**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ»**

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

**ОТЧЕТ  
по лабораторной работе**№2  
по дисциплине «Операционные системы»

Выполнила:

Студентка группы НПИбд-01–21

Студенческий билет № 1032216538

Абдуллина Ляйсан Раисовна

« 22 » апреля 2022г.

Москва 2022

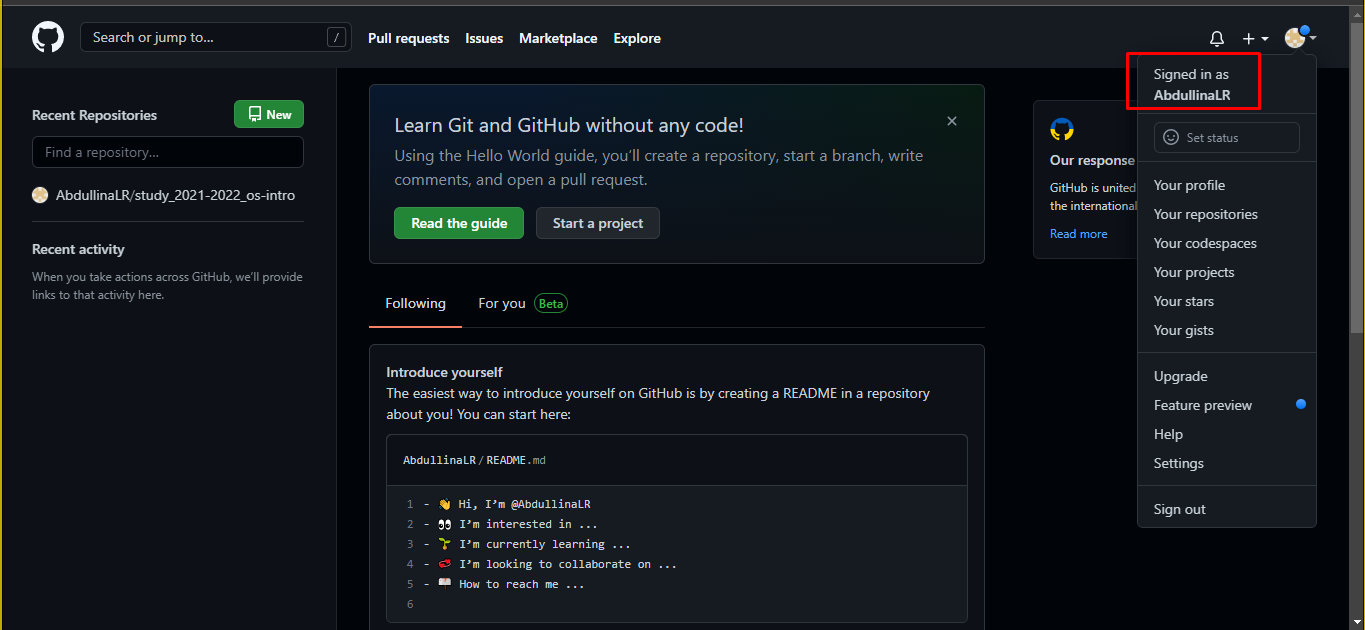
**Цель работы:**

Изучение идеологии и применения средств контроля версий, а также освоение умения по работе с git.

**Ход работы:**

**[1] Настройка GitHub**

Создадим учётную запись на <https://github.com> и заполним основные данные. (скриншот 1.1)



* 1. Успешная регистрация на GitHub’е с использованием своего имени, согласно Соглашении об именовании.

**[2] Установка программного обеспечения**

Поскольку программное обеспечение удалено из репозитория, мы должны установить его вручную, используя следующие команды(скриншоты 2.1–2.3):

cd /tmp

wget --no-check-certificate -q https://raw.github.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh

chmod +x gitflow-installer.sh

sudo ./gitflow-installer.sh install stable

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание2.1 Ввод выше представленных команд в консоль. Введение пароля для пользователя. Установка gitflow.

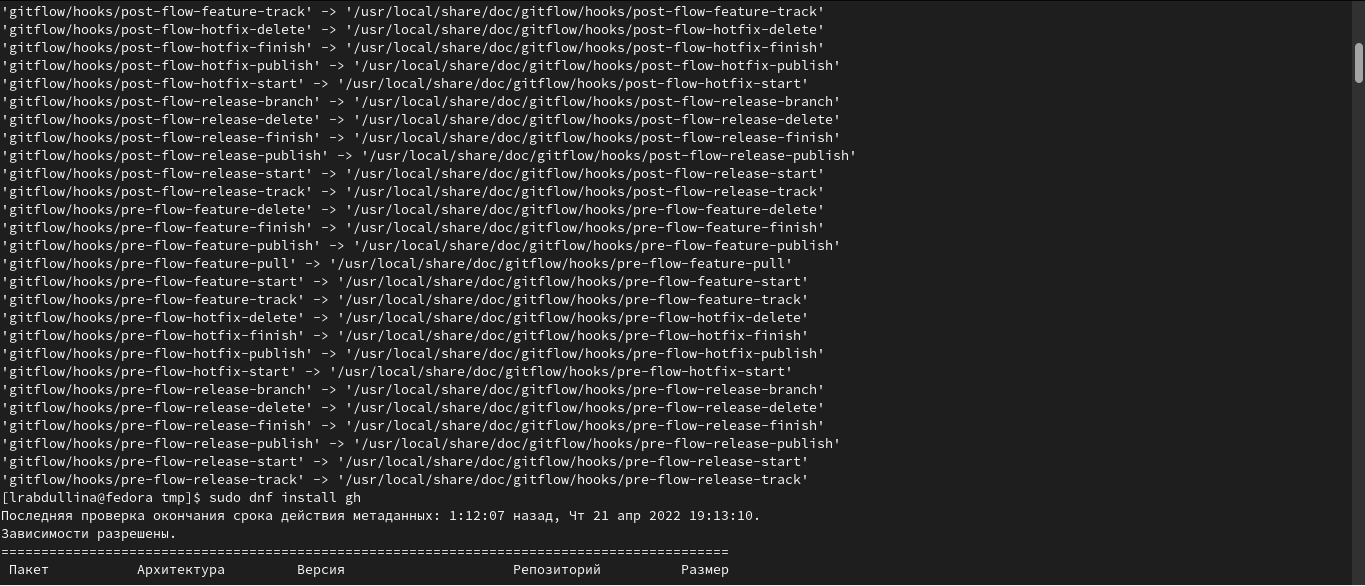
2.2 Продолжение установки GitFlow и создание каталога /usr/local/share/doc/gitflow/hooks

Изображение выглядит как текст

Автоматически созданное описание2.3 Завершение установки gitflow

**[3] Установка gh в Fedora Linux**

С помощью команды, которая приведена ниже, устанавливаем gh в Fedora Linux. (скриншоты 3.1–3.2)

****

sudo dnf install gh

3.1 Ввод команды для установки gh в Fedora Linux

Изображение выглядит как текст

Автоматически созданное описание3.2 Установка gh в Fedora Linux, вывод информации

**[4] Базовая настройка git**

Используя следующие команды, зададим имя и почту владельца репозитория (заранее будем использовать ту почту, с которой мы регистрировались на GitHub), а также настроим utf-8 в выводе сообщений git. (скриншот 4.1)

git config --global user.name AbdullinaLR

git config --global user.email owlwithcookie@gmail.com

git config --global core.quotepath false

Изображение выглядит как текст

Автоматически созданное описание

4.1 Задаем имя и почту владельца репозитория, используя свои личные данные, а также настраиваем вывод сообщений

Настраиваем верификацию и подписание коммитов git, задаем имя начальной ветки (master), а также параметры autocrlf и autosafecrlf, используя следущий набор команд: (скриншот 4.2)

git config --global init.defaultBranch master

git config --global core.autocrlf input

git config --global core.safecrlf warn

Изображение выглядит как текст

Автоматически созданное описание  
4.2 Выполнение команд: настраивание верификации, подписание коммитов, имя начальной ветки – master.

**[5] Создание ключа ssh**

Ключ ssh можно создать двумя разными способами:

Первое: по алгоритму rsa с ключём размером 4096 бит:

ssh-keygen -t rsa -b 4096

Второе: по алгоритму ed25519:

ssh-keygen -t ed25519

Мы опробовали 2 способа, чтобы убедиться в их работоспособности и создали, тес самым, ключ ssh (скриншоты 5.1 – 5.2)

Изображение выглядит как текст

Автоматически созданное описание

5.1 Ввод первого способа, создание ключа, вывод картинки, соответствующей ключу

Изображение выглядит как текст

Автоматически созданное описание

5.2 Ввод второго способа, перезапись ключа, повторный вывод картинки (уже другой)

**[6] Создаём ключи pgp**

Генерируем ключ, используя следующую команду (скриншот 6.1):

gpg --full-generate-key

Из предложенных опций выбираем (скриншоты 6.1 - 6.3):

– тип RSA and RSA;

– размер 4096;

– срок действия : 0

GPG запросит личную информацию, которая сохранится в ключе:

– Имя

– Адрес электронной почты (он соответствует адресу, используемому на GitHub)

– Комментарий

Изображение выглядит как текст

Автоматически созданное описание

6.1 Генерация ключа, создание каталога

Изображение выглядит как текст

Автоматически созданное описание

6.2 Выбор нужной информации, заполнение личных данных

Изображение выглядит как текст

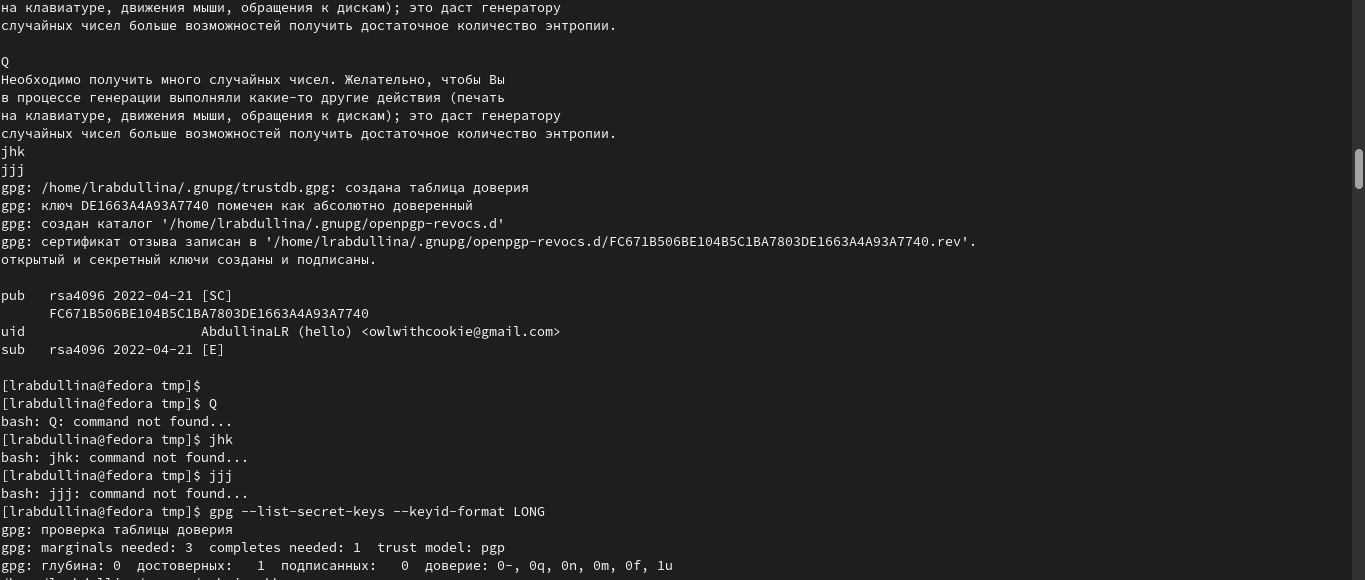
Автоматически созданное описание

6.3 Процесс генерации

**[7] Добавление PGP ключа в GitHub**

Выводим список ключей и копируем отпечаток приватного ключа(скриншоты 7.1–7.2):

gpg --list-secret-keys --keyid-format LONG



7.1 Ввод команды для вывода списка ключей

Изображение выглядит как текст

Автоматически созданное описание

7.2 Список ключей

С помощью формата строки находим отпечаток ключа (скриншот 7.3):



7.3 Копирование отпечатка ключа

Копируем сгенерированный PGP ключ в буфер обмена при помощи следующей команды: (скриншоты 7.4–7.5)

gpg --armor --export | xclip -sel clip

Изображение выглядит как текст

Автоматически созданное описание

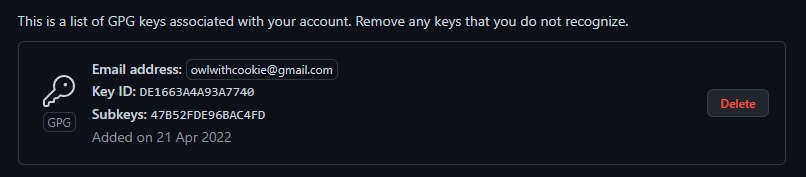
7.4 Ввод команды для копирования PGP ключа в буфер обмена и установка библиотеки xclip

Изображение выглядит как текст

Автоматически созданное описание

7.5 Копирование ключа в буфер обмена

Переходим в настройки GitHub (https://github.com/settings/keys), нажимаем на кнопку New GPG key и вставляем полученный ключ в поле ввода.



7.6 Добавление ключа PGP в GitGub

**[8] Настройка автоматических подписей коммитов git**

Используя введёный мною email, укажим Git применять его при подписи коммитов (скриншот 8.1):

git config --global user.signingkey

git config --global commit.gpgsign true

git config --global gpg.program $(which gpg2)

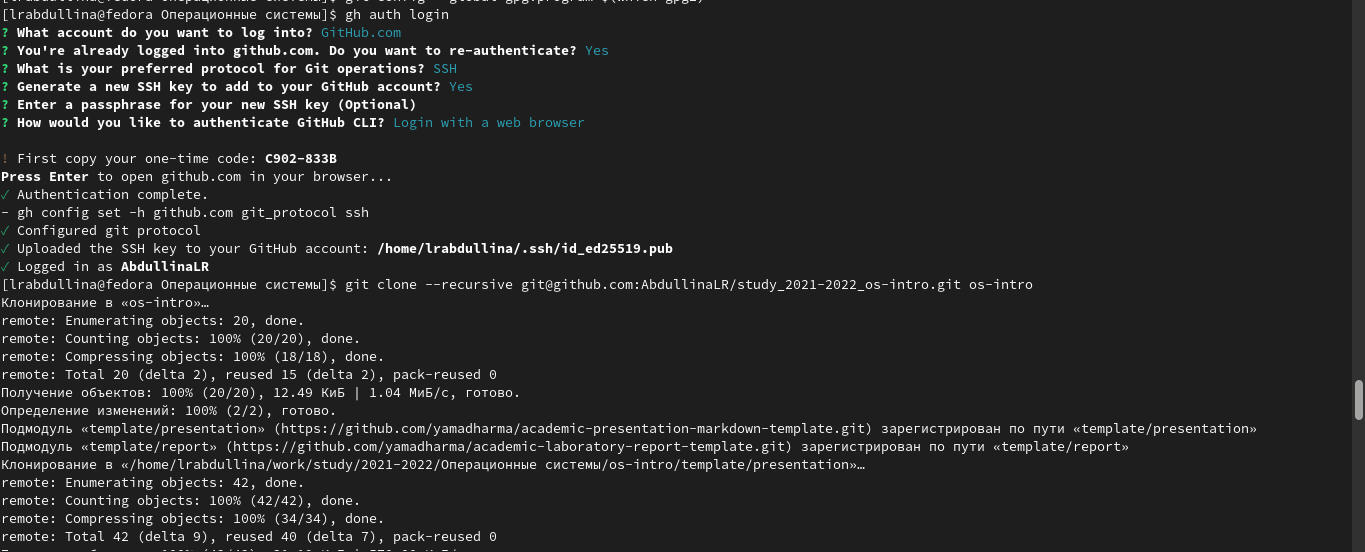
**Изображение выглядит как текст

Автоматически созданное описание**

8.1 Настройка автоматических подписей коммитов git

**[9] Настройка gh**

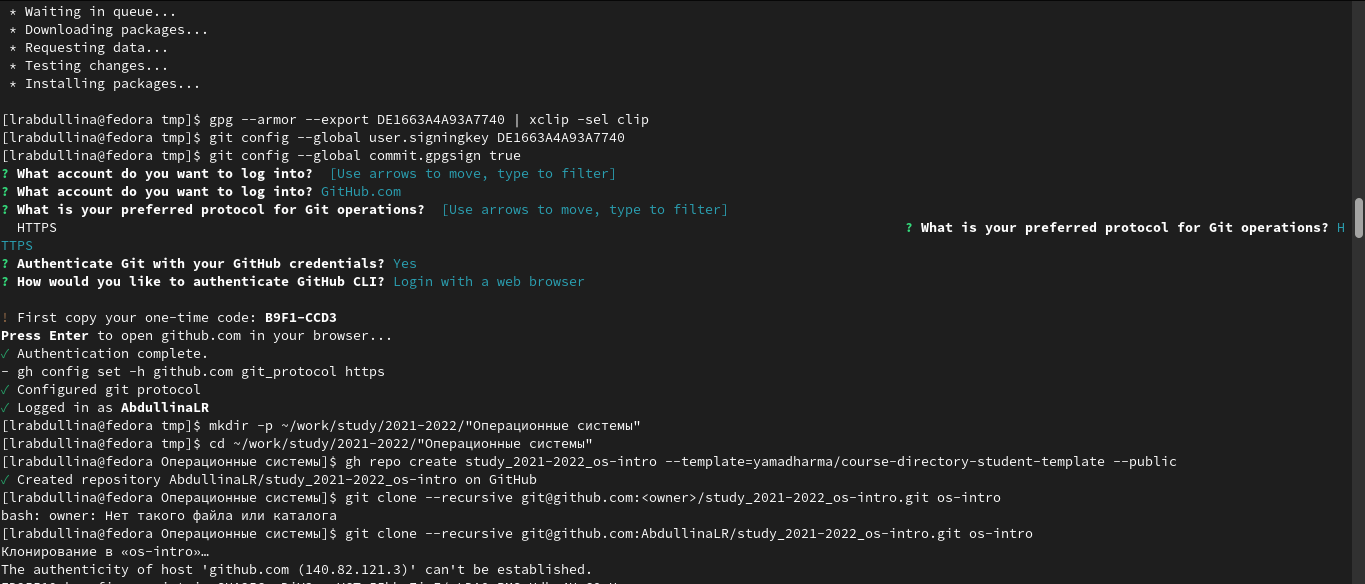
Авторизуемся и настроим все необхоимое (скриншот 9.1):



9.1 Авторизация через консоль

**[10] Создание репозитория курса на основе шаблона**

Необходимо создать шаблон рабочего пространства. Для 2021–2022 учебного года и предмета «Операционные системы» (код предмета os-intro) создание репозитория примет следующий вид: (скриншоты 10.1 – 10.2)

****

mkdir -p ~/work/study/2021-2022/"Операционные системы"

cd ~/work/study/2021-2022/"Операционные системы"

gh repo create study\_2021-2022\_os-intro --template=yamadharma/course-directory-student-template –public

git clone --recursive git@github.comAbdukkinaLR:/study\_2021-2022\_os-intro.git os-intro

Изображение выглядит как текст

Автоматически созданное описание10.1 Создание нового каталога «Операционные системы» и переход в этот каталог

10. 2 Клонирование информации

**[11] Настройка каталога курса**

Перейдём в каталог курса, удалим лишние файлы, создадим необходимые каталоги и отправим файлы на сервер, с помощью следующих команд: (скриншоты 11.1–11.2)

cd ~/work/study/2021-2022/"Операционные системы"/os-intro

rm package.json

make COURSE=os-intro

git add .

git commit -am 'feat(main): make course structure'

git push

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание11.1 Переход в каталог курса, удаление лишних файлы, создание необходимых каталогов.

11.2 Отправка файлов на сервер

**Контрольные вопросы:**

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

*Система контроля версий – это набор программного обеспечения, который предназначается для работы нескольких человек над одним проектом*.

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

*Хранилище – сервер, на котором хранится вся история изменений проекта.*

*Commit – фиксация «дельта-изменений», то есть изменений с последней версии с его последующей записью как версия в истории.*

*История – список всех изменений проекта (с возможностью «откатить» в любую точку изменений)*

*Рабочая копия – все файлы проекта, с которыми происходит основная работа.*

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

*В централизованных VCS необходим центральный репозиторий для хранения файлов. Например, Subversion. В децентрализованных VCS центральный репозиторий не обязателен. Например, Git.*

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

*Инициализация системы управления версиями git через git init. Работа над проектом использую git-flow для отдельных частей проекта. Git commit – для фиксации изменений. При необходимости использование удаленного сервера для хранения с помощью remote и git push. Удаленный сервер также позволяет работать с нескольких устройств с использованием git pull.*

5. Опишите порядок работы с общим хранилищем VCS.

*При существующей версии проекта в хранилище, скопировать его оттуда через git pull. Использовать git-flow для работы над частями проекта. После окончания работы нужно зафиксировать изменения через got commit. Загрузить в хранилище через git push.*

6. Каковы основные задачи, решаемые инструментальным средством git?

*Ведение истории изменений, совмещение версий, веток и пр., фиксирование изменений, а также откат к прошлым версиям.*

7. Назовите и дайте краткую характеристику командам git.

*Git init – инициализация проекта с системой контроля версий*

*Git commit – фиксация изменений в отслеживаемом файле*

*Git add – добавление файла/директории в систему контроля версий как отслеживаемое*

*Git push – загрузка локальной версии на сервер*

*Git pull – выгрузка локальной версии на сервер*

*Git fetch – «часть» команды git pull, которая собирает актуальную версию (но не вносит в работу)*

*Git merge – слияние веток*

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

*При работе с локальным репозиторием все изменения хранятся локально и не выгружаются на удаленный сервер. Не нужно использование команд push, pull и т. п. При работе с удаленным репозиторием для отображения изменения, они должны быть выгружены на удаленный сервер (с помощью команды pull).*

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

*Ветви позволяют «разделять» части работы, работать отдельно над каждой имплементацией. Использование ветвей дает возможность комфортной проверки и обработки новой информации, которую ввели в основную ветвь.*

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

*Можно игнорировать с помощью .gitignore файла. В нем указываются пути, названия и другие идентификации нежелательных объектов (которые не будут учитываться в commit). Это полезно для устранения «остаточных» файлов, которые более не являются полезными. А также это могут быть конфиденциальные данные, например пароли.*

**Вывод:**

В ходе лабораторной работы мы изучили идеологию и применение средств контроля версий, а также освоение умения по работе с git.