

Отчёт по лабораторной работе №2

Дисциплина: Операционные Системы

Зуева Дарья Тимуровна, НПМбв-01-20

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	1. Базовая настройка git	7
3.2	3. Добавление ssh ключа в Github	8
3.3	4. Создание GPG ключа	8
3.4	5. Добавление GPG ключа на Github	9
3.5	6. Настройка автоматических подписей коммитов git	10
3.6	7. Настройка gh	10
3.7	8. Создание репозитория курса на основе шаблона	11
3.8	9. Настройка каталога курса	11
3.9	10. Контрольные вопросы	12
3.9.1	1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?	12
3.9.2	2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.	13
3.9.3	3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.	13
3.9.4	4. Опишите действия с VCS при единоличной работе с хранилищем.	14
3.9.5	5. Опишите порядок работы с общим хранилищем VCS.	14
3.9.6	6. Каковы основные задачи, решаемые инструментальным средством git?	15
3.9.7	7. Назовите и дайте краткую характеристику командам git.	16
3.9.8	8. Приведите примеры использования при работе с локальным и удалённым репозиториями.	16
3.9.9	9. Что такое и зачем могут быть нужны ветви (branches)?	17
3.9.10	10. Как и зачем можно игнорировать некоторые файлы при commit?	17
4	Вывод	19

Список иллюстраций

Список таблиц

1 Цель работы

Цель работы – изучить идеологию и применение средств контроля версий.
Освоить умения по работе с git.

2 Задание

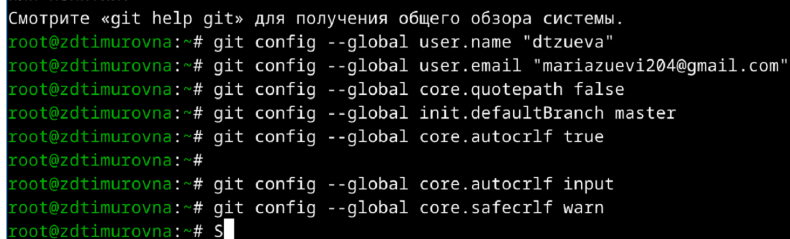
1. Базовая настройка git
2. Создание ключа ssh
3. Добавление ssh ключа в Github
4. Создание GPG ключа
5. Добавление GPG ключа на Github
6. Настройка автоматических подписей коммитов git
7. Настройка gh
8. Создание репозитория курса на основе шаблона
9. Настройка каталога курса
10. Контрольные вопросы

3 Выполнение лабораторной работы

3.1 1. Базовая настройка git

Для самой базовой настройки нужно выполнить следующий ряд команд:

```
git config --global user.name "dtzueva"
git config --global user.email "mariazuevi204@gmail.com"
git config --global core.quotepath false
git config --global init.defaultBranch master
git config --global core.autocrlf true
git config --global core.autocrlf input
git config --global core.safecrlf warn
```



```
Смотрите «git help git» для получения общего обзора системы.
root@zdtimurovna:~# git config --global user.name "dtzueva"
root@zdtimurovna:~# git config --global user.email "mariazuevi204@gmail.com"
root@zdtimurovna:~# git config --global core.quotepath false
root@zdtimurovna:~# git config --global init.defaultBranch master
root@zdtimurovna:~# git config --global core.autocrlf true
root@zdtimurovna:~#
root@zdtimurovna:~# git config --global core.autocrlf input
root@zdtimurovna:~# git config --global core.safecrlf warn
root@zdtimurovna:~# S
```

2. Создание ключа

ssh Создание ключей ssh возможно при помощи многих алгоритмов, например с алгоритмом rsa и размером 4096 (современный стандарт безопасности) и с ал-

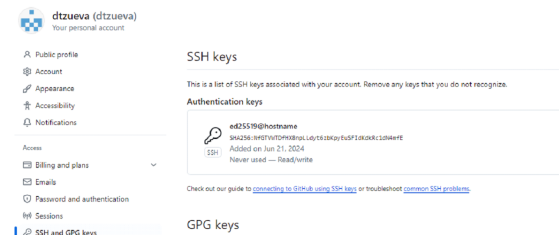
горитмом ed25519

```
root@zdtimurovna:~# 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 (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:gggnvqZE1d/u/zQ5J1z/2NWp0UT70AUUSySgCcaywwA root@zdtimurovna
The key's randomart image is:
+---[RSA 4096]---+
|E. .o . . .o=. |
| . o... o o.. |
|o .+ + o . . |
|.+.+. . . . |
| o. ....S. . o..|
|. . . . .+o.+ |
| + . . .Boo+ |
|+ . . .B+o |
|. . . .+o o |
+-----[SHA256]-----+
```

3.2 3. Добавление ssh ключа в Github

Для начала регистрирую новую учетную запись в github. Затем, после всего

нужного ряда действий, добавляем ssh ключ в Github.



3.3 4. Создание GPG ключа

При создании ключа используется команда `gpg --full-generate-key`, после которой можно настроить автоматическую подпись коммитов git.


```

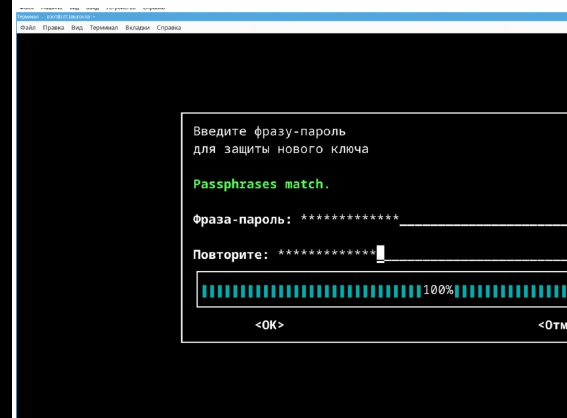
сообщения при создании ключа. время истекло
root@zdtimurovna:~# gpg --full-generate-key
gpg (GnuPG) 2.4.4; Copyright (C) 2024 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

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

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: /root/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/root/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/root/.gnupg/openpgp-revocs.d/947ACB889562892F20FF98CBDE2E4800C8D9DE5E.rev'
открытый и секретный ключи созданы и подписаны.

pub  rsa4096 2024-06-21 [SC]
     947ACB889562892F20FF98CBDE2E4800C8D9DE5E

```



3.4 5. Добавление GPG ключа на Github

```


root@zdtimurovna:~# 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
[keyboard]
-----
sec   rsa4096/DE2E4800C8D9DE5E 2024-06-21 [SC]
      947ACB889562892F20FF98CBDE2E4800C8D9DE5E
uid           [ абсолютно ] dtzueva <retern2003rus@gmail.com>
ssb   rsa4096/09F7654170E3DE73 2024-06-21 [E]

```

Выведем список всех gpg ключей:

GPG keys

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



gpg dtzueva

Email address: retern2003rus@gmail.com

Key ID: DE2E4800C8D9DE5E

Subkeys: 09F7654170E3DE73

Added on Jun 21, 2024

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

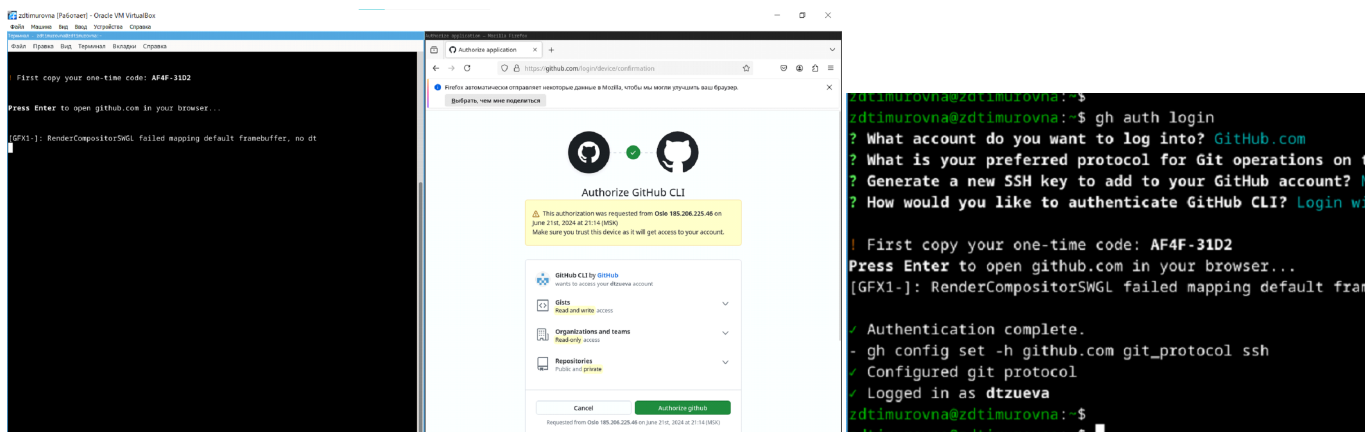
Добавим нужный ключ на Github аналогично ключу ssh.

3.5 6. Настройка автоматических подписей коммитов git

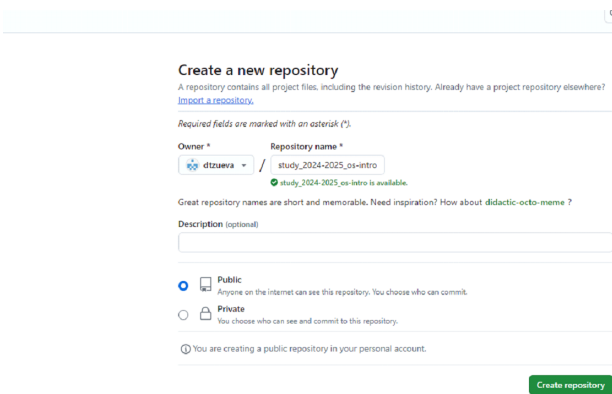
Для настройки автоматических подписей коммитов надо выполнить следующий ряд команд:

```
git config --global user.signingkey DE2E4800C8D9DE5E
git config --global commit.gpgsign true
git config --global gpg.program $(which gpg2)
```

3.6 7. Настройка gh



3.7 8. Создание репозитория курса на основе шаблона



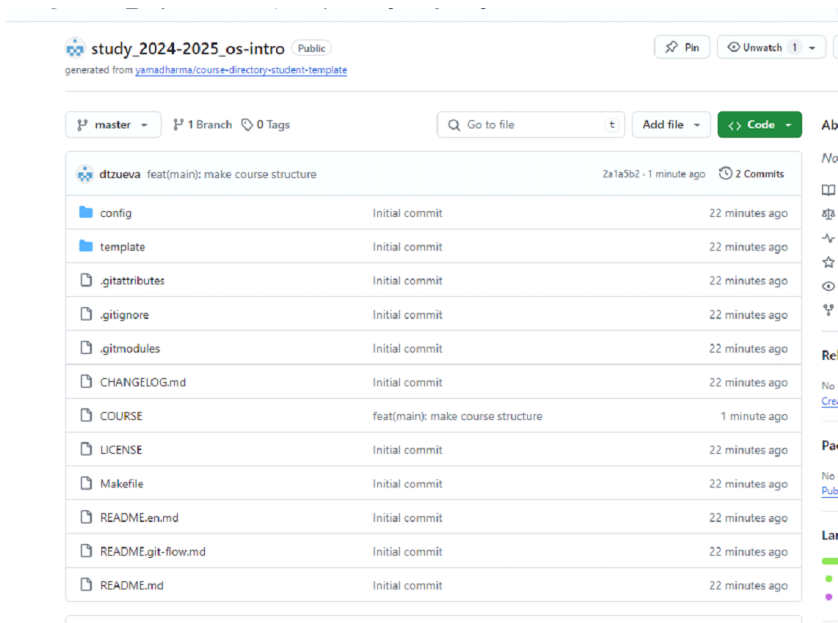
Создадим новый репозиторий на основе шаблона.

```
zdtimurovna@zdtimurovna:~$ git clone --recursive git@github.com:yanadharma/academic-laboratory-report-template.git
Cloning into «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), 18.59 KiB | 18.59 MiB/c, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yanadharma/academic-presentation-markdown-template.git) запущено.
Подмодуль «template/report» (https://github.com/yanadharma/academic-laboratory-report-template.git) запущено.
Клонирование в «/home/zdtimurovna/work/study/2024-2025/Операционные системы/os-intro/template/presentation»...
remote: Enumerating objects: 95, done.
remote: Counting objects: 100% (95/95), done.
remote: Compressing objects: 100% (67/67), done.
remote: Total 95 (delta 34), reused 87 (delta 26), pack-reused 0
Получение объектов: 100% (95/95), 96.99 KiB | 183.00 KiB/c, готово.
Определение изменений: 100% (34/34), готово.
Клонирование в «/home/zdtimurovna/work/study/2024-2025/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 126, done.
remote: Counting objects: 100% (126/126), done.
remote: Compressing objects: 100% (97/97), done.
remote: Total 126 (delta 52), reused 108 (delta 34), pack-reused 0
Получение объектов: 100% (126/126), 335.80 KiB | 411.00 KiB/c, готово.
Определение изменений: 100% (32/32), готово.
Submodule path 'template/presentation': checked out '4ba1761813e197400e8443ff1ca72c60a304f24c'
Submodule path 'template/report': checked out '7c31ab8e5df8c0b2d67caeb8a19ef8028ced88e'
zdtimurovna@zdtimurovna:~$ cd ~/work/study/2024-2025/Операционные системы
zdtimurovna@zdtimurovna:~/work/study/2024-2025/Операционные системы$ mkdir -p ~/work/study/2024-2025/Операционные системы
zdtimurovna@zdtimurovna:~/work/study/2024-2025/Операционные системы$ cd ~/work/study/2024-2025/Операционные системы
```

3.8 9. Настройка каталога курса

Настроим каталог курса по средствам удаления и модифицирования файлов.

```
zdtimurovna@zdtimurovna:~/work/study/2024-2025/Операционные системы/os-intro$ git add .
zdtimurovna@zdtimurovna:~/work/study/2024-2025/Операционные системы/os-intro$ git commit -am 'feat: make course structure'
[master 2a1a5b2] feat(main): make course structure
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
zdtimurovna@zdtimurovna:~/work/study/2024-2025/Операционные системы/os-intro$ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При скатии изменений используется до 4 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 205 Bytes | 205.00 KiB/c, готово.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:dtzueva/study_2024-2025_os-intro.git
1fb8290..2a1a5b2 master -> master
zdtimurovna@zdtimurovna:~/work/study/2024-2025/Операционные системы/os-intro$
```



3.9 10. Контрольные вопросы

3.9.1 1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

Системы контроля версий (VCS) — это инструменты, предназначенные для управления изменениями в файлах и коде в процессе разработки программного обеспечения. Они позволяют:

- **Отслеживать изменения:** сохранять историю изменений файлов.
- **Управлять версиями:** создавать и переключаться между различными версиями проекта.
- **Совместно работать:** поддерживать коллективную работу над проектом, синхронизировать изменения между участниками.
- **Восстанавливать состояние:** возвращаться к предыдущим состояниям файлов.
- **Создавать ветки:** параллельно работать над различными функциональными возможностями и багфиксами.

3.9.2 2. Объясните следующие понятия VCS и их отношения:

хранилище, commit, история, рабочая копия.

- **Хранилище (repository):** основное место, где хранятся все файлы проекта и их истории изменений. Хранилище может быть локальным (на компьютере пользователя) или удалённым (на сервере).
- **Commit:** фиксация изменений. Это операция, при которой сохраняются изменения в файлах и добавляется запись в историю изменений с описанием изменений и временем их внесения.
- **История (history):** последовательность всех коммитов в хранилище, отражающая изменения проекта с течением времени.
- **Рабочая копия (working copy):** текущее состояние файлов проекта на вашем компьютере. Рабочая копия может содержать изменения, которые ещё не зафиксированы в коммите.

3.9.3 3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

Централизованные VCS: - *Описание:* В централизованных VCS есть один центральный сервер, где хранится основное хранилище. Клиенты получают доступ к этому серверу для получения и внесения изменений. - *Примеры:* Subversion (SVN), CVS. - *Преимущества:* Простой и понятный рабочий процесс, контроль над доступом. - *Недостатки:* Зависимость от центрального сервера, ограниченная поддержка работы офлайн.

Децентрализованные VCS: - *Описание:* В децентрализованных VCS каждый разработчик имеет полную копию хранилища, включая всю историю изменений. Это позволяет работать независимо от центрального сервера. - *Примеры:* Git, Mercurial. - *Преимущества:* Возможность работы офлайн, распределённость, повышенная устойчивость к сбоям. - *Недостатки:* Более сложное управление

синхронизацией и объединением изменений.

3.9.4 4. Опишите действия с VCS при единоличной работе с хранилищем.

1. Создание нового хранилища:

```
git init
```

2. Добавление файлов в хранилище:

```
git add <файл>    # Добавить конкретный файл
```

```
git add .          # Добавить все файлы в текущем каталоге
```

3. Создание коммита:

```
git commit -m "Сообщение коммита"
```

4. Просмотр истории изменений:

```
git log
```

5. Создание веток для новых функций или изменений:

```
git branch <имя_ветки>
```

```
git checkout <имя_ветки>
```

3.9.5 5. Опишите порядок работы с общим хранилищем VCS.

1. Клонирование удалённого хранилища:

```
git clone <URL>
```

2. Создание новой ветки для работы над задачей:

```
git checkout -b <имя_ветки>
```

3. Внесение изменений и создание коммитов:

```
git add <файл>
```

```
git commit -m "Описание изменений"
```

4. Обновление локальной копии перед отправкой изменений:

```
git fetch origin
```

```
git merge origin/main      # Предполагая, что основная ветка называется "main"
```

5. Отправка изменений в удалённое хранилище:

```
git push origin <имя_ветки>
```

6. Создание pull request для объединения изменений с основной веткой через интерфейс хостинга, например, GitHub или GitLab.

3.9.6 6. Каковы основные задачи, решаемые инструментальным средством git?

- **Отслеживание изменений в файлах:** фиксация изменений, создание коммитов.
- **Управление версиями:** создание и переключение между версиями проекта.
- **Работа с ветками:** создание, слияние, удаление веток для параллельной работы.
- **Синхронизация с удалёнными хранилищами:** клонирование, отправка и получение изменений.
- **Объединение изменений:** автоматическое и ручное слияние веток.
- **Разрешение конфликтов:** выявление и устранение конфликтов при слиянии изменений.

3.9.7 7. Назовите и дайте краткую характеристику командам git.

`git init`: Инициализация нового локального репозитория.
`git clone <URL>`: Клонирование удалённого репозитория.
`git add <файл>`: Добавление изменений в рабочей копии для следующего коммита.
`git commit -m "сообщение"`: Создание нового коммита с описанием изменений.
`git status`: Проверка состояния рабочей копии и индекса.
`git log`: Просмотр истории коммитов.
`git branch <имя_ветки>`: Создание новой ветки.
`git checkout <имя_ветки>`: Переключение на другую ветку.
`git merge <имя_ветки>`: Слияние указанной ветки с текущей.
`git push`: Отправка изменений в удалённый репозиторий.
`git pull`: Получение и объединение изменений из удалённого репозитория.
`git fetch`: Получение изменений из удалённого репозитория без слияния.

3.9.8 8. Приведите примеры использования при работе с локальным и удалённым репозиториями.

Локальный репозиторий:

```
git init
git add .
git commit -m "Initial commit"
```

Удалённый репозиторий:

```
git clone https://github.com/user/repo.git
cd repo
git checkout -b new-feature
git add .
```



```
git commit -m "Add new feature"
git push origin new-feature
```

3.9.9 9. Что такое и зачем могут быть нужны ветви (branches)?

Ветви (branches) — это параллельные версии хранилища, позволяющие работать над различными задачами независимо друг от друга. Ветви нужны для:

- **Изолирования разработки:** Разработка новых функций, исправление багов или эксперименты могут вестись независимо от основной ветки.
- **Упрощения слияния:** Легкое объединение изменений после завершения работы.
- **Обеспечения стабильности:** Основная ветка (например, main) остаётся стабильной и используется для выпуска релизов, а разработка ведётся в отдельных ветках.

3.9.10 10. Как и зачем можно игнорировать некоторые файлы при commit?

Иногда требуется игнорировать определённые файлы или каталоги, чтобы они не были добавлены в репозиторий. Это может быть полезно для:

Исключения временных файлов: Логи, временные файлы, кэш.

Секретной информации: Конфигурационные файлы с паролями или ключами.

Для игнорирования файлов используется файл .gitignore. Пример содержимого .gitignore:

```
# Игнорировать все файлы .log
*.log

# Игнорировать директорию temp
temp/
```

```
# Игнорировать файл config.yaml  
config.yaml
```

Добавление файлов в `.gitignore` гарантирует, что они не будут отслеживаться Git и не попадут в репозиторий.

4 Вывод

В ходе лабораторной работы были изучены базовые навыки работы с git и его идеология, а также его настройка и на практическом примере создала репозиторий и выполнила коммит.