

# **Отчёт по лабораторной работе №5**

**Операционные системы**

Закиров Нурислам Дамирович

# 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 ключа SSH . . . . .	8
3.4	Создание ключа GPG . . . . .	9
3.5	Настроить подписи Git . . . . .	10
3.6	Настройка gh . . . . .	11
3.7	Создание шаблона для рабочего пространства . . . . .	13
3.8	Настройка каталога курса . . . . .	14
<b>4</b>	<b>Выводы</b>	<b>16</b>
<b>5</b>	<b>Ответы на контрольные вопросы.</b>	<b>17</b>
	<b>Список литературы</b>	<b>20</b>

## Список иллюстраций

3.1	Установка git и gh . . . . .	7
3.2	Базовая настройка git . . . . .	8
3.3	Создание ключа SSH . . . . .	8
3.4	Создание ключа pgr . . . . .	9
3.5	Копирование ключа в буфер обмена . . . . .	9
3.6	Окно New GPG key в GitHub . . . . .	10
3.7	Окно GitHub . . . . .	10
3.8	Настройка подписей коммитов git . . . . .	11
3.9	Авторизация gh . . . . .	11
3.10	Авторизация через браузер . . . . .	12
3.11	Успешная авторизация . . . . .	13
3.12	Успешная авторизация . . . . .	13
3.13	Создание репозитории курса . . . . .	13
3.14	Создание репозитории курса . . . . .	14
3.15	Перемещение по каталогам . . . . .	14
3.16	Удаление файлов . . . . .	14
3.17	Создание каталогов . . . . .	14
3.18	Отправка файлов на сервер . . . . .	15

## **Список таблиц**

# 1 Цель работы

Целью данной лабораторной работы является изучение идеологии и применение средств контроля версий и приобретение знаний по работе с git.

## 2 Задание

1. Зарегистрироваться на GitHub
2. Создать базовую конфигурацию для работы с git
3. Создать ключ SSH
4. Создать ключ GPG
5. Настроить подписи Git
6. Создать локальный каталог для выполнения заданий по предмету.

## 3 Выполнение лабораторной работы

### 3.1 Установка программного обеспечения

Заранее регистрация в git уже была успешно выполнена, поэтому сразу переходим к следующим этапам. Устанавливаем git при помощи `dnf install git` и устанавливаем gh при помощи `dnf install gh`(рис. fig. 3.1).

```
[ndzakirov@ndzakirov ~]$ sudo -i
[sudo] пароль для ndzakirov:
[root@ndzakirov ~]# dnf install git
Последняя проверка окончания срока действия метаданных:
2024 18:21:13.
Пакет git-2.43.0-1.fc38.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
[root@ndzakirov ~]# dnf install gh
Последняя проверка окончания срока действия метаданных:
2024 18:21:13.
Пакет gh-2.36.0-1.fc38.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
```

Рис. 3.1: Установка git и gh

### 3.2 Базовая настройка git

Далее настраиваем верификацию и подписание коммитов git, задаем имя начальной ветки master, параметр autocrlf и safecrlf(рис. fig. 3.2).

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

Рис. 3.2: Базовая настройка git

### 3.3 Создание ключа SSH

Создаем ключ ssh по алгоритму rsa с ключём размером 4096 бит и по алгоритму ed25519 (рис. fig. 3.3).

```
[root@ndzakirov ~]# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
Enter passphrase (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:Tx1z91A4J5hzS2oivTmg76UuNUtHQMksZcIt2FU08E
The key's randomart image is:
+--[ED25519 256]--+
|  + +=E    o  ..|
|  . = * =.. + *..|
|    *   oo   o=o=..|
|    .   o +.o+.o.|
|    .S+.=.   .|
|    . +o=    |
|    + +o.    |
|    . oo     |
|    ++      |
+-----[SHA256]-----+
```

Рис. 3.3: Создание ключа SSH

Следующим этапом мы генерируем ключ pgr выбирая следующие опции: тип RSA and RSA, размер 4096, срок действия неограниченный (рис. fig. 3.4).



### 3.4 Создание ключа GPG

```
[root@endzakirov ~]# gpg --full-generate-key
gpg (GnuPG) 2.4.0; Copyright (C) 2021 Free Software Foundat
This is free software: you are free to change and redistrib
There is NO WARRANTY, to the extent permitted by law.

gpg: создан каталог '/root/.gnupg'
gpg: создан щит с ключами '/root/.gnupg/pubring.kbx'
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)
Срок действия ключа не ограничен
```

Рис. 3.4: Создание ключа gpg

После чего, мы копируем данный нам ключ командой `gpg --armor --export | xclip -sel clip` в буфер обмена, где вместо “PGP Fingerprint” вставляем данные значения.(рис. fig. 3.5).

```
[root@endzakirov ~]# gpg --armor --export BFA077A162927416 | xclip -sel clip
[root@endzakirov ~]#
```

Рис. 3.5: Копирование ключа в буфер обмена

Теперь вставляем данный ключ при помощи комбинации `Ctrl+V` на официальном сайте GitHub в настройках GitHub, нажав на кнопку New GPG key(рис. fig. 3.6).

## Add new GPG key

Title

Key

-----BEGIN PGP PUBLIC KEY BLOCK-----

```
mQINBGXTo3QBEADHObWDanhIrtHo5MG/Wpe+QXazFT5Drftm+gX6Ltgf9NgHetd
n5voLHQlkw072nvumM+Sj17bwuKYhBiqePWG2scRZTgvtYCSBESbMnErud9pFvm5
VcKEcF3D7hB8YCYRzjzz2yD+5n3wOyeAjtWLMQ9lltm5wIIgD9Vp2wd23htosjL
PkZx9vKNL9un6HQ/A4goPgHFuZiZKU3esRSulrrsRac6mfMEldgAdB065Ht/EqF
2Q0YpNRvufxCVbyICITX640TV7DSXA7ZjqUzif6xsAsmZse2EUdyhtZ48rPSjLOf
npclQ8m/wypdogC0UMw6sVrmit9h8AapHhDWbdGYoa7v9tTSHppjxrA+L/qTEyXz
EKD/YliSeoTjj7RxqpJwOFFTVCEnfW148v/ImbET9aW8RWmEB8AniQnZ+GdQtMh
```

Add GPG key

Рис. 3.6: Окно New GPG key в GitHub

Ключ успешно привязался(рис. fig. 3.7).

## GPG keys

New GPG key

This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.



Email address: Nuric35737@gmail.com

Key ID: BFA077A162927416

Subkeys: F94291E96EBEBA24

Added on Feb 19, 2024

Delete

Learn how to [generate a GPG key and add it to your account](#).

Рис. 3.7: Окно GitHub

## 3.5 Настроить подписи Git

Следующим этапом мы используем введённый email и указываем Git применять его при подписи коммитов (рис. fig. 3.8).

```
[root@ndzakirov ~]# git config --global user.signingkey BFA077A162927416
[root@ndzakirov ~]# git config --global commit.gpgsign true
[root@ndzakirov ~]# git config --global gpg.program $(which gpg2)
[root@ndzakirov ~]#
```

Рис. 3.8: Настройка подписей коммитов git

## 3.6 Настройка gh

Далее мы авторизируемся в терминале через команду `gh auth login`. Отвечаем на все наводящие вопросы (рис. 3.9).

```
[root@ndzakirov ~]# gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? HTTPS
X Sorry, your reply was invalid: "н" is not a valid answer, please try again
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 27C9-5FE2
Press Enter to open github.com in your browser...
Running Firefox as root in a regular user's session is not supported. ($?)
TTY is /run/user/1001/.mutter-Xwaylandauth.0QQAJ2 which is owned by ndzakirov
/usr/bin/xdg-open: строка 881: x-www-browser: команда не найдена
Running Firefox as root in a regular user's session is not supported. ($?)
TTY is /run/user/1001/.mutter-Xwaylandauth.0QQAJ2 which is owned by ndzakirov
/usr/bin/xdg-open: строка 881: iceweasel: команда не найдена
/usr/bin/xdg-open: строка 881: seamonkey: команда не найдена
/usr/bin/xdg-open: строка 881: mozilla: команда не найдена
VMware: No 3D enabled (0, Выполнено).
libEGL warning: egl: failed to create dri2 screen


(epiphany:5252): Gtk-CRITICAL **: 22:05:54.360: Unable to register the application: GDBus.Error:org.freedesktop.DBus.Error.NameHasNoOwner: Could not activate remote peer: unit failed.

** (process:2): WARNING **: 22:05:54.823: Failed to get atspi registered listeners: GDBus.Error:org.freedesktop.DBus.Error.NameHasNoOwner: Could not activate remote peer: unit failed.

(epiphany:5252): Gtk-CRITICAL **: 22:05:59.768: Unable to register the application: GDBus.Error:org.freedesktop.DBus.Error.NameHasNoOwner: Could not activate remote peer: unit failed.
```

Рис. 3.9: Авторизация gh

После входим в свой аккаунт GitHub через браузер (рис. 3.10).



## Sign in to GitHub

Username or email address

Password

[Forgot password?](#)

Sign in

New to GitHub? [Create an account](#)

Рис. 3.10: Авторизация через браузер

После авторизации, Git просит ввести код, который мы заранее скопировали с терминала и вставили в браузер, после чего мы успешно авторизовались (рис. fig. 3.11).

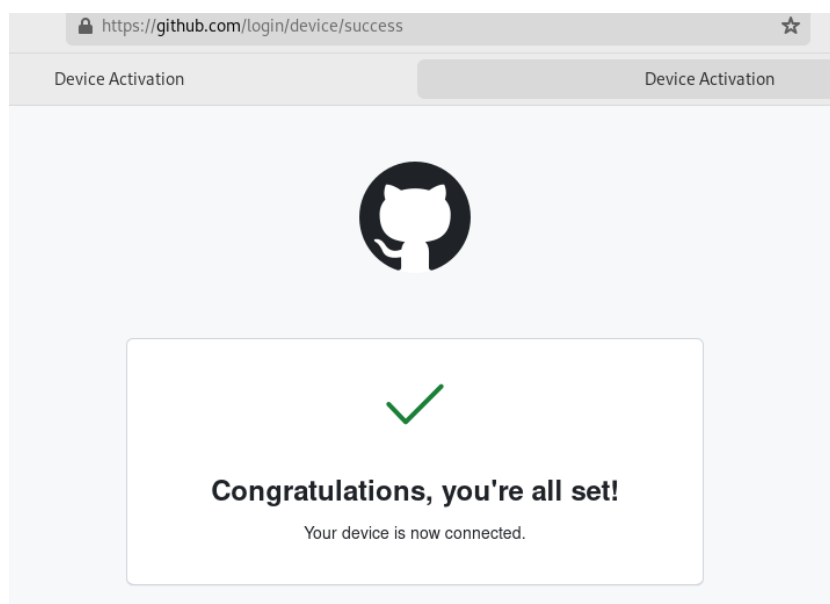


Рис. 3.11: Успешная авторизация

В терминале также подтверждается наша авторизация(рис. fig. 3.12).

```
gh config set --hostname github.com --git-protocol https
✓ Configured git protocol
! Authentication credentials saved in plain text
✓ Logged in as Nurislam0323
[root@ndzakirov ~]#
[root@ndzakirov ~]#
[root@ndzakirov ~]#
```

Рис. 3.12: Успешная авторизация

## 3.7 Создание шаблона для рабочего пространства

Далее нам необходимо создать шаблон рабочего пространства. При помощи `mkdir -p ~/work/study/2023-2024/“Операционные системы”` мы создаем каталог, после чего переходим в нее. Создаем шаблон при помощи `gh repo create`(рис. fig. 3.13).

```
[root@ndzakirov Операционные системы]# gh repo create study_2023-2024_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository Nurislam0323/study_2023-2024_os-intro on GitHub
[root@ndzakirov Операционные системы]#
```

Рис. 3.13: Создание репозитории курса

После чего мы клонируем репозиторий курса на нашу систему при помощи `git clone`(рис. fig. 3.14).

```
[ndzakirov@ndzakirov Операционные системы]$ git clone --recursive https://github.com/Nurislam0323/study_2023-2024_os-intro.git os-intro
Клонирование в «os-intro»...
remote: Enumerating objects: 32, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 32 (delta 1), reused 18 (delta 0), pack-reused 0
Получение объектов: 100% (32/32), 18.60 КиБ | 1.09 МиБ/с, готово.
Обработка изменений: 100% (1/1), готово.
```

Рис. 3.14: Создание репозитории курса

## 3.8 Настройка каталога курса

Переходим в созданную папку `os-intro` при помощи `cd`(рис. fig. 3.15).

```
[ndzakirov@ndzakirov Операционные системы]$ cd os-intro/
[ndzakirov@ndzakirov os-intro]$
```

Рис. 3.15: Перемещение по каталогам

Удаляем лишние файлы при помощи `rm` (рис. fig. 3.16).

```
[ndzakirov@ndzakirov os-intro]$ rm package.json
```

Рис. 3.16: Удаление файлов

Создаем необходимые каталоги при помощи `echo` и `make`(рис. fig. 3.17).

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

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

Рис. 3.17: Создание каталогов

Отправляем файлы на сервер GitHub при помощи git add, git commit и git push(рис. fig. 3.18).

```
[ndzakirov@ndzakirov os-intro]$ git add .
[ndzakirov@ndzakirov os-intro]$ git commit -am 'feat(main): make course structure'
Author identity unknown

*** Пожалуйста, скажите мне кто вы есть.

Запустите

  git config --global user.email "you@example.com"
  git config --global user.name "Ваше Имя"

для указания идентификационных данных аккаунта по умолчанию.
Пропустите параметр --global для указания данных только для этого репозитория.

fatal: не удалось выполнить автоопределение адреса электронной почты (получено «ndzakirov@ndzakirov.(none)»)
[ndzakirov@ndzakirov os-intro]$ git config --global user.name "Nurislam Zakirov"
[ndzakirov@ndzakirov os-intro]$ git config --global user.email "nuric35737@gmail.com"
[ndzakirov@ndzakirov os-intro]$ git commit -am 'feat(main): make course structure'
[master fded8d6] feat(main): make course structure
 2 files changed, 1 insertion(+), 14 deletions(-)
 delete mode 100644 package.json
[ndzakirov@ndzakirov os-intro]$ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 3 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 293 байта | 293.00 КиБ/с, готово.
Всего 3 (изменений 1), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
```

Рис. 3.18: Отправка файлов на сервер

## 4 Выводы

В ходе данной лабораторной работы я изучил идеологии и применения средств контроля версий, а также приобрел знания по работе с git.



## 5 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.

## Список литературы

1. Лабораторная работа № 2 [Электронный ресурс] URL: <https://esystem.rudn.ru/mod/page/view.php?id=12345>