

# **Отчёт по лабораторной работе №2**

**дисциплина: Операционные системы**

Сычев Егор Олегович

# Содержание

|   |                                |    |
|---|--------------------------------|----|
| 1 | Цель работы                    | 5  |
| 2 | Выполнение лабораторной работы | 6  |
| 3 | Контрольные вопросы            | 12 |
| 4 | Вывод                          | 16 |

## Список иллюстраций

|      |   |    |
|------|---|----|
| 2.1  | ssh github . . . . .                    | 6  |
| 2.2  | gpg –full-generate-key . . . . .        | 6  |
| 2.3  | Копирование . . . . .                   | 7  |
| 2.4  | Копирование . . . . .                   | 7  |
| 2.5  | github gpg key . . . . .                | 7  |
| 2.6  | Настройка автомат. подписей . . . . .   | 7  |
| 2.7  | gh auth login . . . . .                 | 8  |
| 2.8  | github авторизация . . . . .            | 8  |
| 2.9  | Создание репозитория . . . . .          | 9  |
| 2.10 | Клонирование . . . . .                  | 10 |
| 2.11 | Настройка каталога . . . . .            | 10 |
| 2.12 | git add, git commit, git push . . . . . | 10 |
| 2.13 | git add, git commit, git push . . . . . | 10 |
| 2.14 | Получившийся репозиторий . . . . .      | 11 |

## **Список таблиц**

# 1 Цель работы

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

## 2 Выполнение лабораторной работы

1. Установка git,gh. Базовая настройка git. Создание ssh ключа. (Выполнено в предыдущих лабораторных работах)

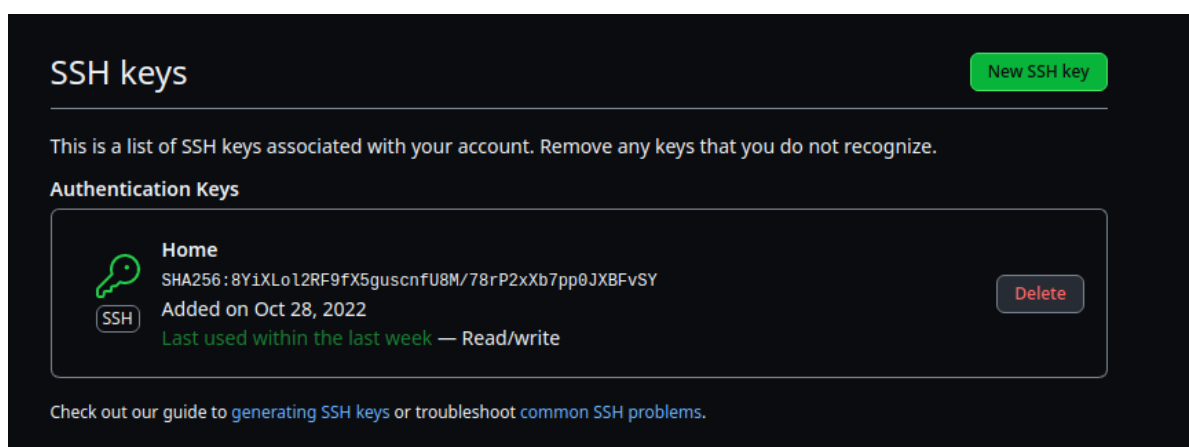


Рис. 2.1: ssh github

2. Создание gpg ключа.

```
[eosihchev@fedora report]$ gpg --full-generate-key
gpg (GnuPG) 2.3.7; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

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
```

Рис. 2.2: gpg --full-generate-key

### 3. Добавление gpg ключа в github.

```
[eosihchev@fedora ~]$ 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,
lu
/home/eosihchev/.gnupg/pubring.kbx
-----
sec   rsa4096/3E70D02667284130 2023-02-17 [SC]
      B64DCA7FD64B5B24666252AE3E70D02667284130
uid           [ абсолютно ] Egor <1132226469@pfur.ru>
ssb   rsa4096/1026378F92854477 2023-02-17 [E]

[eosihchev@fedora ~]$
```

Рис. 2.3: Копирование

```
[eosihchev@fedora ~]$ gpg --armor --export 3E70D02667284130 | xclip -sel clip
[eosihchev@fedora ~]$
```

Рис. 2.4: Копирование

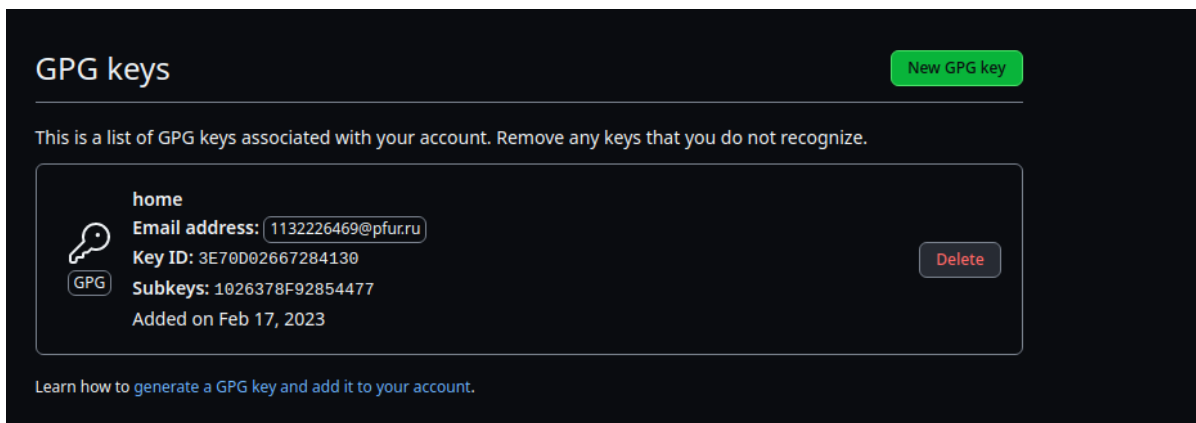


Рис. 2.5: github gpg key

### 4. Настройка автоматических подписей коммитов git.

```
[eosihchev@fedora ~]$ git config --global user.signingkey 3E70D02667284130
[eosihchev@fedora ~]$ git config --global commit.gpgsign true
[eosihchev@fedora ~]$ git config --global gpg.program $(which gpg2)
[eosihchev@fedora ~]$
```

Рис. 2.6: Настройка автомат. подписей

## 5. Настройка gh.

```
[eosihchev@fedora ~]$ gh auth login
bash: gh: команда не найдена...
Установить пакет «gh», предоставляющий команду «gh»? [N/y] y

* Ожидание в очереди...
* Загрузка списка пакетов....
Следующие пакеты должны быть установлены:
gh-2.22.1-1.fc36.x86_64      GitHub's official command line tool
Продолжить с этими изменениями? [N/y] y

* Ожидание в очереди...
* Ожидание аутентификации...
* Ожидание в очереди...
* Загрузка пакетов...
* Запрос данных...
* Проверка изменений...
* Установка пакетов... █
```

Рис. 2.7: gh auth login

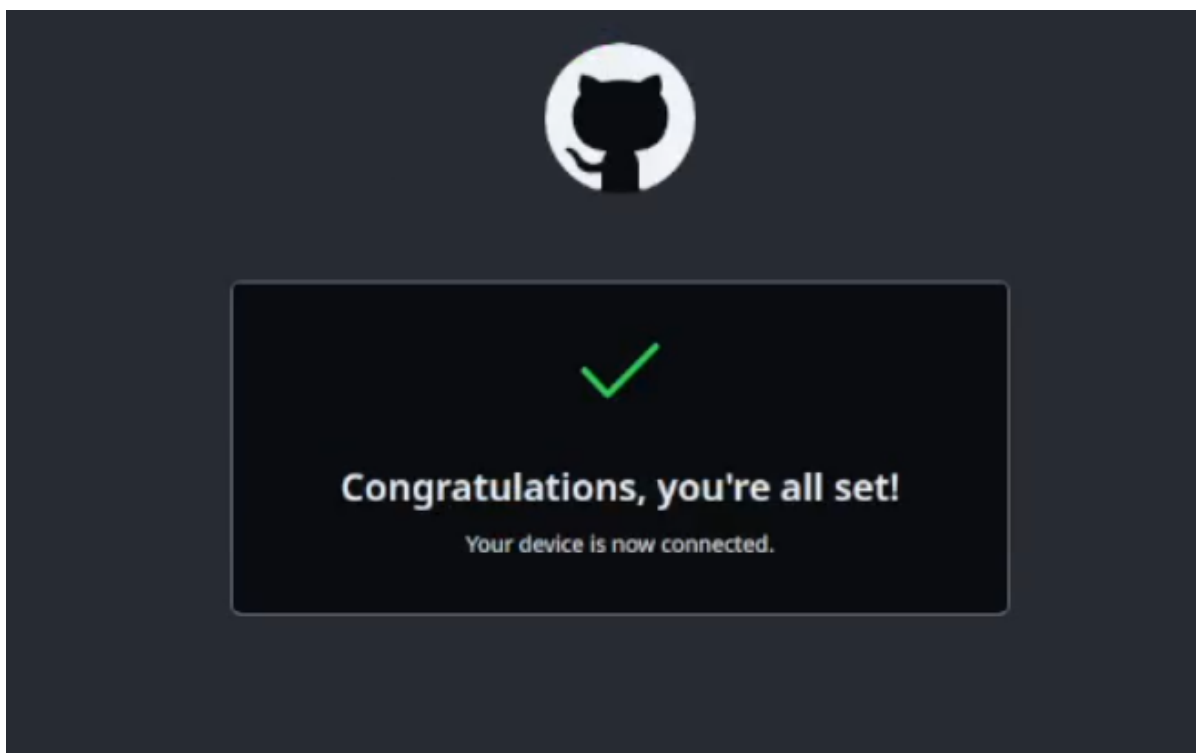


Рис. 2.8: github авторизация





6. Создание репозитория курса на основе шаблона и настройка каталога курса.

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


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


Owner <sup>\*</sup> Repository name <sup>\*</sup>

 eosihchev / study\_2022-2023\_os-intro 


Great repository names are short and memorable. Need inspiration? How about [fluffy-umbrella?](#)

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 yamadharma/course-directory-student-template and not just master.

 You are creating a public repository in your personal account.

[Create repository from template](#)

Рис. 2.9: Создание репозитория

```
eosihchev@fedora:~/work/study/2022-2023/Операционные системы
[eosihchev@fedora ~]$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
[eosihchev@fedora ~]$ cd ~/work/study/2022-2023/"Операционные системы"
[eosihchev@fedora Операционные системы]$ gh repo create study_2022-2023_os-intro --template=yamadharma/course-directory-student-template --public
GraphQL: Could not clone: Name already exists on this account (cloneTemplateRepository)
[eosihchev@fedora Операционные системы]$ git clone --recursive git@github.com:eosihchev/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 КиБ | 16.93 МиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/eosihchev/work/study/2022-2023/Операционные системы/os-intro/template/presentation»...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 82 (delta 28), reused 77 (delta 23), pack-reused 0
Получение объектов: 100% (82/82), 92.90 КиБ | 980.00 КиБ/с, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/home/eosihchev/work/study/2022-2023/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (101/101), done.
remote: Compressing objects: 100% (70/70), done.
remote: Total 101 (delta 40), reused 88 (delta 27), pack-reused 0
Получение объектов: 100% (101/101), 327.25 КиБ | 1.62 МиБ/с, готово.
Определение изменений: 100% (40/40), готово.
Submodule path 'template/presentation': checked out 'b1be3800ee91f5809264cb755d316174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b82e3aef11a33b1e3b2'
[eosihchev@fedora Операционные системы]$
```

Рис. 2.10: Клонирование

```
[eosihchev@fedora Операционные системы]$ cd ~/work/study/2022-2023/"Операционные системы"/os-intro
[eosihchev@fedora os-intro]$ rm package.json
[eosihchev@fedora os-intro]$ echo os-intro > COURSE
[eosihchev@fedora os-intro]$ make
```

Рис. 2.11: Настройка каталога

```
[eosihchev@fedora os-intro]$ git add .
[eosihchev@fedora os-intro]$ git commit -am 'feat(main): make course structure'
```

Рис. 2.12: git add, git commit, git push

```
[eosihchev@fedora os-intro]$ git push
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 343.04 КиБ | 2.54 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:eosihchev/study_2022-2023_os-intro.git
  74e5e32..8b1c2bf master -> master
[eosihchev@fedora os-intro]$
```

Рис. 2.13: git add, git commit, git push

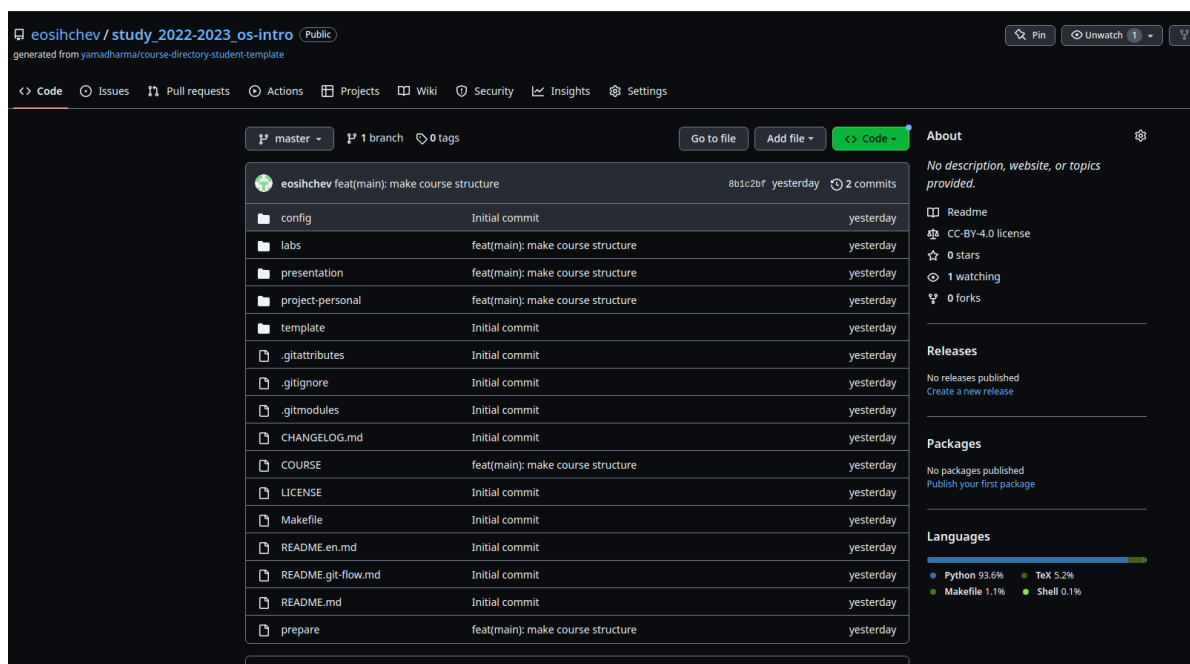


Рис. 2.14: Получившийся репозиторий

### 3 Контрольные вопросы

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

Контроль версий, также известный как управление исходным кодом, — это практика отслеживания изменений программного кода и управления ими. Системы контроля версий — это программные инструменты, помогающие командам разработчиков управлять изменениями в исходном коде с течением времени.

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

Хранилище (repository, сокр. repo), или репозиторий, — место хранения всех версий и служебной информации. • Коммит (commit; редко переводится как «слепок») — 1) синоним версии; 2) создание новой версии («сделать коммит», «закоммитить»). • Рабочая копия (working copy или working tree) — текущее состояние файлов проекта, основанное на версии из хранилища (обычно на последней)

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

Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Пример - Wikipedia. В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. Пример — Bitcoin.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером

4. Опишите действия с VCS при единоличной работе с хранилищем.

Создадим локальный репозиторий. Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория: `git config --global user.name "Имя Фамилия"` `git config --global user.email "work@mail"` и настроив utf-8 в выводе сообщений `git: git config --global quotepath false` Для инициализации локального репозитория, расположенного, например, в каталоге `~/tutorial`, необходимо ввести в командной строке: `cd mkdir tutorial cd tutorial git init`

5. Опишите порядок работы с общим хранилищем VCS.

Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый): `ssh-keygen -C "Имя Фамилия work@mail"` Ключи сохраняться в каталоге `~/.ssh/`. Скопировав из локальной консоли ключ в буфер обмена `cat ~/.ssh/id_rsa.pub | xclip -sel clip` вставляем ключ в появившееся на сайте поле.

6. Каковы основные задачи, решаемые инструментальным средством git?

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. Назовите и дайте краткую характеристику командам git.

Наиболее часто используемые команды git: – создание основного дерева репозитория: `git init`–получение обновлений (изменений)текущего дерева из центрального репозитория: `git pull`–отправка всех произведённых изменений ло-

кального дерева в центральный репозиторий: `git push`–просмотр списка изменённых файлов втекущей директории: `git status`–просмотртекущих изменения: `git diff`–сохранениетекущих 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 (приэтомфайл и/иликаталог 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 натекущей: `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 стекущим 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. Приведите примеры использования при работе с локальным и удалённым репозиториями.

Использования `git` при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий): `git add hello.txt git commit -am'Новый файл`

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

Ветки очень облегчают работу. Они решить такие проблемы как: нужно постоянно создавать архивы с рабочим кодом сложно “переключаться” между архива-

ми сложно перетаскивать изменения между архивами легко что-то напутать или потерять

#### 10. Как и зачем можно игнорировать некоторые файлы при commit?

Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов. Для этого сначала нужно получить список имеющихся шаблонов: `curl -L -s https://www.gitignore.io/api/list` Затем скачать шаблон, например, для С и С++ `curl -L -s https://www.gitignore.io/api/c` » `.gitignore` `curl -L -s https://www.gitignore.io/api/c++` » `.gitignore`

## 4 Вывод

Я изучил идеологию и применение средств контроля версий. И освоил умения по работе с git.