ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

Дисциплина: Операционные системы

Обрезкова Анастасия Владимировна

Содержание

# 1 Цель работы

Изучение идеологии и применения средств контроля версий Git, освоение умения по работе с Git

# 2 Задание

1. Изучить идеологию и применение средств контроля версий.
2. Освоить умения по работе с git.

# 3 Теоретическое введение

## 3.1 Системы контроля версий. Общие понятия

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.

## 3.2 Примеры использования git

1. Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями.
2. Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией.

# 4 Выполнение лабораторной работы

## 4.1 Установка программного обеспечения

1. В прошлом семестре я уже установила git и gh и создала учетную запись на GitHub. (рис. [[1](#fig:001)])

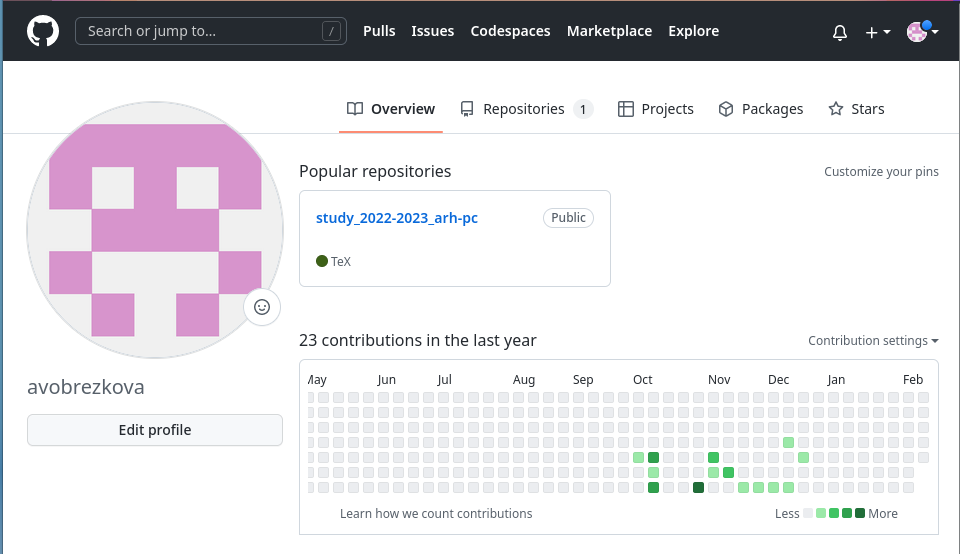


Figure 1: Аккаунт

1. Задала базовую настройку git (имя владельца репозитория, настройка utf-8, установка параметров autocrlf, установка параметров safecrlf. (рис. [[2](#fig:002)])

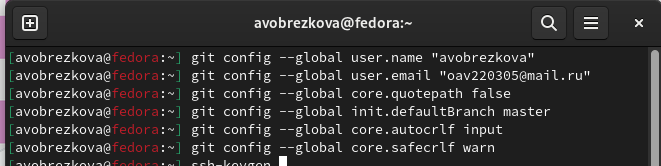


Figure 2: Базовая настройка

1. Создала в прошлом семестре ssh ключ, загрузила его в GitHub. (рис. [[3](#fig:003)], рис. [[4](#fig:004)]; рис. [[5](#fig:005)])

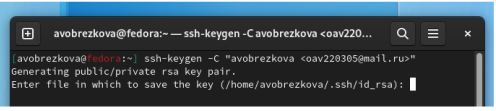


Figure 3: Создание ключа

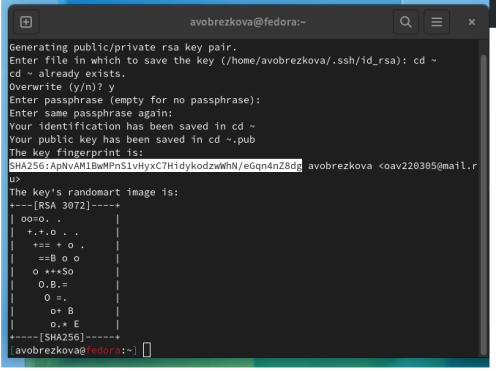


Figure 4: Создание ключа

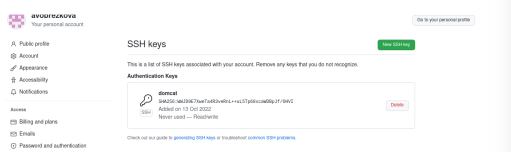


Figure 5: Загруженный на GitHub ключ

1. Создание gpg ключа. (рис. [[6](#fig:006)]; рис. [[7](#fig:007)])

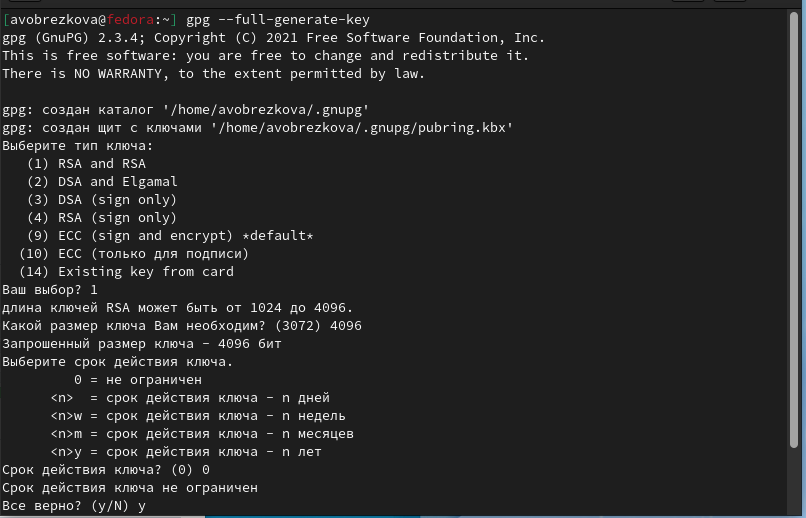


Figure 6: Создание ключа

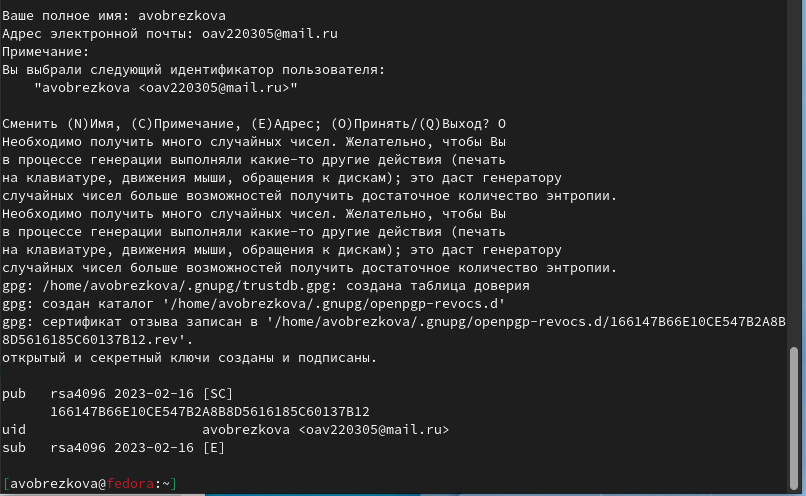


Figure 7: Создание ключа

1. ВЫвила список ключей и скопировала отпечаток приватного ключа, скопировала сгенерированный ключ в буфер обмена и добавила его на GitHub. (рис. [[8](#fig:008)], рис. [[9](#fig:009)])

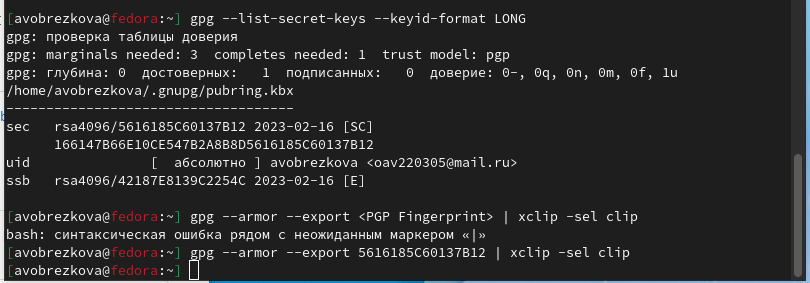


Figure 8: Отпечаток ключа и копирование в буфер обмена

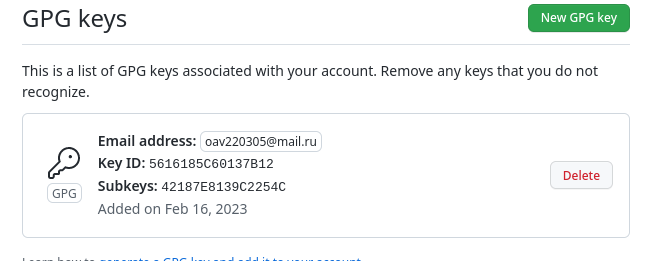


Figure 9: Загрузила ключ на GitHub

1. Настройка автоматических подписей коммитов git. (рис. [[10](#fig:010)])

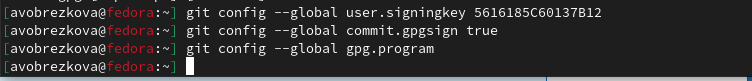


Figure 10: Настройка подписей коммитов

1. Авторизировалась и настроила gh. (рис. [[11](#fig:011)])

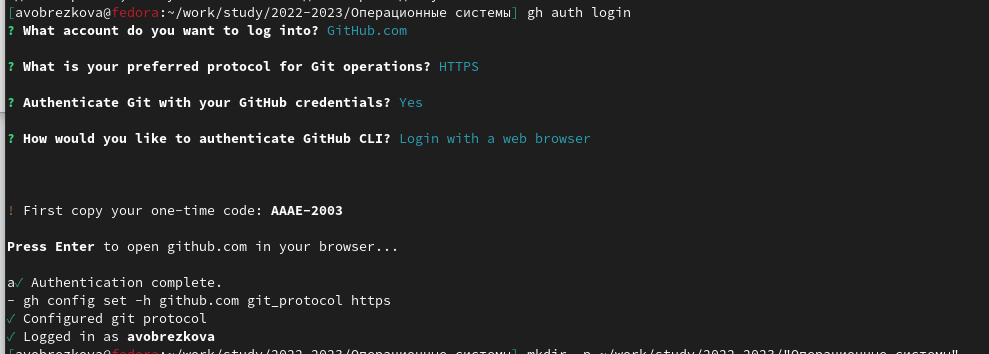


Figure 11: Авторизация

1. Создала репозиторий курса на основе шаблона. (рис. [[12](#fig:012)])

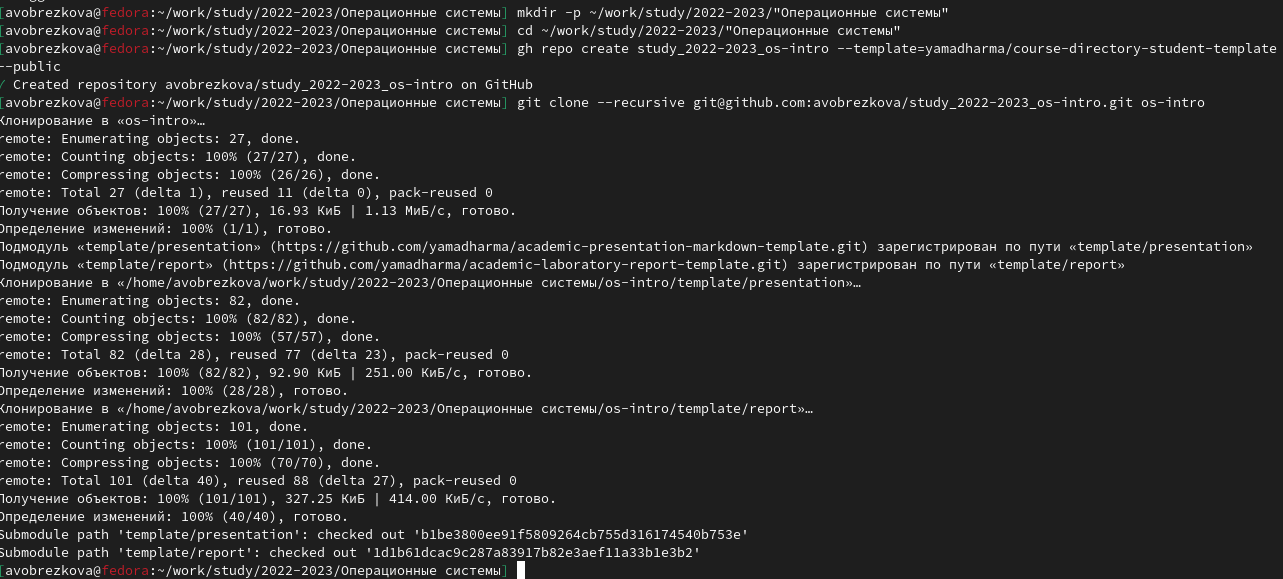


Figure 12: Создание репозитория

1. Настроила каталог курса, удалила лишние файлы, создала каталоги и отправила файлы на сервер. (рис. [[13](#fig:013)]; рис. [[14](#fig:014)]; рис. [[15](#fig:015)])

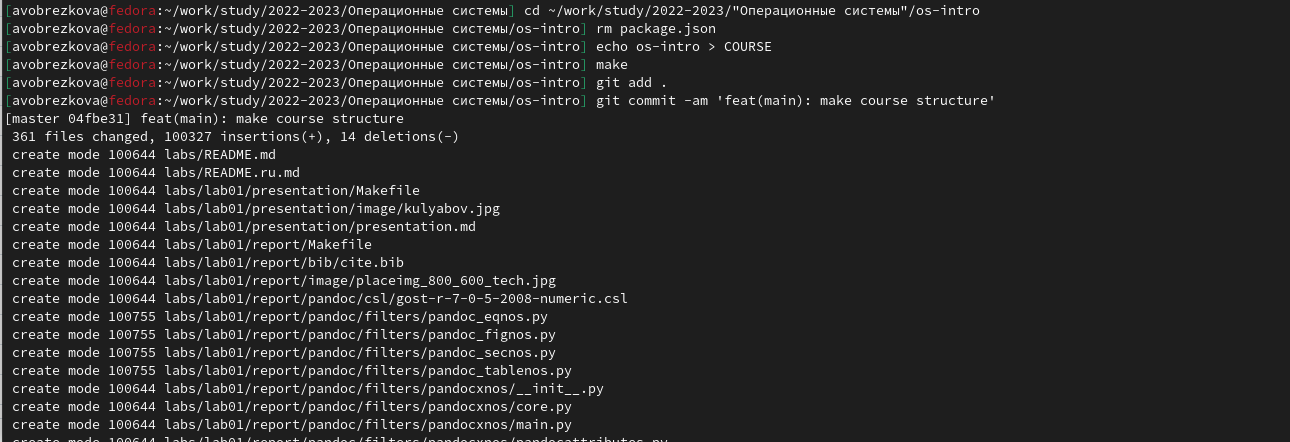


Figure 13: Настройка курса

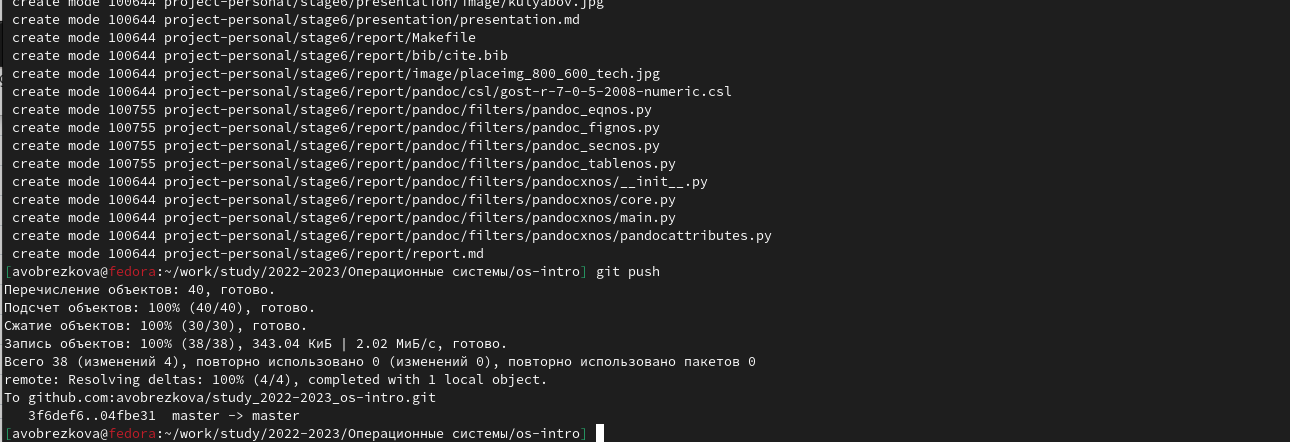


Figure 14: Настройка курса

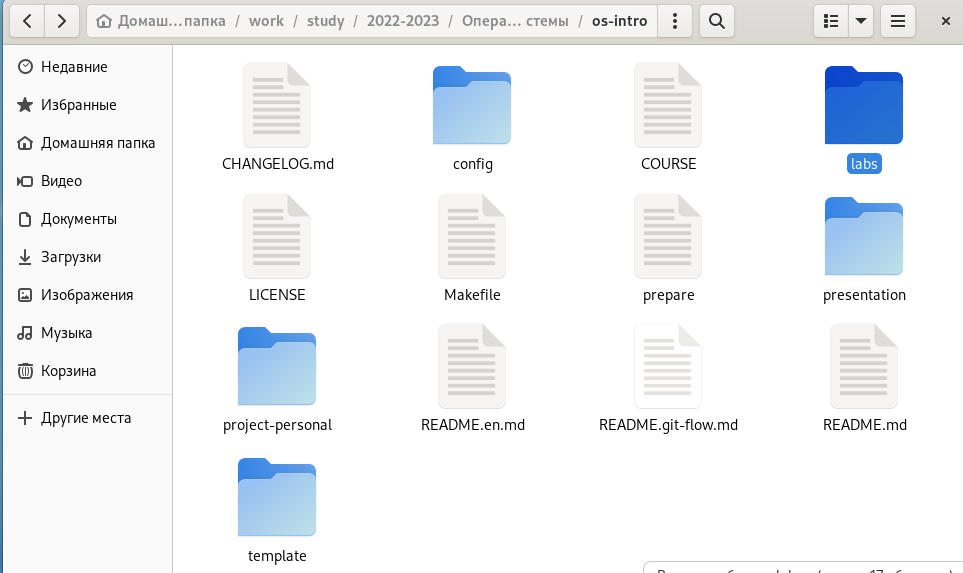


Figure 15: РЕзультат

Данные изменения можно проверить по ссылке: <https://github.com/avobrezkova/study_2022-2023_os-intro/tree/master/labs/lab02>

#Контрольные вопросы

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

Системы контроля версий — это программные инструменты, помогающие командам разработчиков управлять изменениями в исходном коде с течением времени. Какие задачи решает система контроля версий:

• Защищает исходный код от потери. Данные хранятся на удалённом сервере, даже если разработчики удалят файлы с локального компьютера, ониостанутся в репозитории.

• Обеспечивает командную работу.

• Помогает отменить изменения.

• Распределённая работа.

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

Хранилище (repository, сокр. repo), или репозитарий, — место хранения всех версий и служебной информации.

Коммит (commit; редко переводится как «слепок») — 1) синоним версии; 2) создание новой версии («сделать коммит», «закоммитить»).

Рабочая копия (working copy или working tree) — текущее состояние файлов проекта, основанное на версии из хранилища (обычно на последней)

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

Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Пример - Wikipedia.

В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. Пример — Bitcoin.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов.Выполнение большинства функций по управлению версиями осуществляется специальным сервером.

1. Опишите действия с VCS при единоличной работе с хранилищем.

Создадим локальный репозиторий. Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория:

git config –global user.name”Имя Фамилия”.

git config –global user.email”work@mail”

и настроив utf-8 в выводе сообщений git:

git config –global quotepath false

Для инициализации локального репозитория, расположенного, например, в каталоге ~/tutorial, необходимо ввести в командной строке:

сd

mkdir tutorial

cd tutorial

git init

1. Опишите порядок работы с общим хранилищем VCS.

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый): ssh-keygen -C”Имя Фамилия work@mail” Ключи сохраняться в каталоге~/.ssh/. Скопировав из локальной консоли ключ в буфер обмена

сat ~/.ssh/id\_rsa.pub | xclip -sel clip

вставляем ключ в появившееся на сайте поле.

1. Каковы основные задачи, решаемые инструментальным средством git?

У Git две основных задачи:

хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки,

обеспечение удобства командной работы над кодом.

1. Назовите и дайте краткую характеристику командам git.

Наиболее часто используемые команды

• 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 репози-торий:

• git push–просмотр списка изменённых файлов втекущей директории:git status–просмотртекущих изменения:

• git diff–сохранениетекущих 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 (приэтомфайл и/илик аталог 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 натекущей:

• 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 стекущим 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

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

Использования git при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий):

git add hello.txt

git commit -am’Новый файл

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

Ветки очень облегчают работу. Они решить такие проблемы как: нужно постоянно создавать архивы с рабочим кодом сложно “переключаться” между архивами сложно перетаскивать изменения между архивами легко что-то напутать или потерять

1. Как и зачем можно игнорировать некоторые файлы при commit?

Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых придобавлении в репозиторий типов файлов в файл. gitignore с помощью сервисов. Для этого сначала нужно получить список имеющихся шаблонов:

curl -L -s https://www.gitignore.io/api/list

Затем скачать шаблон,например, для C и C++

curl -L -s https://www.gitignore.io/api/c » .gitignore

curl -L -s https://www.gitignore.io/api/c++ » .gitignore

# 5 Выводы

В ходе лабораторной работы я изучила идеологию и применение средств контроля версий git, а также освоила умения по работе с git.

# Список литературы

1. <https://esystem.rudn.ru/mod/page/view.php?id=971076>