

# **Лабораторная работа №2**

**Отчёт**

Борисенкова София Павловна

# Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Ответы на контрольные вопросы	13

## Список иллюстраций

3.1	Установка git . . . . .	7
3.2	Установка gh . . . . .	7
3.3	Указание имени . . . . .	8
3.4	Указание почты . . . . .	8
3.5	Настройка кодировки utf8 . . . . .	8
3.6	Настройка git . . . . .	8
3.7	Создание ключа RSA . . . . .	9
3.8	Создание ключа ed22519 . . . . .	10
3.9	Создание ключа pgr . . . . .	11
3.10	Список pgr ключей . . . . .	11
3.11	Настройка автоматических подписей коммитов git . . . . .	12

## **Список таблиц**

# 1 Цель работы

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

## 2 Задание

Создать базовую конфигурацию для работы с git.

Создать ключ SSH.

Создать ключ PGP.

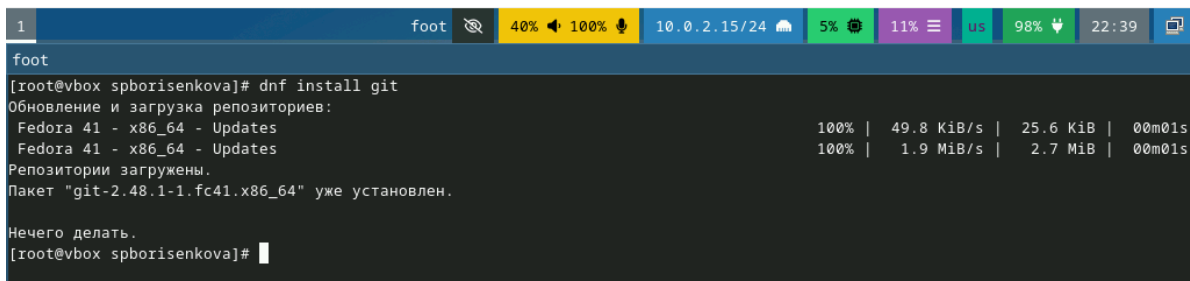
Настроить подписи git.

Зарегистрироваться на Github.

Создать локальный каталог для выполнения заданий по предмету.

### 3 Выполнение лабораторной работы

Для начала установим git. В моём случае он уже установлен (рис. 3.1)

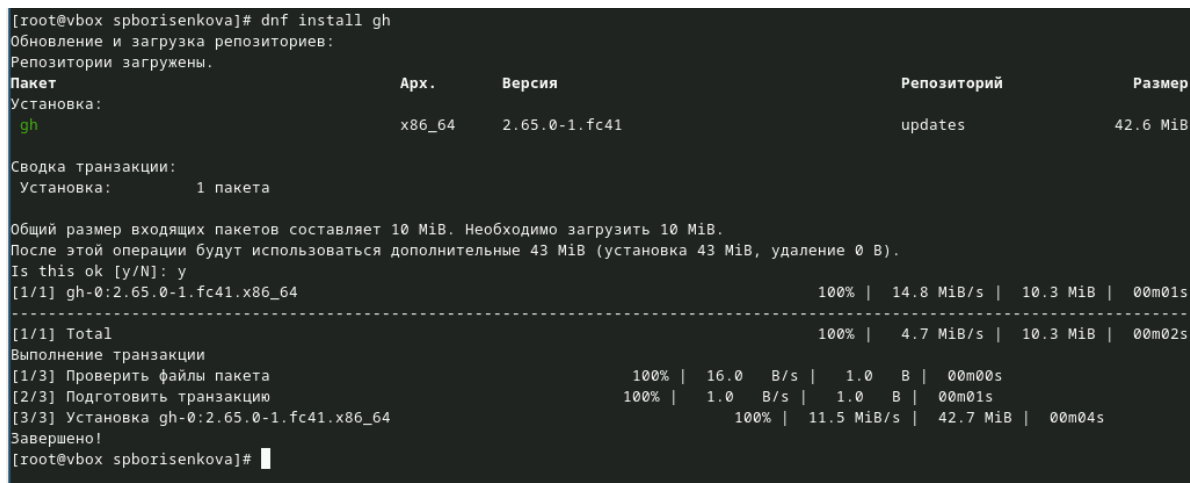


```
1 foot
[roo@vbox spborisenkova]# dnf install git
Обновление и загрузка репозитория:
Fedora 41 - x86_64 - Updates 100% | 49.8 KiB/s | 25.6 KiB | 00m01s
Fedora 41 - x86_64 - Updates 100% | 1.9 MiB/s | 2.7 MiB | 00m01s
Репозитории загружены.
Пакет "git-2.48.1-1.fc41.x86_64" уже установлен.

Нечего делать.
[roo@vbox spborisenkova]#
```

Рис. 3.1: Установка git

Теперь установим gh (рис. 3.2)



```
[roo@vbox spborisenkova]# dnf install gh
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет Арх. Версия Репозиторий Размер
Установка:
gh x86_64 2.65.0-1.fc41 updates 42.6 MiB

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).
Is this ok [y/N]: y
[1/1] gh-0:2.65.0-1.fc41.x86_64 100% | 14.8 MiB/s | 10.3 MiB | 00m01s
-----
[1/1] Total 100% | 4.7 MiB/s | 10.3 MiB | 00m02s
Выполнение транзакции
[1/3] Проверить файлы пакета 100% | 16.0 B/s | 1.0 B | 00m00s
[2/3] Подготовить транзакцию 100% | 1.0 B/s | 1.0 B | 00m01s
[3/3] Установка gh-0:2.65.0-1.fc41.x86_64 100% | 11.5 MiB/s | 42.7 MiB | 00m04s
Завершено!
[roo@vbox spborisenkova]#
```

Рис. 3.2: Установка gh

Далее, зададим имя для владельца репозитория. В данном случае это моё имя (рис. 3.3)

```
[root@vbox os-intro]# git config user.name "Borisenkova Sofia"
```

Рис. 3.3: Указание имени

Теперь зададим почту. Я задала почту, на которую у меня зарегистрирован аккаунт на github (рис. 3.4)

```
[root@vbox os-intro]# git config user.name "sofi-pbor@ya.ru"
```

Рис. 3.4: Указание почты

Настроим кодировку utf8 в выводе сообщений git (рис. 3.5)

```
[root@vbox os-intro]# git config --global init.defaultBranch master
[root@vbox os-intro]# git config --global core.autocrlf input
[root@vbox os-intro]# git config --global core.safecrlf warn
```

Рис. 3.5: Настройка кодировки utf8

Зададим имя начальной ветки, настроим параметры autocrlf и safecrlf (рис. 3.6)

```
[root@vbox os-intro]# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
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
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:wwwOCjdp8LNzlr/DIzBF6rTUckPpHNCLPyp0qsQR2c root@vbox
The key's randomart image is:
+---[RSA 4096]-----+
| . . . . |
| o..+ . |
| .oE.B . |
| . +oB B . |
| . o.X o S . |
| +.O @ o |
| ..= X o |
| + * . o |
| +o..o ..o |
+-----[SHA256]-----+
```

Рис. 3.6: Настройка git



Создадим ключ RSA размером 4096 бит (рис. 3.7)

```
/root/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y
Enter passphrase for "/root/.ssh/id_ed25519" (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:zpXmXo/hcz7f0BrluHZa8XpxmRmQF9yBflFz022uayw root@vbox
The key's randomart image is:
+--[ED25519 256]--+
|      ++B|
|      + +*|
|      . oo+|
|      . . =.|
|      S +  ..B|
|    o +    @+|
|      o . o.= B|
|      . oE+*O+|
|      . oBO*o|
+-----[SHA256]-----+
```

Рис. 3.7: Создание ключа RSA

Теперь создадим ключ по алгоритму ed22519 (рис. 3.8)

```

[root@vbox os-intro]# 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 already exists.
Overwrite (y/n)? y
Enter passphrase for "/root/.ssh/id_ed25519" (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:zpXmXo/hcz7f0BrluHZa8XpxmRmQF9yBf1z022uayw root@vbox
The key's randomart image is:
+--[ED25519 256]--+
|          ++B|
|          + +*|
|          . oo+|
|          . . -.|
|          S + ..B|
|          o +  @+|
|          o . o.= B|
|          . oE!*O|
|          . oBO*o|
+-----[SHA256]-----+

```

Рис. 3.8: Создание ключа ed25519

Теперь создадим ключ `grg`. Выбираем из предложенных вариантов первый тип (RSA and RSA), размер ключа задаём 4096 бит и делаем срок действия ключа неограниченным. После нас попросят ввести свои данные. Мы вводим имя и адрес электронной почты. После этого соглашаемся с генерацией ключа (рис. 3.9)

```
[root@vbox os-intro]# 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:
  (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: Borisenkova Sofiya
Адрес электронной почты: sofi-pbor@ya.ru
```

Рис. 3.9: оздание ключа pgr

Далее, выводим список pgr ключей (рис. 3.10)

```
[root@vbox os-intro]# 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
[keyboxd]
-----
sec   rsa4096/A665AF4A478342ED 2025-03-13 [SC]
      6B886FD8480CD1E6ECE9F370A665AF4A478342ED
uid           [ абсолютно ] Borisenkova Sofiya <sofi-pbor@ya.ru>
ssb   rsa4096/5ED21BCD8E0539B1 2025-03-13 [E]
```

Рис. 3.10: Список pgr ключей

Копируем наш ключ в буфер обмена. Вставляем этот ключ на гитхаб, и задаём ему имя. Я выбрала имя Sway

Теперь производим настройку автоматических подписей (рис. 3.11)

```
[root@vbox os-intro]# git config --global user.signingkey sofi-pbor@ya.ru
[root@vbox os-intro]# git config --global commit.gpgsign true
[root@vbox os-intro]# git config --global gpg.program $(which.gpg2)
-bash: which.gpg2: команда не найдена
[root@vbox os-intro]# git config --global gpg.program $(which gpg2)
```

Рис. 3.11: Настройка автоматических подписей коммитов git

Создаём рабочую директорию курса и переходим в неё. Далее, создаём репозиторий для лабораторных работ из шаблона и клонируем его к себе на компьютер. Переходим в него с помощью `cd` и удаляем ненужные файлы (`package.json`) и создаём необходимые каталоги, записав в файл `COURSE` строку `os-intro` (это наш текущий курс) и прописываем `make prepare` для того, чтобы нужные нам каталоги создались. Теперь добавляем нашу папку для отправки. Делаем коммит, в котором указываем, что мы сделали структуру курса. И отправляем файлы на сервер GitHub с помощью команды `push`. # Выводы

Была произведена установка `git`, проведена его первоначальная настройка, были созданы ключи для авторизации и подписи, а также создан репозиторий курса из предложенного шаблона

## 4 Ответы на контрольные вопросы

1. Системы контроля версий – это системы, в которых мы можем хранить свои проекты и выкладывать их обновления, контролируя релизы и каждые внесённые изменения. Эти системы нужны для работы над проектами, чтобы иметь возможность контролировать версии проектов и в случае командной работы контролировать изменения, внесённые всеми участниками. Также, VCS позволяют откатываться на более ранние версии
2. Хранилище – репозиторий, в нём хранятся все файлы проекта и все его версии  
commit – внесённые изменения в репозитории  
история – это история изменений файлов проекта  
рабочая копия – копия, сделанная из версии репозитория, с которой непосредственно работает сам разработчик
3. Централизованные системы контроля версий имеют один центральный репозиторий, с которым работают все разработчики. Примером является CVS, который является уже устаревшей системой.  
В децентрализованных системах же используется множество репозиториях одного проекта у каждого из разработчиков, при этом репозитории можно объединять брать из каждого только то, что нужно. Примером является знакомый нам Git
4. Создаётся репозиторий, и разрабатывается проект. При внесении изменений файлы отправляются на сервер
5. Разработчик клонирует репозиторий к себе на компьютер, и после внесения

- изменений выгружает их на сервер в качестве отдельной версии. После этого разработчики с более высокими правами могут, например, объединить его версию с текущей
6. Хранение файлов проекта, а также обеспечение командной работы, и контроль за версиями проекта
  7. `git clone` – копирует проект с сервера на компьютер  
`git add` – добавляет папку для выгрузки на сервер  
`git commit` – фиксирует изменения репозитория  
`git push` – выгружает изменения на сервер  
`git pull` – получить изменения с сервера  
`git rm` – удалить файл  
`git status` – получить статус репозитория
  8. С локальным: `git commit -am "added files"` – создаёт коммит С удалённым:  
`git push` – загрузить данные на удалённый сервер
  9. Ветки – это несколько независимых копий проекта, в каждой из которых ведётся разработка какой-то конкретной функции, при этом ветки существуют параллельно. Они нужны, когда нужно параллельно вести разработку нескольких функций, а в конце их можно объединить в одну
  10. Игнорировать файлы можно, внося их в файл `.gitignore`. Игнорировать файлы нужно, когда их не нужно добавлять в репозиторий. Например, это могут быть файлы виртуального окружения (`venv`)