

# **Лабораторная работа №2**

**Операционные системы**

Кучмар София Игоревна

# 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>Контрольные вопросы</b>	<b>11</b>
<b>5</b>	<b>Выводы</b>	<b>14</b>

## Список иллюстраций

3.1	Установим Git . . . . .	7
3.2	Установим gh . . . . .	7
3.3	Базовая настройка git . . . . .	7
3.4	Сгенерируем ключи по алгоритму RSA . . . . .	8
3.5	Сгенерируем ключи по алгоритму ed25519 . . . . .	8
3.6	Сгенерируем PGP ключ . . . . .	8
3.7	Добавим его в наш github . . . . .	9
3.8	Настройка автоматических подписей коммитов Git . . . . .	9
3.9	Авторизация . . . . .	9
3.10	Создадим папку . . . . .	9
3.11	Создадим репозиторий . . . . .	10
3.12	Склонируем репозиторий . . . . .	10
3.13	Настройка каталога курса . . . . .	10
3.14	git add и git commit . . . . .	10
3.15	git push . . . . .	10

## **Список таблиц**

# 1 Цель работы

В рамках данной лабораторной работы мы выполним установку необходимого программного обеспечения, а именно установим Git и GitHub CLI (gh).

## 2 Задание

В процессе выполнения лабораторной работы мы произведем базовую настройку Git, сгенерируем SSH и PGP ключи для безопасной аутентификации. После этого мы настроим GitHub, добавив сгенерированный PGP ключ в нашу учетную запись и настроив автоматические подписи коммитов Git. Затем мы произведем настройку GitHub CLI (gh) для удобной работы с репозиториями. В завершение, мы создадим шаблон для рабочего пространства, создадим репозиторий курса на основе этого шаблона и настроим каталог курса для организации файлов. В итоге мы получим полностью настроенное окружение для контроля версий и совместной работы с кодом на платформе GitHub.

### 3 Выполнение лабораторной работы

Проведём установку программного обеспечения. Установим Git (рис. 3.1).

```
root@sikuchmar:~# dnf install git
Updating and loading repositories:
Fedora 41 - x86_64 - Updates                                100% | 23.0 KiB/s | 15.2 KiB | 00m01s
Fedora 41 - x86_64 - Updates                                100% | 645.7 KiB/s | 2.8 MiB | 00m04s
Repositories loaded.
Пакет "git-2.47.0-1.fc41.x86_64" уже установлен.
Nothing to do.
```

Рис. 3.1: Установим Git

Установим gh (рис. 3.2).

```
root@sikuchmar:~# dnf install gh
Updating and loading repositories:
Repositories loaded.
Package Arch Version Repository Size
Installing: x86_64 2.65.0-1.fc41 updates 42.6 MiB
gh
Transaction Summary:
```

Рис. 3.2: Установим gh

Зададим имя и email владельца репозитория. Настроим utf-8 в выводе сообщений git. Зададим имя начальной ветки (будем называть её master). Установим параметр autocrlf и safecrlf (рис. 3.3).

```
root@sikuchmar:~# git config --global user.name "sikuchmar"
root@sikuchmar:~# git config --global user.email "2450sonia@gmail.com"
root@sikuchmar:~# git config --global core.quotepath false
root@sikuchmar:~# git config --global init.defaultBranch master
root@sikuchmar:~# git config --global core.autocrlf input
root@sikuchmar:~# git config --global core.safecrlf warn
```

Рис. 3.3: Базовая настройка git

Сгенерируем ключи по алгоритму RSA с ключом размером 4096 бит (рис. 3.4).

```

root@sikuchmar:~# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): /root/.ssh/id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:zaNdbSJ6JoW0CocqhTa9MVuF3Konv2bt3zSZE8y0RZg root@sikuchmar
The key's randomart image is:
+---[RSA 4096]-----+
|
| . o o
| o o.E .
| .. .o. =+ .
|.o.+oo. S *=o o
|...Bo . =+++o
|. . = .o +. +B
|. +o . +o o
| ooo... .
+---[SHA256]-----+

```

Рис. 3.4: Сгенерируем ключи по алгоритму RSA

Сгенерируем ключи по алгоритму ed25519 (рис. 3.5).

```

root@sikuchmar:~# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519): /root/.ssh/id_ed25519
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519
Your public key has been saved in /root/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:0IDrW2Vy94jcGnJdkvC00pFiEzyTIIQXkvTe6uSInY0 root@sikuchmar
The key's randomart image is:
+--[ED25519 256]--+
|.O+. o..
|. +o o +o.
|o o o.+. + .
|. . o =.* = .
| o o XS0 =
| o + B = .
| o o + o
|..=+. .
|..Eo.
+---[SHA256]-----+

```

Рис. 3.5: Сгенерируем ключи по алгоритму ed25519

Сгенерируем PGP ключ (рис. 3.6).

```

root@sikuchmar:~# 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 '/root/.gnupg'
Выберите тип ключа:

```

Рис. 3.6: Сгенерируем PGP ключ



Добавим его в наш github (рис. 3.7).

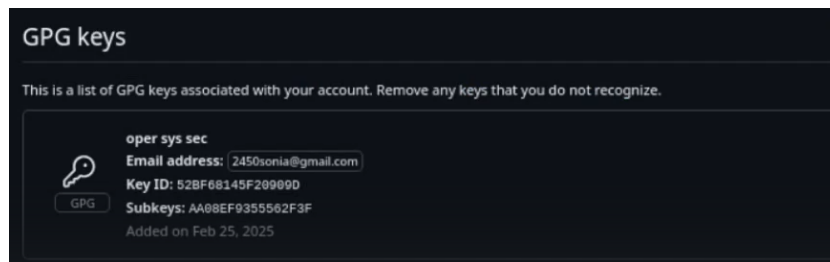


Рис. 3.7: Добавим его в наш github

Настройка автоматических подписей коммитов Git. Используя введенный email, укажем Git применять его при подписи коммитов (рис. 3.8).

```
root@sikuchmar:~# git config --global user.signingkey 575055539F077896
root@sikuchmar:~# git config --global commit.gpgsign true
root@sikuchmar:~# git config --global gpg.program $(which gpg2)
```

Рис. 3.8: Настройка автоматических подписей коммитов Git

Для начала авторизуемся (рис. 3.9).

```
sikuchmar@sikuchmar:~$ 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

! First copy your one-time code: 0EBC-E572
Press Enter to open https://github.com/login/device in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as sikuchmar
```

Рис. 3.9: Авторизация

Необходимо создать шаблон рабочего пространства. Для этого создадим папку (рис. 3.10).

```
sikuchmar@sikuchmar:~$ mkdir -p ~/work/study/2024-2025/"Операционные системы"
```

Рис. 3.10: Создадим папку

Перейдём туда и создадим репозиторий (рис. 3.11).

```
sikuchmar@sikuchmar:~/work/study/2024-2025/Операционные системы$ gh repo create study_2024-2025_os-intro --template=yamadham
a/course-directory-student-template --public
✓ Created repository sikuchmar/study_2024-2025_os-intro on GitHub
https://github.com/sikuchmar/study_2024-2025_os-intro
```

Рис. 3.11: Создадим репозиторий

После склонируем репозиторий (рис. 3.12).

```
sikuchmar@sikuchmar:~/work/study/2024-2025/Операционные системы$ git clone --recursive https://github.com/sikuchmar/study_2024-2025_os-intro.git
Клонирование в «study_2024-2025_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 КиБ | 381.00 КиБ/с, готово.
Обработка изменений: 100% (1/1), готово.
```

Рис. 3.12: Склонируем репозиторий

Перейдем в каталог курса. Удалим лишние файлы и создадим необходимые каталоги (рис. 3.13).

```
sikuchmar@sikuchmar:~/work/study/2024-2025/Операционные системы$ cd study_2024-2025_os-intro
sikuchmar@sikuchmar:~/work/study/2024-2025/Операционные системы/study_2024-2025_os-intro$ rm package.json
sikuchmar@sikuchmar:~/work/study/2024-2025/Операционные системы/study_2024-2025_os-intro$ echo os-intro > COURSE
sikuchmar@sikuchmar:~/work/study/2024-2025/Операционные системы/study_2024-2025_os-intro$ make
Usage:
  make <target>

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

sikuchmar@sikuchmar:~/work/study/2024-2025/Операционные системы/study_2024-2025_os-intro$ make prepare
```

Рис. 3.13: Настройка каталога курса

Отправим файлы на сервер (рис. 3.14) и (рис. 3.15).

```
sikuchmar@sikuchmar:~/work/study/2024-2025/Операционные системы/study_2024-2025_os-intro$ git add .
sikuchmar@sikuchmar:~/work/study/2024-2025/Операционные системы/study_2024-2025_os-intro$ git commit -am 'feat(main): make course structure'
```

Рис. 3.14: git add и git commit

```
sikuchmar@sikuchmar:~/work/study/2024-2025/Операционные системы/study_2024-2025_os-intro$ git push
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 341.66 КиБ | 6.70 МБ/с, готово.
Total 38 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To https://github.com/sikuchmar/study_2024-2025_os-intro.git
 e577977..e6aa956 master -> master
```

Рис. 3.15: git push

## 4 Контрольные вопросы

Что такое системы контроля версий (VCS) и для решения каких задач они предназначены?:

- VCS: Инструменты для отслеживания изменений файлов во времени.
- Задачи: Отслеживание, возврат, сравнение версий, совместная работа, ветвление, резервное копирование.

Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

- Хранилище: Все версии проекта + история.
- Commit: Снимок состояния проекта в момент времени.
- История: Последовательность коммитов.
- Рабочая копия: Локальная копия для работы.

Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

- Централизованные: Единое хранилище (SVN, CVS).
- Децентрализованные: Полная копия хранилища у каждого (Git, Mercurial).

Опишите действия с VCS при единоличной работе с хранилищем.

1. Создать/Клонировать.
2. Изменить.

3. Проверить.
4. Добавить в индекс.
5. Закоммитить.
6. (Push для удаленного).

Опишите порядок работы с общим хранилищем VCS.

1. Клонировать.
2. Создать ветку.
3. Изменить.
4. Проверить.
5. Добавить в индекс.
6. Закоммитить.
7. Push.
8. Pull Request.
9. Ревью.
10. Слияние.
11. Удалить ветку.
12. Pull.

Каковы основные задачи, решаемые инструментальным средством git?

- Версионный контроль, совместная работа, управление ветками, отслеживание изменений, разрешение конфликтов, возврат к версиям.

Назовите и дайте краткую характеристику командам git.

- git init: Создать репо локально.
- git clone: Скачать репо удаленно.
- git add: Добавить в индекс сейчас.
- git commit: Зафиксировать изменения локально.
- git push: Отправить на сервер вверх.

- `git pull`: Получить с сервера вниз.
- `git status`: Состояние сейчас показывает.
- `git log`: История коммитов отображается.
- `git branch`: Ветки создать, переключить.
- `git checkout`: Переключиться между ветками.
- `git merge`: Объединить ветки вместе.
- `git diff`: Различия файлы покажет.

Приведите примеры использования при работе с локальным и удалённым репозиториями

- (Локально): `git init`, `git add file.txt`, `git commit -m "Initial"`, `git log`.
- (Удаленно): `git clone URL`, `git remote add origin URL`, `git push origin main`, `git pull origin main`.

Что такое и зачем могут быть нужны ветви (branches)?

- Независимые линии разработки.
- Нужны для: параллельной работы, экспериментов, разделения задач.

Как и зачем можно игнорировать некоторые файлы при `commit`?

- Исключение из коммитов временных, конфиденциальных и больших файлов.
- Сокращение размера репозитория и предотвращение конфликтов.

## 5 Выводы

В ходе выполнения лабораторной работы было успешно создано и настроено окружение для эффективной работы с системой контроля версий Git и платформой GitHub. В частности, были выполнены следующие задачи:

- Установлены необходимые инструменты: Git и GitHub CLI (gh).
- Выполнена базовая настройка Git и сгенерированы ключи SSH и PGP для безопасной аутентификации.
- Настроен аккаунт GitHub с добавлением PGP-ключа и включены автоматические подписи коммитов.
- Настроен GitHub CLI для удобного взаимодействия с репозиториями.
- Создан шаблон рабочего пространства, создан на его основе репозиторий курса и настроена структура каталогов для организации файлов.

В результате получено полностью готовое к использованию окружение для контроля версий и совместной разработки кода на платформе GitHub.