РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

ОТЧЕТ

по лабораторной работе № 2

дисциплина: Операционные системы

Студент: Крутова Екатерина Дмитриевна

Группа: НПИбд-01-21

**МОСКВА**

2020 г.

**Цель:**

– Изучить идеологию и применение средств контроля версий.

– Освоить умения по работе с git.

**Ход работы**:

1. Установка программного обеспечения

Изображение выглядит как текст

Автоматически созданное описание

* 1. Установка git-flow в Fedora Linux



* 1. Установка gh в Fedora Linux

1. Базовая настройка git

Изображение выглядит как текст, табличка

Автоматически созданное описание

* 1. Задавание имени и email владельца репозитория, настройка utf-8 в выводе сообщений git, настройка верификации и подписания коммитов git, задавание имени начальной ветки (будем называть её master), параметра autocrlf, параметра safecrlf:

1. Создание ключа shh

Изображение выглядит как текст

Автоматически созданное описание

* 1. Создание по алгоритму rsa с ключём размером 4096 бит

1. Создание ключа pgp



* 1. Генерируем ключ

1. Добавление PGP ключа в GitHub

Изображение выглядит как текст

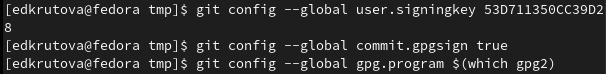
Автоматически созданное описание

* 1. Вывод списка ключей и копирование отпечаток приватного ключа



* 1. Копирование сгенерированного PGP ключа в буфер обмена (чтобы вставить его в настройки GitHub (https://github.com/settings/keys))

1. Настройка автоматических подписей коммитов git



* 1. Используя введёный email, указывание Git применять его при подписи коммитов

1. Настройка gh

Изображение выглядит как текст

Автоматически созданное описание

* 1. Авторизация, ответы на вопросы утилиты

1. Сознание репозитория курса на основе шаблона

Изображение выглядит как текст

Автоматически созданное описание

* 1. Создание шаблона рабочего пространства, создание требуемого вида

1. Настройка каталога курса



* 1. Переход в каталог курса



* 1. Удаление лишних файлов



* 1. Создание необходимых каталогов



* 1. Отправка файлов на сервер

**Контрольные вопросы:**

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

**Система контроля версий (Version Control System, VCS) — программное обеспечение для облегчения работы с изменяющейся информацией. VCS позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.**

1. 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", т.е. изменений с последнего commit’a с его последующей записью как версии в истории.**

**История — список всех изменений проекта с возможностью отката в любую точку истории.**

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

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

**В централизованных VCS необходим центральный репозиторий для хранения файлов. Примером таковых могут служить CVS и Subversion. В децентрализованных VCS наличие центрального репозитория не обязательно. Децентрализованными VCS являются Git, Bazaar и Mercurial.**

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

**Инициализация системы управления версиями git через git init. Работа над проектом используя 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**

1. 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 зафиксировать изменения через git commit и загрузить в хранилище через git push.**

1. Каковы основные задачи, решаемые инструментальным средством git?

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

1. Назовите и дайте краткую характеристику командам git.

* **git init — инициализация проекта с системой контроля версий**
* **git add — добавление файла/директории в систему контроля версий как отслеживаемое**
* **git commit — фиксация изменений в отслеживаемых файлах**
* **git push — загрузка локальной версии на сервер**
* **git pull — выгрузка актуальной версии с сервера**
* **git fetch — "часть" команды git pull, которая собирает актуальную версию, но не вносит её в работу**
* **git merge — слияние вето**

1. 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 требуется использование команд push, pull, remote и т.д. При работе с удаленным репозиторием для отображения изменения на удаленном репозитории и его актуализации, последние изменения должны быть загружены на удаленный сервер.**

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

**Ветви позволяют "разделять" части работы и работать отдельно над каждой имплементацией. Использование ветвей дает возможность комфортной ревизии и обработки нововведений в основную ветвь, которая чаще всего является релизной.**

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

**Игнорирование файлов при commit происходит с помощью .gitignore файла. В нем указываются пути, названия, расширения и другие идентификации нежелательных объектов, которые не будут учитываться в commit. Это полезно для исключения как "мусорных" файлов, которые не являются значимой частью проекта, а также конфиденциальных файлов, которые содержат в себе приватную информацию, такую как пароли и токены.**

**ВЫВОД:**

Изучили идеологию и применение средств контроля версий и освоить умения по работе с git.