

# **Лабораторная работа №2**

**Отчёт**

Мошаров Денис Максимович

# 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>Выводы</b>	<b>18</b>
<b>5</b>	<b>Ответы на контрольные вопросы</b>	<b>19</b>
	<b>Список литературы</b>	<b>21</b>

# Список иллюстраций

3.1	Установка git . . . . .	7
3.2	Установка gh . . . . .	8
3.3	Указание имени . . . . .	8
3.4	Указание почты . . . . .	9
3.5	Настройка кодировки utf8 . . . . .	9
3.6	Настройка git . . . . .	9
3.7	Создание ключа RSA . . . . .	9
3.8	Создание ключа ed22519 . . . . .	10
3.9	Создание ключа pgr (1) . . . . .	11
3.10	Создание ключа pgr (2) . . . . .	12
3.11	Список pgr ключей . . . . .	13
3.12	Копирование ключа . . . . .	13
3.13	Вставка ключа в GitHub . . . . .	14
3.14	Настройка автоматических подписей коммитов git . . . . .	14
3.15	Авторизация в gh . . . . .	15
3.16	Создание рабочей директории и переход в неё . . . . .	15
3.17	Создание репозитория курса . . . . .	15
3.18	Клонирование репозитория . . . . .	15
3.19	Удаление ненужных файлов и использование make . . . . .	16
3.20	Использование git add . . . . .	16
3.21	Использование git commit . . . . .	16
3.22	Использование git push . . . . .	17

## Список таблиц

# 1 Цель работы

Изучить идеологию и применение средств контроля версий. Освоить умения по работе с git [tuis?]

## 2 Задание

Создать базовую конфигурацию для работы с git.

Создать ключ SSH.

Создать ключ PGP.

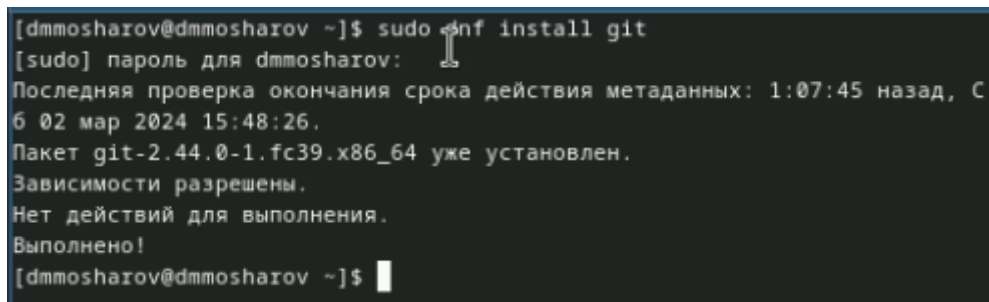
Настроить подписи git.

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)



```
[dmmosharov@dmmosharov ~]$ sudo dnf install git
[sudo] пароль для dmmosharov: 
Последняя проверка окончания срока действия метаданных: 1:07:45 назад, С
6 02 мар 2024 15:48:26.
Пакет git-2.44.0-1.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
[dmmosharov@dmmosharov ~]$
```

Рис. 3.1: Установка git

Теперь установим gh (рис. 3.2)

```
[dmmosharov@dmmosharov ~]$ sudo dnf install git
[sudo] пароль для dmmosharov: 
Последняя проверка окончания срока действия метаданных: 1:07:45 назад, С
6 02 мар 2024 15:48:26.
Пакет git-2.44.0-1.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
[dmmosharov@dmmosharov ~]$ sudo dnf install gh
Последняя проверка окончания срока действия метаданных: 1:07:58 назад, С
6 02 мар 2024 15:48:26.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
=====
Установка:
gh          x86_64       2.43.1-1.fc39  updates      9.1 М
=====
Результат транзакции
=====
Установка 1 Пакет

Объем загрузки: 9.1 М
Объем изменений: 46 М
Продолжить? [д/Н]: у
Загрузка пакетов:
gh-2.43.1-1.fc39.x86_64.rpm                30 MB/s | 9.1 MB    00:00
-----
Общий размер                               9.7 MB/s | 9.1 MB    00:00
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно.
Выполнение транзакции
Подготовка      :                               1/1
Установка       : gh-2.43.1-1.fc39.x86_64    1/1
Запуск скрипта  : gh-2.43.1-1.fc39.x86_64    1/1
Проверка        : gh-2.43.1-1.fc39.x86_64    1/1

Установлен:
gh-2.43.1-1.fc39.x86_64

Выполнено!
[dmmosharov@dmmosharov ~]$
```

Рис. 3.2: Установка gh

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

```
[dmmosharov@dmmosharov ~]$ git config --global user.name "Denis Mosharov"
```

Рис. 3.3: Указание имени

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



```
[dmmosharov@dmmosharov ~]$ git config --global user.email denis.krosh@mail.com
```

Рис. 3.4: Указание почты

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

Настройка кодировки utf8

Рис. 3.5: Настройка кодировки utf8

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

```
[dmmosharov@dmmosharov ~]$ git config --global core.quotepath false
```

Рис. 3.6: Настройка git

Создадим ключ RSA размером 4096 бит (рис. 3.7)

```
[dmmosharov@dmmosharov ~]$ git config --global init.defaultBranch master  
[dmmosharov@dmmosharov ~]$ git config --global core.autocrlf input  
[dmmosharov@dmmosharov ~]$ git config --global core.safecrlf warn
```

Рис. 3.7: Создание ключа RSA

Теперь создадим ключ по алгоритму ed22519 (рис. 3.8)

```

[dmmosharov@dmmosharov ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/dmmosharov/.ssh/id_rsa):
Created directory '/home/dmmosharov/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dmmosharov/.ssh/id_rsa
Your public key has been saved in /home/dmmosharov/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:jzxs5MGWlzaNdGjpaSlyGok31jzqfd+G/xq+kh1fbJw dmmosharov@dmmosharov
The key's randomart image is:
+---[RSA 4096]-----+
|
|      .
|      o
|      . * = =
|      . B S X o o .
|      o # X o .E=
|      o B . +.+
|      . o .. +oo..
|      . . . .+*+.
+-----[SHA256]-----+
[dmmosharov@dmmosharov ~]$ █

```

Рис. 3.8: Создание ключа ed22519

Теперь создадим ключ gpg. Выбираем из предложенных вариантов первый тип (RSA and RSA), размер ключа задаём 4096 бит и делаем срок действия ключа неограниченным (рис. 3.9)

```

[dmmosharov@dmmosharov ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/dmmosharov/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dmmosharov/.ssh/id_ed25519
Your public key has been saved in /home/dmmosharov/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:G7wtH+VWAawPh73zAkS/m1YkSrIayewZXF3hDBBxXY dmmosharov@dmmosharov
The key's randomart image is:
+--[ED25519 256]--+
|      .=*o.      |
|      ..*.E      |
|      o B +..    |
|      .o 0 = ..   |
|      =SB *.o.    |
|      X=.o*.      |
|      ++oo=oo     |
|      . oo+o. .   |
|      .. .        |
+-----[SHA256]-----+
[dmmosharov@dmmosharov ~]$ 

```

Рис. 3.9: создание ключа pgp (1)

После нас попросят ввести свои данные. Мы вводим имя и адрес электронной почты. После этого соглашаемся с генерацией ключа (рис. 3.10)

```
[dmmosharov@dmmosharov ~]$ gpg --full-generate-key
gpg (GnuPG) 2.4.3; Copyright (C) 2023 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 '/home/dmmosharov/.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
```

Рис. 3.10: оздание ключа gpg (2)

Далее, выводим список gpg ключей (рис. 3.11)

```

Ваше полное имя: Denis Mosharov
Адрес электронной почты: denis.krosh@gmail.com
Примечание:
Вы выбрали следующий идентификатор пользователя:
    "Denis Mosharov <denis.krosh@gmail.com>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? o
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
gpg: /home/dmmosharov/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/dmmosharov/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/dmmosharov/.gnupg/openpgp-revocs.d/C5A5E18801E856C24D9FC527C8C74DB743B86ED7.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2024-03-02 [SC]
      C5A5E18801E856C24D9FC527C8C74DB743B86ED7
uid           Denis Mosharov <denis.krosh@gmail.com>
sub   rsa4096 2024-03-02 [E]

```

Рис. 3.11: Список ргр ключей

Копируем наш ключ в буфер обмена (рис. 3.12)

```

[dmmosharov@dmmosharov ~]$ 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/C8C74DB743B86ED7 2024-03-02 [SC]
      C5A5E18801E856C24D9FC527C8C74DB743B86ED7
uid           [ абсолютно ] Denis Mosharov <denis.krosh@gmail.com>
ssb   rsa4096/60C53B1D8EDF51E4 2024-03-02 [E]

```

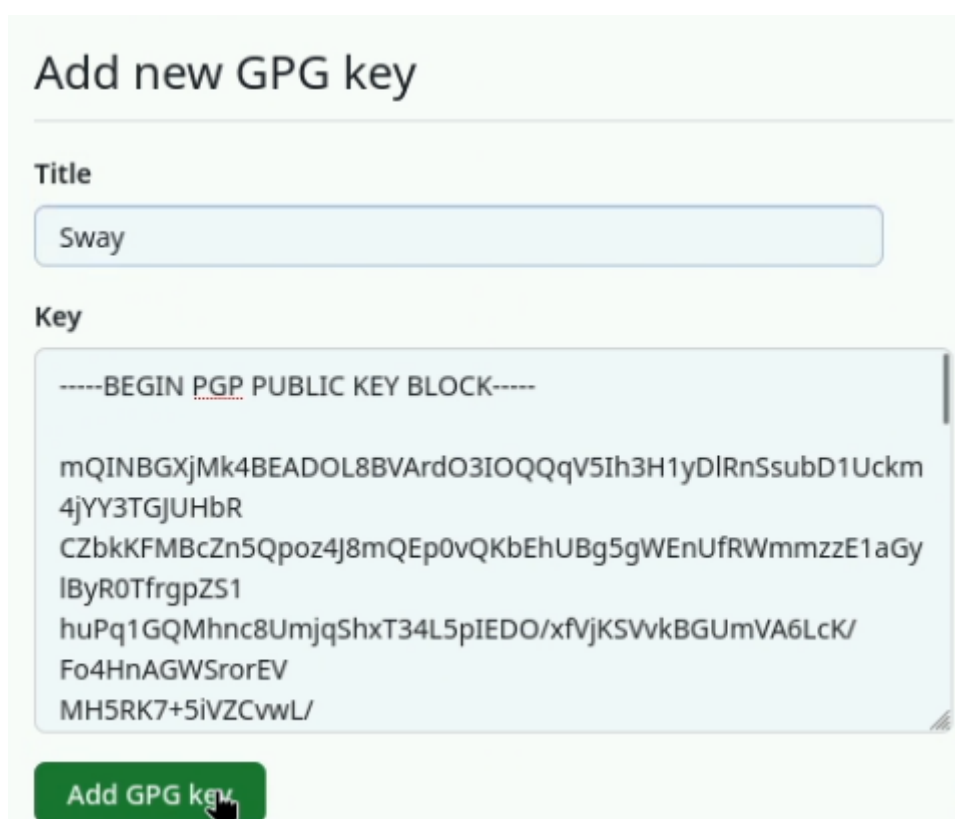
Рис. 3.12: Копирование ключа

Вставляем этот ключ на гитхаб, и задаём ему имя. Я выбрал имя Sway (рис. 3.13)

```
[dmmosharov@dmmosharov ~]$ gpg --armor --export denis.krosh@gmail.com  
xclip -sel clip
```

Рис. 3.13: Вставка ключа в GitHub

Теперь производим настройку автоматических подписей (рис. 3.14)



**Add new GPG key**

**Title**

Sway

**Key**

```
-----BEGIN PGP PUBLIC KEY BLOCK-----  
  
mQINBGXjMk4BEADOL8BVArDO3IOQQqV5Ih3H1yDlRnSsubD1Uckm  
4jYY3TGJUHbR  
CZbkKFMBCzn5Qpoz4J8mQEp0vQKbEhUBg5gWEnUfRWmmzzE1aGy  
lByR0TfrgpZS1  
huPq1GQMhnc8UmjqShxT34L5pIEDO/xfVjKSVvkBGUmVA6LcK/  
Fo4HnAGWSrorEV  
MH5RK7+5iVZCvWL/
```

**Add GPG key**

Рис. 3.14: Настройка автоматических подписей коммитов git

После, нам нужно авторизоваться в github с помощью gh. Мы выбираем сайт для авторизации (GitHub.com), после выбираем предпочитаемый протокол (SSH), публичный SSH ключ (id\_rsa.pub), и имя для ключа (Sway). В качестве способа авторизации выбираем авторизацию через браузер (рис. 3.15)

```
[dmmosharov@dmmosharov ~]$ git config --global user.signingkey denis.krosh@gmail.com
[dmmosharov@dmmosharov ~]$ git config --global commit.gpgsign true
[dmmosharov@dmmosharov ~]$ git config --global gpg.program $(which gpg2)
```

Рис. 3.15: Авторизация в gh

Теперь создаём рабочую директорию курса и переходим в неё (рис. 3.16)

```
[dmmosharov@dmmosharov ~]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /home/dmmosharov/.ssh/id_rsa.pub
? Title for your SSH key: Sway
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 8140-FE47
Press Enter to open github.com in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
✓ Uploaded the SSH key to your GitHub account: /home/dmmosharov/.ssh/id_rsa.pub
✓ Logged in as theraf1u1
[dmmosharov@dmmosharov ~]$
```

Рис. 3.16: Создание рабочей директории и переход в неё

Далее, создаём репозиторий для лабораторных работ из шаблона (рис. 3.17)

```
[dmmosharov@dmmosharov ~]$ mkdir -p ~/work/study/2023-2024/"Операционные системы"
[dmmosharov@dmmosharov ~]$ cd ~/work/study/2023-2024/"Операционные системы"
```

Рис. 3.17: Создание репозитория курса

И клонируем его к себе на компьютер (рис. 3.18)

```
[dmmosharov@dmmosharov Операционные системы]$ gh repo create study_2023-2024_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository theraf1u1/study_2023-2024_os-intro on GitHub
https://github.com/theraf1u1/study_2023-2024_os-intro
```

Рис. 3.18: Клонирование репозитория

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 (рис. 3.19)

```
[dmmosharov@dmmosharov Операционные системы]$ git clone --recursive git@github.com:theraflu1/study_2023-2024_os-intro.git os-intro
Клонирование в «os-intro»...
```

Рис. 3.19: Удаление ненужных файлов и использование `make`

Теперь добавляем нашу папку для отправки (рис. 3.20)

```
[dmmosharov@dmmosharov ~]$ cd ~/work/study/2023-2024/"Операционные системы"/os-intro
[dmmosharov@dmmosharov os-intro]$ rm package.json
[dmmosharov@dmmosharov os-intro]$ echo os-intro > COURSE
[dmmosharov@dmmosharov os-intro]$ make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare        Generate directories structure
  submodule      Update submodules
[dmmosharov@dmmosharov os-intro]$ make prepare
```

Рис. 3.20: Использование `git add`

Делаем коммит, в котором указываем, что мы сделали структуру курса (рис. 3.21)

```
[dmmosharov@dmmosharov os-intro]$ git add .
```

Рис. 3.21: Использование `git commit`

И отправляем файлы на сервер GitHub с помощью команды `push` (рис. 3.22)



```

_eqnos.py
create mode 100755 project-personal/stage5/report/pandoc/filters/pandoc
_fignos.py
create mode 100755 project-personal/stage5/report/pandoc/filters/pandoc
_secnos.py
create mode 100755 project-personal/stage5/report/pandoc/filters/pandoc
_tablenos.py
create mode 100644 project-personal/stage5/report/pandoc/filters/pandoc
xnos/__init__.py
create mode 100644 project-personal/stage5/report/pandoc/filters/pandoc
xnos/core.py
create mode 100644 project-personal/stage5/report/pandoc/filters/pandoc
xnos/main.py
create mode 100644 project-personal/stage5/report/pandoc/filters/pandoc
xnos/pandocattributes.py
create mode 100644 project-personal/stage5/report/report.md
create mode 100644 project-personal/stage6/presentation/Makefile
create mode 100644 project-personal/stage6/presentation/image/kulyabov.
jpg
create mode 100644 project-personal/stage6/presentation/presentation.md
create mode 100644 project-personal/stage6/report/Makefile
create mode 100644 project-personal/stage6/report/bib/cite.bib
create mode 100644 project-personal/stage6/report/image/placeimg_800_60
0_tech.jpg
create mode 100644 project-personal/stage6/report/pandoc/csl/gost-r-7-0
-5-2008-numeric.csl
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc

```

Рис. 3.22: Использование git push

## 4 Выводы

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

## 5 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 внесе-

- ния 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`)

## **Список литературы**