

# **Лабораторная работа №2**

**Операционные системы**

Кирилюк Светлана

# Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	16

# Список иллюстраций

3.1	1	. . . . .	7
3.2	2	. . . . .	7
3.3	3	. . . . .	8
3.4	4	. . . . .	8
3.5	5	. . . . .	8
3.6	6	. . . . .	9
3.7	7	. . . . .	9
3.8	8	. . . . .	9
3.9	9	. . . . .	10
3.10	10	. . . . .	10
3.11	11	. . . . .	10
3.12	12	. . . . .	11
3.13	13	. . . . .	11
3.14	14	. . . . .	12
3.15	15	. . . . .	12
3.16	16	. . . . .	13
3.17	17	. . . . .	13

## Список таблиц

# 1 Цель работы

Изучить идеологию и применение средств контроля версий. Освоить умения по работе с git.

## 2 Задание

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

### 3 Выполнение лабораторной работы

В первую очередь я установила git (рис. 3.1) и gh (рис. 3.2).

```
[sakirilyuk@sakirilyuk ~]$ sudo -i
[sudo] пароль для sakirilyuk:
[root@sakirilyuk ~]# dnf install git
```

Рис. 3.1: 1

```
[root@sakirilyuk ~]# dnf install gh
Последняя проверка окончания срока действия метаданных: 2:51:
29 назад, Ср 15 фев 2023 00:04:52.
Зависимости разрешены.
=====
Пакет  Архитектура Версия                Репозиторий  Размер
=====
Установка:
gh      x86_64        2.22.1-1.fc37      updates     8.3 М
```

Рис. 3.2: 2

Затем я задала имя и email владельца репозитория (рис. 3.3) и настроила utf-8 в выводе сообщений git (рис. 3.4).

```
[root@sakirilyuk ~]# git config --global user.name "sakirilyuk"
[root@sakirilyuk ~]# git config --global user.email "lana.kirilyuk@internet.ru"
```

Рис. 3.3: 3

```
[root@sakirilyuk ~]# git config --global core.quotepath false
[root@sakirilyuk ~]#
```

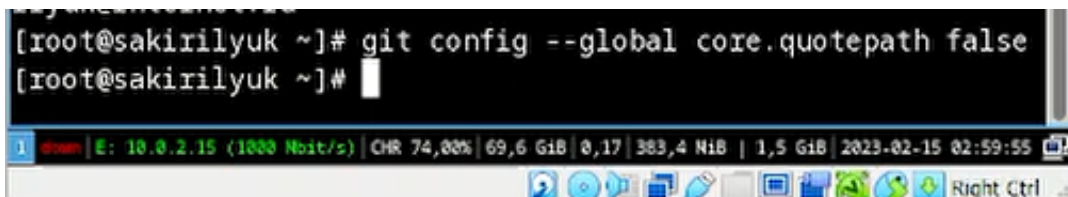


Рис. 3.4: 4

Я сгенерировала ргр ключ (рис. 3.5), вывела список ключей и скопировала отпечаток приватного ключа (рис. 3.6), чтобы скопировать сгенерированный PGP ключ в буфер обмена (рис. 3.7). Затем я вставила полученный ключ в строку на GitHub (рис. 3.8).

```
[root@sakirilyuk ~]# gpg --full-generate-key
```

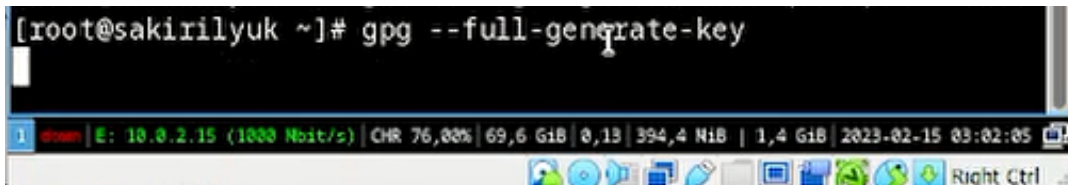


Рис. 3.5: 5



```
[root@sakirilyuk ~]# 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
/root/.gnupg/pubring.kbx
-----
sec rsa4096/AB00AF2BC7FDEE26 2023-02-15 [SC]
```

Рис. 3.6: 6

```
[root@sakirilyuk ~]# gpg --armor --export AB00AF2BC7FDEE26
1
```

Рис. 3.7: 7

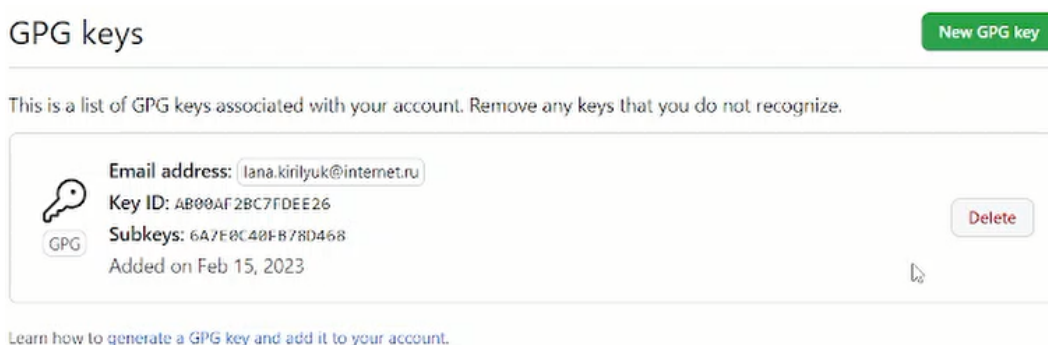


Рис. 3.8: 8

Также я настроила автоматические подписи коммитов (рис. 3.9), задала параметры autocrlf и safecrlf и сгенерировала ключи по двум алгоритмам (рис. 3.10), (рис. 3.11).

```
[root@sakirilyuk ~]# git config --global user.signingkey AB00AF2BC7FDEE26
[root@sakirilyuk ~]# git config --global commit.gpgsign true
[root@sakirilyuk ~]# git config --global gpg.program $(which gpg2)
[root@sakirilyuk ~]#
```

Рис. 3.9: 9

```
al init.defaultBranch maste
al core.autocrlf input
al core.safecrlf warn
[root@sakirilyuk ~]# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
```

Рис. 3.10: 10

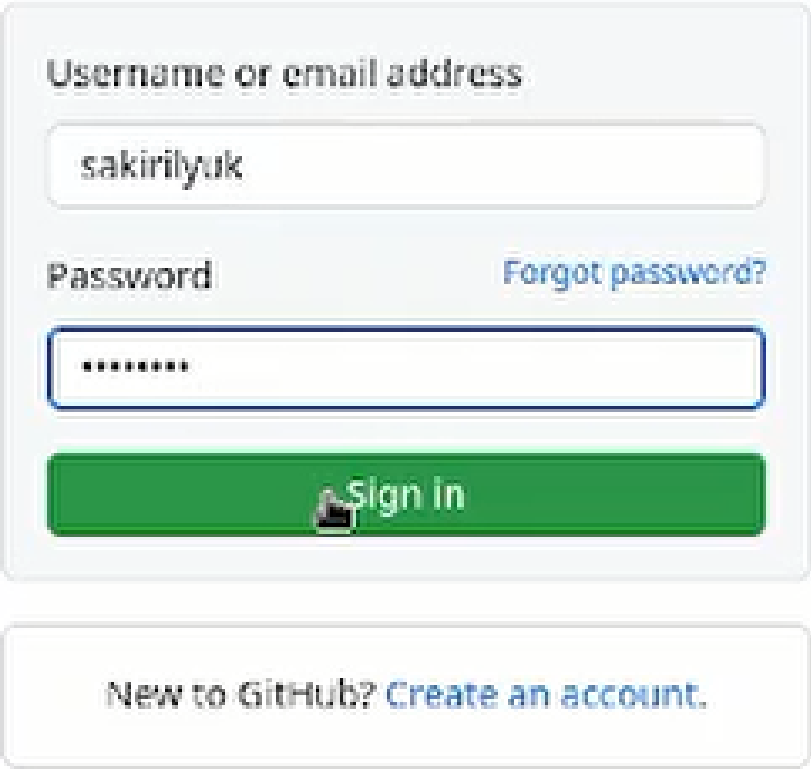
```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519
Your public key has been saved in /root/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:el14YTq3LX1+j08cWldEvIH5wNndfqBTQZT/TieJNCo root@sakirilyuk.sakirilyuk.net
The key's randomart image is:
+--[ED25519 256]--+
|      .o0++      |
|      *+++      |
|      o0o++      |
|      +=. ++     |
|      S +o++ .+  |
|      .E..=.+o = |
|      . . . o o0+ |
|      . . . .++*  |
|      . o*       |
+-----[SHA256]-----+
[root@sakirilyuk ~]#
```

Рис. 3.11: 11

Далее я авторизовалась (рис. 3.12), (рис. 3.13). Создала шаблон репозитория (рис. 3.14), (рис. 3.15).

```
[sakirilyuk@sakirilyuk ~]$ gh auth login
? What account do you want to log into? [Use arrows to move,
  type to filter]
> GitHub.com
  GitHub Enterprise Server
```

Рис. 3.12: 12



The image shows the GitHub login interface. It features a light gray rounded rectangle containing the login fields. At the top, the text "Username or email address" is displayed. Below it is a text input field containing the username "sakirilyuk". Underneath the username field is the "Password" label, followed by a password input field with masked characters. To the right of the password field is a blue link that says "Forgot password?". Below the password field is a green "Sign in" button with a white GitHub Octocat icon. At the bottom of the login box is a link that says "New to GitHub? Create an account."

Рис. 3.13: 13

Create a new repository from course-directory-student-template

The new repository will start with the same files and folders as [yamadharm/course-directory-student-template](#).

Owner \* sakirilyuk / Repository name \* study\_2022-2023\_os-intro ✓

Great repository names: study\_2022-2023\_os-intro is available. Piration? How about potential-enigma?

Description (optional)

☒ Public  
Anyone on the internet can see this repository. You choose who can commit.

☐ Private  
You choose who can see and commit to this repository.

☐ Include all branches  
Copy all branches from yamadharm/course-directory-student-template and not just master.

📌 You are creating a public repository in your personal account.

Create repository from template

Рис. 3.14: 14

```
[sakirilyuk@sakirilyuk Операционные системы]$ git clone --recursive git@github.com:sakirilyuk/study_2022-2023_os-intro.git os-intro
Клонирование в «os-intro»...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 27 (delta 1), reused 11 (delta 0), pack-reused 0
Получение объектов: 100% (27/27), 16.93 КБ | 1.06 МБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharm/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
```

Рис. 3.15: 15

Перейдя в каталог курса я удалила лишние файлы и создала необходимые каталоги (рис. 3.16), затем отправила файлы на сервер (рис. 3.17).

```

study/2022-2023/"Операционные системы"/os-intro
package.json
os-intro > COURSE

add .
commit -am 'feat(main): make course structure'

```

Рис. 3.16: 16

```

[sakirilyuk@sakirilyuk os-intro]$ git push
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (38/38), готово.
Запись объектов: 100% (38/38), 342.40 КиБ | 2.00 МиБ/с, готово.
Всего 38 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:sakirilyuk/study_2022-2023_os-intro.git
  531120a..9e3c5b6  master -> master
[sakirilyuk@sakirilyuk os-intro]$

```

Рис. 3.17: 17

Ответы на контрольные вопросы: 1) Система контроля версий (VCS) — это место хранения кода. Она нужна для разработки продуктов (для хранения кода, синхронизации работы нескольких человек, создания релизов).

2) 2.1 Хранилище (репозиторий) - центральное место, хранящее не только файлы, но и историю. Доступ к репозиторию осуществляется через сеть, выступая в роли сервера и инструмента контроля версий, выступающего в роли клиента. 2.2 Commit - это команда Git для записи индексированных изменений в репозиторий. 2.3 Рабочая копия – это снимок хранилища, личное рабочее место, где разработчики могут выполнять свою работу, оставаясь изолированными от остальной части команды.

3) 3.1 Централизованные VCS Одно основное хранилище всего проекта; Каждый пользователь копирует себе необходимые ему файлы из этого репозитория.

тория, изменяет и, затем, добавляет свои изменения обратно. 3.2 Децентрализованные VCS У каждого пользователя свой вариант (возможно не один) репозитория; Присутствует возможность добавлять и забирать изменения из любого репозитория.

6)Git — это система управления версиями. У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.

7) 7.1 Создание основного дерева репозитория: `git init` 7.2 Получение обновлений (изменений) текущего дерева из центрального репозитория: `git pull` 7.3 Отправка всех произведённых изменений локального дерева в центральный репозиторий: `git push` 7.4 Просмотр списка изменённых файлов в текущей директории: `git status` 7.5 Просмотр текущих изменений: `git diff` 7.6 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`). 7.7 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`), отправка изменений конкретной ветки в центральный репозиторий (`git push origin имя_ветки`), слияние ветки с текущим деревом (`git merge --no-ff имя_ветки`) 7.8 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) Ветка (англ. branch) — это последовательность коммитов, в которой ведётся параллельная разработка какого-либо функционала Основная ветка – master Ветки в GIT. Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций.

10)Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты

## 4 Выводы

В ходе работы я изучила идеологию и применение средств контроля версий. Освоила умения по работе с git.