Лабараторная работа 2

Первоначальна настройка git

Хрусталев Влад Николаевич

Содержание

1 Цель работы **4**

2 Выполнение лабораторной работы **5**

3 Выводы **14**

Список иллюстраций

2.1 Установка пакетов 5
2.2 Первичная настройка git 5
2.3 Создание SSH ключей 6
2.4 Создание PGP ключей 6
2.5 Использование PGP ключей 7
2.6 Окончательная настройка git 7
2.7 копирование репозитория 7
2.8 Организация рабочего пространства 8
2.9 Настройка рабочего пространства(1) 8
2.10 Настройка рабочего пространства(2) 9

1 Цель работы

Изучить идеологию и применение средств контроля версий.

Освоить умения по работе с git.

2 Выполнение лабораторной работы

Установим git и gh.(рис. fig. 2.1)

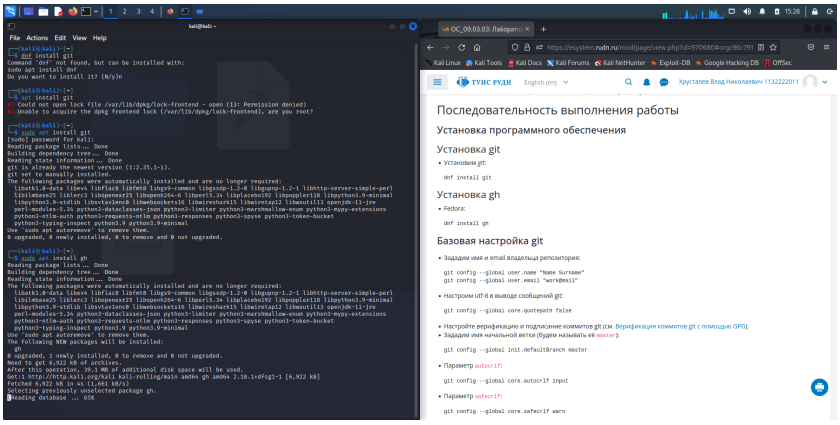

Рис. 2.1: Установка пакетов

Первичная настройка git по инструкции(рис. fig. 2.2)

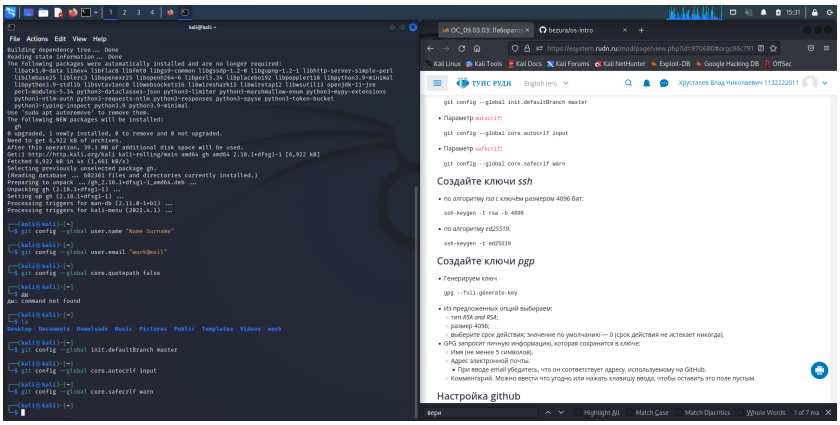

Рис. 2.2: Первичная настройка git

Создание двух ключей с алгоритмами rsa 4096 и ed25519(рис. fig. 2.3)

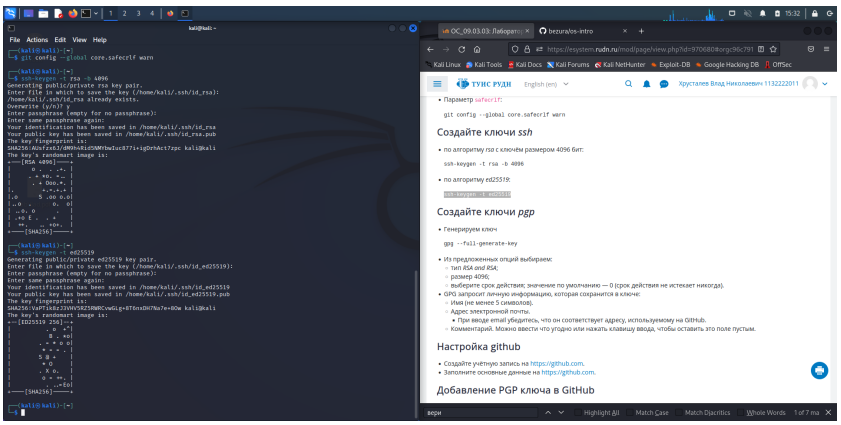

Рис. 2.3: Создание SSH ключей

Создание PGP ключа(рис. fig. 2.4)

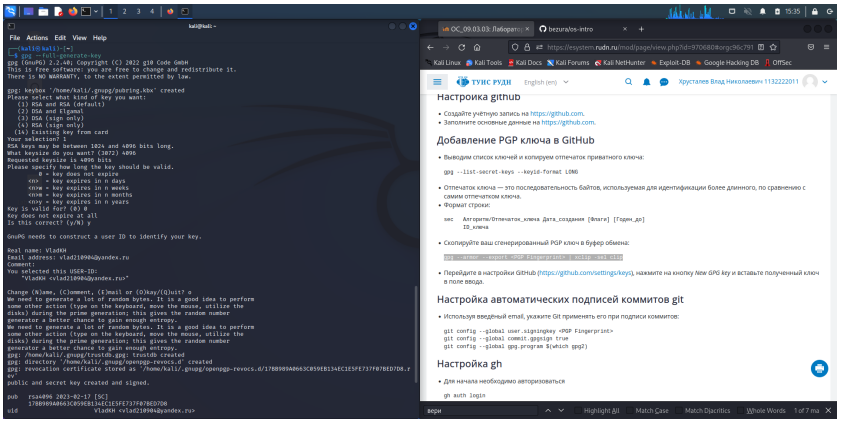

Рис. 2.4: Создание PGP ключей

Теперь скопируем готовый ключ и добавим на сайте github в GPG keys.(рис. fig. 2.5)

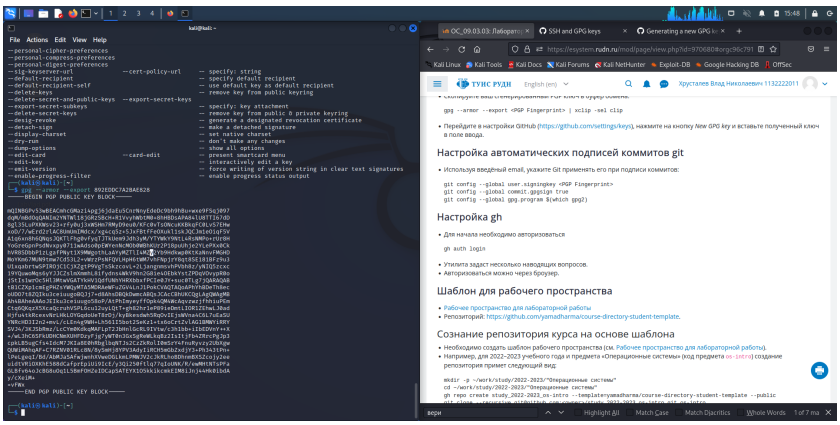

Рис. 2.5: Использование PGP ключей

Завершим настройку git.(рис. fig. 2.6)

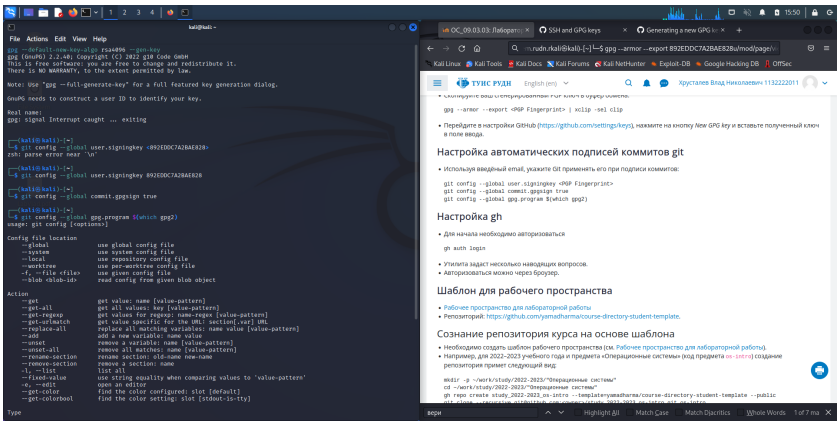

Рис. 2.6: Окончательная настройка git

Теперь открою репозиторий-шаблон и скопирую к себе.(рис. fig. 2.7).

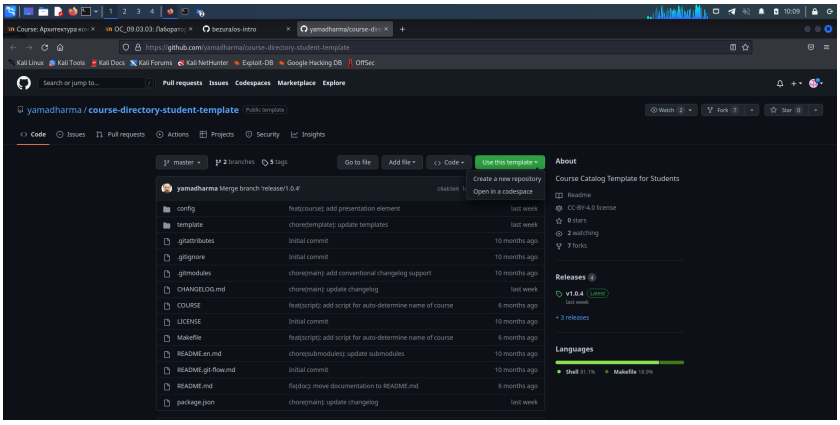

Рис. 2.7: копирование репозитория

Создадим нужную иерархию и скопируем репозиторий(рис. fig. 2.8).

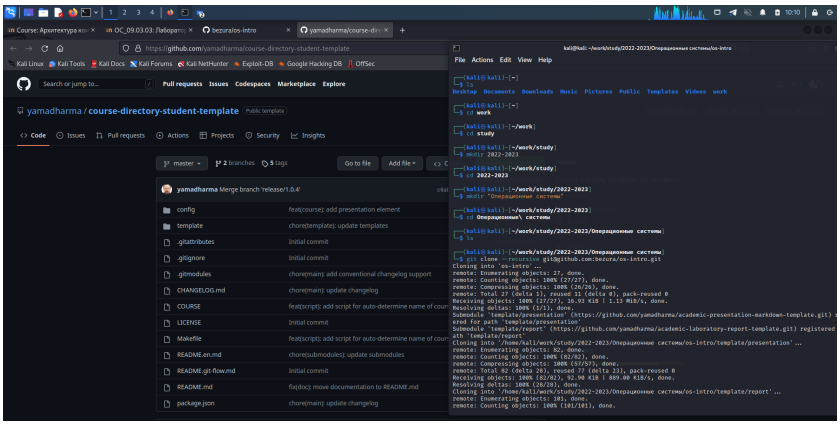

Рис. 2.8: Организация рабочего пространства

Удалим ненужные файлы и запустим файл make для настройки репозитория(рис. fig. 2.9).

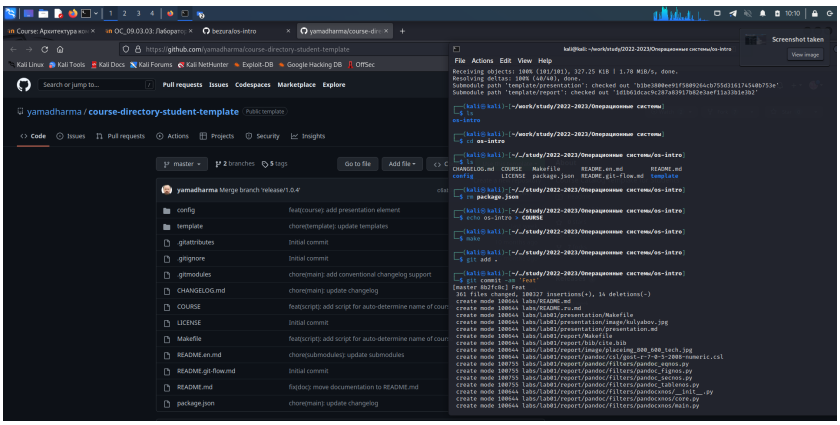

Рис. 2.9: Настройка рабочего пространства(1)

Далее загрузим обратно в гитхаб(рис. fig. 2.10)

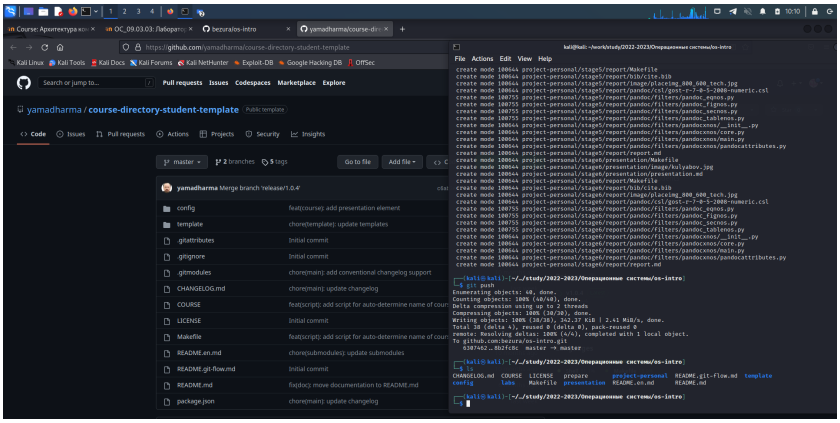

Рис. 2.10: Настройка рабочего пространства(2)

#Контрольные вопросы

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

Место где храняться какие-либо данные и к которому можно получить доступ, менять(сохраняя историю изменений). Используется для написания больших программ командой разрабочиков и в многих других задачах

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

Хранилище - то место что является эталоном

Commit - коментарий к изменным файлам

История - кто когда и что редактировал

Рабочая копия - копия на ПК, в которой производятся изменения, а далее обратно выгружается на сервер

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

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.

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

Один человек загрузил и выгрузил обратно. Хранилище у него на пк(вероятнее всего)

5)Опишите порядок работы с общим хранилищем VCS.

Хоть сколько человек загрузили и обновии данные. Сохраняется кто что изменял.

6)Каковы основные задачи, решаемые инструментальным средством git?

История изменнений, лёгкость правок и отката.

7)Назовите и дайте краткую характеристику командам git.

```
Перечислим 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

Просмотр текущих изменений:
```

```
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 катало

    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 реда

    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

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 репозиторием

Локальный репозиторий - фрилансер один пишет код.

Удалённый репозиторий - например на гитхабе репозиторий большого проекта команды

9)Что такое и зачем могут быть нужны ветви (branches)?
История изменений, правок - даёёт удобство. Или же можно посмотреть вариацию изме

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

Не изменные файлы остаются такими же

3 Выводы

На данной лабороторной, я закрепил знания полученные в прошлом семестре по работе с git.