

Шаблон отчёта по лабораторной работе № 2

Markdown

Скобеева Алиса Алексеевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Контрольные вопросы	15
5	Выводы	17

Список иллюстраций

3.1	Вводим необходимую команду	7
3.2	Вводим необходимую команду	7
3.3	Вводим необходимые команды для установки	8
3.4	Вводим ssh-keygen -t ed25519	8
3.5	Ввод и выполнение команды	9
3.6	Выбираем необходимые параметры	9
3.7	Ввод команды	10
3.8	Пробуем это сделать	10
3.9	Копируем ключ вручную	11
3.10	Вставляем в необходимое окно ключ	12
3.11	Используя введенный email, указываем Git применять его при под- писи коммитов	12
3.12	Вводим необходимую команду	13
3.13	Вводим необходимые команды	14

Список таблиц

1 Цель работы

Научиться работать с Markdown и составлять с помощью него отчеты к лабораторным работам.

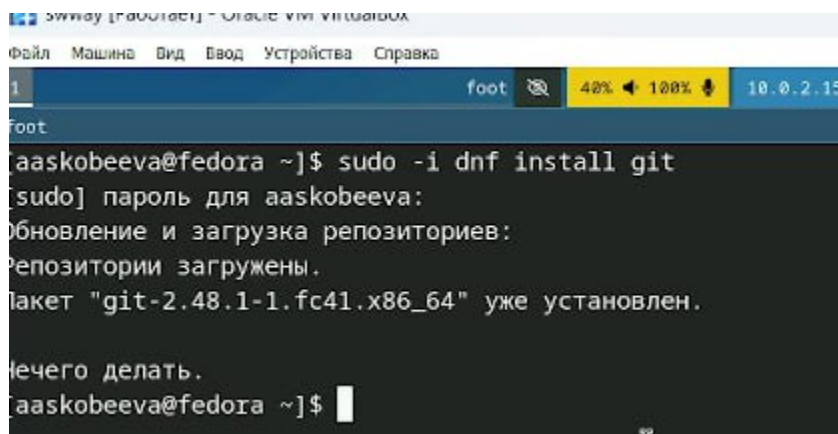
2 Задание

Сделать отчет к лабораторной работе № 2.

3 Выполнение лабораторной работы

1. Установка программного обеспечения

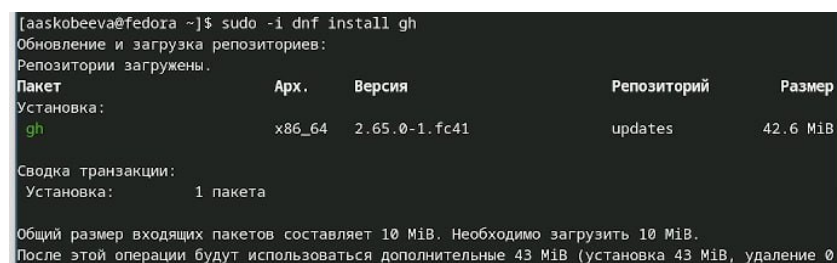
Устанавливаем git.



```
[aaskobeeva@fedora ~]$ sudo -i dnf install git
[sudo] пароль для aaskobeeva:
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет "git-2.48.1-1.fc41.x86_64" уже установлен.
Нечего делать.
[aaskobeeva@fedora ~]$
```

Рис. 3.1: Вводим необходимую команду

Устанавливаем gh.

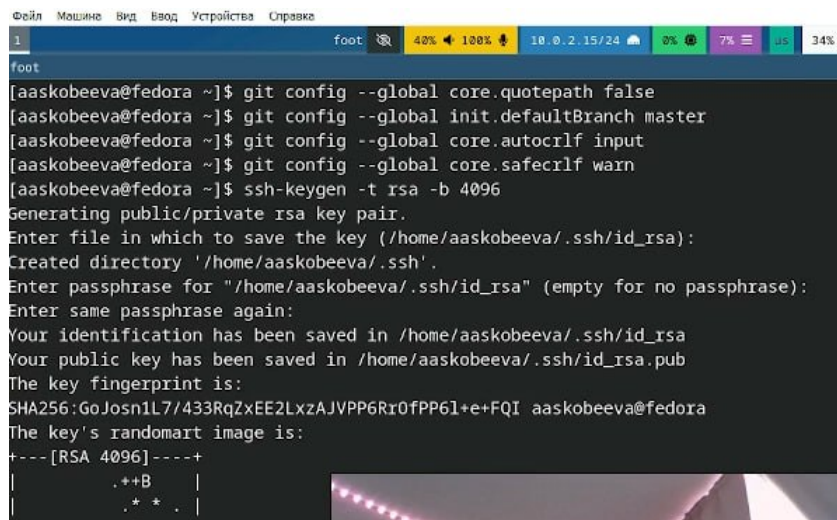


```
[aaskobeeva@fedora ~]$ sudo -i dnf install gh
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет
Установка:
  gh
  Арх.      Версия      Репозиторий      Размер
  x86_64    2.65.0-1.fc41    updates          42.6 MiB
Сводка транзакции:
  Установка:      1 пакета
Общий размер входящих пакетов составляет 10 MiB. Необходимо загрузить 10 MiB.
После этой операции будут использоваться дополнительные 43 MiB (установка 43 MiB, удаление 0
```

Рис. 3.2: Вводим необходимую команду

2. Базовая настройка git. Задаем имя и email владельца репозитория, настраиваем utf-8 в выводе сообщений git, настраиваем верификацию и подписание

коммитов, задаем имя начальной ветки, устанавливаем параметр autocrlf, а также параметр safecrlf.



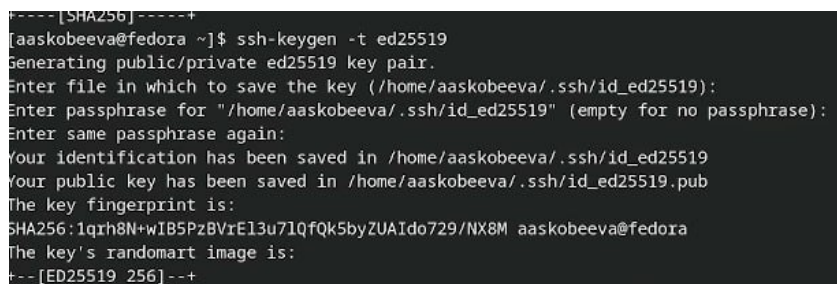
```
1
foot
[aaskobeeva@fedora ~]$ git config --global core.quotepath false
[aaskobeeva@fedora ~]$ git config --global init.defaultBranch master
[aaskobeeva@fedora ~]$ git config --global core.autocrlf input
[aaskobeeva@fedora ~]$ git config --global core.safecrlf warn
[aaskobeeva@fedora ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/aaskobeeva/.ssh/id_rsa):
Created directory '/home/aaskobeeva/.ssh'.
Enter passphrase for "/home/aaskobeeva/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aaskobeeva/.ssh/id_rsa
Your public key has been saved in /home/aaskobeeva/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:GoJsn1L7/433RqZxEE2LxzAJVPP6RrOfPP6l+e+FQI aaskobeeva@fedora
The key's randomart image is:
+---[RSA 4096]-----+
|          .++B      |
|         .+ * .    |
|          .+ * .    |
|          .+ * .    |
|          .+ * .    |
|          .+ * .    |
|          .+ * .    |
|          .+ * .    |
|          .+ * .    |
|          .+ * .    |
+---[RSA 4096]-----+
```

Рис. 3.3: Вводим необходимые команды для установки

3. Создаем ключи ssh

По алгоритму rsa с ключем размером 4096 бит(см. изображение 3 выше)

По алгоритму ed25519:



```
-----[SHA256]-----+
[aaskobeeva@fedora ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/aaskobeeva/.ssh/id_ed25519):
Enter passphrase for "/home/aaskobeeva/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/aaskobeeva/.ssh/id_ed25519
Your public key has been saved in /home/aaskobeeva/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:1qrh8N+wIB5PzBVrE13u7lQfQk5byZUAIdo729/NX8M aaskobeeva@fedora
The key's randomart image is:
+---[ED25519 256]---+
```

Рис. 3.4: Вводим ssh-keygen -t ed25519

4. Создание ключей pgp

Генерируем ключ с помощью команды gpg --full-generate-key


```
[aaskobeeva@fedora ~]$ gpg --full-generate-key
gpg (GnuPG) 2.4.5; Copyright (C) 2024 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 '/home/aaskobeeva/.gnupg'
Выберите тип ключа:
(1) RSA and RSA
```

Рис. 3.5: Ввод и выполнение команды

Иллюстрация выбора из предложенных опций:

```
Файл  Машина  Вид  Ввод  Устройства  Справка
1      foot
foot
(4) RSA (sign only)
(9) ECC (sign and encrypt) *default*
(10) ECC (только для подписи)
(14) Existing key from card
Ваш выбор? 1
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
```

Рис. 3.6: Выбираем необходимые параметры

5. Настройка github

Поскольку учетная запись у меня была создана и заполнена ранее, данный пункт пропускаем.

6. Добавление PGP ключа в GitHub

Выводим список ключей, копируем отпечаток приватного ключа - в нашем случае это адрес электронной почты.

```
[aaskobeeva@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, 0n, 0m,
0f, 1u
[keyboxd]
```

Рис. 3.7: Ввод команды

Копируем сгенерированный PGP ключ в буфер обмена.

```
завершено!
[aaskobeeva@fedora ~]$ gpg --armor
[aaskobeeva@fedora ~]$ gpg --armor
port skobeevaalisa@gmail.com | xcl
el clip
```

Рис. 3.8: Пробуем это сделать

Поскольку с помощью команды скопировать ключ не вышло, скопируем его вручную, воспользовавшись командой cat:

```
savepшeno:
[aaskobeeva@fedora ~]$ gpg --armor --ex
[aaskobeeva@fedora ~]$ gpg --armor --ex
port skobeevaalisa@gmail.com | xclip -s
el clip
[aaskobeeva@fedora ~]$ gpg --armor --ex
port skobeevaalisa@gmail.com | cat
-----BEGIN PGP PUBLIC KEY BLOCK-----
[REDACTED]
nQINB6fEaKwBEAChAXzhu4apCCT3msmYqbU2ydm
iATWYGpZpkXnAz5YYHYXIyAN7[REDACTED]
Ly6vs3kFIrJtVKPh0SexBCla50ZjP1YIR[REDACTED]
INyyJ4BZHpp9Yi133PQLh7Zjw[REDACTED]
iklkk0A7Y1MV80Pwrfo+yjb91Jq7BEFF6[REDACTED]
44mMjTYhZ9YQF2+5Sa+r8ETvS[REDACTED]
77WAVrW86omm6pyQN3EUI5cZgcZxzGTTx[REDACTED]
zwwwuWVKUfUWHxYVnAu+DMQxcj[REDACTED]
[REDACTED]
[REDACTED]
```

Рис. 3.9: Копируем ключ вручную

Вставляем ключ в настройках GitHub:

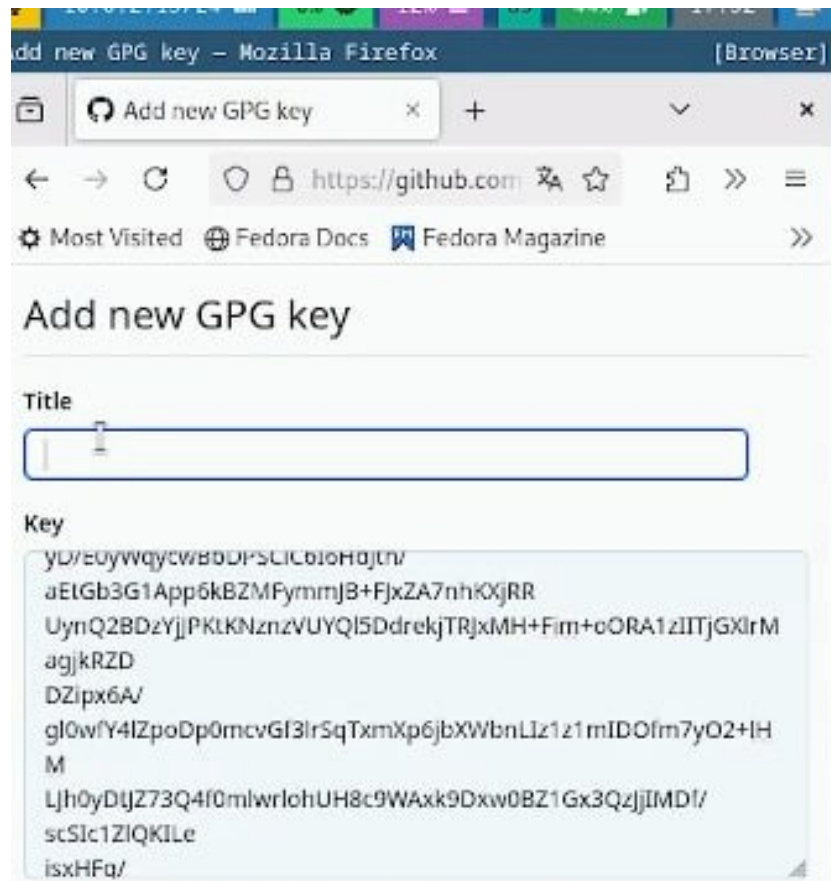


Рис. 3.10: Вставляем в необходимое окно ключ

7. Настройка автоматических подписей коммитов git

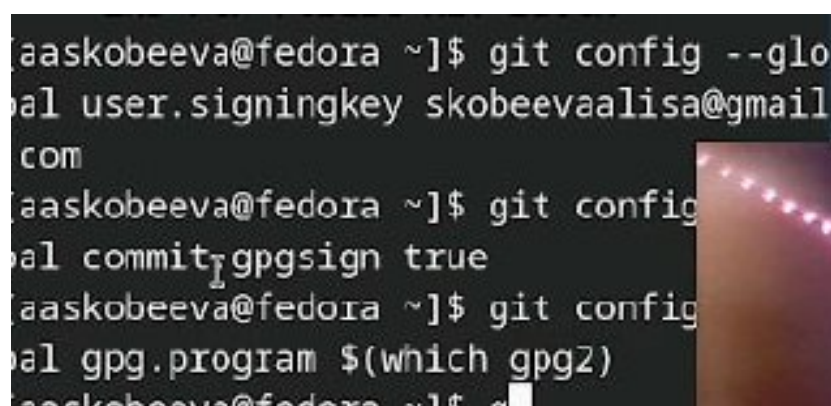


Рис. 3.11: Используя введенный email, указываем Git применять его при подписи коммитов

8. Создание репозитория курса

Вводим необходимые команды для создания каталогов и подключения рабочего пространства:

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

Рис. 3.12: Вводим необходимую команду

9. Настройка каталога курса

Переходим в каталог курса, удаляем лишние файлы и создаем необходимые каталоги

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

Targets:
  list                               List
```

Рис. 3.13: Вводим необходимые команды

В конце отправляем файлы на сервер с помощью команд:

```
git add . git commit -am 'feat(main): make course structure' git push
```

4 Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

Системы контроля версий (VCS) - это системы для отслеживания изменений файлов проекта. Предназначаются для следующих задач: совместная разработка, версионность, откат к предыдущим версиям, управление ветвями.

2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

- Хранилище: База данных со всеми версиями проекта.
- Commit: Запись изменений с комментарием.
- История: Последовательность коммитов.
- Рабочая копия: Локальная копия файлов для работы. (Отношения: Рабочая копия -> Commit -> История в Хранилище).

3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

- Централизованные VCS: Одно центральное хранилище. Пример: SVN.
- Децентрализованные VCS: Полная локальная копия хранилища у каждого. Пример: Git.

4. Опишите действия с VCS при единоличной работе с хранилищем.

Единоличная работа: git init, git add, git commit, git log, git checkout.

5. Опишите порядок работы с общим хранилищем VCS.

Общее хранилище: git clone, git branch, git checkout, git add, git commit, git push, pull request, code review, merge.

6. Каковы основные задачи, решаемые инструментальным средством git?

Git: Версионность, совместная разработка, управление ветвями, удаленное хранение, разрешение конфликтов.

7. Назовите и дайте краткую характеристику командам git.

git init (создание), git clone (копирование), git add (добавление), git commit (фиксация), git push (отправка), git pull (получение), git branch (ветки), git merge (слияние).

8. Приведите примеры использования при работе с локальным и удалённым репозиториями.

- Локальный: git init, git add ., git commit -m "Initial commit".
- Удаленный: git clone , git push origin .

9. Что такое и зачем могут быть нужны ветви (branches)?

Ветви: Параллельные линии разработки. Зачем: разработка новых функций, эксперименты, управление релизами.

10. Как и зачем можно игнорировать некоторые файлы при commit?

Игнорирование файлов: Файл .gitignore. Зачем: избегать захламления хранилища, безопасность, производительность.

5 Выводы

Мы научились работать с Markdown и составлять отчеты с его помощью.