

# **Отчёт по лабораторной работе №2.**

**дисциплина: операционные системы**

Тимофеева Екатерина Николаевна

# Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	13
5	Выполнение контрольных вопросов.	14
	Список литературы	18

## Список иллюстраций

3.1	Регистрация на сайте . . . . .	7
3.2	Конфигурация git . . . . .	7
3.3	Генерация ключей . . . . .	8
3.4	Копирование ключа . . . . .	8
3.5	Загрузка сгенерённого ключа . . . . .	9
3.6	Создание ключа . . . . .	9
3.7	Вывод списка ключей и копирование . . . . .	10
3.8	Копирование сгенерированного ключа в буфер обмена . . . . .	10
3.9	Настройка автоматических подписей коммитов . . . . .	10
3.10	Авторизация . . . . .	11
3.11	Создание репозитория . . . . .	11
3.12	Клонирование . . . . .	12
3.13	Настройка каталога курса . . . . .	12

## **Список таблиц**

# 1 Цель работы

Целью работы является изучение идеологии и применение средств контроля версий. Приобретение практических навыков 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

№1. Создаём учётную запись на сайте и заполняем основные данные. (рис. 3.1).

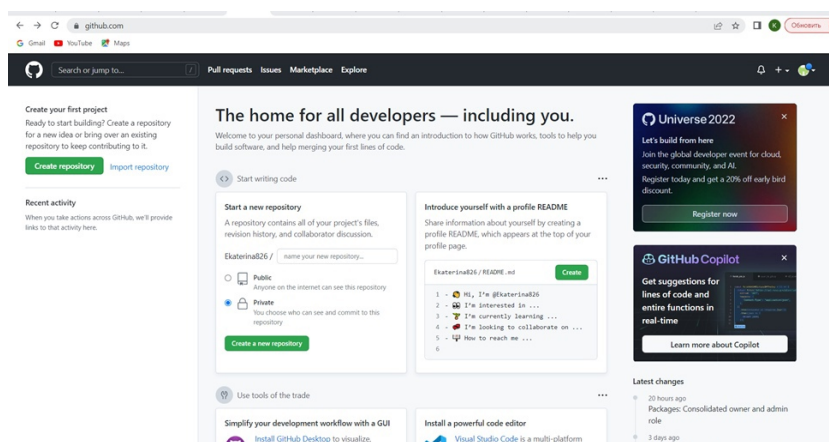


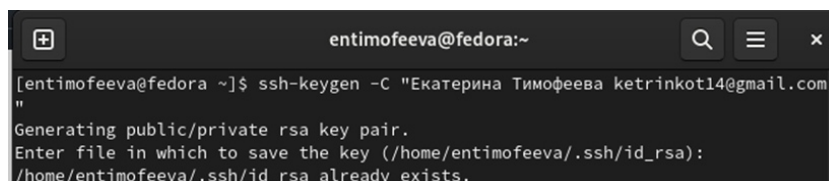
Рис. 3.1: Регистрация на сайте

Сначала сделаем предварительную конфигурацию git. Откроем терминал и введём следующие команды, указав имя и email владельца репозитория. (рис. 3.2)

```
entimofeeva@fedora:~ — ssh-keygen -C Екатерина Тимофеев...
[entimofeeva@fedora ~]$ git config --global user.name "<Ekaterina826>"
[entimofeeva@fedora ~]$ git config --global user.email "<ketrinkot14@gmail.com>"
[entimofeeva@fedora ~]$ git config --global core.quotepath false
[entimofeeva@fedora ~]$ git config --global init.defaultBranch master
[entimofeeva@fedora ~]$ git config --global core.autocrlf input
[entimofeeva@fedora ~]$ git config --global core.safecrlf warn
```

Рис. 3.2: Конфигурация git

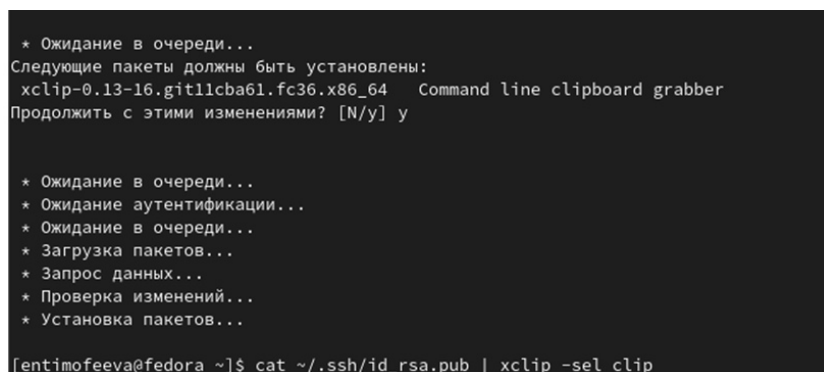
Для последующей идентификации пользователя на сервере сгенерируем пару ключей (приватный и открытый). (рис. 3.3).



```
entimofeeva@fedora:~  
[entimofeeva@fedora ~]$ ssh-keygen -C "Екатерина Тимофеева ketrinkot14@gmail.com"  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/entimofeeva/.ssh/id_rsa):  
/home/entimofeeva/.ssh/id_rsa already exists.
```

Рис. 3.3: Генерация ключей

Далее необходимо загрузить сгенерённый открытый ключ. Для этого зайдём на сайт под своей учётной записью и перейдём в меню Setting. После этого выберем в боковом меню SSH and GPG keys и нажмём кнопку New SSH key, скопировав из локальной консоли ключ в буфер обмена, вставляем ключ в появившееся на сайте поле и указываем для ключа имя. (рис. 3.4), (рис. 3.5)



```
* Ожидание в очереди...  
Следующие пакеты должны быть установлены:  
xclip-0.13-16.git11cba61.fc36.x86_64 Command line clipboard grabber  
Продолжить с этими изменениями? [N/y] y  
  
* Ожидание в очереди...  
* Ожидание аутентификации...  
* Ожидание в очереди...  
* Загрузка пакетов...  
* Запрос данных...  
* Проверка изменений...  
* Установка пакетов...  
[entimofeeva@fedora ~]$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
```

Рис. 3.4: Копирование ключа



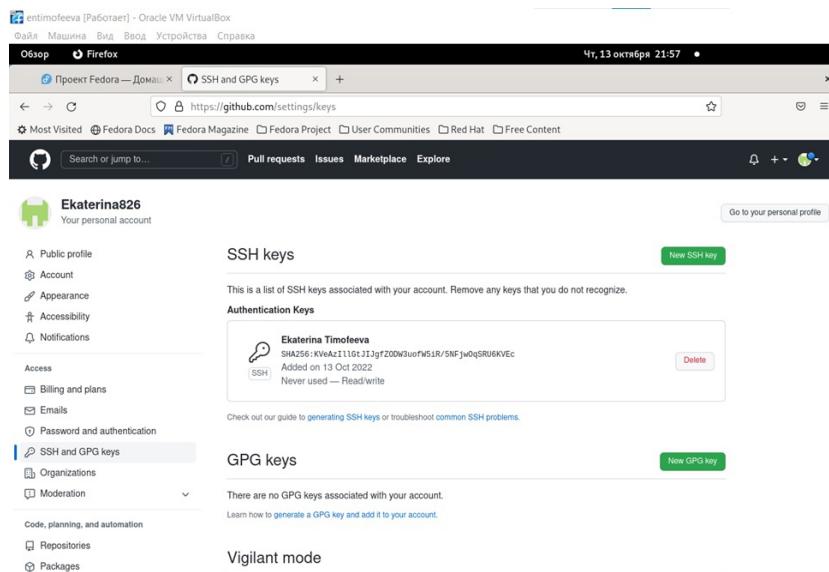


Рис. 3.5: Загрузка сгенерённого ключа

Генерируем pgr-ключ. (рис. 3.6), (рис. 3.7), (рис. 3.8)

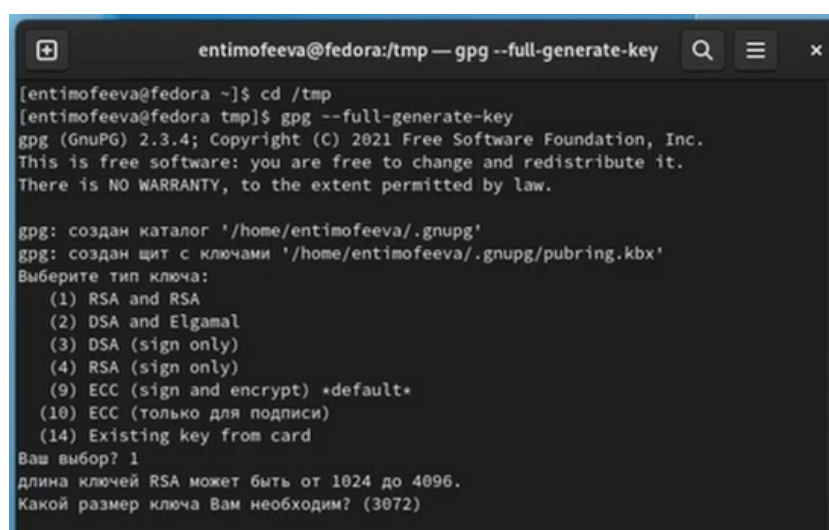


Рис. 3.6: Создание ключа

```
entimofeeva@fedora:/tmp
gpg: /home/entimofeeva/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/entimofeeva/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/entimofeeva/.gnupg/openpgp-revocs.d/5CDE9AA32976E354222CF9204A5F6F1BB03E9533.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2023-02-16 [SC]
      5CDE9AA32976E354222CF9204A5F6F1BB03E9533
uid           Ekaterina <ketrinkot14@gmail.com>
sub   rsa4096 2023-02-16 [E]

[entimofeeva@fedora tmp]$ 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
, lu
/home/entimofeeva/.gnupg/pubring.kbx
-----
sec   rsa4096/4A5F6F1BB03E9533 2023-02-16 [SC]
      5CDE9AA32976E354222CF9204A5F6F1BB03E9533
uid           [ абсолютно ] Ekaterina <ketrinkot14@gmail.com>
ssb   rsa4096/E521FE38E30E9124 2023-02-16 [E]

[entimofeeva@fedora tmp]$
```

Рис. 3.7: Вывод списка ключей и копирование

```
ssb   rsa4096/E521FE38E30E9124 2023-02-16 [E]

[entimofeeva@fedora tmp]$ gpg --armor --export 4A5F6F1BB03E9533 | xclip -sel cli
p
[entimofeeva@fedora tmp]$
```

Рис. 3.8: Копирование сгенерированного ключа в буфер обмена

Заходим на gh, вставляем скопированный ключ и добавляем его.

Настроим автоматические подписи коммитов git, используя введенный email, укажем Git при подписи коммитов. (рис. 3.9)

```
p
[entimofeeva@fedora tmp]$ git config --global user.signingkey 4A5F6F1BB03E9533
[entimofeeva@fedora tmp]$ git config --global commit.gpgsign true
[entimofeeva@fedora tmp]$ git config --global gpg.program $(which gpg2)
[entimofeeva@fedora tmp]$
```

Рис. 3.9: Настройка автоматических подписей коммитов

Настраиваем GitHub, для этого необходимо авторизоваться, вводим нужные команды. Затем в терминале нам приходит код, который нужно вставить в форму, которая открывается при переходе по ссылке. (рис. 3.10)

```
[entimofeeva@fedora tmp]$ gh auth login
bash: gh: команда не найдена...
Установить пакет «gh», предоставляющий команду «gh»? [N/y] y

* Ожидание в очереди...
* Загрузка списка пакетов...
Следующие пакеты должны быть установлены:
gh-2.22.1-1.fc36.x86_64      GitHub's official command line tool
Продолжить с этими изменениями? [N/y] y

* Ожидание в очереди...
* Ожидание аутентификации...
* Ожидание в очереди...
* Загрузка пакетов...
* Запрос данных...
* Проверка изменений...
* Установка пакетов...

! First copy your one-time code: 5284-3F9D
Open this URL to continue in your web browser: https://github.com/login/device
```

Рис. 3.10: Авторизация

Создаём репозиторий курса, с помощью команд, вводимых в терминале. (рис. 3.11), (рис. 3.12)

```
entimofeeva@fedora:~/work/study/2022-2023/Операционные системы
[entimofeeva@fedora ~]$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
[entimofeeva@fedora ~]$ cd ~/work/study/2022-2023/"Операционные системы"
[entimofeeva@fedora Операционные системы]$ gh repo create study_2022-2023_os-intro
"--public", "--private", or "--internal" required when not running interactively
Usage: gh repo create [<name>] [flags]

Flags:
  --add-readme      Add a README file to the new repository
  -c, --clone       Clone the new repository to the current directory
  -d, --description string Description of the repository
  --disable-issues  Disable issues in the new repository
  --disable-wiki    Disable wiki in the new repository
  -g, --gitignore string Specify a gitignore template for the repository
  -h, --homepage URL Repository home page URL
  --include-all-branches Include all branches from template repository
  --internal        Make the new repository internal
  -l, --license string Specify an Open Source License for the repository
  --private        Make the new repository private
  --public         Make the new repository public
  --push           Push local commits to the new repository
  -r, --remote string Specify remote name for the new repository
  -s, --source string Specify path to local repository to use as source
  -t, --team name   The name of the organization team to be granted access
  -p, --template repository Make the new repository based on a template repository

[entimofeeva@fedora Операционные системы]$ gh repo create study_2022-2023_os-intro --template=yamadharma/course-directory-student-template --public
Created repository Ekaterina826/study_2022-2023_os-intro on GitHub
[entimofeeva@fedora Операционные системы]$
```

Рис. 3.11: Создание репозитория

```

[entimofeeva@fedora Операционные системы]$ git clone --recursive git@github.com:Ekaterina826/study_2022-2023_os-intro.git os-intro
Клонирование в «os-intro»...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 27 (delta 1), reused 11 (delta 0), pack-reused 0
Получение объектов: 100% (27/27), 16.93 КиБ | 199.00 КиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по п
ути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «templ
ate/report»
Клонирование в «/home/entimofeeva/work/study/2022-2023/Операционные системы/os-intro/template/presentation»...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 82 (delta 28), reused 77 (delta 23), pack-reused 0
Получение объектов: 100% (82/82), 92.90 КиБ | 99.00 КиБ/с, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/home/entimofeeva/work/study/2022-2023/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (101/101), done.
remote: Compressing objects: 100% (70/70), done.
remote: Total 101 (delta 40), reused 88 (delta 27), pack-reused 0
Получение объектов: 100% (101/101), 327.25 КиБ | 119.00 КиБ/с, готово.
Определение изменений: 100% (40/40), готово.
Submodule path 'template/presentation': checked out 'b1be3800ee91f5809264cb755d316174540b753e'
Submodule path 'template/report': checked out '1d1b61dca9c287a83917b82e3ae11a33b1e3b2'
[entimofeeva@fedora Операционные системы]$

```

Рис. 3.12: Клонирование

Переходим в каталог курса, удаляем лишние файлы, создаём необходимые каталоги и отправляем файлы на сервер. (рис. 3.13)

```

[entimofeeva@fedora Операционные системы]$ cd ~/work/study/2022-2023/"Операционные системы"/os-intro
[entimofeeva@fedora os-intro]$ rm package.json
[entimofeeva@fedora os-intro]$ echo os-intro > COURSE
[entimofeeva@fedora os-intro]$ make
[entimofeeva@fedora os-intro]$ git add .
[entimofeeva@fedora os-intro]$ git commit -am 'feat(main): make course structure'

```

Рис. 3.13: Настройка каталога курса

## 4 Выводы

Мы изучили идеологии и применение средств контроля версий. Приобрели практических навыков по работе с git.

## 5 Выполнение контрольных вопросов.

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначены? Система контроля версий — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое. Системы контроля версий (Version Control System, VCS) применяются для:
  - Хранение полной истории изменений
  - причин всех производимых изменений
  - Откат изменений, если что-то пошло не так
  - Поиск причины и ответственного за появления ошибок в программе
  - Совместная работа группы над одним проектом
  - Возможность изменять код, не мешая работе других пользователей
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Репозиторий - хранилище версий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией. Commit — отслеживание изменений, сохраняет разницу в изменениях Рабочая копия - копия проекта, связанная с репозиторием (текущее состояние файлов проекта, основанное на версии из хранилища (обычно на последней)) История хранит все изменения в проекте и позволяет при необходимости обратиться к нужным данным.
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные

VCS (Subversion; CVS; TFS; VAULT; AccuRev): • Одно основное хранилище всего проекта • Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно

Децентрализованные VCS (Git; Mercurial; Bazaar): • У каждого пользователя свой вариант (возможно не один) репозитория • Присутствует возможность добавлять и забирать изменения из любого репозитория

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

4. Опишите действия с VCS при единоличной работе с хранилищем. Сначала создаем и подключаем удаленный репозиторий. Затем по мере изменения проекта отправлять эти изменения на сервер.
5. Опишите порядок работы с общим хранилищем VCS. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент.
6. Каковы основные задачи, решаемые инструментальным средством git? Первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.
7. Назовите и дайте краткую характеристику командам git. Наиболее часто используемые команды git: • 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_ветки` – принудительное удаление локальной ветки: `git branch -D имя_ветки` – удаление ветки с центрального репозитория: `git push origin :имя_ветки`

8. Приведите примеры использования при работе с локальным и удалённым репозиториями. `git push -all` (`push origin master/любой branch`)
9. Что такое и зачем могут быть нужны ветви (branches)? Ветвление («ветка», `branch`) — один из параллельных участков истории в одном хранилище, исходящих из одной версии (точки ветвления). • Обычно есть главная ветка (`master`), или ствол (`trunk`). • Между ветками, то есть их концами, возможно слияние. Используются для разработки новых функций.
10. Как и зачем можно игнорировать некоторые файлы при `commit`? Во время



работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл `.gitignore` с помощью сервисов.

## **Список литературы**

Кулябов Д.С. “Материалы к лабораторной работе”