

Ministerul Educației al Republicii Moldova

Universitatea Tehnică a Moldovei

Catedra: Automatica și Tehnologii Informaționale

RAPORT

Lucrare de laborator Nr.2
la M.I.D.P.S.

A efectuat:

st. gr. TI-144
I. Stratan

A verificat:

lector.univ.
I.Cojanu
lector.sup.
S.Cojocar

Chișinău 2016

LUCRAREA DE LABORATOR 2

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

Laboratory Requirements:

- *Basic Level* (nota 5 || 6) :
 - conecteaza-te la server folosind SSH
 - compileaza cel putin 2 sample programs din setul HelloWorldPrograms folosind CLI
 - executa primul commit folosind VCS
- *Normal Level* (nota 7 || 8):
 - initializeaza un nou repository
 - configureaza-ti VCS
 - crearea branch-urilor (creeaza cel putin 2 branches)
 - commit pe ambele branch-uri (cel putin 1 commit per branch)
- *Advanced Level* (grade 9 || 10):
 - seteaza un branch to track a remote origin pe care vei putea sa faci push (ex. Github, Bitbucket or custom server)
 - reseteaza un branch la commit-ul anterior
 - merge 2 branches
 - conflict solving between 2 branches
- *Bonus Point:*
 - Scribe un script care va compila HelloWorldPrograms projects.

Taskurile folosite :

```
git config --global user.name "You name"
git config --global user.email "You email"
git config --list
cd ~/.ssh
ssh-keygen
cat ~/.ssh/id_rsa.pub
git init
git status
git clone
git add .
git commit -m "message"
git branch name
git checkout nume
git branch -a
git branch -vv
git push origin master
```

Analiza lucrări de laborator :

Linkul meu <https://github.com/StratanIon/MIDPSv1>

- Am folosit aceste două comenzi pentru a configura github , adica am introdus numele meu de pe github și pe care email a fost înregistrat acest acaunt.

```
git config --global user.name "You name"
```

```
git config --global user.email "You email"
```

- Am folosit această comandă git config --list pentru a vedea lista de configurări.
- Aceste trei comenzi c d ~/.ssh , ssh-keygen , cat ~/.ssh/id_rsa.pub le-am folosit pentru a genera o cheie ssh și ultima comandă îmi afișează cheia ssh , care am introdus-o pe github . Această cheie ne dă posibilitatea de a lucra cu repozițoriu nostru în oricare timp , fără a introduce de fiecare dată numele și parola la github.
- Comanda git init inițializează mapa noastră ,care crează o mapă nevizibilă cu numele .git , unde în această mapă conține toate resursele pe care le folosim .
- Comanda git status ne reprezintă statutul mapei noastre în momentul de timp dat.
- Comanda git clone, realizează clonarea mapei noastre , sau poate să cloneze alte fișiere de pe github.
- Comanda git add. , adauga in local server fișierele noi . Punctul reprezintă adaugarea tuturor fișierilor , dar ca să adăugăm un singur fișier trebuie sa scriem git add numele fișierului.
- Comanda git commit -m " " , crează un commit cu mesajul dat , adică ne arată ce sa modificat într-un interval de timp.
- Comanda git branch , reprezintă niște ramuri a proiectului nostru, ramura principală este master , și putem să creem multe ramuri . Fiecare ramură reprezintă niște schimbări ale noastre care nu poate să influențeze direct la proiectul principal . Adica fiecare programist își are misiunea lui și fiecare program creat de el asupra proiectului este inclus în branch lui.
- Comanda git checkout nume , aici se realizează interschimbările între branchuri.
- Comanda git branch -a , ne arată totalitatea branchurilor.
- Comanda git branch --v, ne arată totalitatea branchurilor cu comitele lor respective.
- Comanda git push origin master , realizează introducerea fișierilor pe github din mapa ta locală.

Conectare la server folosind SSH :

Folosesc comanda : `cd ~/.ssh` , apoi `ssh-keygen` , pentru a genera o cheie.

```
ION@Jony MINGW64 ~/Desktop/MIDPS (master)
$ cd ~/.ssh

ION@Jony MINGW64 ~/.ssh
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/ION/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/ION/.ssh/id_rsa.
Your public key has been saved in /c/Users/ION/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:Ta4IhXFR712I4YSJvYo391YaCxF3yHEw2a8de7hX7fw ION@Jony
The key's randomart image is:
+---[RSA 2048]-----+
|      . 0=. +**      |
|      +. =+==0.      |
|      . .  +=....    |
|      . 0= . .0       |
|      .. .S.+ .0 +.   |
|      ..+.0.. 0 + +   |
|      ..0.0 =  =.     |
|      = . +          |
|      . .E           |
+---[SHA256]-----+
```

Ca sa obtin cheia scriu : cat ~/.ssh/id_rsa.pub

```
I0N@Jony MINGW64 ~/.ssh  
$ cat ~/.ssh/id_rsa.pub  
ssh-rsa AAAA8C3NzaC1yc2EAAAADAQABAAQAC1/PQO3TfBaX/cZr9eTA/oRpk2TULFy1NR5MD8tAICKsbB6G6aEI0Som  
k82KjP9lT7u9bKGPGXVxtQGgs/h0iSegnfUAq/Mwqm04htvgheQd3DDJqWD0/8Yd6wEEv03IoaSRn1WzDe1gUM1jBtQTka  
II/MDiRUBySm0jDceJQv00gcxMU9gTMJ5b3U5aHFU8RDQgDiZL1YCvi9ydsy+HKK5-F4TF8rh2fwg/xJg15Fsu1DHPDU9E  
EGIIbTR/+t/8vPDsrAsA5ew3Izc/Qy+AazD3w1kqC63AcXRMR4htHO67q8SUCkgTq+IEoHWSf5w123eatYI  
mNN TON@Jony
```

Și plasez cheia pe Github :

Personal settings

Profile

Account settings

Emails

Notification center

Billing

SSH keys

Security

OAuth applications

Personal access tokens

Repositories

Organizations

SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Stratan Ion key

c7:3c:0b:1c:3a:35:ed:de:90:16:4c:fa:20:5a:48:ef

Added on 15 Feb 2016 — Last used within the last day

Delete

?

 Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH Problems](#).

Execut primul comit :

```
ION@Jony MINGW64 /e/MIDPS (master)
$ git commit -m "Primul meu commit"
[master (root-commit) da20651] Primul meu commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 First.txt
```


Inițializez un nou repozitoriu :

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

Repository name

 StratanIon ▾

 /

MIDPSv1 ✓

Great repository names are short and memorable. Need inspiration? How about [vigilant-couscous](#).

Description (optional)

☒ Public

Anyone can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.


Add .gitignore: **None** ▾

 |

Add a license: **None** ▾ ⓘ

Create repository

Quick setup — if you've done this kind of thing before

 Set up in Desktop

 or

HTTPS

SSH

git@github.com:StratanIon/MIDPSv1.git

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# MIDPSv1" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:StratanIon/MIDPSv1.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:StratanIon/MIDPSv1.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

Configurez VCS :

```
ION@Jony MINGW64 /e/MIDPS (master)
$ git config --global user.name "StratanIon"

ION@Jony MINGW64 /e/MIDPS (master)
$ git config --global user.email "95stratosfera@mail.ru"
```

```
ION@Jony MINGW64 /e/MIDPS (master)
$ git config --list
core.symlinks=false
core.autocrlf=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
diff.astextplain.textconv=astextplain
rebase.autosquash=true
user.email=95stratosfera@mail.ru
user.name=StratanIon
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
core.hidedotfiles=dotGitOnly
```

Creiez două branchuri :

```
ION@Jony MINGW64 /e/MIDPS (master)
$ git branch one-ver

ION@Jony MINGW64 /e/MIDPS (master)
$ git branch two-ver
```

```
ION@Jony MINGW64 /e/MIDPS (master)
$ git branch -a
* master
  one-ver
  two-ver
```

Commit la aceste două branchuri :

```
ION@Jony MINGW64 /e/MIDPS (one-ver)
$ git commit -m "text nou"
[one-ver 2a54312] text nou
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 ab.txt
```

```
ION@Jony MINGW64 /e/MIDPS (two-ver)
$ git commit -m "text in al doilea branch"
[two-ver 3e525a8] text in al doilea branch
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 ba.txt
```

Concluzie

În urma efectuării acestui laborator , am înțeles cum funcționează GITHUB. M-am folosit de unele comenzi pentru a îndeplini sarcinile , comenzile nu sunt grele. Acest mediu este foarte bun, deoarece este folosit la proiecte mari în companii, și e comod de a lucra cu el, pentru că îți aduce folos și îți economisește mult timp.