

Отчёт по лабораторной работе №2

Операционные системы

Гасанова Шакира Чингизовна

Содержание

Цель работы	5
Задание	6
Теоретическое введение	7
Выполнение лабораторной работы	9
Установка программного обеспечения	9
Базовая настройка git	10
Создание ключа SSH	10
Создание ключа GPG	11
Регистрация на GitHub	12
Добавление ключа GPG в GitHub	13
Настройка автоматических подписей коммитов git	14
Настройка gh	14
Создание репозитория курса на основе шаблона	16
Выводы	18
Ответы на контрольные вопросы.	19
Список литературы	22

Список иллюстраций

1	Установка программного обеспечения	9
2	Установка программного обеспечения	9
3	Ввод имени и email	10
4	Настройка utf-8	10
5	Задаю параметры	10
6	Ключ по алгоритму rsa с ключём размером 4096 бит	11
7	Ключ по алгоритму ed25519	11
8	Создание gpg ключа	12
9	Заполнение личной информации, ввод фразы-пароля	12
10	Вход в аккаунт гитхаба	13
11	Копирование отпечатка приватного ключа	13
12	Копирование ключа	13
13	Создание gpg ключа на гитхаб	14
14	Настройка подписей git	14
15	Авторизация на gh	14
16	Копирование одноразового кода	15
17	Завершение авторизации на gh	15
18	Завершение авторизации на gh	16
19	Создание директории	16
20	Создание репозитория и клонирование в директорию	17
21	Удаление файла и создание каталогов	17
22	Отправка файлов на сервер	17

Список таблиц

Цель работы

Целью данной лабораторной работы являются изучение идеологии и применения средств контроля версий и освоение умений по работе с git.

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.

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.

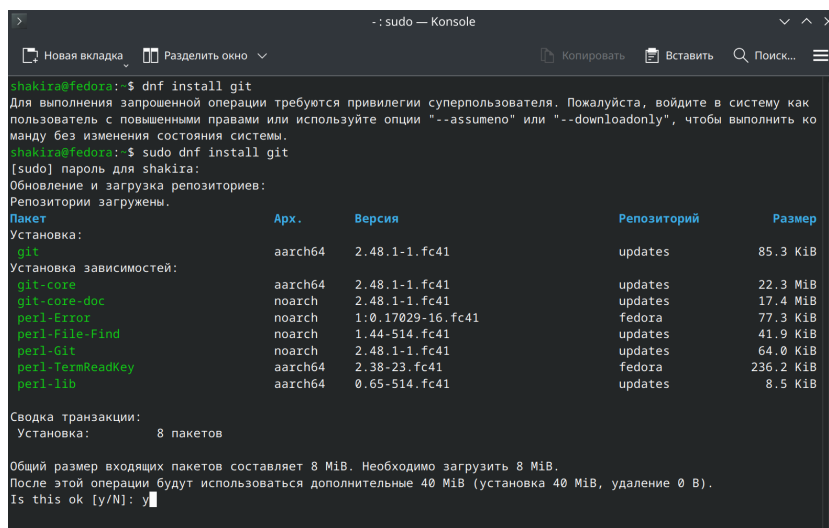
В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

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.

Выполнение лабораторной работы

Установка программного обеспечения

Устанавливаю программное обеспечение git и gh через терминал (рис. fig:001), (рис. @fig:002).

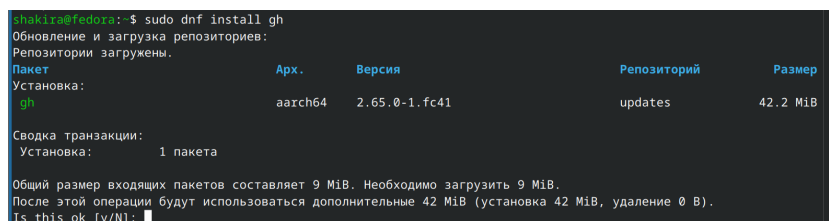


```
shakira@fedora: ~$ dnf install git
Для выполнения запрошенной операции требуются привилегии суперпользователя. Пожалуйста, войдите в систему как
пользователь с повышенными правами или используйте опции "--assumeno" или "--downloadonly", чтобы выполнить ко
манду без изменения состояния системы.
shakira@fedora: ~$ sudo dnf install git
[sudo] пароль для shakira:
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет                                Арх.      Версия      Репозиторий    Размер
Установка:
git                                  aarch64   2.48.1-1.fc41  updates       85.3 KiB
Установка зависимостей:
git-core                            aarch64   2.48.1-1.fc41  updates       22.3 MiB
git-core-doc                        noarch    2.48.1-1.fc41  updates       17.4 MiB
perl-Error                          noarch    1:0.17029-16.fc41  fedora        77.3 KiB
perl-File-Find                     noarch    1.44-514.fc41  updates       41.9 KiB
perl-Git                            noarch    2.48.1-1.fc41  updates       64.0 KiB
perl-TermReadKey                   aarch64   2.38-23.fc41  fedora        236.2 KiB
perl-lib                            aarch64   0.65-514.fc41  updates       8.5 KiB

Сводка транзакции:
Установка:      8 пакетов

Общий размер входящих пакетов составляет 8 MiB. Необходимо загрузить 8 MiB.
После этой операции будут использоваться дополнительные 40 MiB (установка 40 MiB, удаление 0 B).
Is this ok [y/N]: y
```

Рис. 1: Установка программного обеспечения



```
shakira@fedora: ~$ sudo dnf install gh
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет                                Арх.      Версия      Репозиторий    Размер
Установка:
gh                                  aarch64   2.65.0-1.fc41  updates       42.2 MiB

Сводка транзакции:
Установка:      1 пакета

Общий размер входящих пакетов составляет 9 MiB. Необходимо загрузить 9 MiB.
После этой операции будут использоваться дополнительные 42 MiB (установка 42 MiB, удаление 0 B).
Is this ok [y/N]: y
```

Рис. 2: Установка программного обеспечения

Базовая настройка git

Задаю в качестве имени и email владельца репозитория свои данные (рис. @fig:003).

```
shakira@fedora: $ git config --global user.name "Shakira Gasanova"
shakira@fedora: $ git config --global user.email "shakiragasc4@gmail.com"
shakira@fedora: $
```

Рис. 3: Ввод имени и email

Настраиваю utf-8 в выводе сообщений гит (рис. @fig:004).

```
shakira@fedora: $ git config --global core.quotePath false
shakira@fedora: $
```

Рис. 4: Настройка utf-8

Задаю имя начальной ветки, параметр autocrlf и параметр safecrlf (рис. @fig:005).

```
shakira@fedora: $ git config --global core.quotePath false
shakira@fedora: $ git config --global init.defaultBranch master
shakira@fedora: $ git config --global core.autocrlf input
shakira@fedora: $ git config --global core.safecrlf warn
shakira@fedora: $
```

Рис. 5: Задаю параметры

Создание ключа SSH

Создаю ключи ssh по алгоритмам rsa с ключём размером 4096 бит и ed25519 (рис. @fig:006), (рис. @fig:007).

```
shakira@fedora:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/shakira/.ssh/id_rsa): /home/shakira/.ssh/id_rsa
Created directory '/home/shakira/.ssh'.
Enter passphrase for '/home/shakira/.ssh/id_rsa' (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/shakira/.ssh/id_rsa
Your public key has been saved in /home/shakira/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:0mXf+t6m7AFfOk0QhFY16zF/ZAHMcYPmRTVZGD1/nEc shakira@fedora
The key's randomart image is:
+---[RSA 4096]-----+
|          o+BB|
|          o+*+|
|          o o+.=|
|         . o . o.+E|
|        . S  ..o@+|
|         . .B.=|
|          . +.|
|          o .o|
|          o*+|
+-----[SHA256]-----+
shakira@fedora:~$
```

Рис. 6: Ключ по алгоритму rsa с ключём размером 4096 бит

```
shakira@fedora:~$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/shakira/.ssh/id_ed25519): /home/shakira/.ssh/id_ed25519
Enter passphrase for "/home/shakira/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/shakira/.ssh/id_ed25519
Your public key has been saved in /home/shakira/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:0qxYbtaAmmP1oI8Cq1NsoWKeiQHENRDk4Q+DQ2b0Ik shakira@fedora
The key's randomart image is:
+--[ED25519 256]--+
|B=o|
|E.=+|
|o+o..|
|.o..|
|... S|
|.o+...|
|O*+ .o+|
|%%.+o...|
|#++oo|
+-----[SHA256]-----+
shakira@fedora:~$
```

Рис. 7: Ключ по алгоритму ed25519

Создание ключа GPG

Создаю ключ gpg: выбираю нужный тип, задаю максимальную длину ключа и выбираю неограниченный срок действия (по умолчанию) (рис. @fig:008).

```
shakira@fedora:~$ 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 '/home/shakira/.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
  <p> = срок действия ключа - п дней
  <p>w = срок действия ключа - п недель
  <p>m = срок действия ключа - п месяцев
  <p>y = срок действия ключа - п лет
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (y/N) y
```

Рис. 8: Создание gpg ключа

Далее отвечаю на вопросы о личной информации и заполняю. После этого требуется ввести фразу-пароль, ввожу (рис. @fig:009).

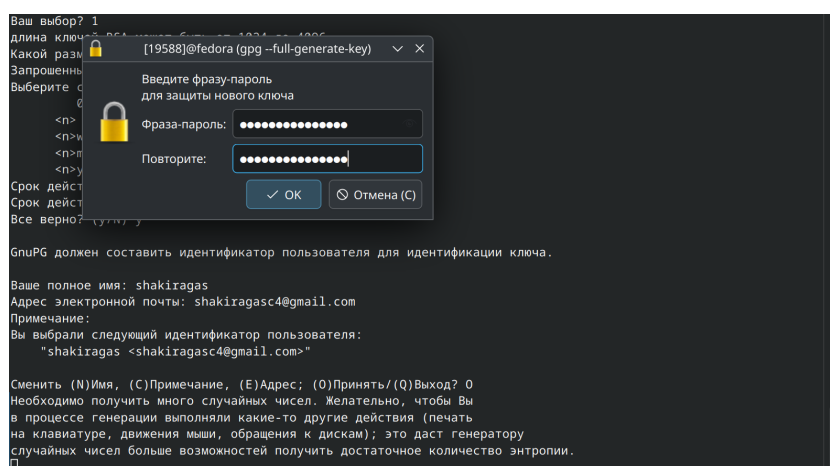


Рис. 9: Заполнение личной информации, ввод фразы-пароля

Регистрация на GitHub

У меня уже есть аккаунт на гитхабе, соответственно все основные данные я уже заполняла. Поэтому просто вхожу в аккаунт (рис. @fig:021).

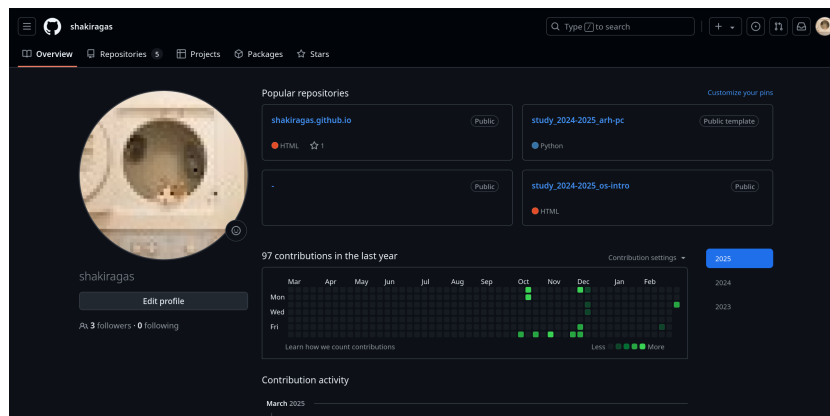


Рис. 10: Вход в аккаунт гитхаба

Добавление ключа GPG в GitHub

Вывожу список ключей и копирую отпечаток приватного ключа (рис. @fig:010).

```
shakira@fedora:~$ 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   rsa4096/7DAAA2F8225038E7 2025-03-04 [SC]
      B87DF0537F076DA6509B533C7DAAA2F8225038E7
uid   [ абсолютно ] shakiragas <shakiragasc4@gmail.com>
ssb   rsa4096/E934A34949C5E5FC 2025-03-04 [E]
shakira@fedora:~$
```

Рис. 11: Копирование отпечатка приватного ключа

Далее копирую сгенерированный PGP ключ в буфер обмена (рис. @fig:022).

```
shakira@fedora:~$ gpg --armor --export 7DAAA2F8225038E7 | xclip -sel clip
```

Рис. 12: Копирование ключа

Перехожу в настройки гитхаба и вставляю скопированный ключ в поле создания gpg ключа (рис. @fig:011).

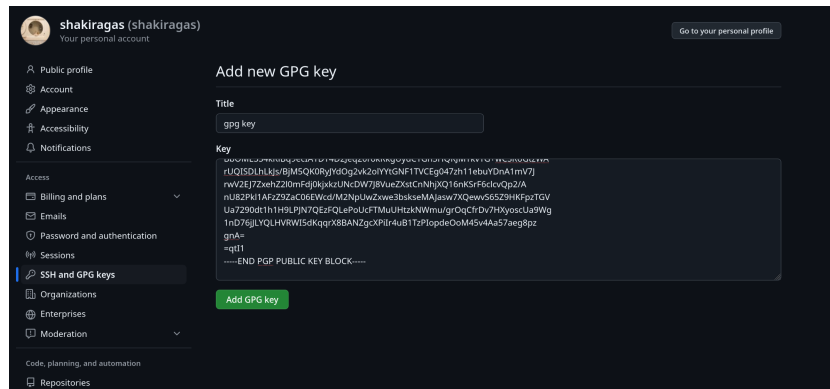


Рис. 13: Создание gpg ключа на гитхаб

Настройка автоматических подписей коммитов git

Настраиваю автоматические подписи коммитов git: используя введенный ранее email, указываю git использовать его при создании подписей коммитов (рис. @fig:012).

```
shakira@fedora: $ gpg --armor --export 7DAAA2F8225038E7 | xclip -sel clip
shakira@fedora: $ git config --global user.signinkey 7DAAA2F8225038E7
shakira@fedora: $ git config --global commit.gpgsign true
shakira@fedora: $ git config --global gpg.program $(which gpg2)
shakira@fedora: $
```

Рис. 14: Настройка подписей git

Настройка gh

Авторизовываюсь в gh, отвечаю на вопрос (рис. @fig:013).

```
shakira@fedora: $ gh auth login
? Where do you use GitHub? [Use arrows to move, type to filter]
  > GitHub.com
  Other
```

Рис. 15: Авторизация на gh

После ответов на все вопросы получаю одноразовый код, который копирую для дальнейших действий (рис. @fig:014).

```
shakirag@fedora:~$ gh auth login
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? HTTPS
? 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: B2D6-17A7
Press Enter to open https://github.com/login/device in your browser...
```

Рис. 16: Копирование одноразового кода

Затем, после нажатия enter, перехожу в браузер на гитхаб, куда ввожу скопированный код для успешной авторизации (рис. @fig:015), (рис. @fig:016).

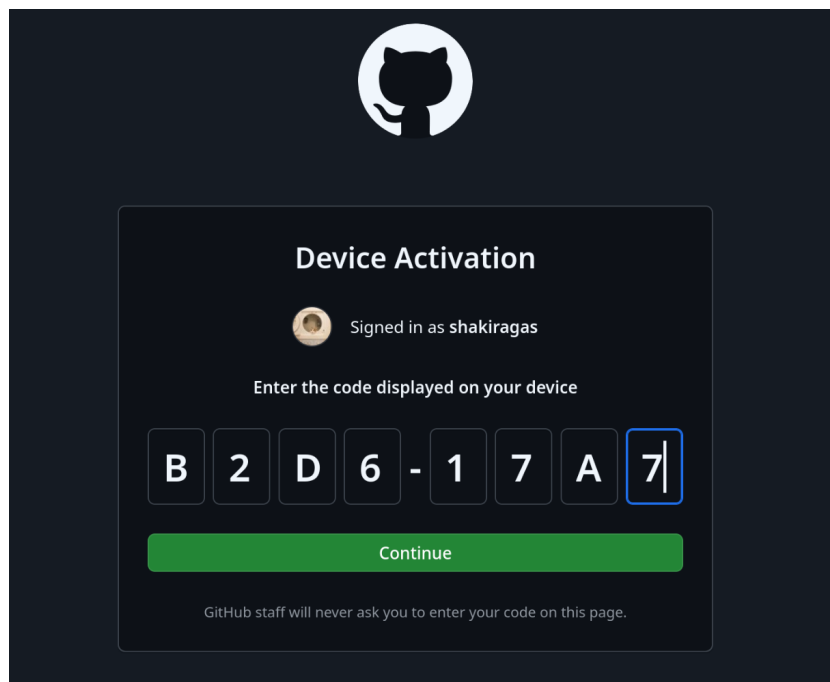


Рис. 17: Завершение авторизации на gh

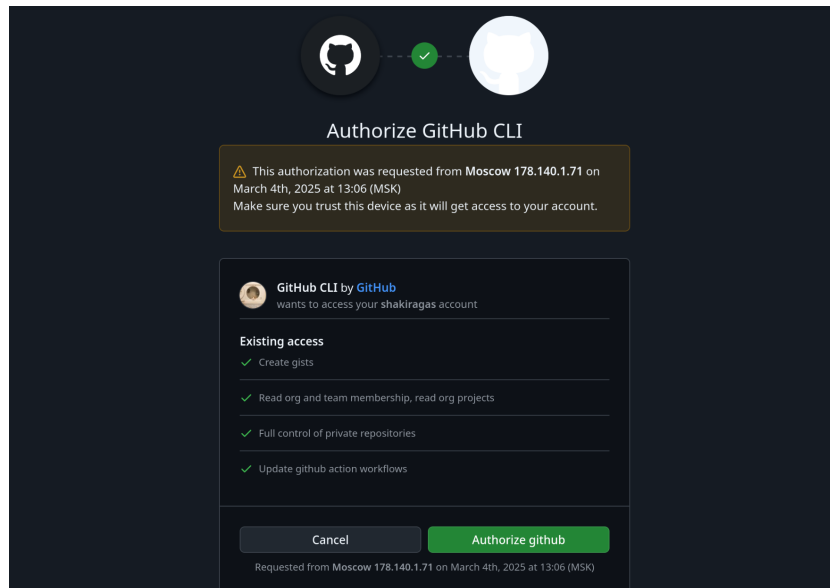


Рис. 18: Завершение авторизации на gh

Создание репозитория курса на основе шаблона

Приступаю к созданию репозитория на основе шаблона. Сначала создаю директорию с помощью утилиты `mkdir`, затем перехожу туда с помощью `cd` (рис. @fig:017).

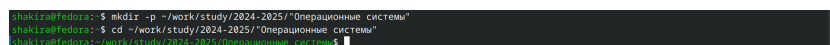


Рис. 19: Создание директории

Далее ввожу команду `gh repo create study_2022-2023_os-intro --template=yamadharm/course-directory-student-template --public`, чтобы создать репозиторий на основе шаблона курса, после чего клонирую его к себе в директорию (рис. @fig:018).


```

shakira@fedora:~/work/study/2024-2025/Операционные системы$ gh repo create study_2024-2025-os-intro --template=yamadharma/course-directory-student-template -- public
accepts at most 1 arg(s), received 2
shakira@fedora:~/work/study/2024-2025/Операционные системы$ git clone --recursive git@github.com:shakiragas/study_2024-2025-os-intro.git os-intro
fatal: репозиторий 'git@github.com: не существует
bash: shakiragas/study_2024-2025-os-intro.git: Not a valid name
shakira@fedora:~/work/study/2024-2025/Операционные системы$ git clone --recursive https://github.com/shakiragas/study_2024-2025-os-intro.git os-intro
Клонирование в os-intro...
remote: Enumerating objects: 36, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 36 (delta 1), reused 21 (delta 0), pack-reused 0 (from 0)
Получение объектов: 100% (36/36), 19.98 KiB | 461.00 KiB/c, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) записан в репозиторий по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) записан в репозиторий по пути «template/report»
Клонирование в «/home/shakira/work/study/2024-2025/Операционные системы/os-intro/template/presentation»...
remote: Enumerating objects: 111, done.
remote: Counting objects: 100% (111/111), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 111 (delta 42), reused 100 (delta 31), pack-reused 0 (from 0)
Получение объектов: 100% (111/111), 102.17 KiB | 1.23 MiB/c, готово.
Определение изменений: 100% (42/42), готово.
Клонирование в «/home/shakira/work/study/2024-2025/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 142, done.
remote: Counting objects: 100% (142/142), done.
remote: Compressing objects: 100% (97/97), done.
remote: Total 142 (delta 68), reused 131 (delta 39), pack-reused 0 (from 0)
Получение объектов: 100% (142/142), 341.09 KiB | 2.35 MiB/c, готово.
Определение изменений: 100% (68/68), готово.
Submodule path «template/presentation»: checked out «c9b2712b4b2d431ad5886c9c72a82bd2fca1d4a6»
Submodule path «template/report»: checked out «c26e22ef7e7b3a84957082ef561ab185f5c748»
shakira@fedora:~/work/study/2024-2025/Операционные системы$

```

Рис. 20: Создание репозитория и клонирование в директорию

Перехожу в каталог курса, удаляю ненужный файл и создаю необходимые каталоги. Для этого требуется установить пакет “make” (рис. @fig:019).

```

shakira@fedora:~/work/study/2024-2025/Операционные системы/os-intro$ rm package.json
shakira@fedora:~/work/study/2024-2025/Операционные системы/os-intro$ echo os-intro > COURSE
shakira@fedora:~/work/study/2024-2025/Операционные системы/os-intro$ make
bash: make: команда не найдена...
Установить пакет «make», предоставляющий команду «make»? [N/y] y

* Ожидание в очереди...
* Загрузка списка пакетов...
Следующие пакеты должны быть установлены:
make-1:4.4-1.fc41.rpm: A GNU tool which simplifies the build process for users
Продолжить с этими изменениями? [N/y] y

* Ожидание в очереди...
* Ожидание аутентификации...
* Ожидание в очереди...
* Загрузка пакетов...
* Запрос данных...
* Проверка изменений...
* Установка пакетов...
Installing...
make: make <target>

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

shakira@fedora:~/work/study/2024-2025/Операционные системы/os-intro$

```

Рис. 21: Удаление файла и создание каталогов

После этого отправляю файлы на сервер с помощью соответствующих команд (рис. @fig:020).

```

shakira@fedora:~/work/study/2024-2025/Операционные системы/os-intro$ echo os-intro > COURSE
shakira@fedora:~/work/study/2024-2025/Операционные системы/os-intro$ make
Usage:
  make <target>

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

shakira@fedora:~/work/study/2024-2025/Операционные системы/os-intro$ git commit -am 'feat(main): make course structure'
[master bc8111b] feat(main): make course structure
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
shakira@fedora:~/work/study/2024-2025/Операционные системы/os-intro$ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 8 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 952 байта | 952.00 КиБ/с, 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 https://github.com/shakiragas/study_2024-2025-os-intro.git
4ac9833..bc8111b master -> master
shakira@fedora:~/work/study/2024-2025/Операционные системы/os-intro$

```

Рис. 22: Отправка файлов на сервер

Выводы

При выполнении данной лабораторной работы я изучила идеологии и применения средств контроля версий и освоение умений по работе с git.

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>