

Отчет по лабораторной работе №2

Операционные системы

Киселева Елизавета Александровна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Установка программного обеспечения	7
3.2	Базовая настройка git	7
3.3	Создание ключа SSH	8
3.4	Создание ключа GPG	10
3.5	Добавление ключа GPG в Github	10
3.6	Настроить подписи Git	11
3.7	Настройка gh	11
3.8	Создание репозитория курса на основе шаблона	12
4	Выводы	14
5	Ответы на контрольные вопросы.	15
	Список литературы	18

Список иллюстраций

3.1	Установка git и gh	7
3.2	Задаю имя и email владельца репозитория	8
3.3	Настройка utf-8 в выводе сообщений git	8
3.4	Задаю имя начальной ветки	8
3.5	Задаю параметры autocrlf и safecrlf	8
3.6	Генерация ssh ключа по алгоритму rsa	9
3.7	Генерация ssh ключа по алгоритму ed25519	9
3.8	Генерация ключа	10
3.9	Вывод списка ключей	10
3.10	Добавление нового PGP ключа	11
3.11	Настройка подписей Git	11
3.12	Авторизация в gh	12
3.13	Создание репозитория	12
3.14	Удаление файлов и создание каталогов	13
3.15	Отправка файлов на сервер	13

Список таблиц

1 Цель работы

Цель данной лабораторной работы – изучение идеологии и применения средств контроля версий, освоение умения по работе с git.

2 Задание

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

3 Выполнение лабораторной работы

3.1 Установка программного обеспечения

Устанавливаю необходимое программное обеспечение git и gh через терминал с помощью команд: `dnf install git` и `dnf install gh` (рис. 3.1).

```
eakiseleva@eakiseleva ~]$ sudo -i
[root@eakiseleva ~]# dnf -y install git
Обновление и загрузка репозитория:
Fedora 41 - x86_64 - Updates      100% | 32.9 KiB/s | 25.9 KiB | 00m01s
Fedora 41 - x86_64 - Updates      100% | 2.6 MiB/s | 2.7 MiB | 00m01s
Репозитории загружены.
Пакет "git-2.48.1-1.fc41.x86_64" уже установлен.

Нечего делать.
[root@eakiseleva ~]# dnf -y install gh
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет                                Арх.    Версия                Репозиторий    Размер
Установка:
gh                                  x86_64  2.65.0-1.fc41         updates        42.6 MiB

Сводка транзакции:
Установка:      1 пакета

Общий размер входящих пакетов составляет 10 MiB. Необходимо загрузить 10 MiB.
После этой операции будут использоваться дополнительные 43 MiB (установка 43 MiB, удаление 0 B).
[1/1] gh-0:2.65.0-1.fc41.x86_64      100% | 6.5 MiB/s | 10.3 MiB | 00m02s
-----
[1/1] Total                          100% | 4.5 MiB/s | 10.3 MiB | 00m02s
```

Рис. 3.1: Установка git и gh

3.2 Базовая настройка git

Задаю в качестве имени и email владельца репозитория свои имя, фамилию и электронную почту (рис. 3.2).

```
[root@eakiseleva ~]# git config --global user.name "Liza kiseleva"
[root@eakiseleva ~]# git config --global user.email "lk3093398@gmail.com"
[root@eakiseleva ~]#
```

Рис. 3.2: Задаю имя и email владельца репозитория

Настраиваю utf-8 в выводе сообщений git для их корректного отображения (рис. 3.3).

```
[root@eakiseleva ~]# git config --global core.quotepath false
[root@eakiseleva ~]#
```

Рис. 3.3: Настройка utf-8 в выводе сообщений git

Начальной ветке задаю имя master (рис. 3.4).

```
[root@eakiseleva ~]# git config --global init.defaultBranch master
[root@eakiseleva ~]#
```

Рис. 3.4: Задаю имя начальной ветки

Задаю параметры autocrlf и safecrlf для корректного отображения конца строки (рис. 3.5).

```
[root@eakiseleva ~]# git config --global core.autocrlf input
[root@eakiseleva ~]# git config --global core.safecrlf warn
[root@eakiseleva ~]#
```

Рис. 3.5: Задаю параметры autocrlf и safecrlf

3.3 Создание ключа SSH

Создаю ключ ssh размером 4096 бит по алгоритму rsa (рис. 3.6).


```

[root@eakiseleva ~]# 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 for "/root/.ssh/id_rsa" (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:fPiYn/SRNQueEI51sWlsCcVUS3V9Jp0y4r4X0U/zbH8 root@eakiseleva
The key's randomart image is:
+---[RSA 4096]-----+
|      .++..oo+|
|      oo*+.=|
|      o 0+.=|
|      . =.=o ...|
|      S.+...o+o|
|      =.o.= o=|
|      o.o =...|
|      o.o.. E|
|      +o. .|
|      +-----[SHA256]-----+
[root@eakiseleva ~]#

```

Рис. 3.6: Генерация ssh ключа по алгоритму rsa

Создаю ключ ssh по алгоритму ed25519 (рис. 3.7).

```

[root@eakiseleva ~]# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
Enter passphrase for "/root/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519
Your public key has been saved in /root/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:ErgYoKJdhola60QUEbtBYXTGoCCpakYs45aj5HJmPg root@eakiseleva
The key's randomart image is:
+--[ED25519 256]--+
|+XB+.          |
|B*O+oo         |
|@== * .        |
|@*O* . .       |
|*+o . . S      |
|++ o .         |
|=.*            |
|.O +          |
|. E .          |
|+-----[SHA256]-----+
[root@eakiseleva ~]#

```

Рис. 3.7: Генерация ssh ключа по алгоритму ed25519

3.4 Создание ключа GPG

Генерирую ключ GPG, затем выбираю тип ключа RSA and RSA, задаю максимальную длину ключа: 4096, оставляю неограниченный срок действия ключа. Далее отвечаю на вопросы программы о личной информации (рис. 3.8).

```
[root@eakiseleva ~]# 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   rsa4096/51C58C4712275F18 2025-03-06 [SC]
      2374A25E5C2BD28432BC06C851C58C4712275F18
uid           [ абсолютно ] Liza kiseleva <lk3093398@gmail.com>
ssb   rsa4096/A27F39CBC58C3614 2025-03-06 [E]
```

Рис. 3.8: Генерация ключа

3.5 Добавление ключа GPG в Github

Вывожу список созданных ключей в терминал, ищу в результате запроса отпечаток ключа (последовательность байтов для идентификации более длинного, по сравнению с самим отпечатком, ключа), он стоит после знака слеша, копирую его в буфер обмена (рис. 3.9).

```
[root@eakiseleva ~]# 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   rsa4096/51C58C4712275F18 2025-03-06 [SC]
      2374A25E5C2BD28432BC06C851C58C4712275F18
uid           [ абсолютно ] Liza kiseleva <lk3093398@gmail.com>
ssb   rsa4096/A27F39CBC58C3614 2025-03-06 [E]
```

Рис. 3.9: Вывод списка ключей

Нажимаю на “New GPG key” и вставляю в поле ключ из буфера обмена (рис. 3.10).

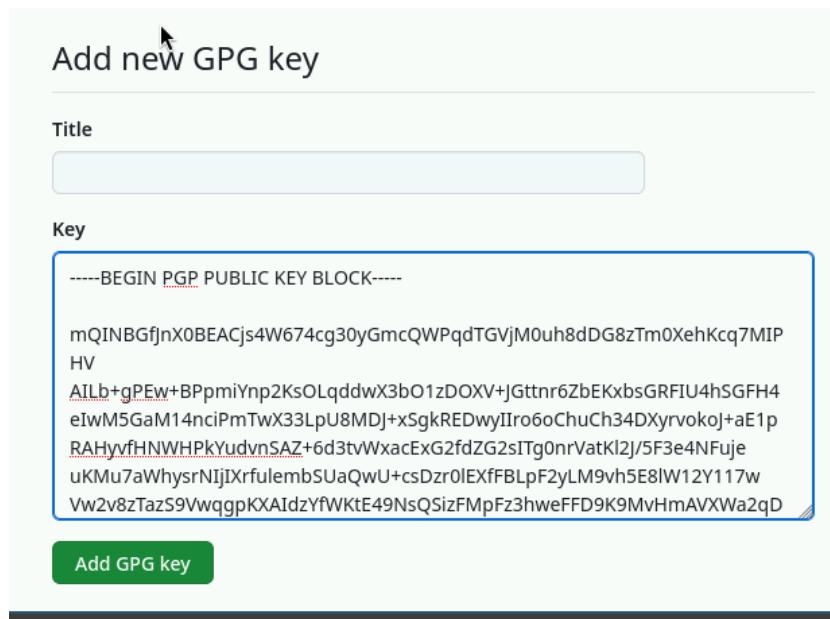


Рис. 3.10: Добавление нового PGP ключа

3.6 Настроить подписи Git

Настраиваю автоматические подписи коммитов git: используя введенный ранее email, указываю git использовать его при создании подписей коммитов (рис. 3.11).

```
eakiseval@eakiseval ~]$ git config --global user.signingkey 3AE07A5AAEFF106
eakiseval@eakiseval ~]$ git config --global commit.gpgsign true
eakiseval@eakiseval ~]$ git config --global gpg.program $(which gpg2)
bash: wich: команда не найдена
eakiseval@eakiseval ~]$ git config --global gpg.program $(which gpg2)
eakiseval@eakiseval ~]$
```

Рис. 3.11: Настройка подписей Git

3.7 Настройка gh

Начинаю авторизацию в gh, отвечаю на наводящие вопросы от утилиты, в конце выбираю авторизоваться через браузер (рис. 3.12).

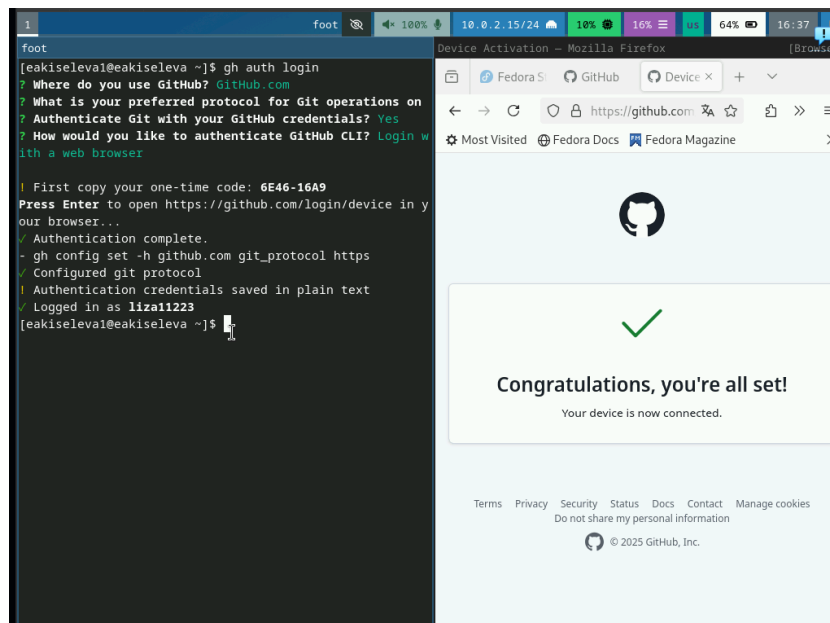


Рис. 3.12: Авторизация в gh

3.8 Создание репозитория курса на основе шаблона

Сначала создаю директорию с помощью утилиты `mkdir` и флага `-p`, который позволяет установить каталоги на всем указанном пути. После этого с помощью утилиты `cd` перехожу в только что созданную директорию “Операционные системы”. Далее в терминале ввожу команду `gh repo create study_2022-2023_os-intro --template yamadharma/course-directory-student-trmplate --public`, чтобы создать репозиторий на основе шаблона репозитория. После этого клонирую репозиторий к себе в директорию, я указываю ссылку с протоколом `https`, а не `ssh`, потому что при авторизации в `gh` выбрала протокол `https` (рис. 3.13).

```
[eakiseleva@eakiseleva ~]$ mkdir -p ~/work/study/2024-2025/ "Операционные системы"
[eakiseleva@eakiseleva ~]$ cd ~/work/study/2024-2025/"Операционные системы"
[eakiseleva@eakiseleva Операционные системы]$ gh repo create study_2024-2025_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository liza11223/study_2024-2025_os-intro on GitHub
https://github.com/liza11223/study_2024-2025_os-intro
```

Рис. 3.13: Создание репозитория

Перехожу в каталог курса с помощью утилиты `cd`, проверяю содержание ка-

талога с помощью утилиты `ls`. Удаляю лишние файлы с помощью утилиты `rm`, далее создаю необходимые каталоги используя `makefile` (рис. 3.14).

```
lisa1@lisa1:~/Операционные системы$ cd study_2024-2025_os-intro
lisa1@lisa1:~/study_2024-2025_os-intro$ rm package.json
lisa1@lisa1:~/study_2024-2025_os-intro$ echo os-intro > COURSE
lisa1@lisa1:~/study_2024-2025_os-intro$ make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare       Generate directories structure
  submodule     Update submules
lisa1@lisa1:~/study_2024-2025_os-intro$
```

Рис. 3.14: Удаление файлов и создание каталогов

Добавляю все новые файлы для отправки на сервер (сохраняю добавленные изменения) с помощью команды `git add` и комментирую их с помощью `git commit` (рис. 3.15).

```
lisa1@lisa1:~/study_2024-2025_os-intro$ git add .
lisa1@lisa1:~/study_2024-2025_os-intro$ git commit -am 'feat(main): make course s
structure'
[master 3eaab6d] feat(main): make course structure
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
lisa1@lisa1:~/study_2024-2025_os-intro$ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 949 байтов | 949.00 КиБ/с, готово.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
to https://github.com/liza11223/study_2024-2025_os-intro.git
 f749c45..3eaab6d master -> master
lisa1@lisa1:~/study_2024-2025_os-intro$
```

Рис. 3.15: Отправка файлов на сервер

4 Выводы

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

5 Ответы на контрольные вопросы.

1. Системы контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Они позволяют хранить несколько версий изменяющейся информации, одного и того же документа, может предоставить доступ к более ранним версиям документа. Используется для работы нескольких человек над проектом, позволяет посмотреть, кто и когда внес какое-либо изменение и т. д. VCS применяются для: Хранения полной истории изменений, сохранения причин всех изменений, поиска причин изменений и совершивших изменение, совместной работы над проектами.
2. Хранилище – репозиторий, хранилище версий, в нем хранятся все документы, включая историю их изменения и прочей служебной информацией. commit – отслеживание изменений, сохраняет разницу в изменениях. История – хранит все изменения в проекте и позволяет при необходимости вернуться/обратиться к нужным данным. Рабочая копия – копия проекта, основанная на версии из хранилища, чаще всего последней версии.
3. Централизованные VCS (например: CVS, TFS, AccuRev) – одно основное хранилище всего проекта. Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет, затем добавляет изменения обратно в хранилище. Децентрализованные VCS (например: Git, Bazaar) – у каждого пользователя свой вариант репозитория (возможно несколько вариантов), есть возможность добавлять и забирать изменения из любого

репозитория. В отличие от классических, в распределенных (децентрализованных) системах контроля версий центральный репозиторий не является обязательным.

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.
7. 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 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_ветк 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.

Список литературы

1. Лабораторная работа № 2 [Электронный ресурс] URL: <https://esystem.rudn.ru/mod/page/view>