**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**Отчет**

**по лабораторной работе №****2**

***дисциплина: Операционные системы***

Студентка: Митичкина Екатерина Павловна

Группа: НПИбд-02-21

№ ст. билета: 1032212278

**МОСКВА**

2022 г.

**Цель работы:**

**–** Изучить идеологию и применение средств контроля версий.

**–** Освоить умения по работе с git.

**Выполнение работы:**

Я ознакомилась с основными командами git, работой различных репозиторий. Далее я приступила к выполнению задания

* Создать базовую конфигурацию для работы с git.
* Создать ключ *SSH*.
* Создать ключ *PGP*.
* Настроить подписи git.
* Зарегистрироваться на *Github*.
* Создать локальный каталог для выполнения заданий по предмету.

Так как у меня уже была учетная запись на github, я пропустила пункт настройки github.

Далее перешла к установке программного обеспечения.

1) Установка git-flow в Fedora Linux

Для этого я запустила код

cd /tmp

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

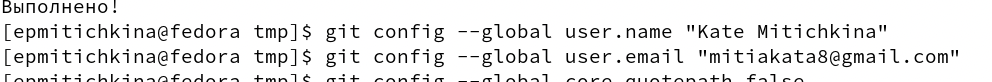
sudo ./gitflow-installer.sh install stable

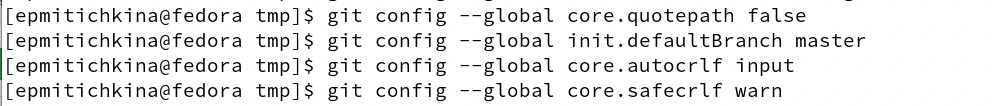
2) Установка gh в Fedora Linux

sudo dnf install gh

Все программное обеспечение загрузилось

Следящий пункт базовая настройка git

Для начала я задала имя и email репозиторию:

Настроила utf-8 в выводе сообщений git, задала имя начальной ветки, параметр autocrlf и safecrlf

После этого создала ключи ssh

* Изображение выглядит как текст

  Автоматически созданное описаниепо алгоритму *rsa* с ключём размером 4096 бит:
* Изображение выглядит как текст

  Автоматически созданное описаниепо алгоритму *ed25519*:

Изображение выглядит как текст

Автоматически созданное описаниеСледующий пункт был создать ключи pgp

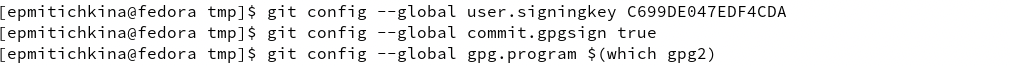
Дальше нужно было добавить PGP ключа в GitHub

Изображение выглядит как текст

Автоматически созданное описаниеВывила список ключей и скопировала отпечаток приватного ключа

Скопировала сгенерированный PGP ключ в буфер обмена:

И вставила полученный код на github

Дальше настроила автоматические подписи коммитов git

Изображение выглядит как текст

Автоматически созданное описаниеПосле нужно было авторизироваться

 Следующие создание репозитория курса на основе шаблона

И последнее это настройка каталога курса

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как стол

Автоматически созданное описаниеПерехожу в нужный каталог и удаляю не нужный файл, далее создаю нужный каталог и отправляю нужный файл на сервер

**Вывод:**

В результате работы изучила идеологию и применение средств контроля версий. Освоила умения по работе с git.

**Ответы на контрольные вопросы**

1) Version Control System — программное обеспечение для облегчения работы с изменяющейся информацией. 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

2) 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 неполную версию изменённых файлов, а производить так называемую дельта-компрессию—сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

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.

3) 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.

4) 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 сообщенийgit:

git config —global quotepath false

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

cd

mkdir tutorial

cd tutorial

git init

5) 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

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

вставляем ключ в появившееся на сайте поле.

6) У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.

7) Основные команды 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 репозиторий: git push–просмотр списка изменённых файлов в текущей директории: git status–просмотртекущих изменения: 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 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

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

git add hello.txt

git commit -am' Новый файл

9) Проблемы, которые решают ветки git:

• нужно постоянно создавать архивы с рабочим кодом

• сложно "переключаться" между архивами

• сложно перетаскивать изменения между архивами

• легко что-то напутать или потерять

10) 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 при добавлении в репозиторий типов файлов в файл gitignore с помощью сервисов. Для этого сначала нужно получить списки меняющихся 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