

Отчет по лабораторной работе №2

Операционные системы

Мустафина Аделя Юрисовна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Установка программного обеспечения	7
3.2	Базовая настройка git	7
3.3	Создание ключа ssh	8
3.4	Создание ключа pgr	10
3.5	Настройка github	12
3.6	Добавление PGP ключа в GitHub	13
3.7	Настройка автоматических подписей коммитов git	14
3.8	Настройка gh	14
3.9	Создание репозитория курса на основе шаблона	17
4	Выводы	20
5	Ответы на контрольные вопросы	21
6	Список литературы	25

Список иллюстраций

3.1	Установка	7
3.2	Имя и email	8
3.3	Базовая настройка	8
3.4	Создание ключа ssh по rsa	9
3.5	Создание ключа ssh по ed25519	9
3.6	Генерирование ключа pgr	11
3.7	Фраза-пароль	12
3.8	Создание ключей	12
3.9	Настройка github	13
3.10	Вывод ключей	13
3.11	Копирование ключей	13
3.12	New GPG key	14
3.13	Настройка подписей git	14
3.14	Авторизация в gh	15
3.15	Авторизация на сайте	16
3.16	Завершение авторизации	17
3.17	Завершение авторизации в терминале	17
3.18	Создание репозитория курса	18
3.19	Клонирование репозитория курса	18
3.20	Редактирование созданного репозитория	18
3.21	Отправка файлов на сервер	19
3.22	Отправка файлов	19

Список таблиц

1 Цель работы

Изучить идеологию и применение средств контроля версий и 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 Выполнение лабораторной работы

3.1 Установка программного обеспечения

Устанавливаю программное обеспечение через терминал с помощью команд `dnf install git` и `dnf install gh` (рис. 3.1).

```
[aymustafina@aymustafina ~]$ 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 команду без изменения состояния системы.
[aymustafina@aymustafina ~]$ sudo -i
[sudo] пароль для аymustafina:
[root@aymustafina ~]# dnf install git
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет "git-2.48.1-1.fc41.x86_64" уже установлен.

Нечего делать.
[root@aymustafina ~]# dnf install gh
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет "gh-2.65.0-1.fc41.x86_64" уже установлен.

Нечего делать.
[root@aymustafina ~]#
```

Рис. 3.1: Установка

3.2 Базовая настройка git

Задаю имя и email владельца репозитория с помощью команд: `git config --global user.name "Name Surname"` `git config --global user.email "work@mail"` (рис. 3.2).

```
Пакет "gh-2.65.0-1.fc41.x86_64" уже установлен.  
  
Нечего делать.  
[root@aymustafina ~]# git config --global user.name "Adelya Mustafina"  
[root@aymustafina ~]# git config --global user.email "aliyamstfn@gmail.com"
```

Рис. 3.2: Имя и email

Настроим utf-8 в выводе сообщений git: `git config --global core.quotepath false`

Зададим имя начальной ветки (будем называть её master): `git config --global init.defaultBranch master`

Параметр `autocrlf`: `git config --global core.autocrlf input`

Параметр `safecrlf`: `git config --global core.safecrlf warn` (рис. 3.3).

```
Нечего делать.  
[root@aymustafina ~]# git config --global user.name "Adelya Mustafina"  
[root@aymustafina ~]# git config --global user.email "aliyamstfn@gmail.com"  
[root@aymustafina ~]# git config --global core.quotepath false  
[root@aymustafina ~]# git config --global init.defaultBranch master  
[root@aymustafina ~]# git config --global core.autocrlf input  
[root@aymustafina ~]# git config --global core.safecrlf warn
```

Рис. 3.3: Базовая настройка

3.3 Создание ключа ssh

по алгоритму `rsa` с ключём размером 4096 бит: `ssh-keygen -t rsa -b 4096` (рис. 3.4).


```

[root@aymustafina ~]# 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:vMx4DQfaqVHM51EUfifXNMB5ikzh+2C6oj1GzLi2XwE root@aymustafina
The key's randomart image is:
+---[RSA 4096]-----+
|           o*oo..|
|          o  .+ o oo|
|         E= =.o = +|
|        =.= +.o + |
|       * S.o+    |
|      . X =+ o   |
|     = =o. .    |
|    o.=. .      |
|   .o=oo.      |
+-----[SHA256]-----+
[root@aymustafina ~]#

```

Рис. 3.4: Создание ключа ssh по rsa

по алгоритму ed25519: `ssh-keygen -t ed25519` (рис. 3.5).

```

[root@aymustafina ~]# 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:FnVwLA4VGS7Q0S6Q5ZXAHV9bDzPiR2PHFKewf0vGACc root@aymustafina
The key's randomart image is:
+--[ED25519 256]--+
| .+*OE=.O+*|
| oo++*==B X+|
| .++..+.+.|
| .oo oo |
| S. .+.|
| . o..|
| . |
| |
+-----[SHA256]-----+
[root@aymustafina ~]#

```

Рис. 3.5: Создание ключа ssh по ed25519

3.4 Создание ключа `pgp`

Генерирую ключ `gpg --full-generate-key`

Из предложенных опций выбираю: тип RSA and RSA; размер 4096; выбрала срок действия; значение по умолчанию — 0 (срок действия не истекает никогда). GPG запросит личную информацию, которая сохранится в ключе: Имя. Адрес электронной почты. При вводе email убеждаюсь, что он соответствует адресу, используемому на GitHub. Комментарий. Нажимаю клавишу ввода, чтобы оставить это поле пустым. (рис. 3.6).

```

[aliy@aliy]# 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>w = срок действия ключа - n недель
  <n>m = 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

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: Adelya Mustafina
Адрес электронной почты: aliyamstfn@gmail.com
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Adelya Mustafina <aliyamstfn@gmail.com>"

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 (печатать

```

Рис. 3.6: Генерирование ключа gpg

Ввожу фразу-пароль для защиты ключа (рис. 3.7).

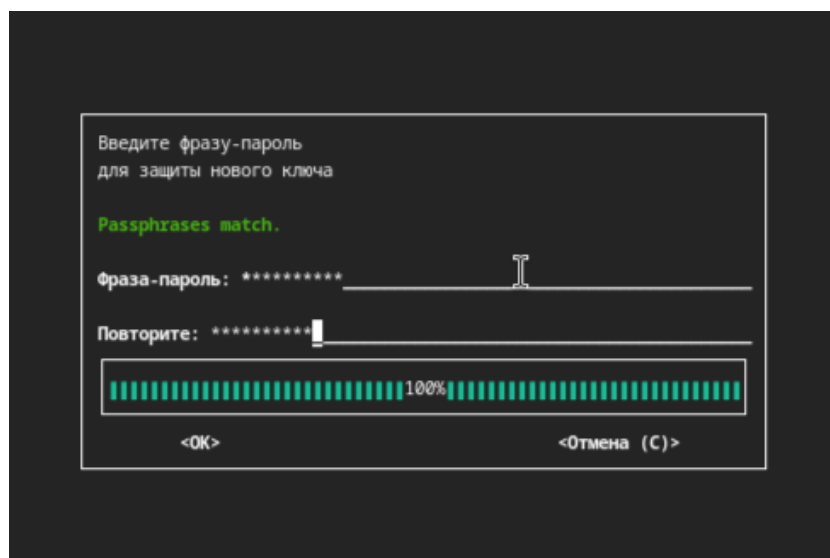


Рис. 3.7: Фраза-пароль

Открытый и секретный ключ созданы (рис. 3.8).

```

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/D751C72AB78BA87B691E
44ED7D8ACD846906604E.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2025-03-02 [SC]
       D751C72AB78BA87B691E44ED7D8ACD846906604E
uid           Adelya Mustafina <aliyamstfn@gmail.com>
sub   rsa4096 2025-03-02 [E]

[root@aymustafina ~]#

```

Рис. 3.8: Создание ключей

3.5 Настройка github

У меня уже был создан аккаунт на github, основные данные аккаунта заполнены (рис. 3.9).

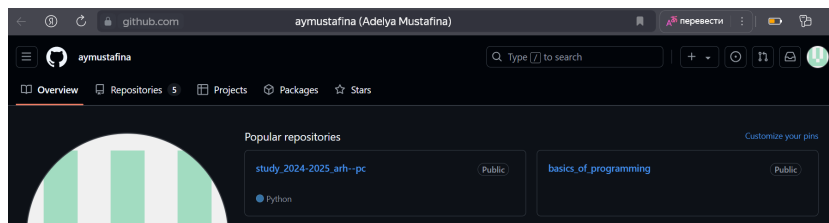


Рис. 3.9: Настройка github

3.6 Добавление PGP ключа в GitHub

Вывожу список ключей и копирую отпечаток приватного ключа: `gpg --list-secret-keys --keyid-format LONG`

Отпечаток ключа — это последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа. (рис. 3.10).

```
[root@aymustafina ~]# 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: 2 подписанных: 0 доверие: 0-,
0q, 0n, 0m, 0f, 2u
[keyboard]
-----
sec   rsa4096/BC82ED8D50F07E5A 2025-03-02 [SC]
      ACDA5D9B96A8FC41F8D33B28BC82ED8D50F07E5A
uid           [ абсолютно ] Adelya Mustafina <aliyamstfn@gmail.com>
ssb   rsa4096/1251AB79DF88E459 2025-03-02 [E]
```

Рис. 3.10: Вывод ключей

Формат строки:

sec Алгоритм/Отпечаток_ключа Дата_создания [Флаги] [Годен_до] ID_ключа
Копирую сгенерированный PGP ключ в буфер обмена: `gpg --armor --export | xclip -sel cli` (рис. 3.11).

```
-bash: xclip: команда не найдена
[root@aymustafina ~]# gpg --armor --export BC82ED8D50F07E5A | xclip -sel clip
```

Рис. 3.11: Копирование ключей

Перехожу в настройки Github, нажимаю кнопку New GPG key и вставляю полученный ключ в поле ввода (рис. 3.12).

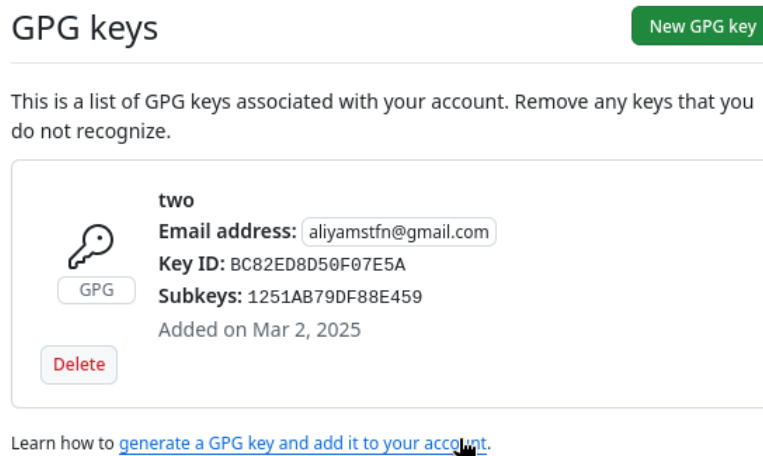


Рис. 3.12: New GPG key

3.7 Настройка автоматических подписей коммитов git

Используя введённый email, укажите Git применять его при подписи коммитов:
git config --global user.signingkey git config --global commit.gpgsign true git config --global gpg.program \$(which gpg2) (рис. 3.13).

```
[root@aymustafina ~]# git config --global user.signingkey BC82ED8D50F07E5A
[root@aymustafina ~]# git config --global commit.gpgsign true
[root@aymustafina ~]# git config --global gpg.program $(which gpg2)
[root@aymustafina ~]# gh auth login
```

Рис. 3.13: Настройка подписей git

3.8 Настройка gh

Сначала аворизуюсь в gh, отвечаю на наводящие вопросы, в конце выбираю авторизацию через браузер (рис. 3.14).

```

[aymustafina@aymustafina ~]$ gh auth status
You are not logged into any GitHub hosts. To log in, run: gh auth login
[aymustafina@aymustafina ~]$ sudo -i
[sudo] пароль для аymustafina:
[root@aymustafina ~]# ls ~/.ssh/id_ed25519.pub
/root/.ssh/id_ed25519.pub
[root@aymustafina ~]# cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIP6MSzUHKw7vFXhur5sS2QVjybWxA8AB+B1zZa9+15Y root@aymustafina
[root@aymustafina ~]# ssh -T git@github.com
Hi aymustafina! You've successfully authenticated, but GitHub does not provide shell access.
[root@aymustafina ~]# gh auth login --web
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /root/.ssh/id_ed25519.pub
? Title for your SSH key: new

! First copy your one-time code: 3A34-99D3
Press Enter to open https://github.com/login/device in your browser...

```

Рис. 3.14: Авторизация в gh

Завершаю авторизацию на сайте (рис. 3.15).

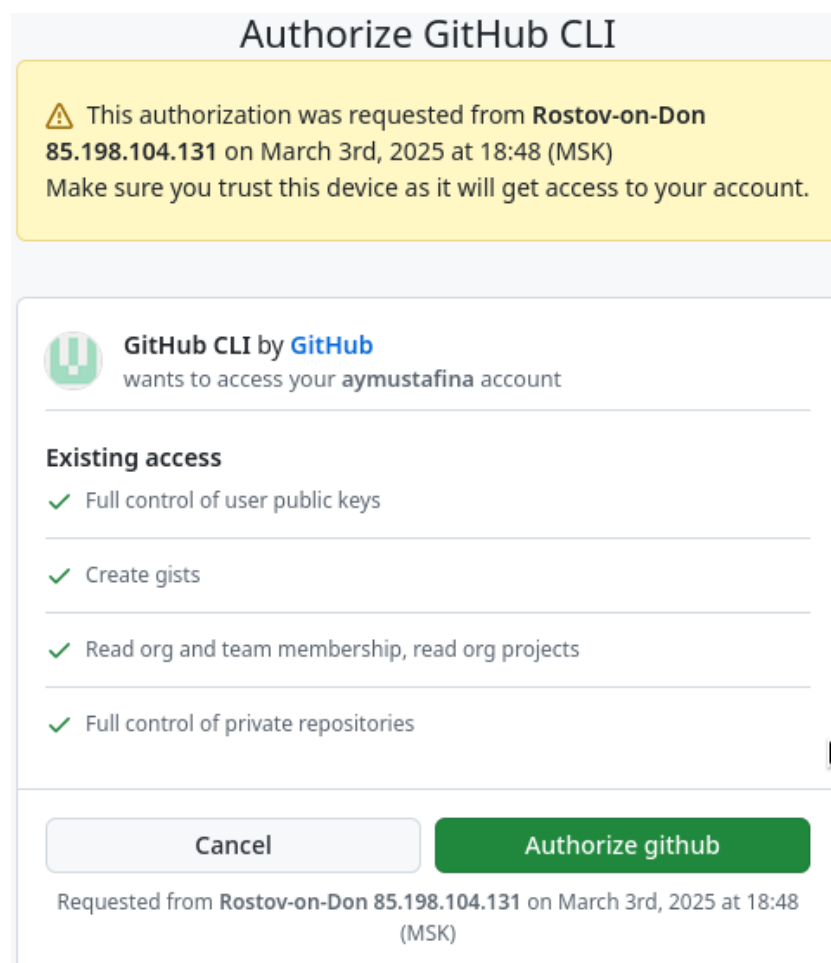


Рис. 3.15: Авторизация на сайте

Вижу сообщение о завершении авторизации на сайте (рис. 3.16).

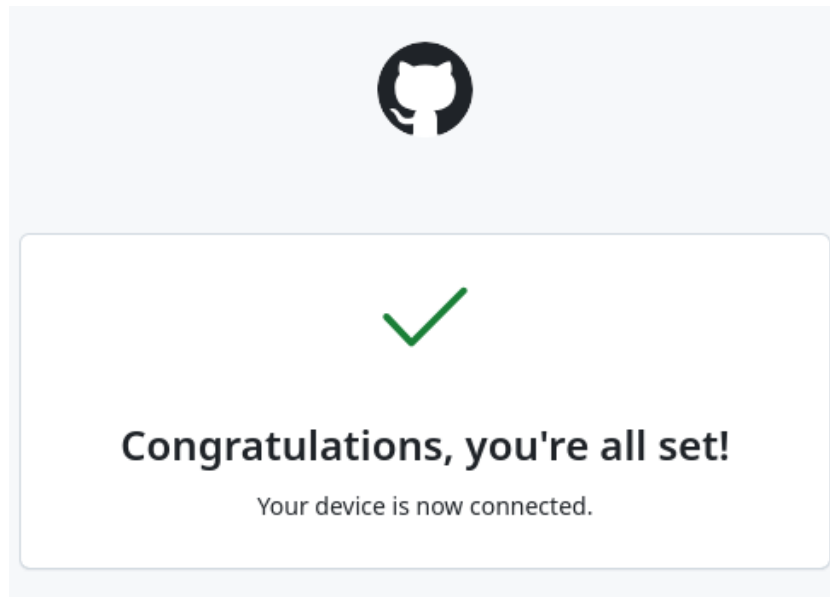


Рис. 3.16: Завершение авторизации

Видю сообщениe о завершении авторизации в терминале (рис. 3.17).

```
Press Enter to open https://github.com/login/device in your browser...
Authorization required, but no authorization protocol specified

Error: cannot open display: :0
✓ Authentication complete.
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
! Authentication credentials saved in plain text
✓ SSH key already existed on your GitHub account: /root/.ssh/id_ed25519.pub
✓ Logged in as aymustafina
[root@aymustafina ~]#
```

Рис. 3.17: Завершение авторизации в терминале

3.9 Создание репозитория курса на основе шаблона

Сначала создаю директорию с помощью утилиты `mkdir` и с помощью `cd` перехожу в созданную директорию:

```
mkdir -p ~/work/study/2024-2025/“Операционные системы” cd ~/work/study/2024-2025/“Операционные системы” gh repo create study_2024-2025_os-intro --template=yamadharma/course-directory-student-template --public git clone --recursive git@github.com:/study_2024-2025_os-intro.git os-intro (рис. 3.18).
```

```
[root@aymustafina ~]# mkdir -p ~/work/study/2024-2025/"Операционные системы"
[root@aymustafina ~]# cd ~/work/study/2024-2025/"Операционные системы"
[root@aymustafina Операционные системы]# gh repo create study_2022-2023_os-intro --template=yamadharma/course-directory-student-template --public
Created repository aymustafina/study_2022-2023_os-intro on GitHub
https://github.com/aymustafina/study_2022-2023_os-intro
[root@aymustafina Операционные системы]# gh repo create study_2024-2025_os-intro --template=yamadharma/course-directory-student-template --public
Created repository aymustafina/study_2024-2025_os-intro on GitHub
https://github.com/aymustafina/study_2024-2025_os-intro
```

Рис. 3.18: Создание репозитория курса

Клонирование репозитория (рис. 3.19).

```
[root@aymustafina Операционные системы]# git clone --recursive git@github.com:aymustafina/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% (35/35), done.
remote: Total 36 (delta 1), reused 21 (delta 0), pack-reused 0 (from 0)
Получение объектов: 100% (36/36), 19.38 Кб | 1.49 Мб/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»...
Клонирование в «/root/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 Кб | 1.04 Мб/с, готово.
Определение изменений: 100% (42/42), готово.
Клонирование в «/root/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 60), reused 121 (delta 39), pack-reused 0 (from 0)
Получение объектов: 100% (142/142), 341.09 Кб | 1.79 Мб/с, готово.
Определение изменений: 100% (60/60), готово.
Submodule path 'template/presentation': checked out 'c9b2712b4b2d431ad5086c9c72a82bd2fca1d4a6'
Submodule path 'template/report': checked out 'c26e22effe7b3e84957d7d82ef561ab185f5c748'
[root@aymustafina Операционные системы]#
```

Рис. 3.19: Клонирование репозитория курса

Перехожу в каталог курса с помощью `cd`. Удаляю лишние файлы `rm package.json`.
Создаю необходимые каталоги: `echo os-intro > COURSE` и `make` (рис. 3.20).

```
[root@aymustafina Операционные системы]# cd ~/work/study/2024-2025/"Операционные системы"/os-intro
[root@aymustafina os-intro]# rm package.json
rm: удалить обычный файл 'package.json'? yes
[root@aymustafina os-intro]# echo os-intro > COURSE
[root@aymustafina os-intro]# make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare        Generate directories structure
  submodule      Update submodules
[root@aymustafina os-intro]#
```

Рис. 3.20: Редактирование созданного репозитория

Добавляю все новые файлы для отправки на сервер и комментирую их: `git add`.
`git commit -am 'feat(main): make course structure'` (рис. 3.21).

```
[root@aymustafina os-intro]# git add .
[root@aymustafina os-intro]# git commit -am 'feat(main): make course structure'
[master 41fd89a] feat(main): make course structure
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
```

Рис. 3.21: Отправка файлов на сервер

Отправляю файлы на сервер с помощью git push (рис. 3.22).

```
[root@aymustafina os-intro]# git push
Перечисление объектов: 39, готово.
Подсчет объектов: 100% (39/39), готово.
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 342.28 КиБ | 562.00 КиБ/с, готово.
Total 38 (delta 4), reused 1 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:aymustafina/study_2024-2025_os-intro.git
 41fd89a..493dd71 master -> master
[root@aymustafina os-intro]#
```

Рис. 3.22: Отправка файлов

4 Выводы

При выполнении данной лабораторной работы я изучила идеологию и применение средств контроля версий, изучила команды для работы с git.

5 Ответы на контрольные вопросы

- Что такое системы контроля версий (VCS) и для решения каких задач они предназначены?

Системы контроля версий(VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Они помогают разработчикам отслеживать изменения и управлять версиями кода. Основная цель VCS - упростить и упорядочить работу над проектом. С его помощью можно легко отслеживать, какие изменения были внесены в проект и кем. Это помогает быстро находить и исправлять ошибки, анализировать причины их возникновения.

- Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

Хранилище - это репозиторий, хранилище версий. В нем хранятся документы и их история изменения. Commit - отслеживание редактирования файла и его изменений. Сохранение разницы между версиями. История - хранение всех изменений в проекте, благодаря чему возможно обратиться к данным, которые были изменены. Рабочая копия - копия проекта, которая используется в данный момент. Обычно последняя измененная версия копия.

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

Централизованные системы контроля версий предполагают хранение проекта на едином сервере. Так чтобы изменить исходный код, разработчику необходимо

скачать с сервера необходимые файлы и изменить, а затем снова вернуть эти файлы на сервер. Примеры централизованных VCS: CVS, Subversion, Perforce.

Децентрализованные системы контроля версий предполагают, что при каждом копировании удаленного репозитория происходит полное копирование данных в локальный репозиторий. И каждая новая копия хранит все данные, хранящиеся в удаленном репозитории. Примеры децентрализованных VCS: Git, Basaar, Mercurial.

- Опишите действия с VCS при единоличной работе с хранилищем.

Создается и подключается удаленный репозиторий. Далее при изменении проекта новые изменения отправляются на сервер.

- Опишите порядок работы с общим хранилищем VCS.

Сначала разработчик клонирует нужный ему удаленный репозиторий. Создает ветку для работы. После внесения каких-либо изменений в код, разработчик отправляет изменения в удаленный репозиторий. При этом все внесенные изменения сохраняются и к ним можно вернуться в любой момент. - Каковы основные задачи, решаемые инструментальным средством git?

управление версиями файлов
отслеживание изменений
резервное копирование и восстановление данных

- Назовите и дайте краткую характеристику командам git.

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`

- Приведите примеры использования при работе с локальным и удалённым репозиториями.

Для локального репозитория: `git init` - создание нового репозитория `git add .` - добавление всех файлов в индекс `git commit -m "Initial commit"` - фиксируем изменения

Для удаленного репозитория: `git clone` - клонирование репозитория `git push origin main` - отправка изменений в удаленный репозиторий `git pull` - получение изменений из удаленного репозитория

- Что такое и зачем могут быть нужны ветви (branches)?

Отдельные линии разработки, позволяющие работать над разными задачами независимо. Нужны для изоляции изменений, тестирования, исправления багов без влияния на основную ветку.

- Как и зачем можно игнорировать некоторые файлы при commit?

Создается файл `.gitignore`, в котором указываются шаблоны файлов/папок, которые следует игнорировать. Это делается для того, чтобы не включать в репозиторий временные файлы, бинарные данные, конфиденциальную информацию, файлы, создаваемые средой разработки.

6 Список литературы

1. Лабораторная работа №2 [Электронный ресурс]. URL: <https://esystem.rudn.ru/mod/page/view.php?id=1224371>