

# **Отчет по лабораторной работе №2**

**Операционные системы**

Ирина Васильевна Панявкина

# 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>
3.1	Установка программного обеспечения . . . . .	7
3.2	Базовая настройка git . . . . .	7
3.3	Создание ключа SSH . . . . .	8
3.4	Создание ключа GPG . . . . .	10
3.5	Регистрация на GitHub . . . . .	11
3.6	Добавление ключа GPG на GitHub . . . . .	12
3.7	Настроить подписи Git . . . . .	15
3.8	Настройка gh . . . . .	15
3.9	Создание репозитория курса на основе шаблона . . . . .	16
<b>4</b>	<b>Выводы</b>	<b>19</b>
<b>5</b>	<b>Ответы на контрольные вопросы</b>	<b>20</b>
	<b>Список литературы</b>	<b>23</b>

# Список иллюстраций

3.1	Установка git и gh . . . . .	7
3.2	Задаю имя и email владельца репозитория . . . . .	8
3.3	Настройка utf-8 в выводе сообщений git . . . . .	8
3.4	Задаю имя начальной ветки . . . . .	8
3.5	Задаю параметры autocrlf и safecrlf . . . . .	8
3.6	Генерация ssh ключа по алгоритму rsa . . . . .	9
3.7	Генерация ssh ключа по алгоритму ed25519 . . . . .	9
3.8	Генерация ключа GPG . . . . .	10
3.9	Защита ключа GPG . . . . .	11
3.10	Аккаунт на GitHub . . . . .	12
3.11	Вывод списка ключей . . . . .	12
3.12	Вывод и копирование ключа в буфер обмена . . . . .	13
3.13	Настройки GitHub . . . . .	14
3.14	Добавление нового GPG ключа . . . . .	14
3.15	Добавленный GPG ключ . . . . .	15
3.16	Настройка подписей Git . . . . .	15
3.17	Авторизация в gh . . . . .	16
3.18	Завершение авторизации . . . . .	16
3.19	Создание репозитория . . . . .	17
3.20	Перемещение между директориями . . . . .	17
3.21	Добавление и комментирование файлов для отправки на сервер .	17
3.22	Отправка файлов на сервер . . . . .	18

## **Список таблиц**

# 1 Цель работы

Цель данной лабораторной работы - изучение идеологии и применения средств контроля версий, освоение умения по работе с git.

## 2 Задание

1. Создать базовую конфигурацию для работы с git
2. Создать ключ SSH
3. Создать ключ GPG
4. Настроить подписи Git
5. Зарегистрироваться на GitHub
6. Создать локальный каталог для выполнения заданий по предмету.

## 3 Выполнение лабораторной работы

### 3.1 Установка программного обеспечения

Устанавливаю необходимое программное обеспечение git и gh через терминал с помощью команд: `dnf install git` и `dnf install gh` (рис.[3.1]).

```
[irinaranyavkina@irinaranyavkina ~]$ sudo dnf -y install git
[sudo] пароль для irinaranyavkina:
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет "git-2.48.1-1.fc41.x86_64" уже установлен.

Нечего делать.
[irinaranyavkina@irinaranyavkina ~]$ sudo dnf -y install gh
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет                                Арх.      Версия
Установка:
gh                                x86_64    2.65.0-1.fc41

Сводка транзакции:
Установка:      1 пакета

Общий размер входящих пакетов составляет 10 MiB. Необходимо загрузить 10 MiB.
После этой операции будут использоваться дополнительные 43 MiB (установка 43 MiB, удаление 0 B).
[1/1] gh-0:2.65.0-1.fc41.x86_64
-----
[1/1] Total
Выполнение транзакции
[1/3] Проверить файлы пакета
[2/3] Подготовить транзакцию
[3/3] Установка gh-0:2.65.0-1.fc41.x86_64
Завершено!
```

Рис. 3.1: Установка git и gh

### 3.2 Базовая настройка git

Задаю в качестве имени и email владельца репозитория свои имя, фамилию и электронную почту (рис.[3.2]).

```
[irinapanyavkina@irinapanyavkina ~]$ git config --global user.name "Irina Panyavkina"
[irinapanyavkina@irinapanyavkina ~]$ git config --global user.email "renrrurenrru@gmail.com"
[irinapanyavkina@irinapanyavkina ~]$
```

Рис. 3.2: Задаю имя и email владельца репозитория

Настраиваю utf-8 в выводе сообщений git для их корректного отображения (рис.[3.3]).

```
[irinapanyavkina@irinapanyavkina ~]$ git config --global core.quotepath false
[irinapanyavkina@irinapanyavkina ~]$
```

Рис. 3.3: Настройка utf-8 в выводе сообщений git

Начальной ветке задаю имя master (рис.[3.4]).

```
[irinapanyavkina@irinapanyavkina ~]$ git config --global init.defaultBranch master
[irinapanyavkina@irinapanyavkina ~]$
```

Рис. 3.4: Задаю имя начальной ветки

Задаю параметры autocrlf и safecrlf для корректного отображения конца строки (рис.[3.5]).

```
[irinapanyavkina@irinapanyavkina ~]$ git config --global core.autocrlf input
[irinapanyavkina@irinapanyavkina ~]$ git config --global core.safecrlf warn
[irinapanyavkina@irinapanyavkina ~]$
```

Рис. 3.5: Задаю параметры autocrlf и safecrlf

### 3.3 Создание ключа SSH

Создаю ключ ssh размером 4096 бит по алгоритму rsa (рис.[3.6]).



```

[irinapanyavkina@irinapanyavkina ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/irinapanyavkina/.ssh/id_rsa):
Created directory '/home/irinapanyavkina/.ssh'.
Enter passphrase for "/home/irinapanyavkina/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/irinapanyavkina/.ssh/id_rsa
Your public key has been saved in /home/irinapanyavkina/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:zoKwokUxLQxc0M6Pd167UjaENgSc19Te/4zjMMga2bs irinapanyavkina@irinapanyavkina
The key's randomart image is:
+---[RSA 4096]-----+
|  ooB.+.. |
|  * * o . |
|  o + + . . |
|  + = . . . |
|  o = o $ . |
|  * + B o .. |
|  + + = = o + o + |
|  o o o + .ooo |
|o      .... Eo.. |
+-----[SHA256]-----+
[irinapanyavkina@irinapanyavkina ~]$

```

Рис. 3.6: Генерация ssh ключа по алгоритму rsa

Создаю ключ ssh по алгоритму ed25519 (рис.[3.7]).

```

[irinapanyavkina@irinapanyavkina ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/irinapanyavkina/.ssh/id_ed25519):
Enter passphrase for "/home/irinapanyavkina/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/irinapanyavkina/.ssh/id_ed25519
Your public key has been saved in /home/irinapanyavkina/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:0boUT6/kZxb4MzrQ9A0q/UQP1sd1xm4IKYNNgDuEvQ irinapanyavkina@irinapanyavkina
The key's randomart image is:
+--[ED25519 256]--+
| o... ..o |
|o . + o . . o |
|.. E. * . ....o |
|.. ...*.. o |
|.. .+S* = . |
|. o==o+ o |
| ooo+.. |
|. +..o |
|. +* o |
+-----[SHA256]-----+
[irinapanyavkina@irinapanyavkina ~]$

```

Рис. 3.7: Генерация ssh ключа по алгоритму ed25519

## 3.4 Создание ключа GPG

Генерирую ключ GPG, затем выбираю тип ключа RSA и RSA, задаю максимальную длину ключа 4096, оставляю неограниченный срок действия ключа. Затем даю ответы на вопросы программы о личной информации (рис.[3.8]).

```
[irinapanyavkina@irinapanyavkina ~]$ gpg --full-generate-key
gpg (GnuPG) 2.4.5; Copyright (C) 2024 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/irinapanyavkina/.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: IrinaPanyavkina
Адрес электронной почты: renrrurenrru@gmail.com
```

Рис. 3.8: Генерация ключа GPG

Ввожу фразу-пароль для защиты нового ключа (рис.[3.9]).

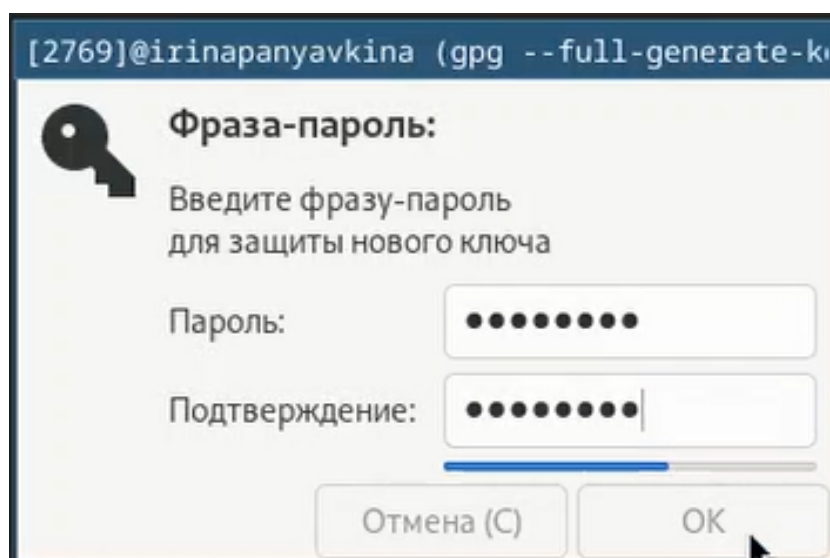


Рис. 3.9: Защита ключа GPG

## 3.5 Регистрация на GitHub

Аккаунт на GitHub я уже создавала, поэтому основные данные уже заполнены и проведена его настройка, тогда просто вхожу в свой аккаунт (рис.[3.10]).

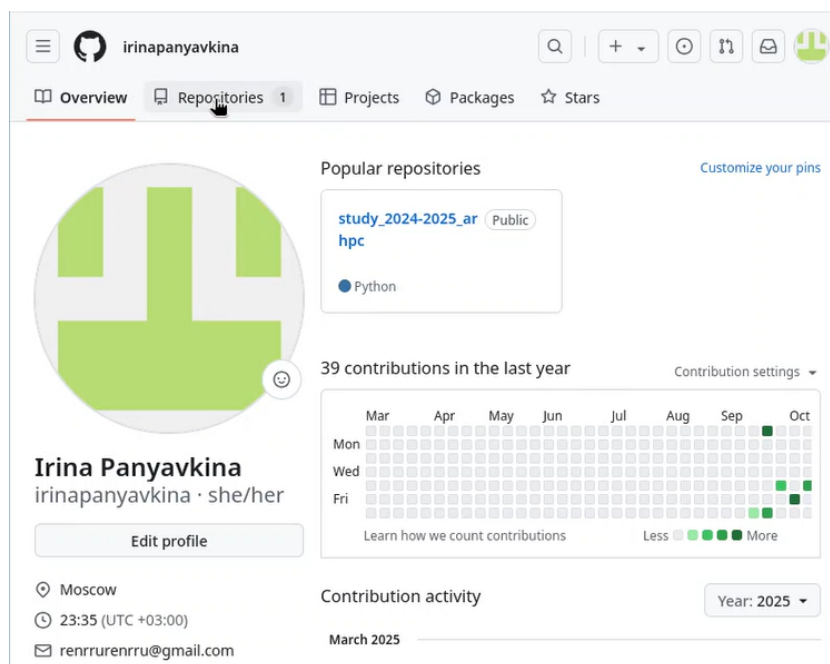


Рис. 3.10: Аккаунт на GitHub

## 3.6 Добавление ключа GPG на GitHub

Вывожу список созданных ключей в терминал, ищу в результате запроса отпечаток ключа (последовательность байтов для идентификации более длинного, по сравнению с самим отпечатком, ключа), он находится после знака слеш, копирую его в буфер обмена.(рис.[3.11]).

```
[irinapanyavkina@irinapanyavkina ~]$ 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/E805258AA4D557F8 2025-03-03 [SC]
      DA08F64B2C7AA788BD300CE2E805258AA4D557F8
uid   [ абсолютно ] IrinaPanyavkina <renrrurenrru@gmail.com>
ssb   rsa4096/6CA8625CCFE2CDC9 2025-03-03 [E]
```

Рис. 3.11: Вывод списка ключей

К сожалению, скопировать ключ с помощью утилиты xclip, введя следующую

команду `gpg --armor --export` отпечаток ключа | `xclip -sel clip`, мне не удалось, как бы я не пыталась (происходила вставка предыдущего скопированного элемента), поэтому пришлось сделать это вручную с помощью команды для просмотра буфера обмена `xclip -sel clip -o`. (рис.[3.12]).

```
[irinapanyavkina@irinapanyavkina ~]$ gpg --armor --export E805258AA40557F8 | xclip -sel clip
[irinapanyavkina@irinapanyavkina ~]$ xclip -sel clip -o
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGfGEaYBEACbN8rch0w9/Mi3egdhx9OU6gmzwMNCOWlpqCDFzhqr15K63f
QVxk49r9x610ds+UErDbLBDDtqUmg1AzLpgj1B490YKUuTb5Lre7xH1b56wh1011
E/a69SmcGGz8mT6XF7D1FA+YwvYT1jBwmfB+00R8KfXqitfoU6JSou5fpaVRjYpN
7b81vYocslkKH29A72+3egqsCucpRvMcqMiW150XrncGs0a22cAgAju8oXFJ13vq
CUa07/LtyTgbtnRVdAPX1tMnMAj0gnZ/uzqx28w9wcZDeSDk12V2tLkak2e2GLc8
n6k/cg1D384T0ZnUmuXmwOQBZAdLvhpbBpBL2PQI96yOUUxn6DND1qwkfRvQeIy7
DeAb09jCTLw6qyc1rHYHepAgNnV2f7MnUL5f5myIwWZ29aw6XdYrvLXoX/1eQM/
kdGdJB4XTck+4jorHyasM12pZNy0GeK1ped0j91/VTLrU8oFIp9ov1Iz1xQCXLeR
wJol50413234YTT5wWUCJoeFwmOae/0f0+Ie5e43eD8v07DccEOXCH31vt1YrK
/8UqkgGSW240vNIImoyw0AH5yyAJExVIt/nLQH03cQ5j1ygFVIRcNv0NVCgdiud
tQ4tjfh/zSe5Zonq1XHoPpzmPyVuiK1gUR61CPQWYVyhFb2ZxEkpvUkBUwARAQAB
tChJcm1uYVBhbn1hdmtpbmEgPHJ1bnJydXJ1bnJydUBnbWlFpbC5jb20+IQJRBMBM
CAA7fIEE2tj2Syx6p4i9MAzi6AU1iqTVV/gFAmfGEaYCGwMFCwkIBwICIGIGFQoJ
CAsCBByCAwECHgcCF4AACGkQ6AU1iqTVV/h4QBAAh1+H88Nkcwbn1vS0IrpLKNB
9/j8gCI6oXst8c8qNPEqLMI+H6MgcQekE1BhAXepe8Q1CTABjUEnePqpCJZ716ja
xL/HixStEs3RcJqdKey+/BsPZ8Wg0RreRhHiC+FTS1b5JLI2GRDOKMHfXxgS0Nf
cRGXit1f1J63B29Tm5iQaQTXg/QbN+3GtuImb8CZ71noQQLfQCVB83L9AI4+2x1g
81d30P0W13Xra2N/ej374YRZB2k11484qYJu1v1z2K3e0J+P0i6eG00/es51bYk
QdRrGu03aeM1XgaBqRMS5SHNDahKI6bzK6QJavBB5eTJdRuwXmWwBpv5u59wvfrvy
gFAkLH9PRaEN57Fe/C9r9NMv0rKCHNqTsg+BMKS9QxpUfC1HRGfU7rbsxc19pqjc
30oEEU2+5D5+YknOpOL6Nk37HMYtkqex3aGu5oJW+UbZAPASnF1uONEjPc/Od0AP
K4KX4gwVPUJ49ALweVxveJjdEplpdqe3Nd13V1RMobbPfpQDSRgiycq8jcpq56JM
/IFyBg2qbvv4MhMCHMedUZYo19WdJ0aXDSL45Xa0P7ku97dRfjT2kA87hmVAmA3p
daviXddag94w7aMGMS0+VCq912yk2Uj0BvSeXqgh8XY01LXEu59srbVJR7wRaTf
h3V0AKP66F2dIEpCWAq5Ag0EZ8YRpgEQAOuvdc0XP1HXqME5a4wXodSPx0b3VL18
Uz/HXz0DhcBt7CNUrQIpZ71qg8xwoQqgToDYTS/0dn/ABYNSpLeQKL1KZJcAQzOc
sYV2IrgaE4y2D9go1RS/aUaVccC94gHit2q1A3CwzMBx86EQ0dPnY1urYsYATpEQ
NcvS512FcXjW+zc17G/+vjrJSpeghK/uuap5tdqs0tn+ubop4oecvq66pJsDQbo
zpTmnBmy4efbNui7GIUNu9YAQ8D90DvWFXSGy0adnuTBkTeND0D8Y3ugxM81Mcj
P1Bv/yRGgplXPK9wMRZaQrE6U0jKTFc6KdRYYRiE6c1h6f8ryuR1cj/kea9E2ddb
stPw3+AhU5+4v90YTnIbUIMkKdvDSMJ09KxZLenhDYAMgp7s3BN4Ry7c48PzpR+
o4dpW4qZGTekQvShXA0g3MuahQ2MSfuxxhZq16dVxb8q41oiQd7JRD/u++VJ5FT0
c6PwfyKI20jhoPIRuRZA4S05AS/4c1+Nb+8FhN67zfOC1T/XrRMykou+/tdPLS39
C9ah7q1UglNeykOyhUzFN3+p6ZooFXsQRv1V6Z5LAGWjuN2w6L/faJpp5wy1yaH
k1ve2skrgxulMtbmxzUU7A7a1RsqurVb9UqmBNmDAh9TZg/UKmp34pkUjVD0T26j
GhrAhXUf1JE1ABEBAAGJAjYEGAETACAWIQTa2PZLLHqniL0wDOLo8S5KpNVX+AUC
```

Рис. 3.12: Вывод и копирование ключа в буфер обмена

Открываю настройки GitHub, ищу среди них добавление GPG ключа (рис.[3.13]).

## 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. 3.13: Настройки GitHub

Нажимаю на “New GPG key” и вставляю в поле ключ из буфера обмена (рис.[3.14]).

### Add new GPG key

Title

Key

```
-----BEGIN PGP PUBLIC KEY BLOCK-----  
MItF/qTyfMs68fePNJqxrS/  
eOhvmPkjUaUXi8YLRRJcJ+MrorjippfCGFLhHN3HD  
kO7SlpE3DIbXUSzuUN5TA8pjdDI0PrKTTsRRlJtfdSW8LpxpeTpW  
K/6I5A3K8h4K  
qkl48PIrCbnN5fia/  
mwcKzbDf7joqbOhbMPQhakoBYNJUPEeQC+iMS4vnxLMbhTI  
etbGNbNsBQ==  
=s/x5
```

Add GPG key

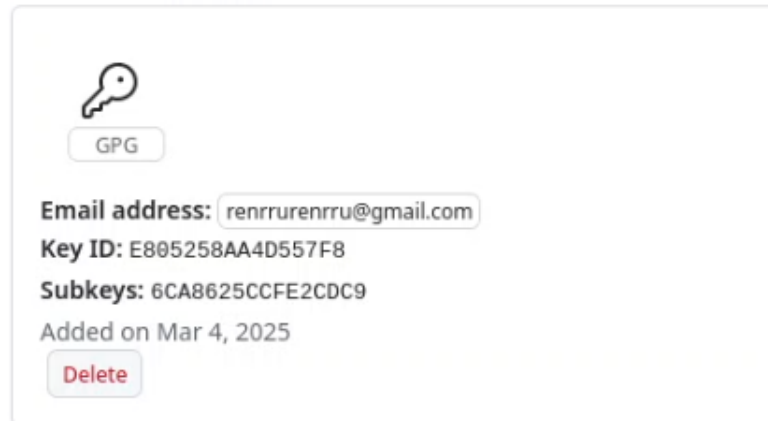
Рис. 3.14: Добавление нового GPG ключа

Ключ GPG успешно добавлен на GitHub (рис.[3.15]).

## GPG keys

[New GPG key](#)

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



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

Рис. 3.15: Добавленный GPG ключ

## 3.7 Настроить подписи Git

Настраиваю автоматические подписи коммитов git: используя введенный ранее email, указываю git использовать его при создании подписей коммитов (рис.[3.16]).

```
[irinapanyavkina@irinapanyavkina ~]$ git config --global user.signingkey E805258AA4D557F8
[irinapanyavkina@irinapanyavkina ~]$ git config --global commit.gpgsign true
[irinapanyavkina@irinapanyavkina ~]$ git config --global gpg.program $(which gpg2)
```

Рис. 3.16: Настройка подписей Git

## 3.8 Настройка gh

Начинаю авторизацию в gh, отвечаю на наводящие вопросы от утилиты, в конце выбираю авторизоваться через браузер (рис.[3.17]).

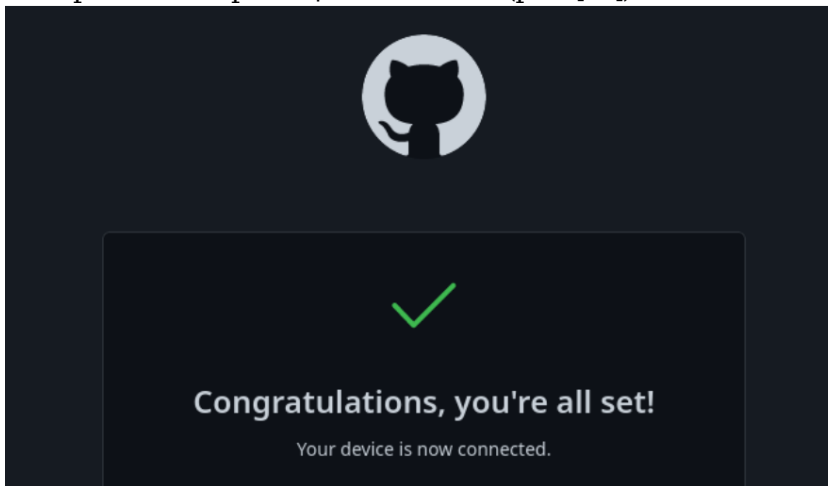
```

[irinapanyavkina@irinapanyavkina ~]$ gh auth login
? Where do you use GitHub? 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. 3.17: Авторизация в gh

Завершаю авторизацию на сайте (рис.[??]).



Вижу сообщение о завершении авторизации под именем irinapanyavkina (рис.[3.18]).

```

✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as irinapanyavkina

```

Рис. 3.18: Завершение авторизации

## 3.9 Создание репозитория курса на основе шаблона

Сначала я создаю директорию с помощью утилиты `mkdir` и флага `-p`, который позволяет установить каталоги на всем указанном пути. Затем с помощью утилиты `cd` перемещаюсь в созданную директорию “Операционные системы”.



После этого в терминале ввожу команду `gh repo create study_2024-2025_os-intro-template yamadharm/course --directory-template --public`, для того чтобы создать репозиторий на основе шаблона репозитория. Далее клонирую репозиторий к себе в директорию, указывая ссылку с протоколом `https`, а не `ssh`, потому что во время авторизации в `gh`, я выбрала протокол `https` (рис.[3.19]).

```
irinapanyavkina@irinapanyavkina:~/Операционные системы$ git clone --recursive https://github.com/irinapanyavkina/study_2024-2025_os-intro.git os-intro
Клонирование в «os-intro»...
remote: Enumerating objects: 36, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 36 (delta 1), reused 21 (delta 0), pack-reused 0 (from 0)
Получение объектов: 100% (36/36), 19.38 KiB | 188.00 KiB/c, готово.
Определение изменений: 100% (1/1), готово.
Подготовка «template/presentation» (https://github.com/yamadharm/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
```

Рис. 3.19: Создание репозитория

Перехожу в каталог курса с помощью утилиты `cd`, проверяю содержание каталога с помощью утилиты `ls` (рис.[3.20]).

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

Рис. 3.20: Перемещение между директориями

Удаляю лишние файлы с помощью утилиты `rm`, затем создаю необходимые каталоги используя `makefile` (рис.[3.21]).

```
irinapanyavkina@irinapanyavkina:~/os-intro$ rm package.json
irinapanyavkina@irinapanyavkina:~/os-intro$ echo os-intro > COURSE
irinapanyavkina@irinapanyavkina:~/os-intro$ make
```

Добавляю все новые файлы для отправки на сервер (сохраняю добавленные изменения) с помощью команды `git add` и комментирую их с помощью команды `git commit` (рис.[3.21]).

```
irinapanyavkina@irinapanyavkina:~/os-intro$ git add .
irinapanyavkina@irinapanyavkina:~/os-intro$ git commit -am 'feat(main): make course structure'
[master 7e75e16] feat(main): make course structure
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
```

Рис. 3.21: Добавление и комментирование файлов для отправки на сервер

Отправляю файлы на сервер с помощью `git push` (рис.[3.22]).

```
[irinapanyavkina@irinapanyavkina os-intro]$ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 951 байт | 951.00 КиБ/с, готово.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/irinapanyavkina/study_2024-2025_os-intro.git
a4c6cbd..7e75e16 master -> master
```

Рис. 3.22: Отправка файлов на сервер

## **4 Выводы**

При выполнении лабораторной работы я изучила идеологию и применение средств контроля версий, освоила умение по работе с git.

## 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 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 (децентрализо-

ванных) 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 вет-

ки: `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.

## **Список литературы**

::: {#esystem.rudn.ru/mod/page/view.php?id=1224311}