

# **Отчет по лабораторной работе No2**

**Операционные системы**

Нелиа Нджову

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>4</b>	<b>Выводы</b>	<b>18</b>
	<b>Список литературы</b>	<b>19</b>

# Список иллюстраций

3.1	установка git . . . . .	7
3.2	установка gh . . . . .	7
3.3	задаю имя и email владельца репозитория и настройка ut-8 . . . .	7
3.4	задаю имя начальной ветке . . . . .	8
3.5	рис.5:задаю параметры autocrlf . . . . .	8
3.6	рис.6:задаю параметры safecrlf . . . . .	8
3.7	генерация ssh по алгоритму rsa . . . . .	8
3.8	генерация ssh по алгоритму ed25519 . . . . .	9
3.9	генерация ключа . . . . .	9
3.10	защита ключа . . . . .	9
3.11	аккаунт на github . . . . .	10
3.12	вывод список ключей . . . . .	10
3.13	копирование ключа . . . . .	10
3.14	настройка github . . . . .	11
3.15	добавление нового pgr ключа . . . . .	11
3.16	добавление ключ gpg . . . . .	12
3.17	настройка подписей . . . . .	12
3.18	авторизация в gh . . . . .	12
3.19	завершение авторизации через браузер . . . . .	13
3.20	завершение авторизации . . . . .	13
3.21	создание репозитория . . . . .	14
3.22	копирование репозитории . . . . .	14
3.23	удаление файлов . . . . .	14
3.24	создание каталога . . . . .	14
3.25	рис.25:отправка файлов на сервере . . . . .	15
3.26	рис.26:коммент . . . . .	15
3.27	отправка файлов на сервере . . . . .	15

## **Список таблиц**

# 1 Цель работы

Цель работы является изучение идеологии и применения средств контроля версий, освоение умения по работе с git

## 2 Задание

1. Создать базовую конфигурацию для работы с git.
2. Базовая настройка git
3. Создать ключ SSH.
4. Создать ключ PGP.
5. Зарегистрироваться на Github.
6. Добавление PGP ключа в GitHub
7. Настроить подписи git.
8. Настройка gh
9. Создание репозитория курса на основе шаблона

## 3 Выполнение лабораторной работы

### 1. Создать базовую конфигурацию для работы с git.

Я устанавливаю git через терминал с помощью `dnf install git`(рис.1)

```
set on most systems).
nelianjovu@nelianjovu:~$ sudo dnf -y install git
[sudo] password for nelianjovu:
Last metadata expiration check: 0:03:31 ago on Sat
Package git-2.43.2-1.fc39.x86_64 is already install
```

Рис. 3.1: установка git

Я устанавливаю gh через терминал с помощью `dnf install gh`(рис.2)

```
nelianjovu@nelianjovu:~$ sudo dnf -y install gh
Last metadata expiration check: 0:03:58 ago on Sat 2
Dependencies resolved.
```

Рис. 3.2: установка gh

### 2. Базовая настройка git

Я задаю имя и email владельца репозитория (свои имя, фамилию и электронную почту) и настраиваю `ut-8` в выводе сообщений git для корректного отображения(рис.3)

```
Complete!
nelianjovu@nelianjovu:~$ git config --global user.name "nelianj"
nelianjovu@nelianjovu:~$ git config --global user.email "1032239033@pfur"
nelianjovu@nelianjovu:~$ git config --global core.quotepath false
nelianjovu@nelianjovu:~$
```

Рис. 3.3: задаю имя и email владельца репозитория и настройка `ut-8`

Я задаю имя master начальной ветке(рис.4)

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

Рис. 3.4: задаю имя начальной ветке

Я задаю параметры autocrlf и safecrlf для корректного отображения конца строки

```
nelianjovu@nelianjovu:~$ git config --global core.autocrlf input
nelianjovu@nelianjovu:~$
```

Рис. 3.5: задаю параметры autocrlf

```
nelianjovu@nelianjovu:~$ git config --global core.safecrlf warn
nelianjovu@nelianjovu:~$
```

Рис. 3.6: задаю параметры safecrlf

### 3. Создать ключ SSH.

Я создаю ключ ssh размером 4096 ,бит по алгоритму rsa(рис.7)

```
nelianjovu@nelianjovu:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/nelianjovu/.ssh/id_rsa):
Created directory '/home/nelianjovu/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

Рис. 3.7: генерация ssh по алгоритму rsa

Я создаю ключ ssh по алгоритму ed25519(рис.8)



```
nelianjovu@nelianjovu:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/nelianjovu/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/nelianjovu/.ssh/id_ed25519
Your public key has been saved in /home/nelianjovu/.ssh/id_ed25519.pub
The fingerprint is:
```

Рис. 3.8: генерация ssh по алгоритму ed25519

#### 4. Создать ключ PGP.

Я генерирую ключ GPG, затем выбираю RSA и тип ключа RSA, устанавливаю максимальную длину ключа-4096 и оставляю ключ на неограниченный срок действия. Далее я отвечаю на вопросы программы о личной информации(рис.9)

```
nelianjovu@nelianjovu:~$ 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.
```

Рис. 3.9: генерация ключа

Я ввожу фразу-пароль для защиты нового ключа(рис.10)

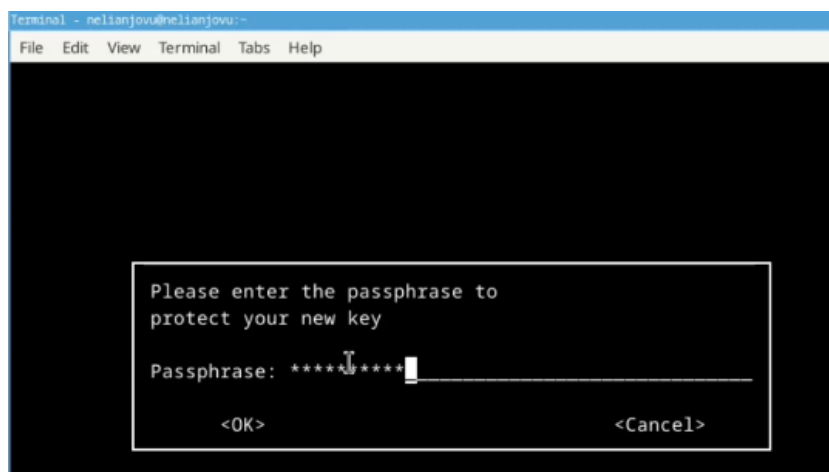


Рис. 3.10: защита ключа

#### 5. Зарегистрироваться на Github.

У меня уже была создана учетная запись на github, и я также заполнила и настроила ее, поэтому я просто вхожу в свою учетную запись(рис.11)

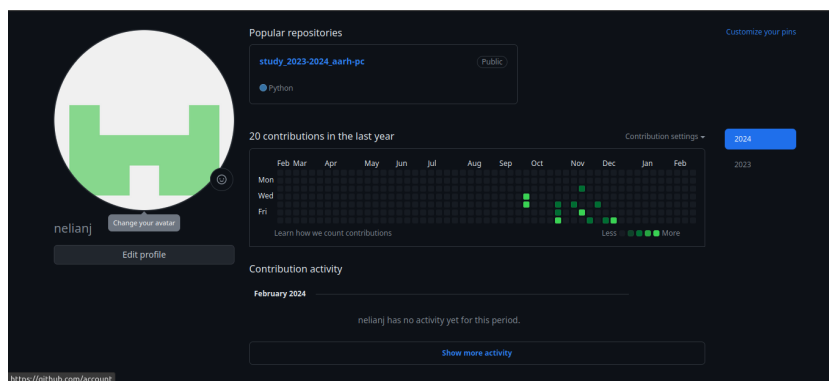


Рис. 3.11: аккаунт на github

## 6. Добавление PGP ключа в GitHub

Я вывожу список созданных ключей на терминал, ищу отпечаток ключа в результате запроса(последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа), он стоит после знака косой черты, копирую его в буфер обмена(рис.12)

```
nelianjovu@nelianjovu:~$ gpg --list-secret-keys --keyid-format LONG
gpg: checking the trustdb
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
[keyboard]
```

Рис. 3.12: вывод список ключей

Я копирую сам ключ gpg в буфер обмена(рис.13)

```
nelianjovu@nelianjovu:~$ gpg --armor --export DAB003B626756E7A | xclip -sel c
lip
nelianjovu@nelianjovu:~$
```

Рис. 3.13: копирование ключа

Я открываю настройка github, ищу среди них добавление gpg ключа(рис.14)

## GPG keys

New GPG key

There are no GPG keys associated with your account.

Learn how to [generate a GPG key and add it to your account](#).

Рис. 3.14: настройка github

Я нажимаю на “New GPG key” и вставляю в поле ключ из буфера обмена(рис.15)

### Add new GPG key

Title

Key

```
IQ75BA+Q9llf8iN8N  
BBxE8cqrOzz2HDyqeQDQFvRiyNS2QRt4fYuXY75opQvqNc+kILKi  
P7XH/ydMghLq  
qQlB1pvWSN2x3Eg3kipPS3wx08vTSDhOKQhzQYVeVxbLI19+27h  
smInplUBeq58M  
uD5XNAcGNFsp5Qs8PwY1cDYnKvcSI2Lix8oupRVPTDutFU9WZw  
JAWauIjQARAQAB  
tBxOam92dU5lbGhIDwxMDMyMjM5MDMzQHBmdXI+iQJRBBM  
BCAA7FiEE5ELnHY2M
```

Add GPG key

1t/s | BAT 72.00% 00:40 | 109.8 GiB | 1.66 | 1.3 GiB | 2.4 GiB | 2024-02-24 13:17:0

Рис. 3.15: добавление нового pgp ключа

Я добавила ключ gpg на github(рис.16)

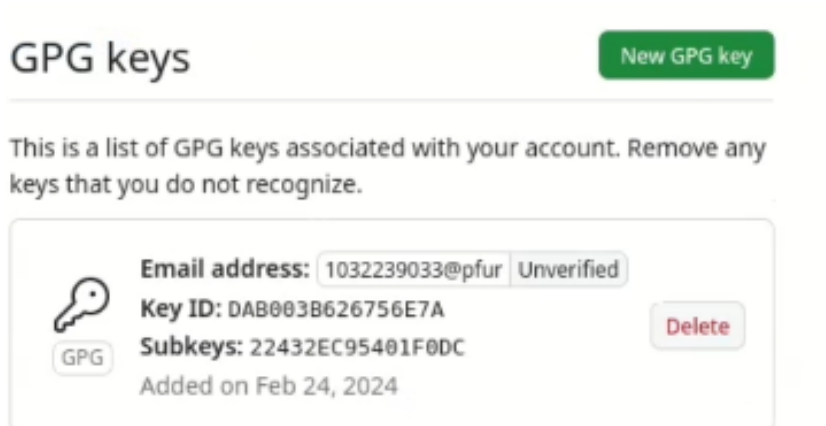


Рис. 3.16: добавление ключ gpg

## 7. Настроить подписи git.

Я настраиваю автоматические подписи коммитов git; используя адрес электронной почты, который я ввел ранее, я говорю git использовать его при создании подписей коммитов(рис.17)

```
nelianjovu@nelianjovu:~$ git config --global user.signingkey DAB003B626756E7A
nelianjovu@nelianjovu:~$ git config --global commit.gpgsign true
nelianjovu@nelianjovu:~$ git config --global gpg.program $(which gpg2)
nelianjovu@nelianjovu:~$
```

Рис. 3.17: настройка подписей

## 8. Настройка gh

Я начинаю авторизацию в gh, отвечаю на наводящие вопросы утилиты и в конце выбираю войти через браузер(рис.18)

```
nelianjovu@nelianjovu:~$ git config --global gh.hostname github.com
nelianjovu@nelianjovu:~$ gh auth login
? What account do you want to log into?
> GitHub.com
  GitHub Enterprise Server
```

Рис. 3.18: авторизация в gh

Я завершаю авторизацию на сайте(рис.19)

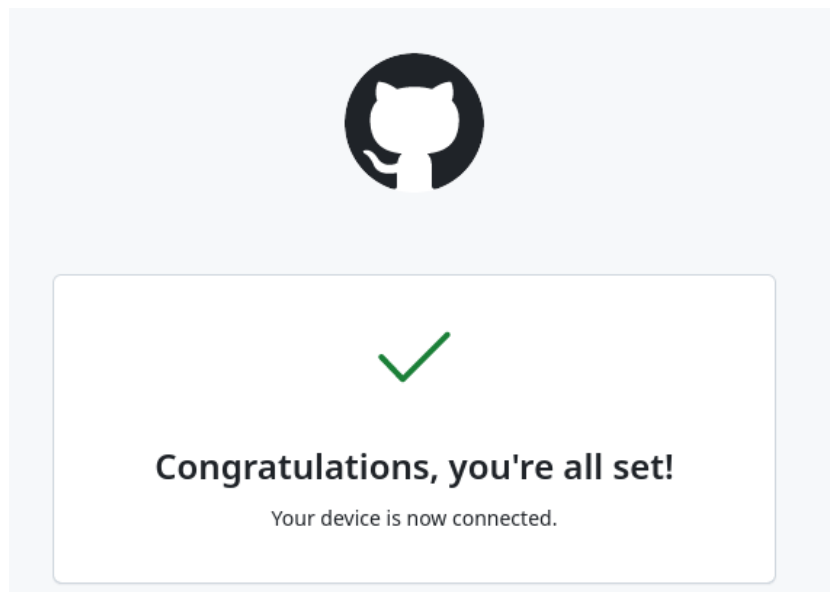


Рис. 3.19: завершение авторизации через браузер

Я вижу сообщение о завершении авторизации под именем nelianj(рис.20)

```
Press Enter to open github.com in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
! Authentication credentials saved in plain text
✓ Logged in as nelianj
```

Рис. 3.20: завершение авторизации

## 9. Создание репозитория курса на основе шаблона

Я создаю директорию с помощью команды `mkdir` и флага `-p`, который позволяет установить каталоги на всем указанном пути. Потом я перехожу в только созданную директорию “Операционные системы” с помощью команды `cd`. После этого я ввожу команду `'gh repo create study_2023-2024_os-intro --template=yamadharma/course-directory-student-template --public'`, чтобы создать репозиторий на основе шаблона репозитория(рис.21)

```

nelianjovu@nelianjovu:~$ mkdir -p ~/work/study/2023-2024/Операционные системы
nelianjovu@nelianjovu:~$ cd ~/work/study/2023-2024/Операционные системы
nelianjovu@nelianjovu:~/work/study/2023-2024/Операционные системы$ gh repo create study_2023-2024_os-intro --template=yamadharma/cours
e-structure-student-template --public

```

Рис. 3.21: создание репозитория

Я копирую репозиторий к себе в директорию, я указываю ссылку с проколотом https, а не ssh, потому что при авторизации в gh выбрала протокол https(рис.22)

```

nelianjovu@nelianjovu:~/work/study/2023-2024/Операционные системы$
git clone --recursive https://github.com/nelianj/study_2023-2024_
os-intro.git os-intro
Cloning into 'os-intro'...
remote: Enumerating objects: 32, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (31/31), done

```

Рис. 3.22: копирование репозитории

Я захожу в каталог курсов с помощью команды cd и удаляю файл package.json с помощью команды rm(рис.23)

```

nelianjovu@nelianjovu:~/work/study/2023-2024/Операционные системы$
cd os-intro
nelianjovu@nelianjovu:~/work/study/2023-2024/Операционные системы/
os-intro$ rm package.json
nelianjovu@nelianjovu:~/work/study/2023-2024/Операционные системы/

```

Рис. 3.23: удаление файлов

Я создаю необходимо каталог используя makefile(рис.24)

```

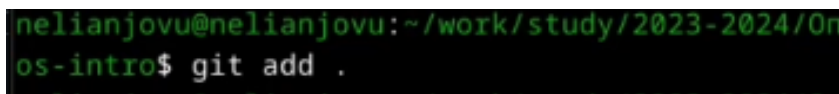
nelianjovu@nelianjovu:~/work/study/2023-2024/Операционные системы/
os-intro$ echo os-intro > COURSE
nelianjovu@nelianjovu:~/work/study/2023-2024/Операционные системы/
os-intro$ make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare       Generate directories structure
  submodule     Update submules

```

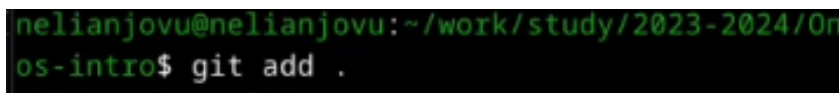
Рис. 3.24: создание каталога

Я добавляю все новые файлы для отправки на сервер(сохраняю добавление изменения) с помощью команды git add и комментирую их с помощью git commit



```
nelianjovu@nelianjovu:~/work/study/2023-2024/Операционные системы/
os-intro$ git add .
```

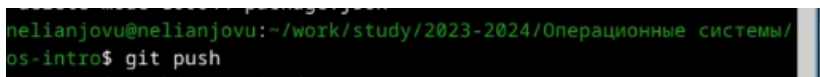
Рис. 3.25: рис.25:отправка файлов на сервере



```
nelianjovu@nelianjovu:~/work/study/2023-2024/Операционные системы/
os-intro$ git add .
```

Рис. 3.26: рис.26:коммент

Я отправляю файлы на сервер с помощью git push(рис.27)



```
nelianjovu@nelianjovu:~/work/study/2023-2024/Операционные системы/
os-intro$ git push
```

Рис. 3.27: отправка файлов на сервере

### *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 применяются для: Хранения полной 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 документы, включая историю их изменения и прочей служебной информацией.commit – отслеживание изменений, сохраняет разницу в изменениях.

История – 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 распределенных (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 сохраненные 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.

## 4 Выводы

Выполняя эту лабораторную работу, я изучил идеологию и применение инструментов контроля версий, а также освоил умение работать с git.

# Список литературы

1.Лабораторная работа № 2