

# **Отчёт по лабораторной работе №2**

**Дисциплина: Архитектура Компьютеров и Операционные Системы**

Вакутайпа Милдред

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
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
<b>4</b>	<b>Выводы</b>	<b>14</b>
<b>5</b>	<b>Ответы на контрольные вопросы</b>	<b>15</b>
<b>6</b>	<b>Список литературы</b>	<b>18</b>

# Список иллюстраций

3.1	Установка git . . . . .	6
3.2	Установка gh . . . . .	6
3.3	имя и email владельца . . . . .	7
3.4	имя начальной ветки и параметры . . . . .	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	Установка xclip . . . . .	10
3.12	Копирование ключ gpg . . . . .	10
3.13	Добавлен ключ gpg . . . . .	10
3.14	указываю Git . . . . .	10
3.15	авторизацию в gh . . . . .	11
3.16	Авторизоваться через браузер. . . . .	11
3.17	Завершена авторизация . . . . .	11
3.18	Создание каталог . . . . .	12
3.19	Создание каталог . . . . .	12
3.20	Удаление файла . . . . .	12
3.21	Созданы необходимых каталогов . . . . .	12
3.22	Отправление файлы на сервер . . . . .	13

# 1 Цель работы

Изучение идеологии, 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.

Устанавливаю git используя “dnf install git”:

```
mwakutaipa@mwakutaipa:~$ sudo -i
[sudo] пароль для mwakutaipa:
root@mwakutaipa:~# dnf install git
Последняя проверка окончания срока действия метаданных: 1:01:58 назад, Ср 21 фев 2024 15:27:59.
Пакет git-2.43.2-1.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
root@mwakutaipa:~#
```

Рис. 3.1: Установление git

С помощью dnf install gh, устанавливаю gh:

```
root@mwakutaipa:~# dnf install gh
Последняя проверка окончания срока действия метаданных: 1:06:30 назад, Ср 21 фев 2024 15:27:59.
Зависимости разрешены.
=====
Пакет                               Архитектура                     Версия
=====
Установка:
gh                                   x86_64                           2.43.1-1.fc39

Результат транзакции
=====
Установка 1 Пакет

Объем загрузки: 9.1 М
Объем изменений: 46 М
Продолжить? [д/Н]:
```

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 параметры 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 параметры

## 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:
SHA256:4r8DQGUJePOG9Uzi8qWJJqSHwEQfF5SL8D7g6wHG6c 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:

```
root@mwakutaipa:~# gpg --full-generate-key
gpg (GnuPG) 2.4.3; Copyright (C) 2023 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: создан каталог '/root/.gnupg'
Выберите тип ключа:
  (1) RSA and RSA
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (9) ECC (sign and encrypt) *default*
 (10) ECC (только для подписи)
 (14) Existing key from card
Ваш выбор? 1
```

Рис. 3.6: Создание ключ gpg

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

```
длина ключей RSA может быть от 1024 до 4096.
Какой размер ключа Вам необходим? (3072) 4096
Запрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
  0 = не ограничен
  <n> = срок действия ключа - n дней
  <n>w = срок действия ключа - n недель
  <n>m = срок действия ключа - n месяцев
  <n>y = срок действия ключа - n лет
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (y/N) y
```

Рис. 3.7: Настройки ключ gpg

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



```

gpgPG 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:

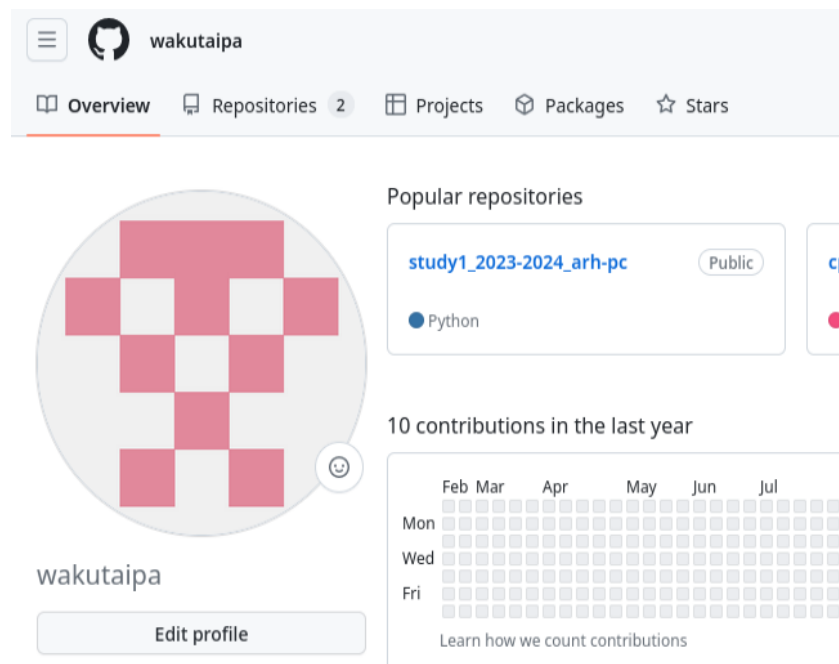


Рис. 3.9: аккаунт на git

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
[keyboard]
-----
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

Устанавливаю xclip:

```
root@mwakutaipa:~# dnf install xclip
Последняя проверка окончания срока действия метаданных: 1:57:05 назад, Ср 21 фев 2024 15:27:59.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
=====
Установка:
xclip      x86_64      0.13-20.git11cba61.fc39  fedora      37 k
Результат транзакции
=====
Установка 1 Пакет
```

Рис. 3.11: Установление xclip

Скопирую сгенерированный gpg ключ в буфер обмена:

```
root@mwakutaipa:~# gpg --armor --export 6A5ADB668F95864D | xclip -sel clip
```

Рис. 3.12: Копирование ключ gpg

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

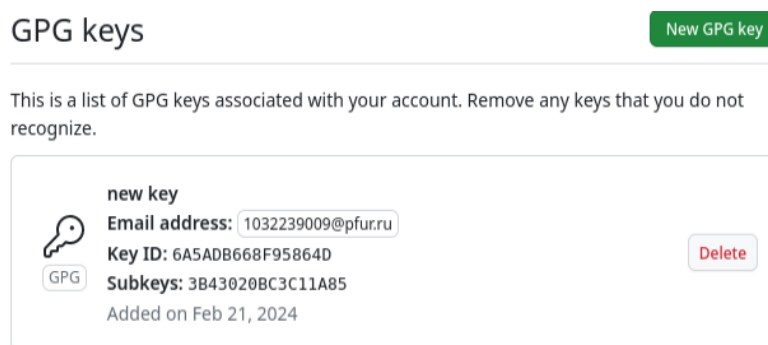


Рис. 3.13: Добавлен ключ gpg

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

```
root@mwakutaipa:~# git config --global user.signingkey 6A5ADB668F95864D
root@mwakutaipa:~# git config --global commit.gpgsign true
root@mwakutaipa:~# git config --global gpg.program $(which gpg2)
```

Рис. 3.14: указываю Git

Начинаю авторизацию в gh используя gh auth login:

```

root@mwakutaipa:~# gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /root/.ssh/id_rsa.pub
? Title for your SSH key: GitHub CLI
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 7435-9D8C
Press Enter to open github.com in your browser

```

Рис. 3.15: авторизацию в gh

Завершаю авторизацию на броузер:

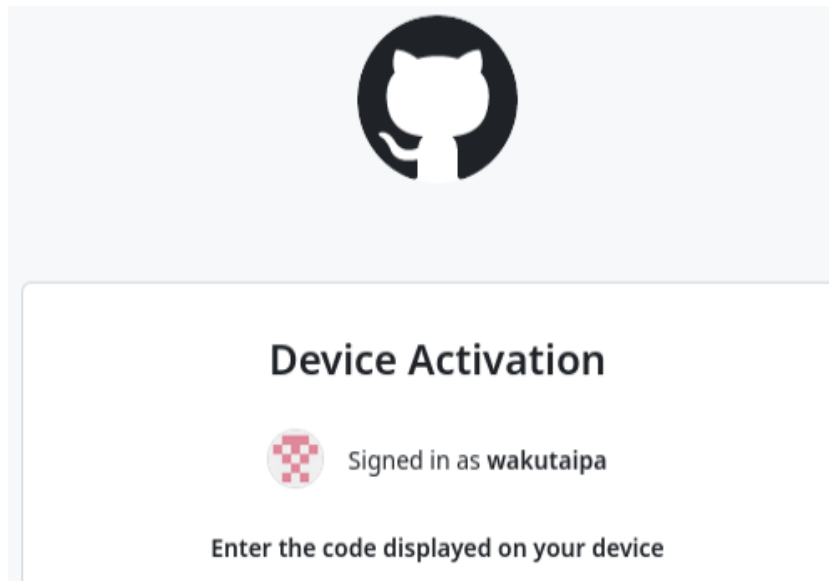


Рис. 3.16: Авторизоваться через броузер.

```

✓ Authentication complete.
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol
! Authentication credentials saved in plain text
✓ Uploaded the SSH key to your GitHub account: /root/.ssh/id_rsa.pub
✓ Logged in as wakutaipa

```

Рис. 3.17: Завершена авторизация

### 3.4 Создание локального каталога для выполнения заданий.

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

```
root@mwakutaipa:~# mkdir -p ~/work/study/2023-2024/"Операционные системы"
root@mwakutaipa:~# cd ~/work/study/2023-2024/"Операционные системы"
```

Рис. 3.18: Создание каталог

Перехожу в созданный каталог:

```
root@mwakutaipa:~/work/study/2023-2024/Операционные системы# gh repo create study_2023-2024-os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository wakutaipa/study_2023-2024_os-intro on GitHub
https://github.com/wakutaipa/study_2023-2024_os-intro
root@mwakutaipa:~/work/study/2023-2024/Операционные системы# git clone --recursive git@github.com:wakutaipa/study_2023-2024_os-intro.git os-intro
Клонирование в «os-intro»...
The authenticity of host 'github.com (140.82.121.4)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvC0qU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

Рис. 3.19: Создание каталог

Удаляю лишние файлы:

```
root@mwakutaipa:~/work/study/2023-2024/Операционные системы# cd os-intro
root@mwakutaipa:~/work/study/2023-2024/Операционные системы/os-intro# rm package.json
rm: удалить обычный файл 'package.json'? y
```

Рис. 3.20: Удаление файла

Создаю еще необходимые каталоги:

```
root@mwakutaipa:~/work/study/2023-2024/Операционные системы/os-intro# echo os-intro > COURSE
root@mwakutaipa:~/work/study/2023-2024/Операционные системы/os-intro# make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare       Generate directories structure
  submodule     Update submodules
```

Рис. 3.21: Создани необходимых каталогов

Отправляю Файлы на сервер:

```
root@mmakutaipa:~/work/study/2023-2024/Операционные системы/os-intro# git add .
root@mmakutaipa:~/work/study/2023-2024/Операционные системы/os-intro# git commit -am 'feat(main): make course structure'
[master 5f9e901] feat(main): make course structure
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
root@mmakutaipa:~/work/study/2023-2024/Операционные системы/os-intro# git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 948 байтов | 237.00 КиБ/с, готово.
Всего 3 (изменений 1), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:wakutaipa/study_2023-2024_os-intro.git
   07d70a8..5f9e901  master -> master
```

Рис. 3.22: Отправление файлы на сервер

## 4 Выводы

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

## 5 Ответы на контрольные вопросы

1. Системы Контроля Версий - Программные инструменты, помогающие командам разработчиков управлять изменениями в исходном коде с течением времени.
2. Хранилище - в нем хранятся все документы, включая историю их изменение и прочей служебной информацией.  
  
commit - отслеживание изменений 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 централизованном VCS например AccuRev, каждый пользователь копирует себе необходимые ему файлы из репозитория, изменяет их а затем добавляет изменения 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 :::