Отчёт по лабораторной работе №2

Дисциплина: Архитектура Компьютеров и Операционные Системы

Вакутайпа Милдред

Содержание

1 Цель работы **4**

2 Задание **5**

3 Выполнение лабораторной работы **6**
3.1 Создание базовой конфигурации для работы с git. 6
3.2 Создание ключ ssh 7
3.3 Создание ключ gpg 8
3.4 Создание локального каталога для выполнения заданий. 11

4 Выводы **14**

5 Ответы на контрольные вопросы **15**

6 Список литературы **18**

Список иллюстраций

3.1 Установление git 6
3.2 Установление gh 6
3.3 имя и email владельца 7
3.4 имя начальной ветки и паррамеры 7
3.5 Создание ключ ssh 7
3.6 Создание ключ gpg 8
3.7 Настройки ключ gpg 8
3.8 личная информация 9
3.9 аккаунт на git 9
3.10 список ключей 9
3.11 Установление xclip 10
3.12 Копирование ключ gpg 10
3.13 Добавлен ключ gpg 10
3.14 указиваю Git 10
3.15 авторизацию в gh 11
3.16 Авторизоваться через броузер. 11
3.17 Завершена авторизация 11
3.18 Создание каталог 12
3.19 Создание каталог 12
3.20 Удаление файла 12
3.21 Создани необходимых каталогов 12
3.22 Отправление файлы на сервер 13

1 Цель работы

Изучение идеалогии, применение средств контроля версий и освоение умения по работе с git.

2 Задание

1. Создать базовую конфигурацию для работы с git.
2. Создать ключ SSH.
3. Создать ключ PGP.
4. Настроить подписи git.
5. Зарегистрироваться на Github.
6. Создать локальный каталог для выполнения заданий по предмету.

3 Выполнение лабораторной работы

3.1 Создание базовой конфигурации для работы с git.

Установливаю git используя “dnf install git”:

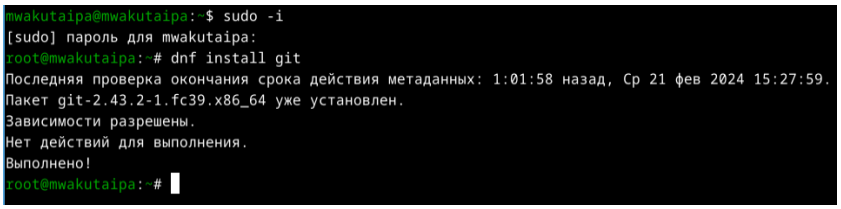

Рис. 3.1: Установление git

С помощью dnf install gh, установливаю gh:

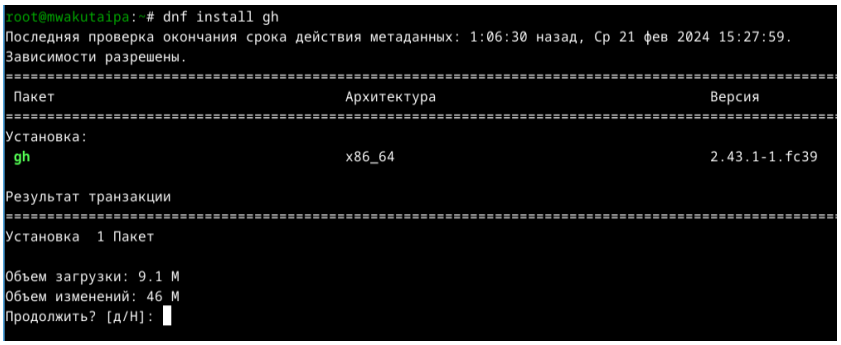

Рис. 3.2: Установление gh

В качестве имя и email владельца репозитории задаю свои имя и email и настраиваю utf-8:

```
root@mwakutaipa:~# git config --global user.name "wakutaipa"
root@mwakutaipa:~# git config --global user.email "1032239009@pfur.ru"
root@mwakutaipa:~# git config --global core.quotepath false
root@mwakutaipa:~#
```

Рис. 3.3: имя и email владельца

Задаю имя начальной ветки и паррамеры autocrlf и safecrlf:

```
root@mwakutaipa:~# git config --global init.defaultBranch master
root@mwakutaipa:~# git config --global core.autocrlf input
root@mwakutaipa:~# git config --global core.safecrlf warn
root@mwakutaipa:~#
```

Рис. 3.4: имя начальной ветки и паррамеры

3.2 Создание ключ ssh

Создаю ключи ssh по алгоритму rsa с размером 4096 бит:

```
root@mwakutaipa:~# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:4r8DQGUJePOG9Uzi8qWJJqSHwEQfF5SL8D7g6wHGs6c root@mwakutai
The key's randomart image is:
+---[RSA 4096]----+
|.. oo*=.         |
| oo *o+ .        |
|o o+.*.=         |
|oo.o+.+ +        |
|+Bo  *.+S        |
|=.=oo.=.         |
| +.+. ..         |
| .+    ..        |
|.E      oo       |
+----[SHA256]-----+
```

Рис. 3.5: Создание ключ ssh

3.3 Создание ключ gpg

Генерирую ключ gpg –full-generate-key:

Рис. 3.6: Создание ключ gpg

Из предложенных опций выбираю тип RSA and RSA; размер 4096; срок действия 0:

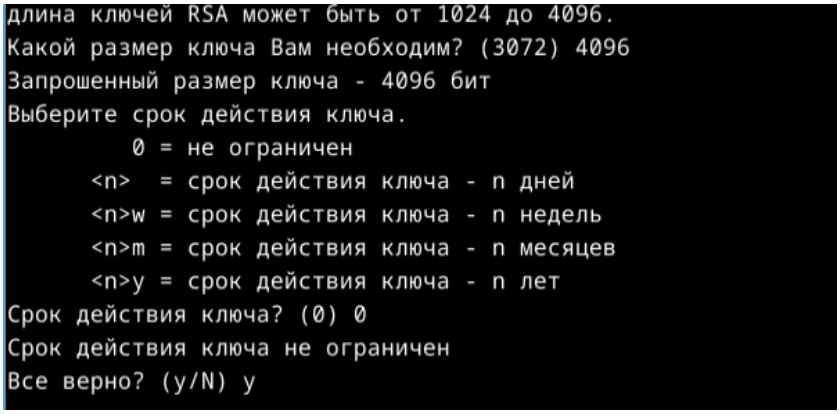

Рис. 3.7: Настройки ключ gpg

GPG запросил личную информацию, которая сохранится в ключе Имя и адрес электронной почты:

```
GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: wakutaipa
Адрес электронной почты: 1032239009@pfur.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
    "wakutaipa <1032239009@pfur.ru>"
```

Рис. 3.8: личная информация

У меня уже есть аккаунт на github, поэтому я вхожу в систему:

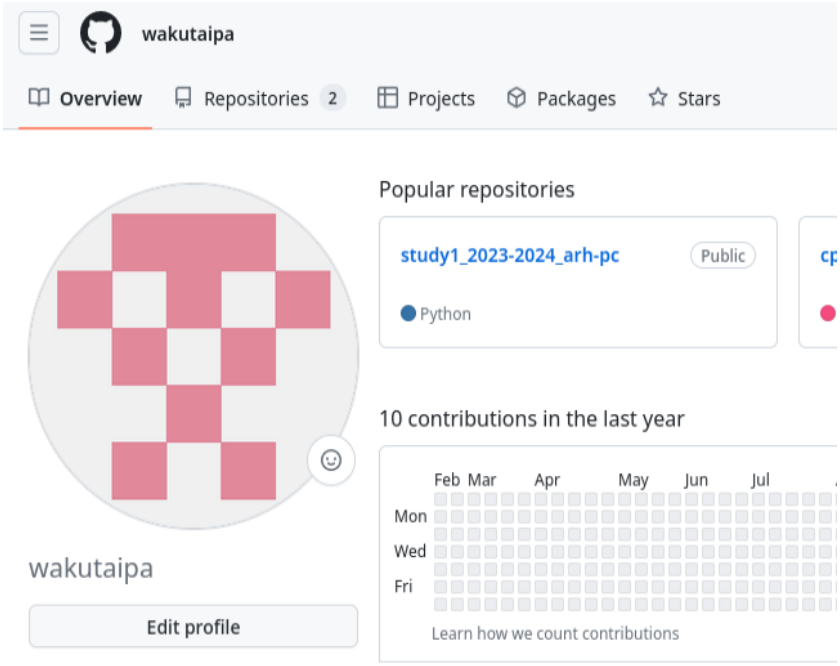

Рис. 3.9: аккаунт на git

Вывожу список ключей:

```
root@mwakutaipa:~# gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: глубина: 0  достоверных:   1  подписанных:   0  доверие: 0-, 0q, 0n, 0m, 0f, 1u
[keyboxd]
---------
sec   rsa4096/6A5ADB668F95864D 2024-02-21 [SC]
      14AA17C9292DC5C5B2E08AA96A5ADB668F95864D
uid                 [  абсолютно ] wakutaipa <1032239009@pfur.ru>
ssb   rsa4096/3B43020BC3C11A85 2024-02-21 [E]
```

Рис. 3.10: список ключей

Установливаю xclip:

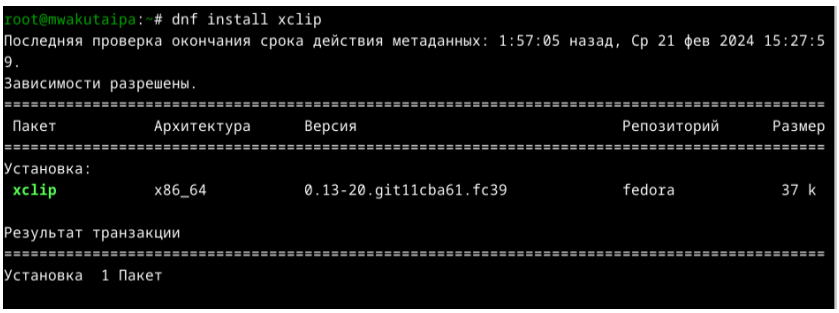

Рис. 3.11: Установление xclip

Скопирую сгенерированный gpg ключ в буфер обмена:

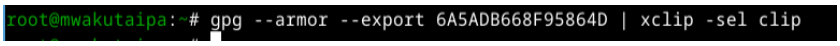

Рис. 3.12: Копирование ключ gpg

Далее перехожу в настройки GitHub, нажимаю на кнопку New GPG key и вставляю полученный ключ:

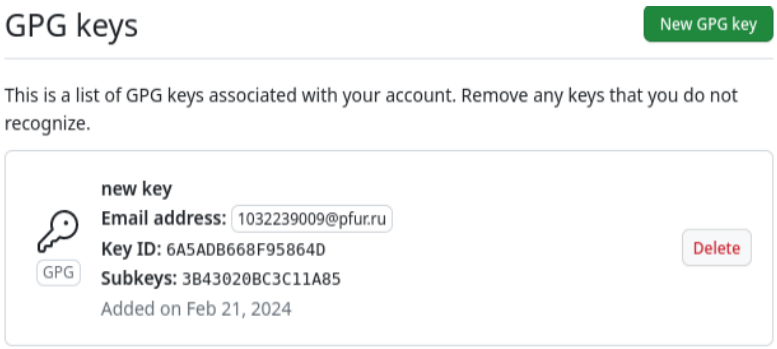

Рис. 3.13: Добавлен ключ gpg

Используя введёный email, указываю Git применять его при подписи коммитов:

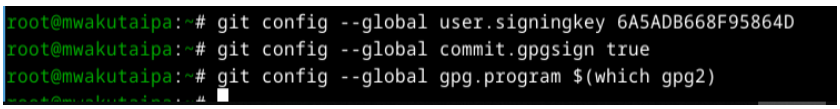

Рис. 3.14: указываю Git

Начинаю авторизацию в gh используя gh auth login:

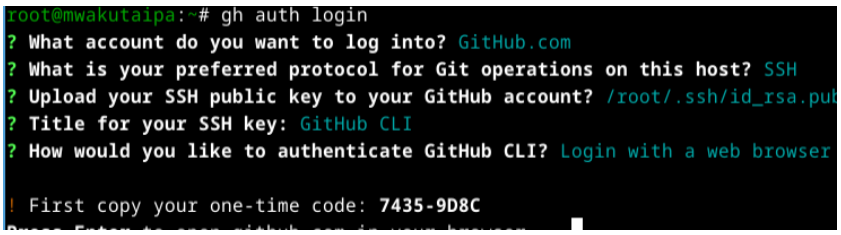

Рис. 3.15: авторизацию в gh

Завершаю авторизацию на броузер:

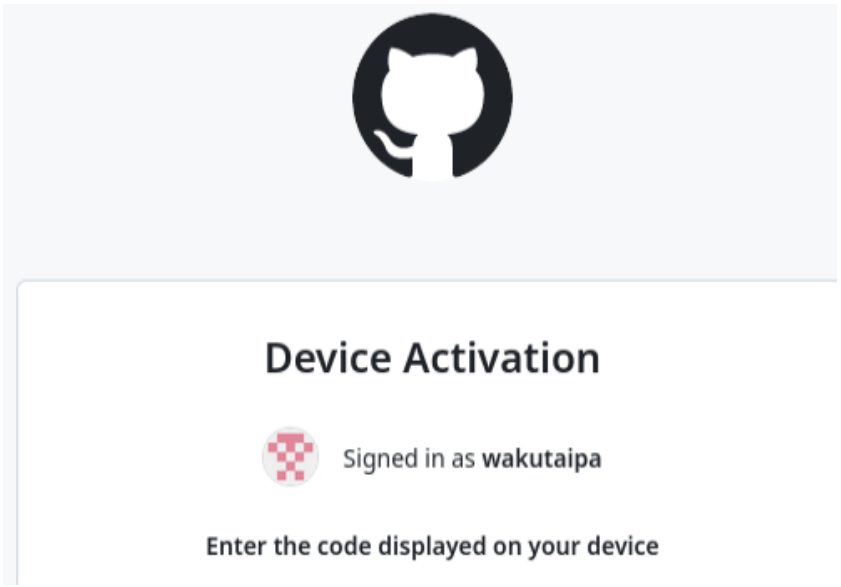

Рис. 3.16: Авторизоваться через броузер.

Рис. 3.17: Завершена авторизация

3.4 Создание локального каталога для выполнения заданий.

Создаю каталог “mkdir -p ~/work/study/2022-2023/”Операционные системы”:

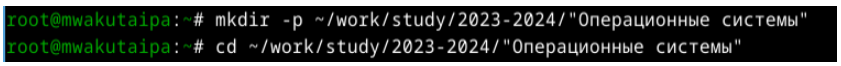

Рис. 3.18: Создание каталог

Перехожу в созданный каталог:

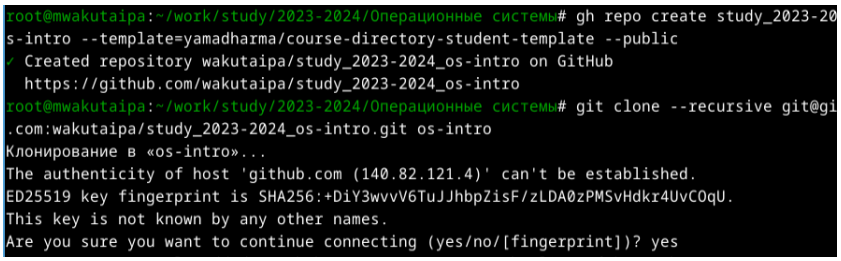

Рис. 3.19: Создание каталог

Удаляю лишные файлы:

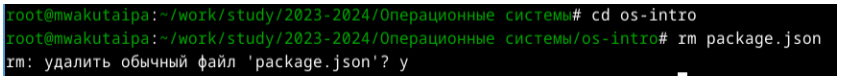

Рис. 3.20: Удаление файла

Создаю еще необходимые каталоги:

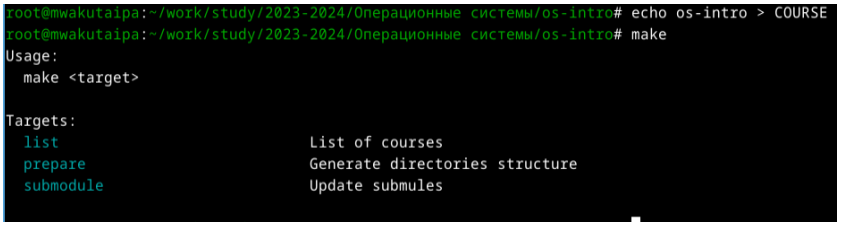

Рис. 3.21: Создани необходимых каталогов

Отправляю Файлы на сервер:

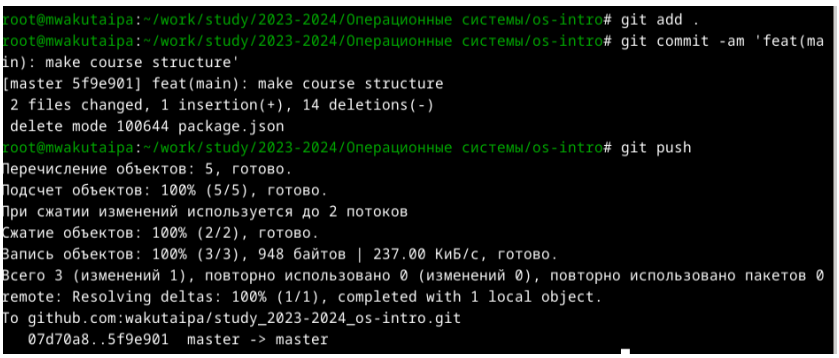

Рис. 3.22: Отправление файлы на сервер

4 Выводы

При выполнении лабораторной работы я изучила идеалогию, применение средств контроля версий и освоеила умение по работе с git.

5 Ответы на контрольные вопросы

1. Системы Контроля Версий - Программные инструменты, помагающие командам разработчиков управлять изменениями в исходном коде с течением времени.

2. Хранилище - в нем храняются все документы, включая историю их изменение и прочей служебной информацией.

   commit - отслеживание измененийб сохраняет разницу в изменениях.

   история - Хранит все изменения в проекте и позволяет при необходимости обратиться к нужным данным.

   рабочая копия- копия проекта основанная на версии из хранилища.

3. В ценрализованном VCS например AccuRev, каждый пользователь копирует себе необходимые ему файлы из репзитория, изменяет их а затем добавляет измениения обратно в хранилище. В децентрализованном VCS например Git, есть возможность добавлять и забирать изменения из любого репозитория.

4. Сначала создается и подключается удаленный репозиторий, затем по мере изменения проекта эти изменения отправляются на сервер.

5. Участник проекта перед началом работы получает нужную ему версию проекта в хранилище, с помощью определенных команд, после внесения изменений пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются. К ним можно вернуться в любой момент.

6. Хранение информации о всех изменениях в вашем коде, обеспечение удобства командной работы над кодом.

Создание основного дерева репозитория: git init

7. 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 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 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. git push -all отправляем из локального репозитория все сохраненные изменения в центральный репозиторий, предварительно создав локальный репозиторий и сделав предварительную конфигурацию.

9. Ветвление - один из параллельных участков в одном хранилище, исходящих из одной версии, обычно есть главная ветка. Между ветками, т. е. их концами возможно их слияние. Используются для разработки новых функций.

10. Во время работы над проектом могут создаваться файлы, которые не следуют добавлять в репозиторий. Например, временные файлы. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов.

6 Список литературы

::: Архитектура ЭВМ :::