

# **Отчет по лабораторной работе №2**

**Операционные системы**

Курилко-Рюмин Евгений Михайлович

# 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 программного обеспечения . . . . .	9
4.2	Базовая настройка git . . . . .	9
4.3	Создание SSH ключа . . . . .	10
4.4	Создание ключа GPG . . . . .	11
4.5	Добавление ключа GPG в Github . . . . .	13
4.6	Настройка gh . . . . .	15
<b>5</b>	<b>Выводы</b>	<b>19</b>
<b>6</b>	<b>Ответы на контрольные вопросы.</b>	<b>20</b>
	<b>Список литературы</b>	<b>23</b>

## Список иллюстраций

4.1	Установка git и gh . . . . .	9
4.2	Настройка git . . . . .	10
4.3	Настройка utf-8 . . . . .	10
4.4	Задача имени для начальной ветки . . . . .	10
4.5	Задача параметров autocrlf и safecrlf . . . . .	10
4.6	Генерация ssh ключа по алгоритму rsa . . . . .	11
4.7	Генерация ssh ключа по алгоритму ed25519 . . . . .	11
4.8	Генерация ключа . . . . .	12
4.9	Защита ключа GPG . . . . .	12
4.10	Учетная запись Github . . . . .	13
4.11	Вывод списка ключей . . . . .	13
4.12	Копирование ключа в буфер обмена . . . . .	14
4.13	Настройки GitHub . . . . .	14
4.14	Добавление нового PGP ключа . . . . .	14
4.15	Добавленный ключ GPG . . . . .	15
4.16	Настройка подписей коммитов git . . . . .	15
4.17	Авторизация в gh . . . . .	15
4.18	Завершение авторизации через браузер . . . . .	16
4.19	Завершение авторизации . . . . .	16
4.20	Создание репозитория . . . . .	17
4.21	Переход в каталог . . . . .	17
4.22	Удаление лишних файлов . . . . .	17
4.23	подготовка файлов к отправлению на сервер . . . . .	17
4.24	отправление файлов на сервер . . . . .	18

## **Список таблиц**

# 1 Цель работы

Цель лабораторной работы заключается в изучение применения средств контроля версий, а также освоение умения по работе с git.

## 2 Задание

1. Создание базовой конфигурацию для работы с git
2. Создание SSH ключ
3. Создание GPG ключ
4. Настройка подписи Git
5. Регистрация на GitHub
6. Создание локального каталога 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 программного обеспечения

Устанавливаю необходимое программное обеспечение git и gh через терминал с помощью команд: `dnf install git` и `dnf install gh` (рис. fig. 4.1).

```
=====
Установка 1 Пакет

Объем загрузки: 8.9 М
Объем изменений: 44 М
Загрузка пакетов:
gh-2.36.0-1.fc38.x86_64.rpm      8.2 MB/s | 8.9 MB   00:01
-----
Общий размер                    5.2 MB/s | 8.9 MB   00:01
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно.
Выполнение транзакции
  Подготовка      :                               1/1
  Установка       : gh-2.36.0-1.fc38.x86_64      1/1
  Запуск скрипта  : gh-2.36.0-1.fc38.x86_64      1/1
  Проверка        : gh-2.36.0-1.fc38.x86_64      1/1

Установлен:
gh-2.36.0-1.fc38.x86_64

Выполнено!
[emkurilkoryumin@fedora ~]$
```

Рис. 4.1: Установка git и gh

### 4.2 Базовая настройка git

Задаю в качестве имени владельца репозитория и его email свое имя, фамилию и почту (рис. fig. 4.2)

```
[emkurilkoryumin@fedora ~]$ git config --global user.name "emkurilkoryumin"
[emkurilkoryumin@fedora ~]$ git config --global user.email "1132232883@pfur.ru"
```

Рис. 4.2: Настройка git

Настраиваю utf-8 в выводе сообщений git для корректного отображения (рис. fig. 4.3)

```
[emkurilkoryumin@fedora ~]$ git config --global core.quotepath false
```

Рис. 4.3: Настройка utf-8

Задаю имя для начальной ветки “master” (рис. fig. 4.4)

```
[emkurilkoryumin@fedora ~]$ git config --global init.defaultBranch master
```

Рис. 4.4: Задача имени для начальной ветки

Далее задаю параметры для autocrlf и safecrlf для корректного отображения конца строки (рис. fig. 4.5)

```
[emkurilkoryumin@fedora ~]$ git config --global core.autocrlf input
[emkurilkoryumin@fedora ~]$ git config --global core.safecrlf warn
```

Рис. 4.5: Задача параметров autocrlf и safecrlf

## 4.3 Создание SSH ключа

Создаю SSH ключ размером 4096 бит по алгоритму RSA (рис. fig. 4.6)

```

[emkurilkoryumin@fedora ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/emkurilkoryumin/.ssh/id_rsa):
/home/emkurilkoryumin/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/emkurilkoryumin/.ssh/id_rsa
Your public key has been saved in /home/emkurilkoryumin/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:gv4gKIdPGA3QqGUQEyPsfnoXt6kYuU25WcAsbEXZFW0 emkurilkoryumin@fedora
The key's randomart image is:
+---[RSA 4096]-----+
|X*  .o .oo |
|+++ .. . E |
|+o  .  . |
|.+. +. |
|o .+.+. S |
| *.oo.oo |
|= +oooo.o |
|,= o+++o |
| oo.=o |
+-----[SHA256]-----+

```

Рис. 4.6: Генерация ssh ключа по алгоритму rsa

Создаю ключ ssh по алгоритму ed25519 (рис. fig. 4.7).

```

[emkurilkoryumin@fedora ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/emkurilkoryumin/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/emkurilkoryumin/.ssh/id_ed25519
Your public key has been saved in /home/emkurilkoryumin/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:cT4nCBXw/mTZcGeZx3QGg1YIe7WVG34KAkGKvU36l0c emkurilkoryumin@fedora
The key's randomart image is:
+--[ED25519 256]--+
|      .o=0.. +=.* |
|      o +.  .+. %. |
|      . + +.+.o B =|
|      B =.*.o +. |
|      o $ B.E. .. |
|      . + * . |
|      . + . |
|      . . |
+-----[SHA256]-----+

```

Рис. 4.7: Генерация ssh ключа по алгоритму ed25519

## 4.4 Создание ключа GPG

Генерирую ключ GPG, затем выбираю тип ключа RSA и RSA, задаю максимальную длину ключа: 4096, оставляю неограниченный срок действия ключа. Далее отвечаю на вопросы программы о личной информации (рис. fig. 4.8).

```
[emkurilkoryumin@fedora ~]$ gpg --full-generate-key
gpg (GnuPG) 2.4.0; Copyright (C) 2021 Free Software Foundation, Inc.
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/emkurilkoryumin/.gnupg'
gpg: создан щит с ключами '/home/emkurilkoryumin/.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) 0
Срок действия ключа не ограничен
Все верно? (y/N) y

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: emkurilkoryumin
Адрес электронной почты: 1132232883@pfur.ru
```

Рис. 4.8: Генерация ключа

Ввожу фразу-пароль для защиты нового ключа (рис. fig. 4.9).

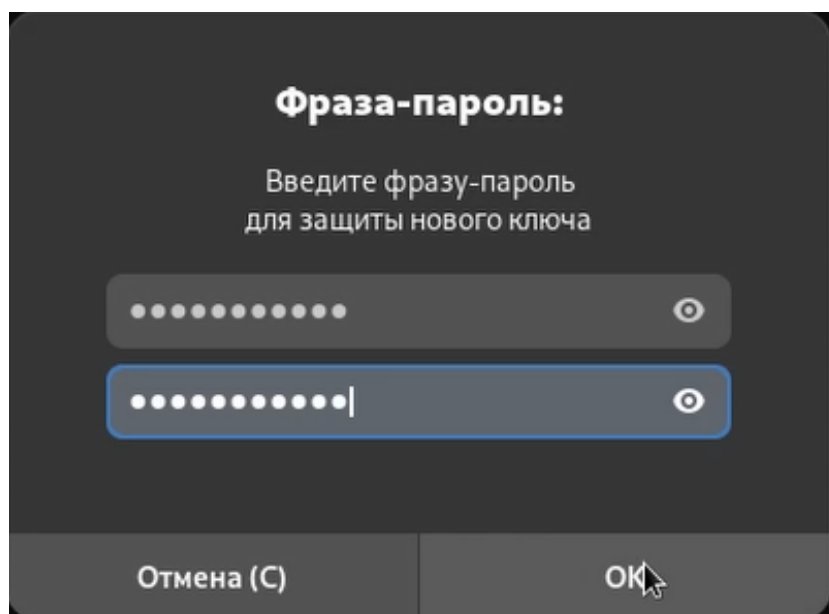


Рис. 4.9: Защита ключа GPG

## ##Создание учётной записи на Github

Так как у меня уже был создан аккаунт на Github,то основные данные аккаунта и его настройка уже была проведена мной,поэтому просто захожу в свой аккаунт

(рис. fig. 4.10).

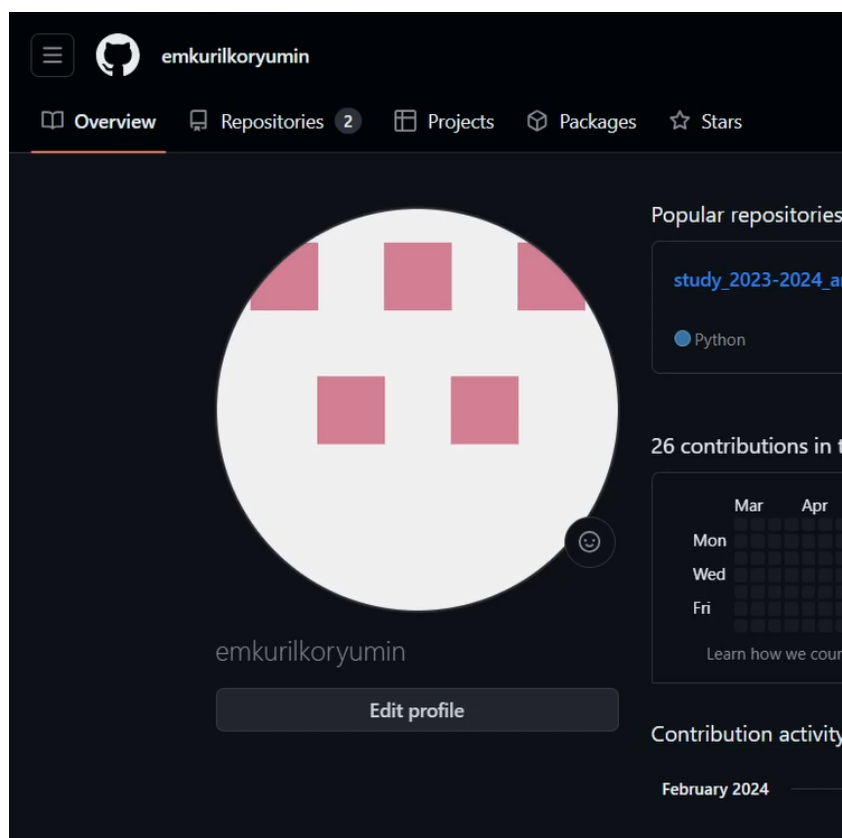


Рис. 4.10: Учетная запись Github

## 4.5 Добавление ключа GPG в Github

Вывожу список созданных ключей в терминал, ищу последовательность байтов для идентификации более длинного, по сравнению с самим отпечатком ключа, копирую его в буфер обмена (рис. fig. 4.11).

```
[emkurilkoryumin@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
/home/emkurilkoryumin/.gnupg/pubring.kbx
-----
sec   rsa4096/10630A924D351AB1 2024-02-26 [SC]
      D170F0A5D8748DC5B592BAD110630A924D351AB1
uid   [ абсолютно ] emkurilkoryumin (...) <1132232883@pfur.ru>
ssb   rsa4096/FBF947D1F5D7213F 2024-02-26 [E]
```

Рис. 4.11: Вывод списка ключей

Ввожу в терминале команду, с помощью которой копирую сам ключ GPG в буфер обмена, за это отвечает утилита xclip (рис. fig. 4.12).

```
[emkuril@koryumin@fedora ~]$ gpg --armor --export D170F0A5DB748DC5B5928AD110638A924D351AB1 | xclip -sel clip
```

Рис. 4.12: Копирование ключа в буфер обмена

Открываю настройки GitHub, ищу среди них добавление GPG ключа (рис. fig. 4.13).

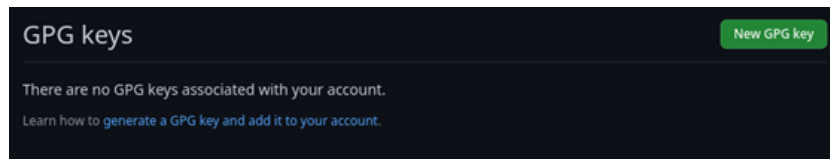


Рис. 4.13: Настройки GitHub

Нажимаю на “New GPG key” и вставляю в поле ключ из буфера обмена (рис. fig. 4.14).

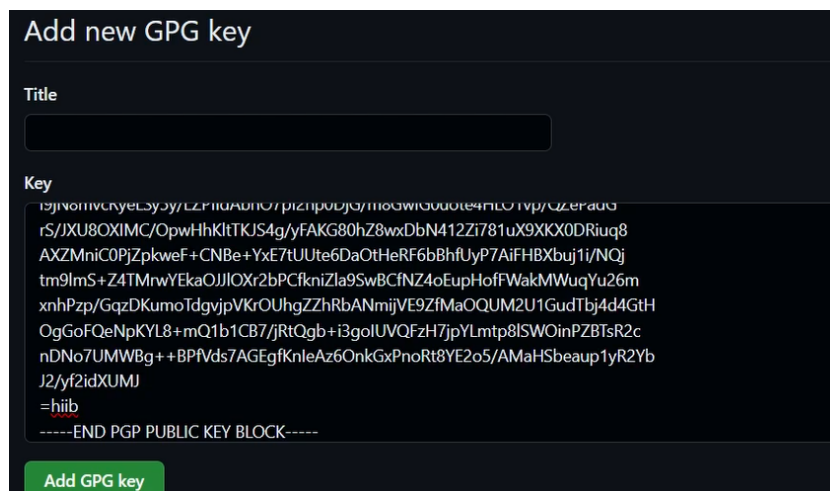


Рис. 4.14: Добавление нового PGP ключа

Ключ GPG добавлен на GitHub (рис. fig. 4.15).

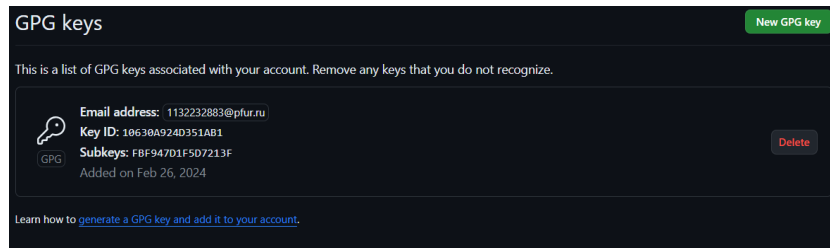


Рис. 4.15: Добавленный ключ GPG

## ##Настройка автоматических подписей коммитов git

Настраиваем автоматическую подпись коммитов git используя введенный ранее email,указываем git который буду использовать при создании подписей коммитов (рис. fig. 4.16).

```
[emkurilkoryumin@fedora ~]$ git config --global user.signingkey D170F0A5DB748DC5B592BAD110630A924D351AB1
[emkurilkoryumin@fedora ~]$ git config --global commit.gpgsign true
[emkurilkoryumin@fedora ~]$ git config --global gpg.program $(which gpg2)
```

Рис. 4.16: Настройка подписей коммитов git

## 4.6 Настройка gh

Начинаю авторизацию в gh, отвечаю на наводящие вопросы от утилиты, в конце выбираю авторизоваться через браузер (рис. fig. 4.17).

```
[emkurilkoryumin@fedora ~]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser
```

Рис. 4.17: Авторизация в gh

Завершаю авторизацию на сайте (рис. fig. 4.18).

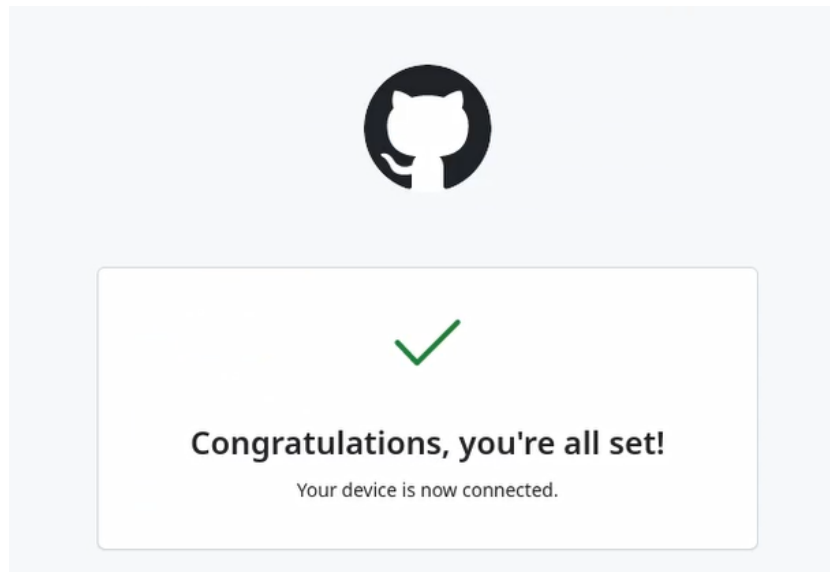


Рис. 4.18: Завершение авторизации через браузер

Видю сообщение о завершении авторизации, в доказательство выводится мое имя emkurilkorymin (рис. fig. 4.19).

```
✓ Authentication complete.  
- gh config set -h github.com git_protocol https  
✓ Configured git protocol  
✓ Logged in as emkurilkorymin
```

Рис. 4.19: Завершение авторизации

#### ##Создание репозитория курса на основе шаблона

Создаю директорию с помощью утилиты `mkdir` и флага `-p`, который позволяет установить каталоги на всем указанном пути. После этого с помощью утилиты `cd` перехожу в созданную директорию. Далее в терминале ввожу команду `gh repo create study_2023-2024_os-intro --template yamadharma/course-directory-student-trmplate --public`, чтобы создать репозиторий на основе шаблона репозитория. После этого клонирую репозиторий к себе в директорию (рис. fig. 4.20).



```
[emkurilkoryumin@fedora Операционные системы]$ git clone --recursive https://github.com/emkurilkoryumin/stude_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 | 312.00 КиБ/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»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/emkurilkoryumin/work/study/2023-2024/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.
Получение объектов: 100% (95/95), 96.99 КиБ | 1023.80 КиБ/с, готово.
Определение изменений: 100% (34/34), готово.
remote: Total 95 (delta 34), reused 87 (delta 26), pack-reused 0
Клонирование в «/home/emkurilkoryumin/work/study/2023-2024/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 126, done.
remote: Counting objects: 100% (126/126), done.
remote: Compressing objects: 100% (87/87), done.
remote: Total 126 (delta 52), reused 108 (delta 34), pack-reused 0
Получение объектов: 100% (126/126), 335.80 КиБ | 1.50 МБ/с, готово.
Определение изменений: 100% (52/52), готово.
submodule path 'template/presentation': checked out '40a1761813e197d00e88443ffica71c66a204f24c'
submodule path 'template/report': checked out '7c31ab8e5dfa8cd2d67caeb8a19ef8028ced88e'
[emkurilkoryumin@fedora Операционные системы]$
```

Рис. 4.20: Создание репозитория

Дальше перехожу в каталог курса применяя утилиту `cd` (рис. fig. 4.21).

```
[emkurilkoryumin@fedora Операционные системы]$ cd ~/work/study/2023-2024/"Операционные системы"/os-intro
[emkurilkoryumin@fedora os-intro]$
```

Рис. 4.21: Переход в каталог

Удаляем лишние файлы с помощью утилиты `rm`, затем создаю каталог используя `makefile` (рис. fig. 4.22).

```
[emkurilkoryumin@fedora os-intro]$ rm package.json
rm: невозможно удалить 'package.json': Нет такого файла или каталога
[emkurilkoryumin@fedora os-intro]$ echo os-intro > COURSE
[emkurilkoryumin@fedora os-intro]$ make
```

Рис. 4.22: Удаление лишних файлов

С помощью команд `git add` и `git commit`, добавляю новые файлы и комментирую их для отправки на сервер (рис. fig. 4.23).

```
[emkurilkoryumin@fedora os-intro]$ git add .
```

Рис. 4.23: подготовка файлов к отправлению на сервер

Отправляю файлы на сервер (рис. fig. 4.24).

```
[emkurilkoryumin@fedora os-intro]$ rm package.json
[emkurilkoryumin@fedora os-intro]$ echo os-intro > COURSE
make
Usage:
  make <target>

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

[emkurilkoryumin@fedora os-intro]$ echo os-intro > COURSE
[emkurilkoryumin@fedora os-intro]$ make
Usage:
  make <target>

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

[emkurilkoryumin@fedora os-intro]$
```

Рис. 4.24: отправление файлов на сервер

## 5 Выводы

В ходе выполнения лабораторной работы я научился применять средства контроля версий, а также получил знания по работе с git.

## 6 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

Архитектура компьютеров и ОС/Электронный ресурс