

# Отчет по лабораторной работе №2

## Операционные системы

Глущенко Евгений Игоревич

### Содержание

1	Цель работы .....	1
2	Задание.....	1
3	Выполнение лабораторной работы .....	1
4	Выводы.....	7
5	Ответы на контрольные вопросы .....	7
	Список литературы.....	9

### 1 Цель работы

Цель данной работы - изучить идеологию и применение средств контроля версий, освоить умения по работе с git.

### 2 Задание

Создать базовую конфигурацию для работы с git.

Создать ключ SSH.

Создать ключ PGP.

Настроить подписи git.

Зарегистрироваться на Github.

Создать локальный каталог для выполнения заданий по предмету.

### 3 Выполнение лабораторной работы

##Установка программного обеспечения

Устанавливаю необходимое обеспечение через терминал (рис fig. 1)

```
eiglushchenko@eiglushchenko:~$ sudo dnf -y install git
[sudo] пароль для eiglushchenko:
Последняя проверка окончания срока действия метаданных: 4:19:02 назад, Пт 01 мар 2024 19:11:34.
Пакет git-2.41.0-2.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
eiglushchenko@eiglushchenko:~$ sudo dnf -y install gh
Последняя проверка окончания срока действия метаданных: 4:19:28 назад, Пт 01 мар 2024 19:11:34.
Зависимости разрешены.
```

*Рис. 1: Установка git и gh*

##Базовая настройка git

Записываю свои данные в строках (почта и имя) (рис fig. 1)

```
eiglushchenko@eiglushchenko:~$ git config --global user.name " Evgenii Glushchenko"
eiglushchenko@eiglushchenko:~$ git config --global user.email "1132239110@pfur.ru"
```

*Рис. 2: Задаю email и имя*

Настраиваю utf-8 в выводе сообщений (рис fig. 3)

```
eiglushchenko@eiglushchenko:~$ git config --global core.quotePath false
```

*Рис. 3: Настраиваю utf-8*

Задаю имя master начальной ветке (рис fig. 4)

```
eiglushchenko@eiglushchenko:~$ git config --global init.defaultBranch master
eiglushchenko@eiglushchenko:~$
```

*Рис. 4: Задаю имя master*

Задаю параметры для корректного отображения (рис fig. 5)

```
eiglushchenko@eiglushchenko:~$ git config --global core.autocrlf input
eiglushchenko@eiglushchenko:~$ git config --global core.safecrlf warn
```

*Рис. 5: Задаю autocrlf и safecrlf*

##Создайте ключи ssh

Создаю ssh ключ размером 4096 бит (рис fig. 6)

```
eiglushchenko@eiglushchenko:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/eiglushchenko/.ssh/id_rsa):
Created directory '/home/eiglushchenko/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/eiglushchenko/.ssh/id_rsa
Your public key has been saved in /home/eiglushchenko/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:278Fqp6AbxEtwwCKEiaxDKI2iKiV9FZLTxleyVYJmtw eiglushchenko@eiglushchenko
The key's randomart image is:
+---[RSA 4096]-----+
|+++. ..oo.. o.oo|
|@= o.. . + = =. |
|O+o oo . * E |
|+... = . |
|. +S . |
|. .. o . . |
|. . . . o . . |
|. . . o . . |
|. . . o . . |
|. . .+ o. |
+---[SHA256]-----+
eiglushchenko@eiglushchenko:~$
```

*Рис. 6: Создания SSH ключа*

Создаю ключ по алгоритму ed25519 (рис fig. 7)

```
eiglushchenko@eiglushchenko:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/eiglushchenko/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/eiglushchenko/.ssh/id_ed25519
Your public key has been saved in /home/eiglushchenko/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:HkFTQNIcnWy9cHU9cNq0Vn7JGTXF0kDXE38uP2DWpAE eiglushchenko@eiglushchenko
The key's randomart image is:
+--[ED25519 256]--+
|      .++E.+oBO|
|      .oo0.*o0|
|      o..* B=|
|      . . #.=|
|      S . B.Oo|
|      . o + .o.|
|      .      ..|
|      .      .|
|      .      .|
+-----[SHA256]-----+
```

Рис. 7: Создание ed25519

##Создайте ключи pgp

Генерирую ключ GPG, затем выбираю тип ключа RSA and RSA, задаю максимальную длину ключа: 4096, оставляю неограниченный срок действия ключа. Далее отвечаю на вопросы программы о личной информации (рис fig. 8)

```
eiglushchenko@eiglushchenko:~$ gpg --full-generate-key
gpg (GnuPG) 2.4.3; Copyright (C) 2023 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: создан каталог '/home/eiglushchenko/.gnupg'
Выберите тип ключа:
  (1) RSA and RSA
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (9) ECC (sign and encrypt) *default*
 (10) ECC (только для подписи)
 (14) Existing key from card
Ваш выбор? 1
длина ключей RSA может быть от 1024 до 4096.
Какой размер ключа Вам необходим? (3072) 4096
Запрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
  0 = не ограничен
  <n> = срок действия ключа - n дней
  <n>w = срок действия ключа - n недель
  <n>m = срок действия ключа - n месяцев
  <n>y = срок действия ключа - n лет
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (y/N) y

GnuPG должен составить идентификатор пользователя для идентификации ключа.
Ваше полное имя: EvgeniiGlushchenko
```

Рис. 8: Генерация ключа

Создаю защитную фразу (рис fig. 9)

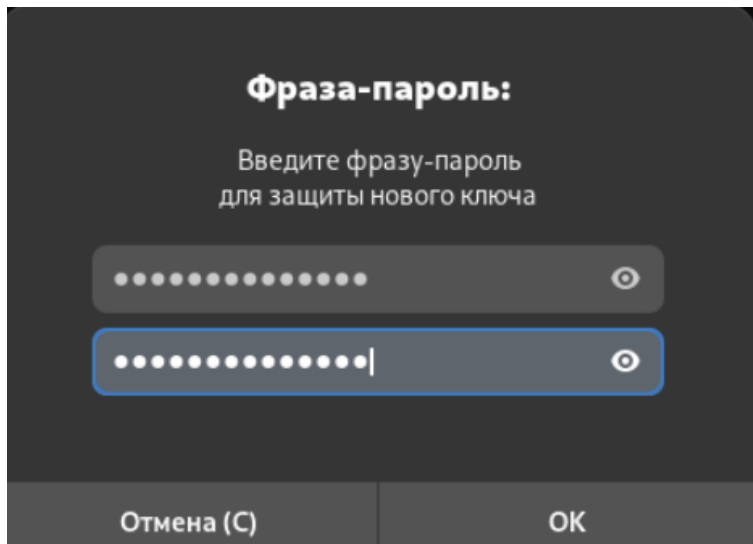


Рис. 9: Пароль для ключа

## ##Настройка github

У меня имеется профиль в гитхабе, работаю на нем (рис fig. 10)

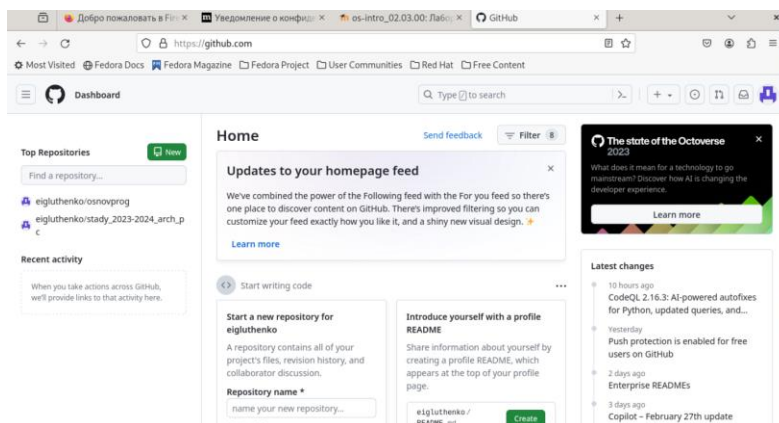


Рис. 10: Профиль в github

## ##Добавление PGP ключа в GitHub

Вывожу список созданных ключей в терминал, ищу в результате запроса отпечаток ключа он стоит после знака слеша, копирую его в буфер обмена (рис fig. 11)

```
eighlushchenko@eighlushchenko:~$ gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: глубина: 0 достоверных: 1 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 1u
[keyboard]
-----
sec   rsa4096/8EF822227A32612 2024-03-01 [SC]
      C6BD0622BE04FC7EBF5181088EF822227A32612
uid           [ абсолютно ] EvgeniiGlushchenko <1132239110@pfur.ru>
ssb   rsa4096/B9DE3E7E571FBCFB 2024-03-01 [E]
eighlushchenko@eighlushchenko:~$
```

Рис. 11: Вывод списка ключей

Копирую ключ с помощью xclip (рис fig. 12)

```
eiglushchenko@eiglushchenko:~$ gpg --armor --export 8EF822227A32612 | xclip -sel clip
eiglushchenko@eiglushchenko:~$
```

Рис. 12: Копирую ключ

Вставляю ключ в GitHub (рис fig. 13)

Add new GPG key

---

Title

Key

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGXiPo8BEADH0eGBMb3oaWww6A6/qmjyvvRZX3y03nwZBbNzyCZf+6jQBU3u
gX9RbuyjzWs6EJEGCvZjvMJEBxMSHx6RZM5hxt02fqkQerLAcrnwhGz0oHuTCNP+
pT54TLmpmgIiucS2HF6uhHBVIQ7x62XCYMTPJ4MajrYe8n4b6MdEorsHNAgtAIq
Ou2ztY1x63KIqXj/MUyAoRTQVxZ+ERwKiMj8xORrvzKDOFyOV1NFtthCUP3c9i0c
eR2yIBIjyLaf/WY4uYOC0Yq/YuwNhl9Y8QJF6y0pzhCvFptsveq80KarsUJWkk
aaFWsUbuDsn4fLBWNvj/GfsfXgqGBVrAKYIpLpgc74KpP37LBHWrEz3d70SV/DjY
2tzGRVMGP2QteWYQVAR3z/Gh0HVU3GanVcLAKK5cYZUoZqU2EzgDjM4cqaVjhaVv
=
```


Рис. 13: Ключ

Ключ добавлен (рис fig. 14)

GPG keys

---

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



Email address: 1132239110@pfur.ru Unverified

Key ID: 8EF822227A32612

Subkeys: B9DE3E7E571FBCFB

Added on Mar 1, 2024

Рис. 14: Проверка ключа

##Настройка автоматических подписей коммитов git

Настраиваю автоматические подписи коммитов git: используя введенный ранее email, указываю git использовать его при создании подписей коммитов (рис fig. 15)

```
eiglushchenko@eiglushchenko:~$ git config --global user.signingkey 8EF822227A32612
eiglushchenko@eiglushchenko:~$ git config --global commit.gpgsign true
error: key does not contain a section: commit
eiglushchenko@eiglushchenko:~$ git config --global commit.gpgsign true
error: key does not contain variable name: commit.
eiglushchenko@eiglushchenko:~$ git config --global commit.gpgsign true
eiglushchenko@eiglushchenko:~$ git config --global pgp.program $(which gpg2)
eiglushchenko@eiglushchenko:~$
```

Рис. 15: Подписи GIT

##Настройка gh

Настраиваю gh, отвечаю на вопросы, авторизуюсь через браузер (рис fig. 16)

```
eiglushchenko@eiglushchenko:~$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser
```

Рис. 16: Настройка gh

Прохожу авторизацию (рис fig. 17)

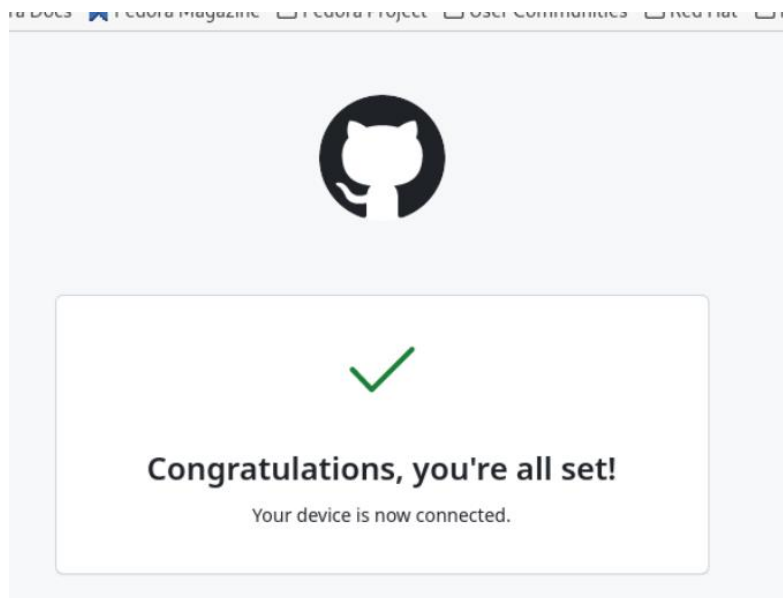


Рис. 17: Завершение авторизации в браузере

Завершаю авторизацию в терминале (рис fig. 18)

```
/ Authentication complete.
- gh config set -h github.com git_protocol https
/ Configured git protocol
/ Logged in as eigluthenko
eiglushchenko@eiglushchenko:~$
```

Рис. 18: Завершение авторизации

##Шаблон для рабочего пространства

Создаю рабочую папку, перехожу в нее, и создаю репозиторий на основе шаблона. (рис fig. 19)

```
eiglushchenko@eiglushchenko:~$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
eiglushchenko@eiglushchenko:~$ cd ~/work/study/2022-2023/"Операционные системы"
eiglushchenko@eiglushchenko:~/work/study/2022-2023/Операционные системы$ gh repo create study_2022-2023_os-intro --template=yamadharma/course-directory-student-template --public
/ Created repository eigluthenko/study_2022-2023_os-intro on GitHub
https://github.com/eigluthenko/study_2022-2023_os-intro
```

Рис. 19: Создание рабочего пространства

Клонирую репозиторий к себе в директорию (рис fig. 20)

```
eiglushchenko@eiglushchenko:~/work/study/2022-2023/Операционные системы$ git clone --recursive https://github.com/eigluthenko/study_2022-2023_os-intro.git os-intro
Клонирование в «os-intro»...
remote: Enumerating objects: 32, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 32 (delta 1), reused 18 (delta 0), pack-reused 0
Получение объектов: 100% (32/32), 18.60 КиБ | 414.00 КиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.
```

*Рис. 20: Создание репозитория*

Перехожу в каталог курса и проверяю его содержание. (рис fig. 21)

```
eiglushchenko@eiglushchenko:~/work/study/2022-2023/Операционные системы$ cd os-intro
eiglushchenko@eiglushchenko:~/work/study/2022-2023/Операционные системы/os-intro$ ls
CHANGELOG.md  COURSE  Makefile  README.en.md  README.md
config        LICENSE  package.json  README.git-flow.md  template
```

*Рис. 21: Перемещение между дерикториями*

Удаляю лишние файлы (рис fig. 22)

```
eiglushchenko@eiglushchenko:~/work/study/2022-2023/Операционные системы/os-intro$ rm package.json
eiglushchenko@eiglushchenko:~/work/study/2022-2023/Операционные системы/os-intro$ echo os-intro > COURSE
eiglushchenko@eiglushchenko:~/work/study/2022-2023/Операционные системы/os-intro$ make
```

*Рис. 22: Удаление*

Отправка файлов на github (рис fig. 23)

```
eiglushchenko@eiglushchenko:~/work/study/2022-2023/Операционные системы/os-intro$ git add .
eiglushchenko@eiglushchenko:~/work/study/2022-2023/Операционные системы/os-intro$ git commit -am 'feat(main): make course structure'
[master c7cald] feat(main): make course structure
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
eiglushchenko@eiglushchenko:~/work/study/2022-2023/Операционные системы/os-intro$
```

*Рис. 23: Отправка файлов*

## 4 Выводы

При выполнении данной работы я изучил основы работы с github.

## 5 Ответы на контрольные вопросы

1. Системы контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Они позволяют хранить несколько версий изменяющейся информации, одного и того же документа, может предоставить доступ к более ранним версиям документа. Используется для работы нескольких человек над проектом, позволяет посмотреть, кто и когда внес какое-либо изменение и т. д. VCS ррименяются для: Хранения понлой истории изменений, сохранения причин всех изменений, поиска причин изменений и совершивших изменение, совместной работы над проектами.
2. Хранилище – репозиторий, хранилище версий, в нем хранятся все доку менты, включая историю их изменения и прочей служебной информацией. commit – отслеживание изменений, сохраняет разницу в изменениях. Ис тория – хранит все изменения в проекте и позволяет при необходимости

вернуться/обратиться к нужным данным. Рабочая копия – копия проекта, основанная на версии из хранилища, чаще всего последней версии.

3. Централизованные VCS (например: CVS, TFS, AccuRev) – одно основное хранилище всего проекта. Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет, затем добавляет изменения обратно в хранилище. Децентрализованные VCS (например: Git, Bazaar) – у каждого пользователя свой вариант репозитория (возможно несколько вариантов), есть возможность добавлять и забирать изменения из любого репозитория. В отличие от классических, в распределённых (децентрализованных) системах контроля версий центральный репозиторий не является обязательным.
4. Сначала создается и подключается удаленный репозиторий, затем по мере изменения проекта эти изменения отправляются на сервер.
5. Участник проекта перед началом работы получает нужную ему версию проекта в хранилище, с помощью определенных команд, после внесения изменений пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются. К ним можно вернуться в любой момент.
6. Хранение информации о всех изменениях в вашем коде, обеспечение удобства командной работы над кодом.
7. Создание основного дерева репозитория: `git init` Получение обновлений (изменений) текущего дерева из центрального репозитория: `git pull` Отправка всех произведённых изменений локального дерева в центральный репозиторий: `git push` Просмотр списка изменённых файлов в текущей директории: `git status` Просмотр текущих изменений: `git diff` Сохранение текущих изменений: добавить все изменённые и/или созданные файлы и/или каталоги: `git add .` добавить конкретные изменённые и/или созданные файлы и/или каталоги: `git add имена_файлов` удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): `git rm имена_файлов` Сохранение добавленных изменений: сохранить все добавленные изменения и все изменённые файлы: `git commit -am 'Описание коммита'` сохранить добавленные изменения с внесением комментария через встроенный редактор: `git commit` создание новой ветки, базирующейся на текущей: `git checkout -b имя_ветки` переключение на некоторую ветку: `git checkout имя_ветки` (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой) отправка изменений конкретной ветки в центральный репозиторий: `git push origin имя_ветки` слияние ветки с текущим деревом: `git merge --no-ff имя_ветки` Удаление ветки: удаление локальной уже слитой с основным деревом ветки: `git branch -d имя_ветки` принудительное удаление локальной ветки: `git branch -D имя_ветки` удаление ветки с центрального репозитория: `git push origin :имя_ветки`
8. `git push -all` отправляем из локального репозитория все сохранённые изменения в центральный репозиторий, предварительно создав локальный репозиторий и сделав предварительную конфигурацию.



9. Ветвление - один из параллельных участков в одном хранилище, исходящих из одной версии, обычно есть главная ветка. Между ветками, т. е. их концами возможно их слияние. Используются для разработки новых функций.
10. Во время работы над проектом могут создаваться файлы, которые не следуют добавлять в репозиторий. Например, временные файлы. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов.

## Список литературы

<https://esystem.rudn.ru/mod/page/view.php?id=1098790>