

# **Отчёт по лабораторной работе № 2**

**ОТЧЁТ**

Саенко Ангелина Андреевна

# Содержание

<b>Цель работы</b>	<b>5</b>
<b>Задание</b>	<b>6</b>
<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>Выводы</b>	<b>16</b>
<b>Ответы на контрольные вопросы</b>	<b>17</b>
<b>Список литературы</b>	<b>19</b>

# Список иллюстраций

1	Установка git. . . . .	7
2	Установка gh. . . . .	8
3	Указание имени . . . . .	8
4	Указание почты. . . . .	8
5	Настройка кодировки utf8. . . . .	9
6	Настройка git . . . . .	9
7	Создание ключа RSA . . . . .	9
8	Создание ключа ed25529. . . . .	9
9	Создание pgr ключа (1) . . . . .	10
10	Создание pgr ключа (2) . . . . .	11
11	Список pgr ключей. . . . .	11
12	Копирование ключа. . . . .	11
13	Вставка ключа в GitHub . . . . .	12
14	Настройка автоматических подписей коммитов git. . . . .	12
15	Авторизация в gh. . . . .	12
16	Создание рабочей директории и переход в неё. . . . .	13
17	Создание репозитория курса. . . . .	13
18	Клонирование репозитория . . . . .	13
19	Удаление ненужных файлов и использование make. . . . .	13
20	Использование git add. . . . .	14
21	Использование git commit . . . . .	14
22	Использование git push . . . . .	15

## **Список таблиц**

# Цель работы

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

# 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.

# Выполнение лабораторной работы

Для начала установим git. В моём случае он уже установлен (рис. [-@fig:001]).

```
[angelina@saasaenko ~]$ dnf install git
Для выполнения запрошенной операции требуется привилегия суперпользователя. Пожалуйста, войдите в систему как пользователь с повышенными правами или используйте опции "--assumeno" или "--downloadonly", чтобы выполнить команду без изменения состояния системы.
[angelina@saasaenko ~]$ sudo -i
[sudo] пароль для angelina:
[root@saasaenko ~]# dnf install git
Обновление и загрузка репозитория:
Fedora 41 - x86_64 - Updates      100% | 9.7 KiB/s | 25.0 KiB | 00m03s
Репозитории загружены.
Пакет "git-2.48.1-1.fc41.x86_64" уже установлен.
```

Рис. 1: Установка git.

Теперь установим gh.

```
[root@aasaenko ~]# dnf install gh
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет                                Arch.  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% | 768.0 KiB/s | 10.3 MiB | 00m14s
-----
[1/1] Total                              100% | 751.7 KiB/s | 10.3 MiB | 00m14s
Выполнение транзакции
[1/3] Проверить файлы пак100% | 16.0 B/s | 1.0 B | 00m00s
[2/3] Подготовить транзак 100% | 1.0 B/s | 1.0 B | 00m01s
[3/3] Установка gh-0:2.65.0-1.fc41 100% | 5.0 MiB/s | 42.7 MiB | 00m09s
Завершено!
[root@aasaenko ~]#
```

Рис. 2: Установка gh.

Далее, зададим имя для владельца репозитория. В данном случае это моё имя.

```
[root@aasaenko ~]# git config --global user.name "Angelina Saenko"
[root@aasaenko ~]# git config --global user.email "angelinasaenko867@gmail.com"
[root@aasaenko ~]#
```

Рис. 3: Указание имени

Теперь зададим почту. Я задаю почту, на которую у меня зарегистрирован аккаунт на GitHub.

```
[root@aasaenko ~]# git config --global user.email "angelinasaenko867@gmail.com"
[root@aasaenko ~]#
```

Рис. 4: Указание почты.

Настроим кодировку utf8 в выводе сообщений git.



```
[root@asasaenko ~]# git config --global user.email "angelinasenko867@gmail.com"
[root@asasaenko ~]#
```

Рис. 5: Настройка кодировки utf8.

Зададим имя начальной ветки, настроим параметры autocrlf и safecrlf

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

Рис. 6: Настройка git

Создадим ключ RSA размером 4096 бит.

```
[root@asasaenko ~]# git config --global init.defaultBranch master
[root@asasaenko ~]# git config --global core.autocrlf input
[root@asasaenko ~]# git config --global core.safecrlf warn
[root@asasaenko ~]#
```

Рис. 7: Создание ключа RSA

Теперь создадим ключ по алгоритму ed25519.

```
[root@asasaenko ~]# 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:
SHA256:9fwQu45QPXVocb84U19X7P0KkSX8l80iduZVj985/GA root@asasaenko
The key's randomart image is:
+----[RSA 4096]-----+
|      .. .ooo.. |
|      . o.o. .o |
|      . oio. .+ |
|      . . +++.* |
|      S o.=0.+* |
|      o oi=Eo+ |
|      o ++oB. |
|      ..o + |
|      . |
+-----[SHA256]-----+
[root@asasaenko ~]#
```

Рис. 8: Создание ключа ed25529.

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

```
[root@aasaenko ~]# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
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:AmZnNYMoIJX8wMjN6V1F4FljQSY65rSj/Cts2EnDxJtU root@aasaenko
The key's randomart image is:
+--[ED25519 256]--+
|o++*o.=8%+.      |
| += E*o*.o        |
|o.o==+==         |
|oo.=+==.         |
|o...+ S          |
|. =              |
|+ +              |
| o .             |
|                 |
+----[SHA256]-----+
[root@aasaenko ~]#
```

Рис. 9: Создание gpg ключа (1)

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

```
[root@saasaenko ~]# 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>m = срок действия ключа - n недель
<n>y = 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. 10: Создание pgr ключа (2)

Далее, выводим список pgr ключей .

```
GnuPG должен составить идентификатор пользователя для идентификации ключа.
Ваше полное имя: Angelina Saenko
Адрес электронной почты: angelinasaenko867@gmail.com
Примечание:
Вы выбрали следующий идентификатор пользователя:
"Angelina Saenko <angelinasaenko867@gmail.com>"
```

Рис. 11: Список pgr ключей.

Копируем наш ключ в буфер обмена

```
[root@saasaenko ~]# gpg --armor --export <angelinasaenko867@gmail.com> | xclip -sel c
lip
```

Рис. 12: Копирование ключа.

Вставляем этот ключ на гитхаб, и задаём ему имя. Я выбрала имя Sway.

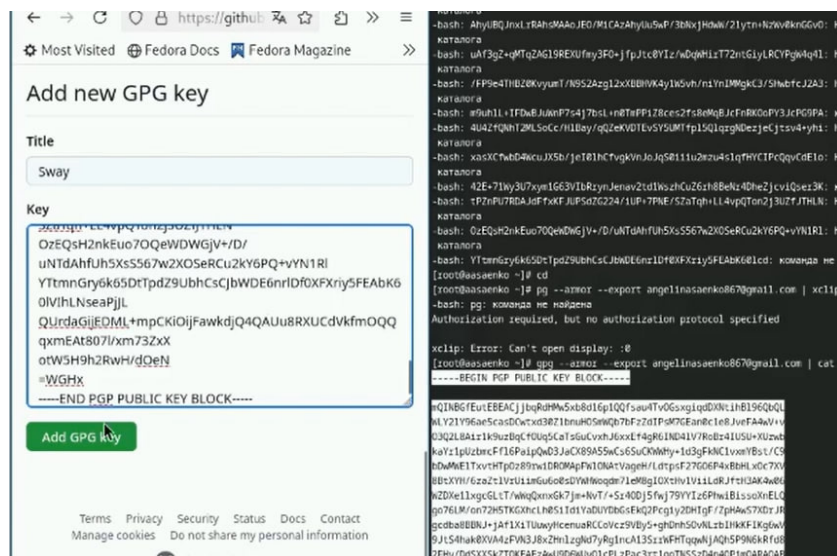


Рис. 13: Вставка ключа в GitHub

Теперь производим настройку автоматических подписей

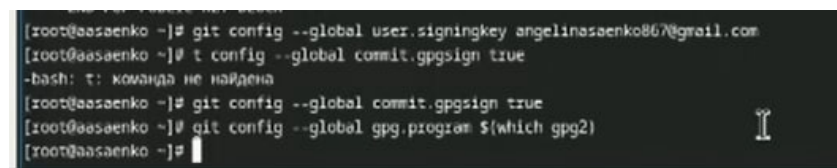


Рис. 14: Настройка автоматических подписей коммитов git.

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 .

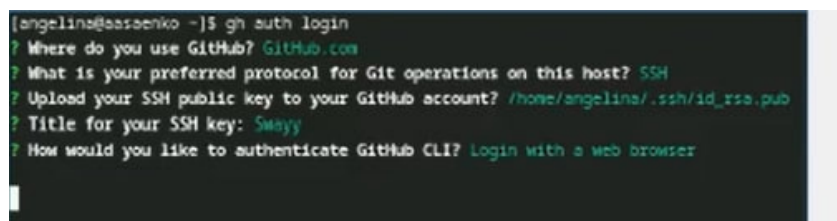


Рис. 15: Авторизация в gh.

Теперь создаём рабочую директорию курса и переходим в неё

```
[angelina@asasaenko ~]$ mkdir -p ~/work/study/2024-2025/"Операционные системы"
[angelina@asasaenko ~]$ cd ~/work/study/2024-2025/"Операционные системы"
[angelina@asasaenko Операционные системы]$
```

Рис. 16: Создание рабочей директории и переход в неё.

Далее, создаём репозиторий для лабораторных работ из шаблона.

```
[angelina@asasaenko Операционные системы]$ gh repo create study_2024-2025_os-intro --template=yanadharma/course-directory-student-template --public
```

Рис. 17: Создание репозитория курса.

И клонируем его к себе на компьютер

```
[angelina@asasaenko Операционные системы]$ git clone --recursive git@github.com:comner/study_2024-2025_os-intro.git os-intro
```

Рис. 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 нам текущий курс) и прописываем `make prepare` для того , чтобы нужные нам каталоги создались

```
[angelina@asasaenko ~]$ cd ~/work/study/2024-2025/"Операционные системы"/os-intro
[angelina@asasaenko os-intro]$ rm package.json
[angelina@asasaenko os-intro]$ echo os-intro > COURSE
[angelina@asasaenko os-intro]$ make
Usage:
  make <target>

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

[angelina@asasaenko os-intro]$ make prepare
[angelina@asasaenko os-intro]$
```

Рис. 19: Удаление ненужных файлов и использование `make`.

Теперь добавляем нашу папку для отправки .

```
[angelina@asasaenko os-intro]$ git add .  
[angelina@asasaenko os-intro]$
```

Рис. 20: Использование git add.

Делаем коммит ,в котором указываем , что мы сделали структуру курса.

```
create node 100644 project-personal/stage4/report/pandoc/filters/pandocxmos/_init_  
_py  
create node 100644 project-personal/stage4/report/pandoc/filters/pandocxmos/core.py  
create node 100644 project-personal/stage4/report/pandoc/filters/pandocxmos/main.py  
create node 100644 project-personal/stage4/report/pandoc/filters/pandocxmos/pandoca  
ttributes.py  
create node 100644 project-personal/stage4/report/report.nd  
create node 100644 project-personal/stage5/presentation/.projectile  
create node 100644 project-personal/stage5/presentation/.texlabroot  
create node 100644 project-personal/stage5/presentation/Makefile  
create node 100644 project-personal/stage5/presentation/image/kulyabov.jpg  
create node 100644 project-personal/stage5/presentation/presentation.nd  
create node 100644 project-personal/stage5/report/Makefile  
create node 100644 project-personal/stage5/report/bib/cite.bib  
create node 100644 project-personal/stage5/report/image/placeholder_800_600_tech.jpg  
create node 100644 project-personal/stage5/report/pandoc/csl/gost-r-7-0-5-2008-nume  
ric.csl  
create node 100755 project-personal/stage5/report/pandoc/filters/pandoc_eqnos.py  
create node 100755 project-personal/stage5/report/pandoc/filters/pandoc_fignos.py  
create node 100755 project-personal/stage5/report/pandoc/filters/pandoc_secnos.py  
create node 100755 project-personal/stage5/report/pandoc/filters/pandoc_tablenos.py  
create node 100644 project-personal/stage5/report/pandoc/filters/pandocxmos/_init_  
_py  
create node 100644 project-personal/stage5/report/pandoc/filters/pandocxmos/core.py  
create node 100644 project-personal/stage5/report/pandoc/filters/pandocxmos/main.py  
create node 100644 project-personal/stage5/report/pandoc/filters/pandocxmos/pandoca  
ttributes.py  
create node 100644 project-personal/stage5/report/report.nd  
create node 100644 project-personal/stage6/presentation/.projectile  
create node 100644 project-personal/stage6/presentation/.texlabroot  
create node 100644 project-personal/stage6/presentation/Makefile  
create node 100644 project-personal/stage6/presentation/image/kulyabov.jpg  
create node 100644 project-personal/stage6/presentation/presentation.nd  
create node 100644 project-personal/stage6/report/Makefile  
create node 100644 project-personal/stage6/report/bib/cite.bib  
create node 100644 project-personal/stage6/report/image/placeholder_800_600_tech.jpg  
create node 100644 project-personal/stage6/report/pandoc/csl/gost-r-7-0-5-2008-nume  
ric.csl  
create node 100755 project-personal/stage6/report/pandoc/filters/pandoc_eqnos.py  
create node 100755 project-personal/stage6/report/pandoc/filters/pandoc_fignos.py
```

Рис. 21: Использование git commit

И отправляем файлы на сервер GitHub с помощью команды push

```
[angelina@asaenko os-intro]$ git push
Переисключение объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
При скачивании изменений используется до 4 потоков
Скачивание объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 342.31 КБ | 2.72 МБ/с, готово.
Total 38 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:AngelinaSaenko/study_2024-2025_os-intro.git
   aa48c80..a1624c0 master -> master
[angelina@asaenko os-intro]$
```

Рис. 22: Использование git 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 .



# 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 позволяет 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 репозиториев 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` - клонирует проект с сервера на компьютер `git commit` - добавляет папку для выгрузки на сервер `git push` - фиксирует изменения репозитория `git pull` - 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 удаленным: `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

[@tanenbaum\_book\_modern-os\_ru][@robbins\_book\_bash\_en][@zarrelli\_book\_mastering-bash\_en][@nam\_ewhbook\_learning-bash\_en] ::: :::