

# **Отчет по лабораторной работе №2**

**Операционные системы**

Абронина Алиса Кирилловна

# 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 ключей ssh . . . . .	8
3.4	Создание ключа pgr . . . . .	9
3.5	Настройка github . . . . .	9
3.6	Добавление ключа pgr в github . . . . .	9
3.7	Настройка автоматических подписей коммитов git . . . . .	11
3.8	Настройка gh . . . . .	11
3.9	Создание репозитория курса на основе шаблона . . . . .	12
<b>4</b>	<b>Ответы на контрольные вопросы.</b>	<b>15</b>
<b>5</b>	<b>Выводы</b>	<b>18</b>

## Список иллюстраций

3.1	Установка git и gh . . . . .	7
3.2	Задаю имя и email . . . . .	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 . . . . .	8
3.7	Генерация ssh ключа по алгоритму ed25519 . . . . .	9
3.8	Генерация pgr ключа . . . . .	9
3.9	Вывол списка ключей . . . . .	10
3.10	Добавление ключа pgr в github . . . . .	10
3.11	Добавление ключа pgr в github . . . . .	11
3.12	Настройка автоматических подписей коммитов git . . . . .	11
3.13	Авторизация . . . . .	11
3.14	Завершение авторизации . . . . .	12
3.15	Завершение авторизации . . . . .	12
3.16	Создание репозитория . . . . .	13
3.17	Перемещение между директориями . . . . .	13
3.18	Удаление файлов и создание каталогов . . . . .	13
3.19	Отправка файлов на сервер . . . . .	14
3.20	Отправка файлов на сервер . . . . .	14

## Список таблиц

# 1 Цель работы

Изучить идеологию и применение средств контроля версий и 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

### 3.1 Установка программного обеспечения

Устанавливаю необходимое программное обеспечение git и gh через терминал (рис. 3.1).

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

Нечего делать.
[root@vbox ~]# dnf install gh
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет "gh-2.65.0-1.fc41.x86_64" уже установлен.

Нечего делать.
[root@vbox ~]#
```

Рис. 3.1: Установка git и gh

### 3.2 Базовая настройка git

Задаю в качестве имени и email владельца репозитория свое имя, фамилию и почту (рис. 3.2).

```
[root@vbox ~]# git config --global user.name "Abronina Alisa"
[root@vbox ~]# git config --global user.email "alisaabronina@gmail.com"
```

Рис. 3.2: Задаю имя и email

Настроиваю utf-8 в выводе сообщений git (рис. 3.3).

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

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

Задаю имя начальной ветки (рис. 3.4).

```
[root@vbox ~]# git config --global init.defaultBranch master
```

Рис. 3.4: Задаю имя начальной ветки

Задаю параметры autocrlf и safecrlf (рис. 3.5).

```
[root@vbox ~]# git config --global core.autocrlf input
[root@vbox ~]# git config --global core.safecrlf warn
```

Рис. 3.5: Задаю параметры autocrlf и safecrlf

### 3.3 Создание ключей ssh

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

```
[root@vbox ~]# 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
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase for "/root/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
```

Рис. 3.6: Генерация ssh ключа по алгоритму rsa

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



```
[root@vbox ~]# 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
/root/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y
```

Рис. 3.7: Генерация ssh ключа по алгоритму ed25519

### 3.4 Создание ключа ргр

Генерирую ключ, затем выбираю тип ключа RSA and RSA, задаю максимальную длину ключа: 4096, оставляю неограниченный срок действия ключа. Далее отвечаю на вопросы программы о личной информации (рис. fig. 3.8).

```
[root@vbox ~]# 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.

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. 3.8: Генерация ргр ключа

### 3.5 Настройка github

У меня уже есть созданный аккаунт на github, соответственно, данные аккаунта я так же не заполняю.

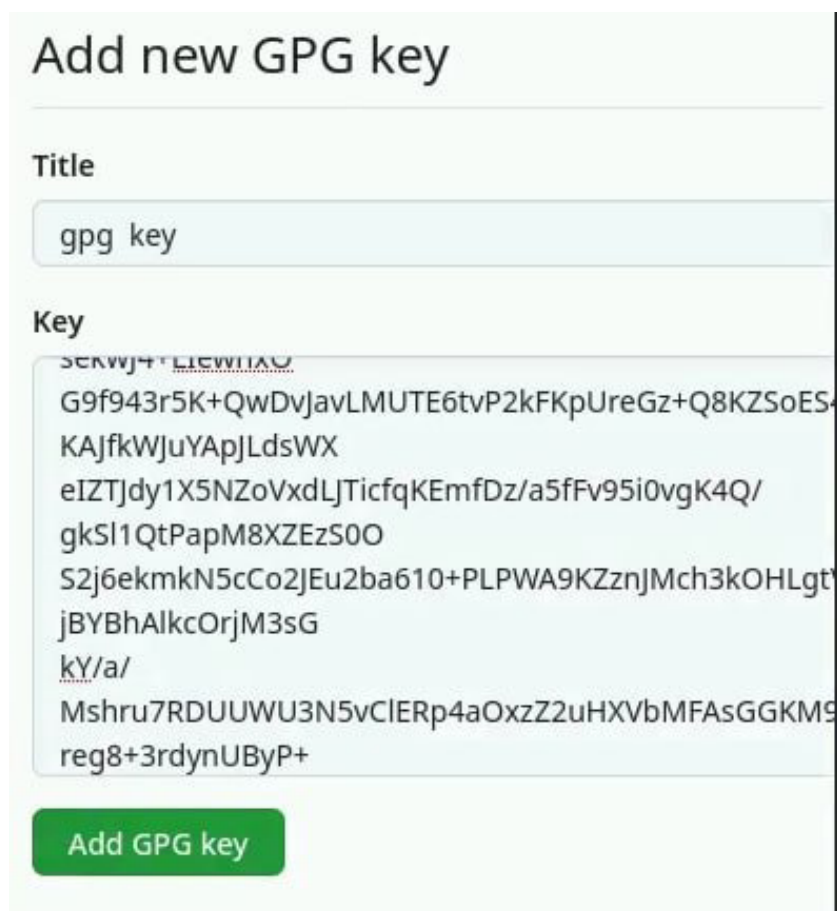
### 3.6 Добавление ключа ргр в github

Выводим список ключей и копируем отпечаток приватного ключа(рис. 3.9).

```
[akabronina@vbox ~]$ 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/D5E0EBA5EFCF398B 2025-03-02 [SC]
      6EE2CFD665F876573A196837D5E0EBA5EFCF398B
uid           [ абсолютно ] AbroninaAlisa <alisaabronina@gmail.com>
ssb   rsa4096/0F37AEC3FDD87276 2025-03-02 [E]
```

Рис. 3.9: Вывол списка ключей

Ввожу в терминале команду, с помощью которой копирую сам ключ GPG в буфер обмена. Открываю настройки GitHub, ищу среди них добавление GPG ключа. Нажимаю на “New GPG key” и вставляю в поле ключ из буфера обмена (рис. fig. 3.10) .



**Add new GPG key**

**Title**

gpg key

**Key**

5ERWJ4+LEWHTAO  
G9f943r5K+QwDvJavLMUTE6tvP2kFKpUreGz+Q8KZSoES-  
KAJfkWJuYApJLdsWX  
eIZTJdy1X5NZoVxdLJTicfqKEmfDz/a5fFv95i0vgK4Q/  
gkSl1QtPapM8XZEzS0O  
S2j6ekmkN5cCo2JEU2ba610+PLPWA9KZznJMch3kOHLgt  
jBYBhAlkcOrjM3sG  
kY/a/  
Mshru7RDUUWU3N5vCIERp4aOxzZ2uHXVbMFAsGGKM9  
reg8+3rdynUBYP+

**Add GPG key**

Рис. 3.10: Добавление ключа pgr в github

Я добавила ключ pgr в github (рис. 3.11).

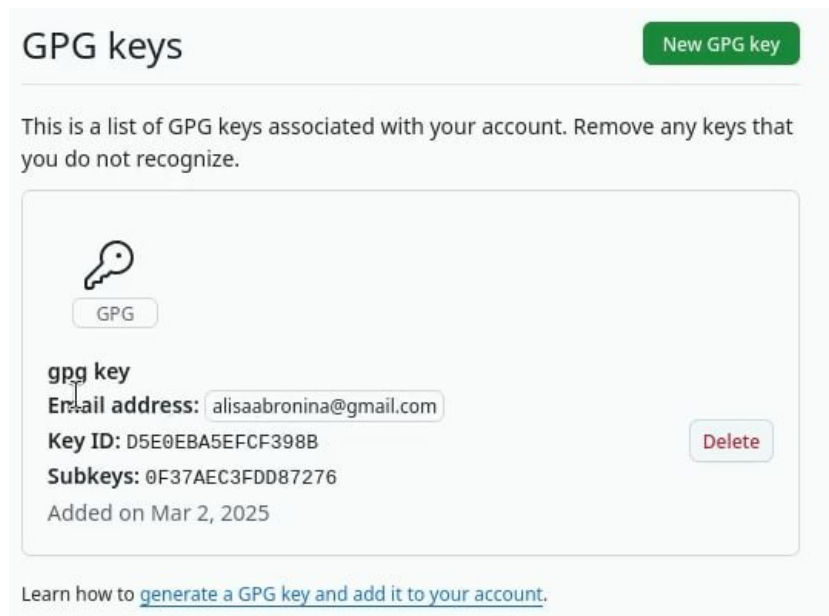


Рис. 3.11: Добавление ключа pgr в github

## 3.7 Настройка автоматических подписей коммитов git

Настроиваю автоматические подписи коммитов git (рис. 3.12).

```
[akabronina@vbox ~]$ git config --global user.signingkey D5E0EBA5EFCF398B
[akabronina@vbox ~]$ git config --global commit.gpgsign true
[akabronina@vbox ~]$ git config gpg.program $(which gpg2)
fatal: not in a git directory
[akabronina@vbox ~]$ git config --global gpg.program $(which gpg2)
```

Рис. 3.12: Настройка автоматических подписей коммитов git

## 3.8 Настройка gh

Начинаю авторизоваться в gh , отвечаю на вопросы (рис. 3.13).

```
[akabronina@vbox ~]$ gh auth login
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host?
PS
? Authenticate Git with your GitHub credentials? (Y/n)
```

Рис. 3.13: Авторизация

Завершаю авторизацию на сайте (рис. 3.14).

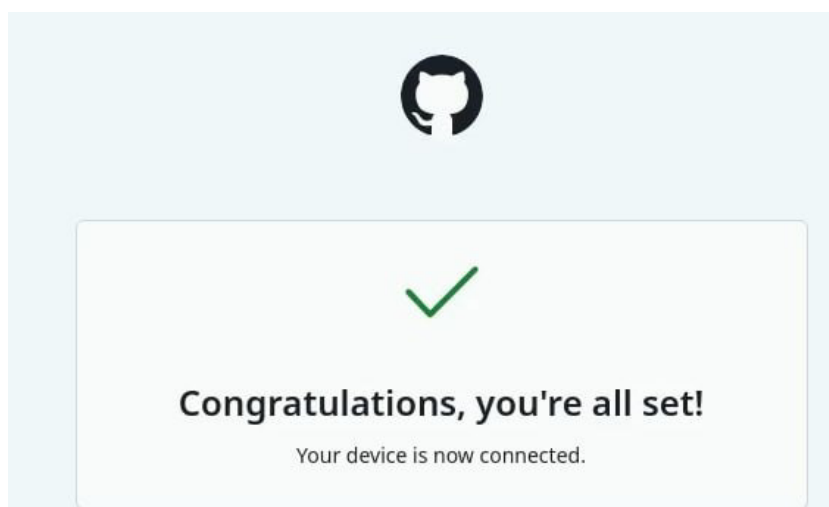


Рис. 3.14: Завершение авторизации

Вижу сообщение о завершении авторизации (рис. 3.15).

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

Рис. 3.15: Завершение авторизации

## 3.9 Создание репозитория курса на основе шаблона

Сначала создаю директорию, после перехожу в нее, затем ввожу команду, которая позволяет создать репозиторий на основе шаблона репозитория. После этого клонирую репозиторий к себе (рис. 3.16).

```
[akabronina@vbox os-intro]$ git clone --recursive https://github.com/akabronina/study_2024-2025_os-intro.git os-intro
Клонирование в «os-intro»...
remote: Enumerating objects: 39, done.
remote: Counting objects: 100% (39/39), done.
remote: Compressing objects: 100% (36/36), done.
remote: Total 39 (delta 2), reused 24 (delta 1), pack-reused 0 (from 0)
Получение объектов: 100% (39/39), 20.25 КиБ | 1.45 МБ/с, готово.
Определение изменений: 100% (2/2), 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/akabronina/work/study/2024-2025/Операционные системы/os-intro/os-intro/template/presentation»...
```

Рис. 3.16: Создание репозитория

Перехожу в каталог курса с помощью утилиты `cd`, проверяю содержание каталога с помощью утилиты `ls` (рис. fig. 3.17).

```
[akabronina@vbox os-intro]$ ls
CHANGELOG.md  COURSE  Makefile  README.git-flow.md  template
config        LICENSE  README.en.md  README.md
```

Рис. 3.17: Перемещение между директориями

Удаляю лишние файлы с помощью утилиты `rm`, далее создаю необходимые каталоги используя `makefile` (рис. fig. 3.18).

```
[akabronina@vbox os-intro]$ echo os-intro > COURSE
[akabronina@vbox os-intro]$ make
Usage:
  make <target>

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

[akabronina@vbox os-intro]$ make prepare
[akabronina@vbox os-intro]$ ls
CHANGELOG.md  labs      prepare      README.en.md  template
config        LICENSE  presentation  README.git-flow.md
COURSE        Makefile  project-personal  README.md
```

Рис. 3.18: Удаление файлов и создание каталогов

Добавляю все новые файлы для отправки на сервер (сохраняю добавленные изменения) с помощью команды `git add` и комментирую их с помощью `git commit` (рис. fig. 3.19).

```
[akabronina@vbox os-intro]$ git add .
[akabronina@vbox os-intro]$ git commit -am 'feat(main): make course structur
e'
```

Рис. 3.19: Отправка файлов на сервер

Отправляю файлы на сервер с помощью git push (рис. fig. 3.20).

```
/__init__.py
create mode 100644 project-personal/stage6/report/pandoc/filters/
/core.py
create mode 100644 project-personal/stage6/report/pandoc/filters/
/main.py
create mode 100644 project-personal/stage6/report/pandoc/filters/
/pandocattributes.py
create mode 100644 project-personal/stage6/report/report.md
[akabronina@vbox os-intro]$ git push
```

Рис. 3.20: Отправка файлов на сервер

## 4 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.

## **5 Выводы**

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