

Отчёт по лабораторной работе №2

Операционные системы

Пономарева Татьяна Александровна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Установка программного обеспечения	8
4.2	Базовая настройка git	8
4.3	Верификация коммитов с помощью PGP	9
4.3.1	Создание ключа	9
4.3.2	Экспорт ключа	9
4.3.3	Добавление PGP ключа в GitHub	10
4.4	Базовая настройка git	11
4.5	Создание ключа ssh	11
4.6	Настройка gh	13
4.7	Настройка автоматических подписей коммитов git	13
4.8	Создание репозитория курса на основе шаблона	14
4.9	Настройка каталога курса	14
5	Контрольные вопросы	15
6	Выводы	18
	Список литературы	19

Список иллюстраций

4.1	Установка git и gh	8
4.2	Задание имени владельца и email, настройка utf-8	8
4.3	Генерация ключа	9
4.4	Вывод списка ключей	10
4.5	Экспорт ключа	10
4.6	Добавление ключа в GitHub	11
4.7	Настройка git	11
4.8	Ключ ssh	12
4.9	Ключ ssh	12
4.10	Добавление ключа в GitHub	13
4.11	Настройка git	13
4.12	Настройка подписей коммитов git	13
4.13	Создание репозитория	14
4.14	Настройка каталога курса	14

Список таблиц

1 Цель работы

Изучить идеологию и применение средств контроля версий и освоить 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 Теоретическое введение

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.

4.3 Верификация коммитов с помощью PGP

4.3.1 Создание ключа

Генерирую ключ, задаю следующие параметры: RSA и RSA, размер 4096, срок действия бессрочен, имя: tanya, email: taonomareva6742@gmail.com, comment: test (рис. 4.3).

```
[taonomareva@taonomareva ~]$ gpg --full-generate-key
gpg: directory '/home/taonomareva/.gnupg' created
Please select what kind of key you want:
(1) RSA and RSA
(2) RSA and ElGamal
(3) DSA (sign only)
(4) RSA (sign only)
(5) ECC (sign and encrypt) "default"
(10) ECC (sign only)
(14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 4096
Requested keysize is 4096 bits
Please specify how long the key should be valid.
0 = key does not expire
<n> = key expires in n days
<w> = key expires in n weeks
<m> = key expires in n months
<y> = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: tanya
Email address: taonomareva6742@gmail.com

GnuPG needs to construct a user ID to identify your key.

Real name: tanya
Email address: taonomareva6742@gmail.com
Comment: test
You selected this USER ID:
"tanya (test) <taonomareva6742@gmail.com>"

Change (Name, Comment, Email or Okey/Quit)?
Change (Name, Comment, Email or Okey/Quit)? 0
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disk) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disk) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: /home/taonomareva/.gnupg/trustdb.gpg: trustdb created
gpg: directory '/home/taonomareva/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/taonomareva/.gnupg/openpgp-revocs.d/21428E8475C38D6A08F5E389C2786F485829698.rev'
public and secret key created and signed.

pub.  rsa4096 2025-03-04 [SC]
uid.  21428E8475C38D6A08F5E389C2786F485829698
uid.  tanya (test) <taonomareva6742@gmail.com>
sub.  rsa4096 2025-03-04 [E]
```

Рис. 4.3: Генерация ключа

4.3.2 Экспорт ключа

Вывожу список ключей (рис. 4.4).

```
[taponomareva@taponomareva ~]$ gpg --list-secret-keys --keyid-format LONG
gpg: checking the trustdb
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
[keyboard]
-----
sec   rsa4096/9C27B6F405B29698 2025-03-04 [SC]
      21428E04475C38D6008F5E389C27B6F405B29698
uid           [ultimate] tanya (test) <taponomareva6742@gmail.com>
ssb   rsa4096/305ADF208800C108 2025-03-04 [E]
```

Рис. 4.4: Вывод списка ключей

Экспортирую ключ по его отпечатку (рис. 4.5).

```
[taponomareva@taponomareva ~]$ gpg --armor --export 9C27B6F405B29698
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGfGkHQBEADF/CwkpHSPmfFvhpwPo11C3E8CaymQJKI/+oWeCOLVDRckaQ/3
X06JA9m61RNfFOiAodQQS1NNzkEhQaHm3131mwrMK98b20Z39aSKjBPgz0JBprA
CY+smbbVKvBzv1x5Gn7ZmtXU5dWsf0PS1VfTTruXyuQMxLH2X83L9HjROyAb482BX
```

Рис. 4.5: Экспорт ключа

4.3.3 Добавление PGP ключа в GitHub

Копирую ключ и добавляю его в настройках профиля на GitHub (рис. 4.6).

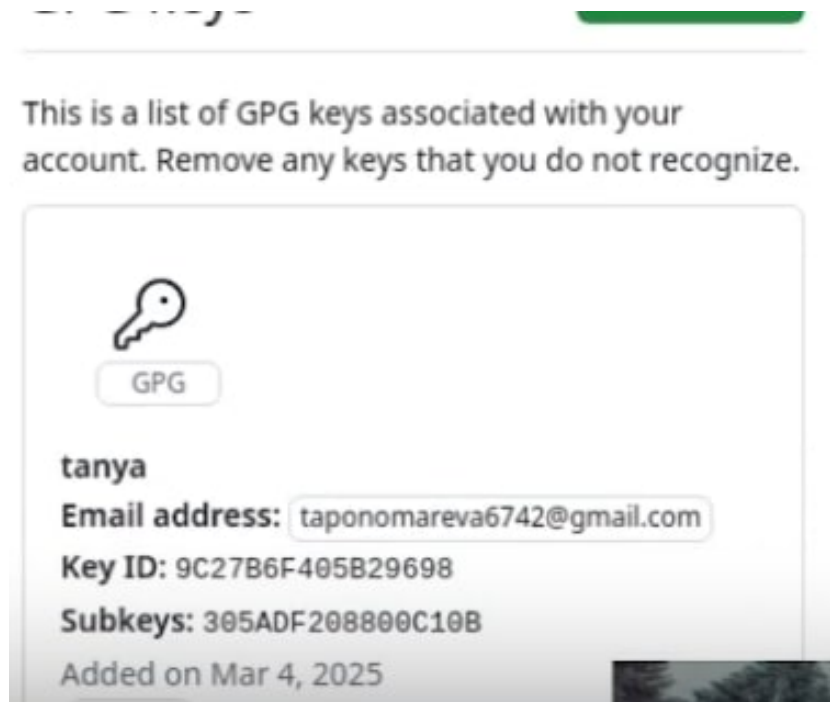


Рис. 4.6: Добавление ключа в GitHub

4.4 Базовая настройка git

Задаю имя начальной ветки (master) (рис. 4.7).

```
[taponomareva@taponomareva ~]$ git config --global init.defaultBranch master
[taponomareva@taponomareva ~]$ git config --global core.autocrlf input
[taponomareva@taponomareva ~]$ git config --global core.safecrlf warn
```

Рис. 4.7: Настройка git

4.5 Создание ключа ssh

Создаю ключ ssh по алгоритму rsa с ключем размером 4096 бит(рис. 4.8).

```
[taonomareva@taonomareva ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/taonomareva/.ssh/id_rsa):
Created directory '/home/taonomareva/.ssh'.
Enter passphrase for "/home/taonomareva/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/taonomareva/.ssh/id_rsa
Your public key has been saved in /home/taonomareva/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:twpQc6++kMqJptj0yo1xdehyqen8haxEFVDAP8qlshDM taonomareva@taonomareva
The key's randomart image is:
+----[RSA 4096]-----+
| ..*+. |
| . 0 =. |
| E . =.0 |
| = 0.+ . |
| = ..S 0 |
| ..00+.0 . |
| .0. .++0.. |
| .0B+00=.. |
| .+*+*+0.+ |
+-----[SHA256]-----+
[taonomareva@taonomareva ~]$
```

Рис. 4.8: Ключ ssh

Создаю ключ ssh по алгоритму ed25519(рис. 4.9).

```
[taonomareva@taonomareva ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/taonomareva/.ssh/id_ed25519):
Enter passphrase for "/home/taonomareva/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/taonomareva/.ssh/id_ed25519
Your public key has been saved in /home/taonomareva/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:caevA1/WSUijvmb0UtE84pjTNY6WkgVAgATVWn0K4 taonomareva@taonomareva
The key's randomart image is:
+--[ED25519 256]--+
| .00++==+.0 |
| o. o++B + |
| . o+B.X . |
| . 00B=0. |
| S..+. 0 |
| Eo.o+ o |
| 000+. |
| . 00.. |
| . 00.. |
+-----[SHA256]-----+
[taonomareva@taonomareva ~]$
```

Рис. 4.9: Ключ ssh

Ключ был добавлен в GitHub (рис. 4.10).

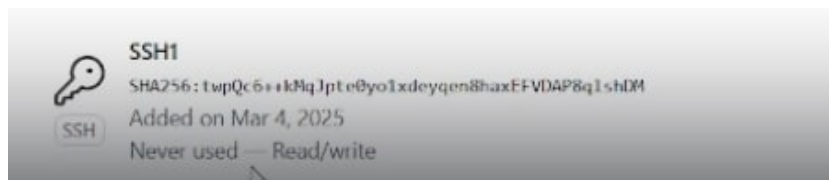


Рис. 4.10: Добавление ключа в GitHub

4.6 Настройка gh

Авторизовываюсь через команду `gh auth login` (рис. 4.11).

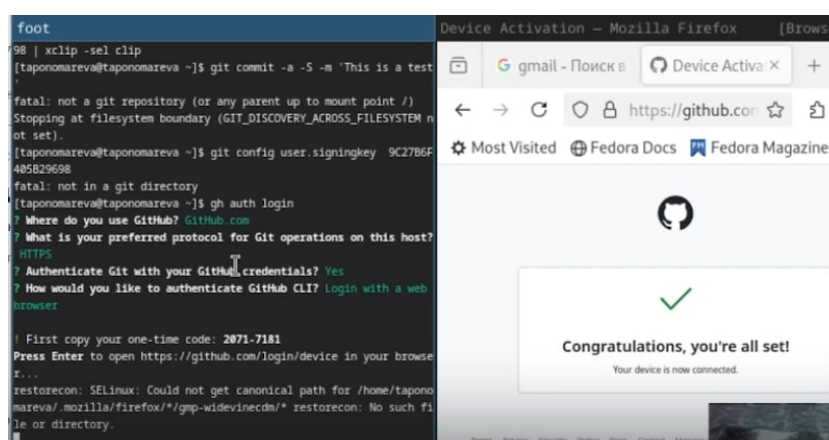


Рис. 4.11: Настройка git

4.7 Настройка автоматических подписей коммитов git

Используя введенный email, указываю Git применять его при подписи коммитов (рис. 4.12).

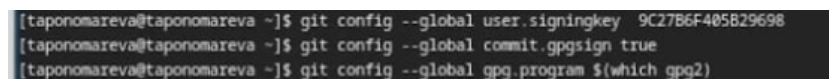


Рис. 4.12: Настройка подписей коммитов git

4.8 Создание репозитория курса на основе шаблона

Создаю репозиторий курса на основе шаблона (рис. 4.13).

```
[taponomareva@taponomareva ~]$ mkdir -p ~/work/study/2024-2025/"Операционные системы"
[taponomareva@taponomareva ~]$ cd ~/work/study/2024-2025/"Операционные системы"
[taponomareva@taponomareva Операционные системы]$ gh repo create study_2024-2025_os-intro --template=yamadharma/course-directory-student-template --public
/ Created repository taponomareva/study_2024-2025_os-intro on GitHub
https://github.com/taponomareva/study_2024-2025_os-intro
[taponomareva@taponomareva Операционные системы]$ git clone --recursive git@github.com:taponomareva/study_2024-2025_os-intro.git os-intro
Cloning into 'os-intro'...
The authenticity of host 'github.com [148.82.121.4]' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvv66WUJhbpZisF/zLDAhzPMSvHdkr4UvC0qU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Рис. 4.13: Создание репозитория

4.9 Настройка каталога курса

Перехожу в каталог и удаляю лишние файлы, затем создаю необходимые каталоги, далее отправляю файлы на сервер (рис. 4.14).

```
[taponomareva@taponomareva Операционные системы]$ cd ~/work/study/2024-2025/"Операционные системы"/os-intro
[taponomareva@taponomareva os-intro]$ rm package.json
[taponomareva@taponomareva os-intro]$ echo os-intro > COURSE
[taponomareva@taponomareva os-intro]$ make
Usage:
  make <target>

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

[taponomareva@taponomareva os-intro]$ prepare
bash: prepare: command not found
[taponomareva@taponomareva os-intro]$ make prepare
[taponomareva@taponomareva os-intro]$ ls
CHANGELOG.md  COURSE  LICENSE  prepare  project-personal  README.git-flow.md  template
config        labs   Makefile  presentation  README.en.md      README.md

[taponomareva@taponomareva os-intro]$ git add .
[taponomareva@taponomareva os-intro]$ git commit -am 'feat(main): make course structure'
[taponomareva@taponomareva os-intro]$ git push
```

Рис. 4.14: Настройка каталога курса

5 Контрольные вопросы

1. Системы контроля версий (VCS) — это инструменты, предназначенные для отслеживания изменений в файлах или наборе файлов. Они позволяют сохранять версии данных и управлять ими, что важно при разработке программного обеспечения, написании текстов или любых других длительных проектных работах, требующих отслеживания изменений. VCS помогают работать с историей изменений, сотрудничать с другими людьми и восстанавливать предыдущие версии файлов.
2. Пояснение понятий VCS:
 - Хранилище (repository) — это место, где хранится вся информация о проекте, включая файлы и историю изменений.
 - Commit — это операция, при которой изменения в рабочих файлах фиксируются в хранилище с добавлением сообщения, объясняющего изменения.
 - История (history) — это последовательность коммитов, представляющая изменения, которые были сделаны в проекте за время его существования.
 - Рабочая копия (working copy) — это локальная версия файлов проекта, с которыми пользователь работает. Изменения сначала происходят в рабочей копии и затем фиксируются в хранилище с помощью коммита.
3. Централизованные и децентрализованные VCS:
 - Централизованные VCS предполагают наличие одного центрального хранилища, к которому подключаются все пользователи. Все изменения сохраняются в этом едином месте. Пример: Subversion (SVN).

- Децентрализованные VCS позволяют каждому пользователю иметь локальное хранилище, синхронизируя изменения между репозиториями. Каждый участник может работать автономно, а затем обмениваться данными с другими. Пример: Git.

4. Примеры VCS:

- Централизованные: Subversion (SVN), CVS.
- Децентрализованные: Git, Mercurial.

5. Действия с VCS при единоличной работе:

Инициализация репозитория (создание нового хранилища). Изменение файлов в рабочей копии. Добавление новых файлов в индекс с помощью команды `git add`. Фиксация изменений в хранилище с помощью `git commit`. Просмотр истории изменений с помощью `git log`.

6. Порядок работы с общим хранилищем VCS:

Клонирование репозитория с удалённого сервера с помощью `git clone`. Синхронизация с удалённым хранилищем с помощью `git fetch` или `git pull`. Редактирование локальных файлов и фиксация изменений через `git commit`. Отправка изменений в удалённое хранилище через `git push`. Разрешение конфликтов при слиянии изменений с помощью `git merge`.

7. Основные задачи, решаемые инструментальным средством Git:

Отслеживание изменений в проекте. Совместная работа с другими пользователями. Ветвление и слияние изменений (работа с ветками). Управление удалёнными репозиториями и синхронизация изменений. Восстановление предыдущих версий файлов и проекта.

8. Команды Git:

`git init` — инициализация нового репозитория. `git clone` — клонирование удалённого репозитория. `git add` — добавление изменений в индекс (подготовка файлов к коммиту). `git commit -m "message"` — фиксация изменений в репозитории с сообщением. `git status` — просмотр текущего состояния рабочей копии. `git log` — просмотр истории коммитов. `git push` — отправка изменений в удалённый репозиторий. `git pull` — получение изменений с удалённого репозитория. `git branch` — создание и управление ветками.

9. Примеры использования Git с локальными и удалёнными репозиториями:

Локальный репозиторий: `git init` — инициализация репозитория, `git commit` — фиксация изменений, `git status` — проверка состояния. Удалённый репозиторий: `git clone` — клонирование удалённого репозитория, `git push` — отправка изменений на сервер, `git pull` — получение обновлений с сервера.

10. Ветви в Git позволяют изолировать изменения в проекте, не влияя на основную рабочую версию. Это полезно для параллельной разработки новых функций, исправлений ошибок или экспериментов, которые затем могут быть объединены в основную ветку (например, `main` или `master`). Ветки позволяют легко работать с несколькими направлениями разработки и избежать конфликтов между различными задачами.

6 Выводы

Была изучена идеология и применение средств контроля версий и освоены умения по работе с git

Список литературы

1. Курс на ТУИС
2. Лабораторная работа №2