

# **Лабораторная работа №2**

**НКАбд-06-23**

Улитина Мария Максимовна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
4.1	Установка git и gh . . . . .	9
4.2	Базовая настройка git . . . . .	9
4.3	Создание ключа ssh . . . . .	10
4.4	Ключи pgr . . . . .	10
4.5	Добавление PGP ключа в GitHub . . . . .	11
4.6	Настройка автоматических подписей коммитов git . . . . .	11
4.7	Настройка gh . . . . .	12
4.8	Создание репозитория курса на основе шаблона . . . . .	12
4.9	Настройка каталога курса . . . . .	12
<b>5</b>	<b>Контрольные вопросы</b>	<b>14</b>
<b>6</b>	<b>Выводы</b>	<b>16</b>
	<b>Список литературы</b>	<b>17</b>

## Список иллюстраций

4.1	git . . . . .	9
4.2	gh . . . . .	9
4.3	имя и email . . . . .	9
4.4	utf-8 . . . . .	9
4.5	имя начальной ветки . . . . .	10
4.6	autocrlf и safecrlf . . . . .	10
4.7	rsa . . . . .	10
4.8	ed25519 . . . . .	10
4.9	ключ pgr . . . . .	10
4.10	тип ключа . . . . .	11
4.11	Список ключей . . . . .	11
4.12	Подпись коммитов . . . . .	11
4.13	Авторизируемся в gh . . . . .	12
4.14	Шаблон для рабочего пространства . . . . .	12
4.15	Шаблон для рабочего пространства . . . . .	12
4.16	Шаблон для рабочего пространства . . . . .	12
4.17	Каталог курса . . . . .	12
4.18	Удаление . . . . .	13
4.19	Создание каталога . . . . .	13
4.20	Создание каталога . . . . .	13
4.21	Отправление файлов на сервер . . . . .	13
4.22	Отправление файлов на сервер . . . . .	13

## Список таблиц

# 1 Цель работы

Изучить идеологию и применение средств контроля версий. 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 Теоретическое введение

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию,

отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.



## 4 Выполнение лабораторной работы

### 4.1 Установка git и gh

Установим git (рис. 4.1).

```
root@10:~# dnf install git
```

Рис. 4.1: git

Установим gh (рис. 4.2).

```
root@10:~# dnf install gh
```

Рис. 4.2: gh

### 4.2 Базовая настройка git

Зададим имя и email владельца репозитория (рис. 4.3).

```
root@10:~# git config --global user.name 'mmulitina'
root@10:~# git config --global user.email 'marieulitinal@yandex.ru'
```

Рис. 4.3: имя и email

Настроим utf-8 в выводе сообщений git (рис. 4.4).

```
root@10:~# git config --global core.quotepath false
```

Рис. 4.4: utf-8

Зададим имя начальной ветки (рис. 4.5).

```
root@10:~# git config --global init.defaultBranch master
```

Рис. 4.5: имя начальной ветки

Настроим параметры autocrlf и safecrlf (рис. 4.6).

```
root@10:~# git config --global core.autocrlf input
root@10:~# git config --global core.safecrlf warn
```

Рис. 4.6: autocrlf и safecrlf

## 4.3 Создание ключа ssh

По алгоритму rsa создадим ключ размером 4096 бит (рис. 4.7).

```
root@10:~# ssh-keygen -t rsa -b 4096
```

Рис. 4.7: rsa

И по алгоритму ed25519 (рис. 4.8).

```
root@10:~# ssh-keygen -t ed25519
```

Рис. 4.8: ed25519

## 4.4 Ключи pgr

Создадим ключ pgr (рис. 4.9).

```
root@10:~# gpg --full-generate-key
```

Рис. 4.9: ключ pgr

Из предложенных опций выбираем тип ключа и его размер(рис. 4.10).

```
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
```

Рис. 4.10: тип ключа

## 4.5 Добавление PGP ключа в GitHub

Выводим список ключей и копируем отпечаток приватного ключа: (рис. 4.11).

```
root@10:~# gpg --list-secret-keys --keyid-format LONG
```

Рис. 4.11: Список ключей

Скопируем полученный ключ и добавим его на github.

## 4.6 Настройка автоматических подписей коммитов git

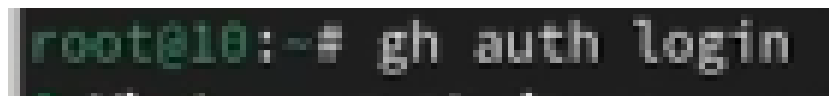
Используя введённый email, укажем Git применять его при подписи коммитов: (рис. 4.12).

```
root@10:~# git config --global user.signingkey 5E8561C1464F430E
root@10:~# git config --global commit.gpgsign true
root@10:~# git config --global gpg.program $(which gpg2)
```

Рис. 4.12: Подпись коммитов

## 4.7 Настройка gh

Авторизируемся в gh (рис. 4.13).

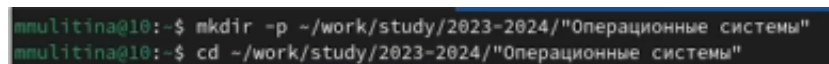


```
root@10:~# gh auth login
```

Рис. 4.13: Авторизируемся в gh

## 4.8 Создание репозитория курса на основе шаблона

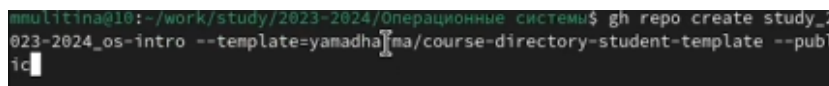
Создадим шаблон для рабочего пространства (рис. 4.14).



```
mmulitina@10:~$ mkdir -p ~/work/study/2023-2024/"Операционные системы"  
mmulitina@10:~$ cd ~/work/study/2023-2024/"Операционные системы"
```

Рис. 4.14: Шаблон для рабочего пространства

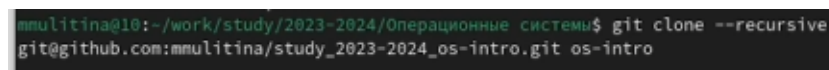
(рис. 4.15).



```
mmulitina@10:~/work/study/2023-2024/Операционные системы$ gh repo create study_2023-2024_os-intro --template=yamadhamma/course-directory-student-template --public
```

Рис. 4.15: Шаблон для рабочего пространства

(рис. 4.16).

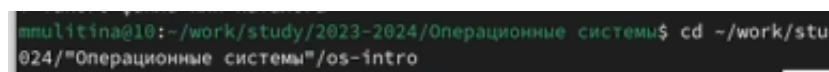


```
mmulitina@10:~/work/study/2023-2024/Операционные системы$ git clone --recursive git@github.com:mmulitina/study_2023-2024_os-intro.git os-intro
```

Рис. 4.16: Шаблон для рабочего пространства

## 4.9 Настройка каталога курса

Перейдем в каталог курса (рис. 4.17).



```
mmulitina@10:~/work/study/2023-2024/Операционные системы$ cd ~/work/study/2023-2024/"Операционные системы"/os-intro
```

Рис. 4.17: Каталог курса

Удалим лишние файлы (рис. 4.18).

```
mmulitina@10:~/work/study/2023-2024/Операционные системы/os-intro$ rm package.js
```

Рис. 4.18: Удаление

Создадим необходимые каталоги (рис. 4.19).

```
mmulitina@10:~/work/study/2023-2024/Операционные системы/os-intro$ echo os-intro  
> COURSE
```

Рис. 4.19: Создание каталога

(рис. 4.20).

```
mmulitina@10:~/work/study/2023-2024/Операционные системы/os-intro$ make  
Usage:  
  make <target>  
  
Targets:  
  list           List of courses  
  prepare        Generate directories structure  
  submodule      Update submules
```

Рис. 4.20: Создание каталога

Отправим файлы на сервер (рис. 4.21).

```
mmulitina@10:~/work/study/2023-2024/Операционные системы/os-intro$ git add .  
mmulitina@10:~/work/study/2023-2024/Операционные системы/os-intro$ git commit -a  
m 'feat(main): make course structure'
```

Рис. 4.21: Отправление файлов на сервер

(рис. 4.22).

```
mmulitina@10:~/work/study/2023-2024/Операционные системы/os-intro$ git push
```

Рис. 4.22: Отправление файлов на сервер

## 5 Контрольные вопросы

1. Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.
2. Хранилище - единый репозиторий для хранения файлов. Commit - сохранить все добавленные изменения и все изменённые файлы. История - история изменений репозитория. Рабочая копия - копия проекта, основанная на версии из хранилища.
3. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным. Классические - CVS, Subversion. Распределённые - Git, Bazaar, Mercurial.
4. Создание и подключение удаленного репозитория, отправка изменений проекта на сервер.
5. Системы контроля версий могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из

участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

6. Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` с различными опциями.
7. Создание основного дерева репозитория `git init`, получение обновлений 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, сохранение текущих и добавленных изменений `git diff`.
8. `git push` - отправление из локального репозитория на удаленный.
9. Для разработки отдельных частей проекта.
10. Для игнорирования файлов, которые не стоит добавлять в репозиторий. С помощью конфигурационного файла `gitignore`.

## 6 Выводы

В процессе выполнения работы я изучила применение средств контроля версий Git.



# Список литературы

1. Лабораторная работа №2.