РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра математики и компьютерных наук

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

дисциплина: Операционные системы

Студент: Морозов Михаил Евгеньевич

Студ. Билет № 1032217050

Группа: НКНбд-01–21

МОСКВА

2021 г

Цель работы: изучить идеологию и применение средств контроля версий. Освоить умения по работе с git.

Выполнение работы:

1.

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 дельтакомпрессию — 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.

После ознакомления с методическим материалом и анализа прочитанного текста немного понимаешь идеологию и применение средств контроля версий.

2.После теории переходим к практике и работе с git.

Установка git-flow в Fedora Linux с помощью команд терминала

(последующие скрины из видео выполнения лабораторной работы 2)

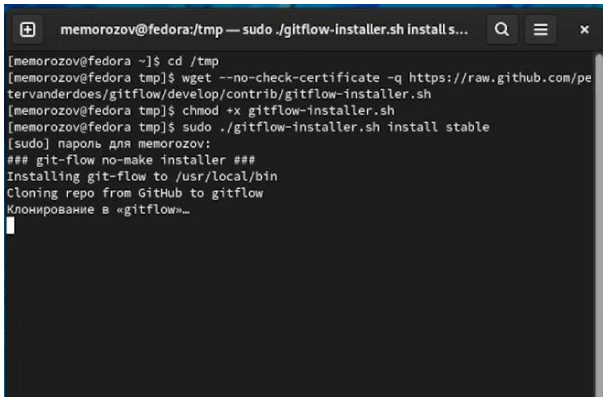

Рис 2.1

После ввода команд в терминал начинается выполнения скрипта , устанавливаем gh в Fedora Linux для дальнейшей работы

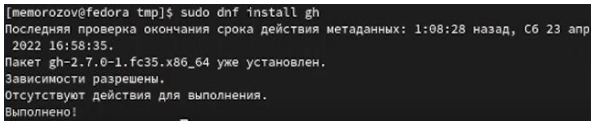

Рис 2.2

Далее переходим к базовой настройке git

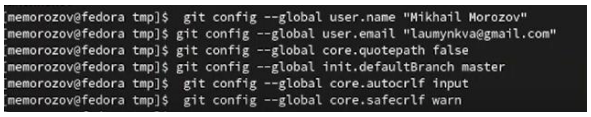

Рис 2.3

Создаем ключи ssh

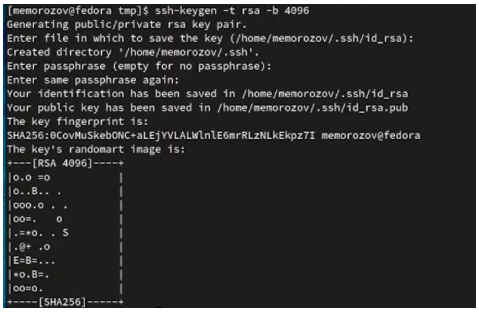

Рис 2.4

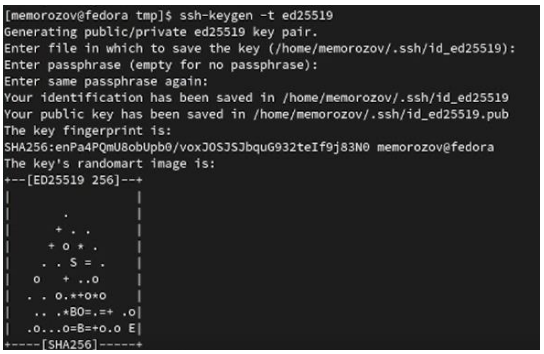

Рис 2.5

Переходим к созданию pgp ключей по формату согласно методическим материалам (RSA,4096,never/name,mail)

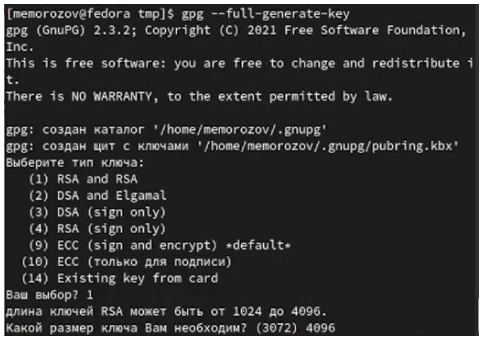

Рис 2.6

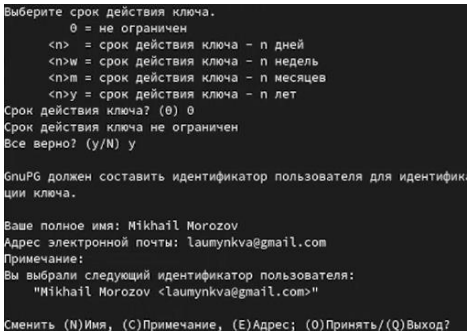

Рис 2.7

```
случайных чисел больше возможностей получить достаточное количес
тво энтропии.
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст гене
ратору
случайных чисел больше возможностей получить достаточное количес
тво энтропии.
gpg: /home/memorozov/.gnupg/trustdb.gpg: создана таблица доверия
gpg: ключ AD38DB37692D4499 помечен как абсолютно доверенный
gpg: создан каталог '/home/memorozov/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/memorozov/.gnupg/openpgp
-revocs.d/B4E5A21988DBD0B720E62890AD38DB37692D4499.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2022-04-23 [SC]
      B4E5A21988DBD0B720E62890AD38DB37692D4499
uid                      Mikhail Morozov <laumynkva@gmail.com>
sub   rsa4096 2022-04-23 [E]
```

Рис 2.8

Добавляем PGP ключ в GitHub

```
[memorozov@fedora tmp]$ gpg --list-secret-keys --keyid-forma
NG
gpg: проверка таблицы доверия
gpg: marginals needed: 3  completes needed: 1  trust model:
gpg: глубина: 0  достоверных:   1  подписанных:   0  доверие
, 0q, 0n, 0m, 0f, 1u
/home/memorozov/.gnupg/pubring.kbx
------------------------------
sec   rsa4096/AD38DB37692D4499 2022-04-23 [SC]
      B4E5A21988DBD0B720E62890AD38DB37692D4499
uid              [  абсолютно ] Mikhail Morozov <laumynkva@
l.com>
ssb   rsa4096/64B24400488DCD4E 2022-04-23 [E]

[memorozov@fedora tmp]$ gpg --armor --export <PGP Fingerprin
 xclip -sel clip
```

Рис 2.9

Связываем свой аккаунт GitHub со своим PGP ключом

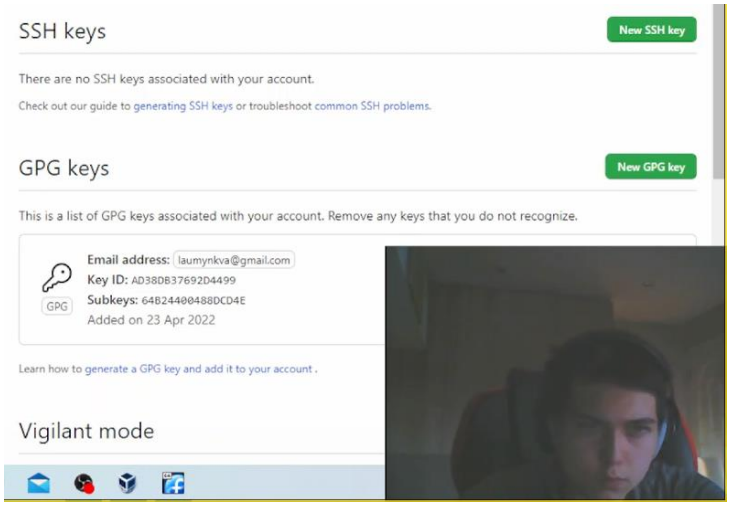

Рис 2.10

Далее настраиваем автоматические подписи коммитов git

```
[memorozov@fedora tmp]$ git config --global user.signingkey AD38DB37692D4499
[memorozov@fedora tmp]$ git config --global commit.gpgsign true
[memorozov@fedora tmp]$ git config --global gpg.program $(which gpg2)
```

Рис 2.11

Далее настройка gh и авторизация через бразуер

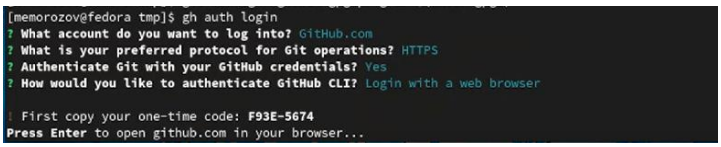

Рис 2.12

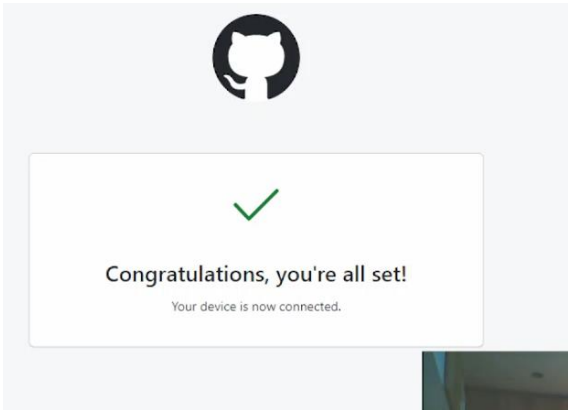

Рис 2.13

Для того чтобы перейти к следующему шагу подключаем к своему аккаунту SSH ключ

Рис 2.14

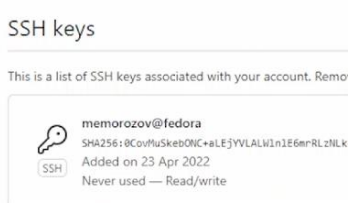

Рис 2.15

Начинаем настройку репозитория курса по средствам введния команд из методического материала

Рис 2.16

Также настраиваем каталог курса

Рис 2.17

Ответы на контрольные вопросы:

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

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.

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

Хранилище-репозиторий-место хранения всех версий и служебной информации

Commit- команда записи индексированных изменений в репозитории

История – место , где сохраняются все коммиты

Рабочая копия – текущее состояние файлов проекта

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

Централизованные системы – это системы в которых одно основное хранилище всего проекта (Subversion)

Децентрализованные системы – это системы в которых каждый пользователь имеет свой вариант репозитория и имеет возможность добавлять и брать изменения из репозиториев (Git)

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

Создаем и подключаем удаленный репозиторий , далее отправляем изменения на сервер.

5. Опишите порядок работы с общим хранилищем VCS.

Пользователь по средствам команд получает нужную ему версию файлов , а после внесения изменений пользователь размещает новую версию в хранилище. При этом предыдущие вресии не удаляются из общего хранилища.

6. Каковы основные задачи, решаемые инструментальным средством git?

Хранение информации об изменениях в коде , обеспечение удобства командной работы над кодом.

7. Назовите и дайте краткую характеристику командам git.

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 репози торий:

git push

– просмотр списка изменённых файлов в текущей директории:

git status

– просмотр текущих 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 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. Приведите примеры использования при работе с локальным и удалённым репозиториями.

отправка изменений конкретной ветки в центральный репозиторий:

git push origin имя_ветки

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

Ветвь – это последовательность коммитов , в которой ведется параллельная разработка какого-либо функционала. Они нужны чтобы несколько программистов могли одновременно работать над проектом.

10. Как и зачем можно игнорировать некоторые файлы при commit

Игнорируемы файлы – как правило артефакты , сборки , файлы , генерируемые машиной из исходных в вашем репозитории , либо файлы которые по какой-либо иной причине не должны попасть в коммит.

Вывод: Я изучил идеологию и применение средств контроля версий. А также приобрел умения по работе с git.