

Отчёт по лабораторной работе №2

Операционные системы

Андреева Софья Владимировна

Содержание

1	Цель работы	4
2	Выполнение работы	5
3	Контрольные вопросы.	11
4	Выводы	14

Список иллюстраций

2.1	Установим git:	5
2.2	Установим gh	5
2.3	Проведем базовую настройку git.	6
2.4	ключ ssh по алгоритму rsa	6
2.5	ключ ssh по алгоритму ed25519	7
2.6	Создадим ключи pgr	7
2.7	список ключей	8
2.8	New GPG key в GitHub	8
2.9	email при подписи коммитов	8
2.10	Настройка gh	9
2.11	Создадим репозиторий	9
2.12	Клонируем репозиторий	9
2.13	создадим необходимые каталоги	10
2.14	Отправим файлы на сервер	10

1 Цель работы

Изучить идеологию и применение средств контроля версий. Освоить умения по работе с git.

2 Выполнение работы

Установим git (рис. fig. 2.1).

```
[svandreeva@fedora ~]$ sudo -i
[sudo] пароль для svandreeva:
[root@fedora ~]# dnf install git
Fedora 39 - x86_64 - Updates
Пакет git-2.43.2-1.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
[root@fedora ~]#
```

Рис. 2.1: Установим git:

Установим gh (рис. fig. 2.2).

```
[root@fedora ~]# dnf install gh
Последняя проверка окончания срока действия метаданных: 0:02:00 назад, Сб 17 фев 2024 21:07:01.
Зависимости разрешены.
=====
Пакет      Архитектура  Версия      Репозиторий  Размер
-----
Установка:
gh         x86_64       2.43.1-1.fc39  updates      9.1 М
=====
Результат транзакции
Установка 1 Пакет
=====
Объем загрузки: 9.1 М
Объем изменений: 46 М
Продолжить? [д/н]: д
Загрузка пакетов:
gh-2.43.1-1.fc39.x86_64.rpm                    5.3 MB/s | 9.1 MB  00:01
-----
Общий размер                                     3.3 MB/s | 9.1 MB  00:02
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно.
Выполнение транзакции
Подготовка      :
Установка       : gh-2.43.1-1.fc39.x86_64      1/1
Запуск скрипта  : gh-2.43.1-1.fc39.x86_64      1/1
Проверка        : gh-2.43.1-1.fc39.x86_64      1/1
Установлен:
gh-2.43.1-1.fc39.x86_64
Выполнено!
```

Рис. 2.2: Установим gh

Проведем базовую настройку git. Зададим имя и email владельца репозитория,

настроим utf-8 в выводе сообщений git, зададим имя начальной ветки, параметр autocrlf и safecrlf (рис. fig. 2.3).

```
[svandreeva@fedora ~]$ git config --global user.name "svandreeva"
[svandreeva@fedora ~]$ git config --global user.email "andreevasofa57@gmail"
[svandreeva@fedora ~]$ git config --global core.quotepath false
[svandreeva@fedora ~]$ git config --global init.defaultBranch master
[svandreeva@fedora ~]$ git config --global core.autocrlf input
[svandreeva@fedora ~]$ git config --global core.safecrlf warn
[svandreeva@fedora ~]$
```

Рис. 2.3: Проведем базовую настройку git.

Создадим ключи ssh: по алгоритму rsa с ключом размером 4096 бит (рис. fig. 2.4).

```
[svandreeva@fedora ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/svandreeva/.ssh/id_rsa):
Created directory '/home/svandreeva/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/svandreeva/.ssh/id_rsa
Your public key has been saved in /home/svandreeva/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:KHLb0XJzI1Lz7TohWRKjSpdozELAZWAXubjLoSpd08g svandreeva@fedora
The key's randomart image is:
+---[RSA 4096]---+
|.==0          |
|ooo  o        |
|o . . o       |
|.O. . . .     |
|.+.+.+.+.S    |
|o.*o+=o+..    |
|o*.++.B.+..   |
|+.E o+ =.o    |
|+      .o.    |
+----[SHA256]-----+
f-----[SHA256]-----+
f-----[SHA256]-----+
```

Рис. 2.4: ключ ssh по алгоритму rsa

И по алгоритму ed25519 (рис. fig. 2.5).

```
[svandreeva@fedora ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/svandreeva/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/svandreeva/.ssh/id_ed25519
Your public key has been saved in /home/svandreeva/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:gTy/P8eIaMm0K9aZX7Wn1bny345PLCtgrFHQFwpMIhQ svandreeva@fedora
The key's randomart image is:
+--[ED25519 256]--+
| .E..oo.  . . |
|  o oo.... |
|   + .... |
|    o .. |
|   So  . |
| . . .+ . . o. |
|   + *.+.+.+.+ |
|  o X ooo += =o |
| . o.o. .o.==* |
+-----[SHA256]-----+
```

Рис. 2.5: ключ ssh по алгоритму ed25519

Создадим ключи ргр. Генерируем ключ, указав его тип, размер, срок действия (рис. fig. 2.6).

```
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

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: svandreeva
Адрес электронной почты: andreevasofa57@gmail.com
Примечание:
Вы выбрали следующий идентификатор пользователя:
"svandreeva <andreevasofa57@gmail.com>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (Q)Принять/(Q)Выход? 0
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
gpg: /home/svandreeva/.gnupg/tzstdb.gpg: создана таблица доверия
gpg: создан каталог '/home/svandreeva/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/svandreeva/.gnupg/openpgp-revocs.d/4CA37ECA1C31ED16054705D0A93240B18774ADBE.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2024-02-17 [SC]
      4CA37ECA1C31ED16054705D0A93240B18774ADBE
uid
sub   rsa4096 2024-02-17 [E]
```

Рис. 2.6: Создадим ключи ргр

У меня уже есть учетная запись в github, поэтому следующим шагом мы добавляем PGP ключ в GitHub, для этого выводим список ключей и копируем отпечаток приватного ключа. (рис. fig. 2.7).

```
[svandreeva@fedora ~]$ 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, 0m, 0f, 1u
[keyboard]
-----
sec   rsa4096/A93240B18774ADBE 2024-02-17 [SC]
      4CA37ECA1C31ED160547D5D0A93240B18774ADBE
uid   [ абсолютно ] svandreeva <andreevasofa57@gmail.com>
ssb   rsa4096/11F978BD3D918B9C 2024-02-17 [E]

[svandreeva@fedora ~]$
```

Рис. 2.7: список ключей

Затем копируем наш сгенерированный PGP ключ в буфер обмена и вставляем его при создании New GPG key в GitHub. (рис. fig. 2.8).

Add new GPG key

Title

main sway

Key

```
M+dh1sBopHEGTbfj1gkbHg9Q/Iz0wRdo4gZQM8po1s3htsDzV6W4iRxEMcMnFD
XMzgpvO8xI2mY35FKHKWouWpghuPtiRUSXDDjDv5yxb9/2Y/hJrLRW5k/XKulc7J
SmgHk6RscLO1B0qn1OPWuACdD66kKT+dlBiGuFGBwIwJ/URszudwv1BYFWpnoOHD
yXh261nnMOMyOodkOj3LRtZuRXfaDUy83+/N8D/PIKaAvTGBc8efuxp8:5PRXSo
SCO8s17EFBO2HL3rZtfturfBKivOlhoACSD6IDAsk0oqNOMFx+PzZAnfc8C4KXa
CX7eoYc7VnEl+VGIHxdKf1PC8ypbxm90rEPVXc4rpxLURFvR6USmCjIGbmMXFxp
I3a9dA==
=ZwMH
-----END PGP PUBLIC KEY BLOCK-----
```

Add GPG key

Рис. 2.8: New GPG key в GitHub

Используя введённый email, укажем Git применять его при подписи коммитов(рис. @fig:009).

```
[svandreeva@fedora ~]$ git config --global user.signingkey 4CA37ECA1C31ED160547D5D0A93240B18774ADBE
[svandreeva@fedora ~]$ git config --global commit.gpgsign true
[svandreeva@fedora ~]$ git config --global gpg.program $(which gpg2)
[svandreeva@fedora ~]$
```

Рис. 2.9: email при подписи коммитов

Настройка gh.Для начала авторизуемся, ответив на несколько наводящих вопросов. (рис. fig. 2.10).

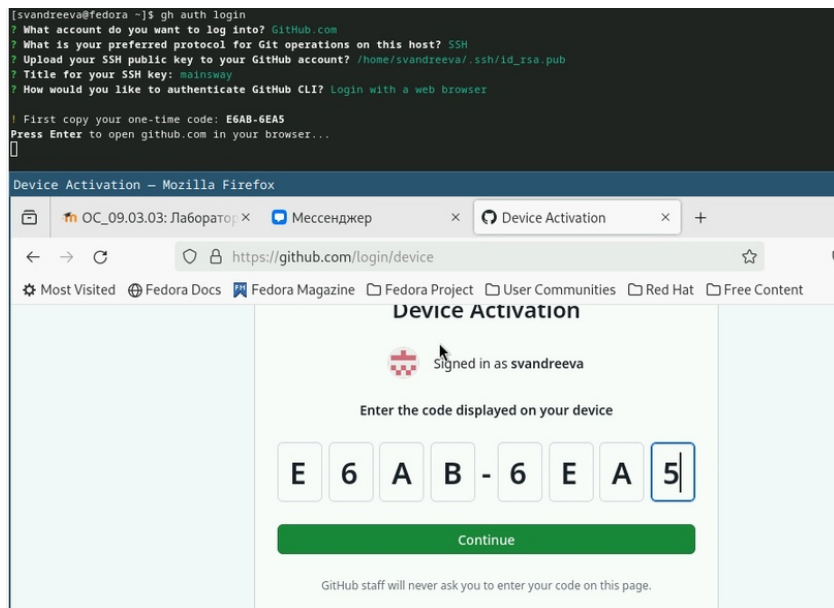


Рис. 2.10: Настройка gh

Создадим репозиторий, предварительно создав рабочее пространство (рис. fig. 2.11).

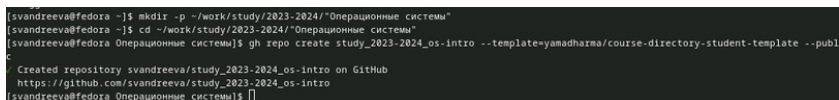


Рис. 2.11: Создадим репозиторий

Клонируем репозиторий (рис. fig. 2.12).

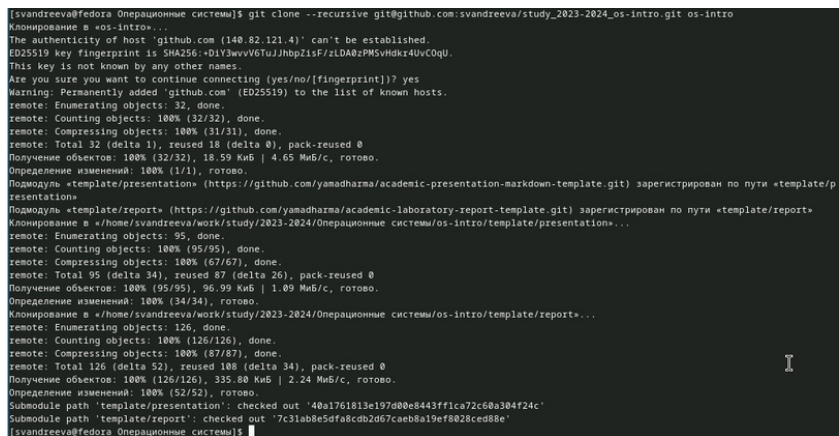


Рис. 2.12: Клонировем репозиторий

Перейдем в каталог курса, удалим лишние файлы и создадим необходимые каталоги(рис. fig. 2.13).

```
[svandreeva@fedora os-intro]$ echo os-intro > COURSE
[svandreeva@fedora os-intro]$ make prepare
[svandreeva@fedora os-intro]$
```

Рис. 2.13: создадим необходимые каталоги

Отправим файлы на сервер (рис. fig. 2.14).

```
[svandreeva@fedora os-intro]$ git add .
[svandreeva@fedora os-intro]$ git commit -am 'feat(main): make course structure'
[master 275c45a] feat(main): make course structure
 2 files changed, 1 insertion(+), 14 deletions(-)
 delete mode 100644 package.json
[svandreeva@fedora 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), 950 байтов | 475.00 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:svandreeva/study_2023-2024_os-intro.git
 55ef62d..275c45a master -> master
[svandreeva@fedora os-intro]$
```

Рис. 2.14: Отправим файлы на сервер

3 Контрольные вопросы.

1. Системы контроля версий (VCS) разработаны специально для того, чтобы максимально упростить и упорядочить работу над проектом (вне зависимости от того, сколько человек в этом участвуют). СКВ дает возможность видеть, кто, когда и какие изменения вносил; позволяет формировать новые ветви проекта, объединять уже имеющиеся; настраивать контроль доступа к проекту; осуществлять откат до предыдущих версий.
2. Основные понятия:
 - Хранилище (repository, сокр. repo), или репозиторий, — место хранения всех версий и служебной информации;
 - Коммит (commit) — 1) синоним версии; 2) создание новой версии («сделать коммит», «закоммитить»);
 - История разработки — совокупность всех версий файлов, над которыми ведется работа. Историей разработки в данном случае будет список изменений: создание файла, добавление изначального текста, исправление опечатки, добавление нового текста, объединение двух версий файла (при выполнении слияния);
 - Рабочая копия (working copy или working tree) — текущее состояние файлов проекта, основанное на версии из хранилища (обычно на последней).
3. Централизованные и децентрализованные VCS:
 - Централизованные VCS - одно основное хранилище всего проекта, где каждый пользователь копирует себе необходимые ему файлы из этого репозитория.

тория, изменяет и, затем, добавляет свои изменения обратно. Например Subversion, CVS, TFS, VAULT, AccuRev;

- Децентрализованные VCS - у каждого пользователя свой вариант (возможно не один) репозитория, присутствует возможность добавлять и забирать изменения из любого репозитория. Например Git, Mercurial, Bazaar.

4. Единоличная работа с хранилищем:

- работа в локальном репозитории;
- сохранение изменений и загрузка на серверов.

5. Работа с общим хранилищем VCS:

- проверка обновлений;
- загрузка обновлений (при наличии);
- работа в локальном репозитории;
- создаются ветвления, если несколько пользователей работают над одним и тем же файлом/документом;
- по результатам различных версий могут происходить слияния в одну ветвь.

6. Основные задачи, решаемые инструментальным средством git:

- хранить информацию о всех изменениях в коде;
- обеспечение удобства командной работы над кодом.

7. Примеры команд git:

- `git pull` - получение обновлений (изменений) текущего дерева из центрального репозитория;
- `git push` - отправка всех произведённых изменений локального дерева в центральный репозиторий;
- `git status` - просмотр списка изменённых файлов в текущей директории;

- `git add` - добавить все изменённые и/или созданные файлы и/или каталоги;
 - `git commit -am 'Описание коммита'` - сохранить все добавленные изменения и все изменённые файлы.
8. Примеры команд для работы с локальным и удалённым репозиториями
`git push -all (push origin master/любой branch)`
9. Ветка (англ. branch) — это последовательность коммитов, в которой ведётся параллельная разработка какого-либо функционала. Основная ветка – `master`. Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка специально для экспериментов.
10. Для игнорирования некоторых файлов можно создать файл `.gitignore` в корневом каталоге репозитория, чтобы сообщить Git, какие файлы и каталоги следует игнорировать при фиксации. Иногда имеется группа файлов, которые не нужно автоматически добавлять в репозиторий. К таким файлам обычно относятся автоматически генерируемые файлы (различные логи, результаты сборки программ и т. п.)

4 Выводы

Я изучила идеологию и применение средств контроля версий. Освоила умения по работе с git.