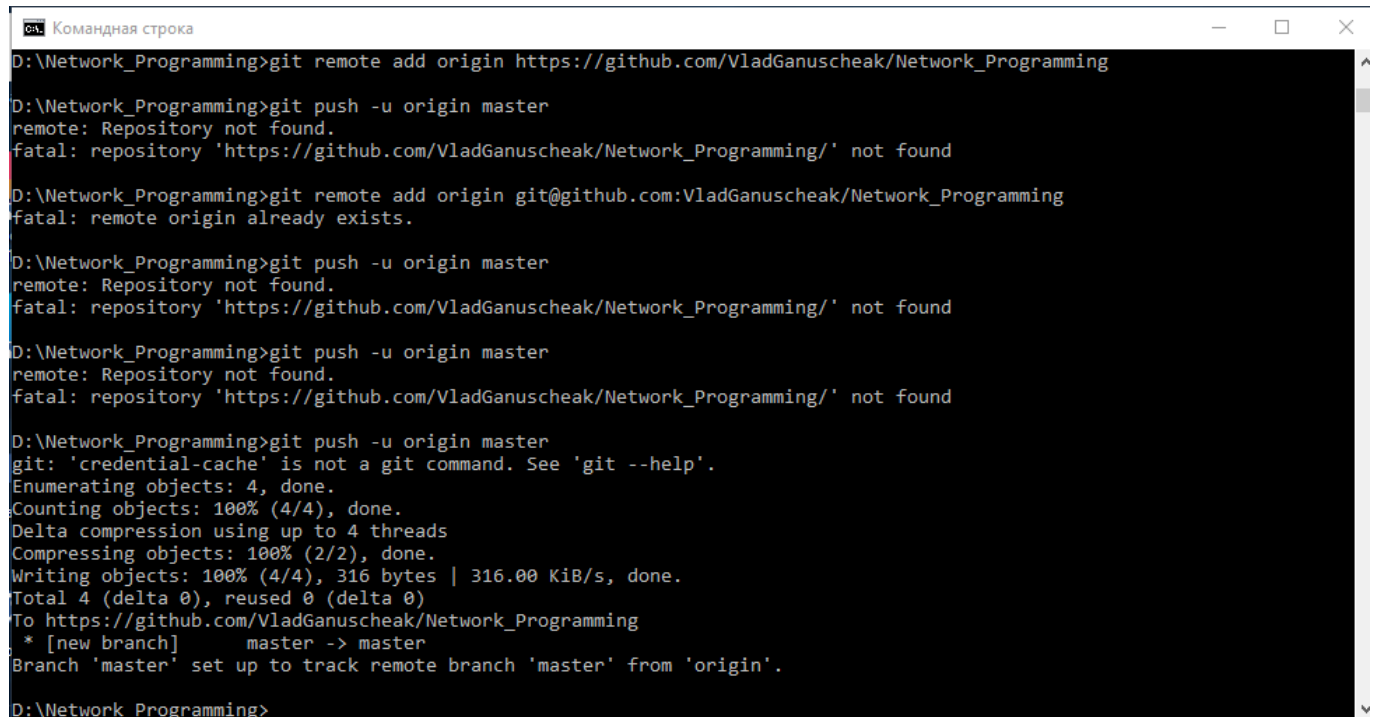
```
git branch dev
git checkout dev
```

Comenzile de mai sus ar fi putut sa fie scrise in forma urmatoare:

```
git checkout -b dev
```

In codul de mai jos le unific ramurile dev cu master si reinnoiesc repozitoriu distant:

```
git push --set-upstream origin dev
git merge dev
git push origin master
```



```
Командная строка
D:\Network_Programming>git remote add origin https://github.com/VladGanuscheak/Network_Programming
D:\Network_Programming>git push -u origin master
remote: Repository not found.
fatal: repository 'https://github.com/VladGanuscheak/Network_Programming/' not found

D:\Network_Programming>git remote add origin git@github.com:VladGanuscheak/Network_Programming
fatal: remote origin already exists.

D:\Network_Programming>git push -u origin master
remote: Repository not found.
fatal: repository 'https://github.com/VladGanuscheak/Network_Programming/' not found

D:\Network_Programming>git push -u origin master
remote: Repository not found.
fatal: repository 'https://github.com/VladGanuscheak/Network_Programming/' not found

D:\Network_Programming>git push -u origin master
git: 'credential-cache' is not a git command. See 'git --help'.
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 316 bytes | 316.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/VladGanuscheak/Network_Programming
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

D:\Network_Programming>
```