

Лабораторная работа №2

Дисциплина: Операционные системы

Панина Жанна Валерьевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Создание ключа SSH	9
4.2	Создание ключа GPG	9
4.3	Настройка подписей Git	11
4.4	Создание локального каталога	12
5	Ответы на контрольные вопросы	14
6	Выводы	18

Список иллюстраций

4.1	Создание ключа SSH	9
4.2	Создание ключа GPG	10
4.3	Ключ GPG	10
4.4	Ключи в GitHub	11
4.5	Настройка подписей Git	11
4.6	Клонирование репозитория	12
4.7	Удаление файлов и создание каталогов	12
4.8	Отправка файлов на сервер	12
4.9	Отправка файлов на сервер	13

Список таблиц

1 Цель работы

Цель работы - изучить идеологию и применения средств контроля версий, освоить умения по работе с git.

2 Задание

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

3 Теоретическое введение

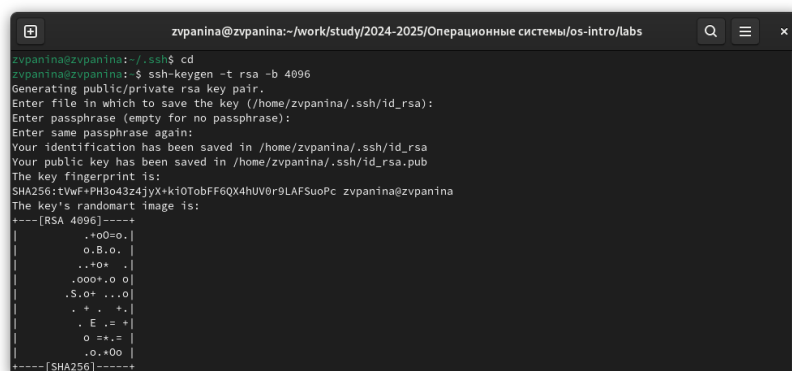
Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. 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. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зави-

симости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом. Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным. Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

4 Выполнение лабораторной работы

4.1 Создание ключа SSH

Поскольку я уже была зарегистрирована на GitHub в первом семестре, начинаю работу с создания ключа SSH (рис. 4.1).



```
zvpanina@zvpanina:~/work/study/2024-2025/Операционные системы/os-intro/labs
zvpanina@zvpanina:~/.ssh$ cd
zvpanina@zvpanina:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/zvpanina/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/zvpanina/.ssh/id_rsa
Your public key has been saved in /home/zvpanina/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:tvWf+PH3o43z4jyX+k1OTobFF6QX4hUV0r9LAFSuoPc zvpanina@zvpanina
The key's randomart image is:
+---[RSA 4096]-----+
|      .+o0=0=0. |
|      o.B.o.  |
|      ..+0+  . |
|      .ooo+.o o |
|      .S.o+ ...o |
|      + + . + |
|      + E . = + |
|      o =+ = |
|      .o.+0o |
+---[SHA256]-----+
```

Рис. 4.1: Создание ключа SSH

4.2 Создание ключа GPG

1. Создаю ключ GPG, указав необходимые параметры (рис. 4.2).

```
zvpanina@zvpanina:~/work/study/2024-2025/Операционные системы/os-intro/labs
zvpanina@zvpanina:~$ gpg --full-generate-key
gpg (GnuPG) 2.4.4; 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.

Выберите тип ключа:
(1) RSA and RSA
(2) DSA and ElGamal
(3) DSA (sign only)
(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
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: Zhanna Panina
Адрес электронной почты: zhanna.panina86@gmail.com
Примечание:
```

Рис. 4.2: Создание ключа GPG

2. Вывожу список ключей и копирую отпечаток приватного ключа. Копирую сгенерированный ключ GPG (рис. 4.3).

```
zvpanina@zvpanina:~/work/study/2024-2025/Операционные системы/os-intro/labs
zvpanina@zvpanina:~$ 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: 2 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 2u
[keyboard]
-----
sec rsa4096/78B3A5684756C90B 2025-03-05 [SC]
    A02F3B0E0E6F43956D55AD17883A5684756C90B
uid          [ a6cometno ] Zhanna Panina <zhanna.panina86@gmail.com>
ssb rsa4096/1AD6A8FD0746F2E7 2025-03-05 [E]

sec rsa4096/4BE6B0B165E541C4 2025-03-04 [SC]
    F6C109C08BD1D58388EBAF914BE6B0B165E541C4
uid          [ a6cometno ] Zhanna <zhanna.panina86@gmail.com>
ssb rsa4096/688A96F8873A87DA 2025-03-04 [E]

zvpanina@zvpanina:~$ gpg --armor --export 78B3A5684756C90B
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINB6ffAbB8EACVcCuHILmryS+hl+/9wQ65NstNT39dstaq50baSx0BQGFwM
R7b/NcM6ArmKpWts7G5jsPswB8RXL725o+//zeCM6v+G107HQ5kcx0SbnqTPf3y
S10L4WLUSDkZyWUBjn4IoAGHkrInsewpt9NB8IH089XZnsHJlfeNCwAMo9RS8N4
nxvcj+/LrxUAzok5uqRnR8XpL3SRsaIt4G2sv+DeENaxYmYdfRkDv98oLhotgYqs
zhvZUkFvyFszBzm59RLxtSu4v9yJuts1DTUrwUebpo61Na1pxZcxwABCK/j2w
3m4PQvzsvk1OpETp2Py0EH8Fu540fK5SEv+gTt89L3vzo732EzWmNe/wBZdov
7HCqqs/gV5Ya+Qckof8SxMKFAGWQWao0wSewzE/3m6FM42zAYdnUTjnv387juc
V89eqSt+1ZKU9CqfbyPun5CU//a059bgl3PU2194kq0SP2T0SSQm87cNEG/n9yB
oLMVPkLwPnFUALW454ybb1LE3aDZ1N3MT5InKm+Hq09VM1M35y2dDpAwLYQGVcP
7AhvCKfx6Lk7Wemu/Pdz93MOCCtWE41b1U7qT65Rb/Wdq+2XUFVnA8/yd49vxiF
BakepXbTsyBCHP+BvoqxLO6H0mUStb989+Hg47m95anbn78Ua6842nyBwARAQAB
tC1a6Fum=guGufuonITdxdGsfubEucfua95hMOZkzz1hnmuyZ9fP0KUCUQT
AQ8A0xYhKAVw4DhvQ5VtVaBX1zpwHvYskLBQ3nyAG0AhsDBQs1CaccA1TC8hUK
```

Рис. 4.3: Ключ GPG

3. Перехожу в настройки GitHub, вставляю ключ (рис. 4.4).

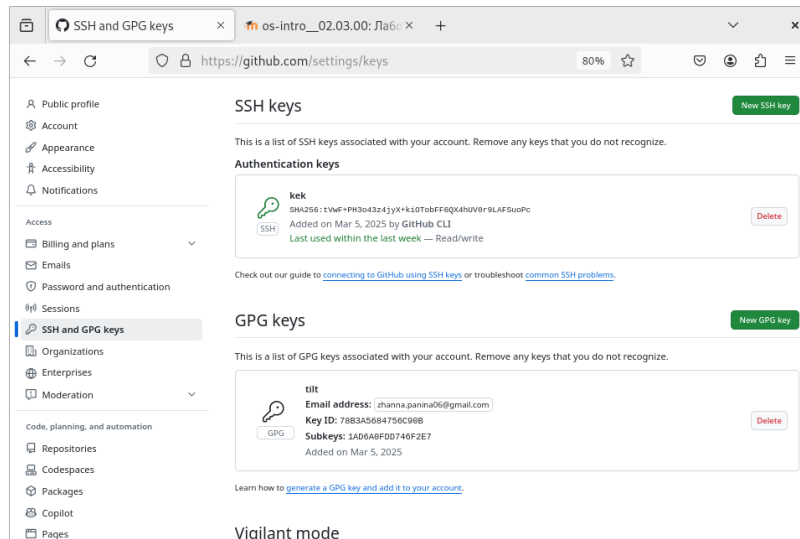


Рис. 4.4: Ключи в GitHub

4.3 Настройка подписей Git

Используя введенный email, указываю Git применять его при подписи коммитов, авторизуюсь (рис. 4.5).

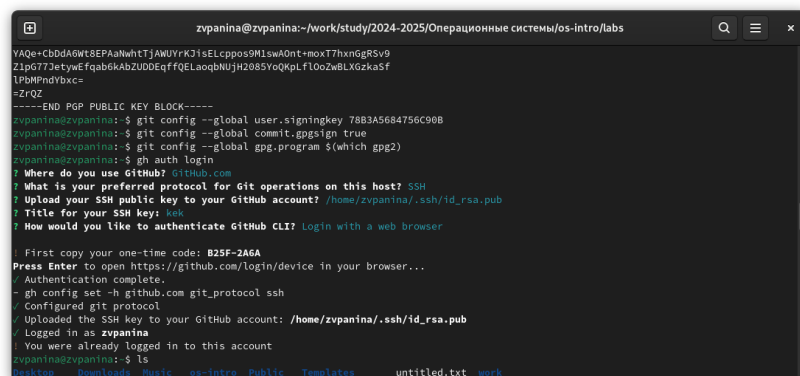


Рис. 4.5: Настройка подписей Git

4.4 Создание локального каталога

1. Создаю каталог “Операционные системы”, перехожу в него и клонирую репозиторий (рис. 4.6).

```
root@zvpalina:~/work/study/2024-2025/Операционные системы# gh repo create study_2024-2025_os-intro --te
mplate=yamadharma/course-directory-student-template --public
/ Created repository zvpalina/study_2024-2025_os-intro on GitHub
https://github.com/zvpalina/study_2024-2025_os-intro
root@zvpalina:~/work/study/2024-2025/Операционные системы# git clone --recurse git@github.com:https:/
/github.com/zvpalina/study_2024-2025_os-intro.git os-intro
Клонирование в «os-intro»...
```

Рис. 4.6: Клонирование репозитория

2. Удаляю лишние файлы, создаю необходимые каталоги (рис. 4.7).

```
zvpalina@zvpalina:~/work/study/2024-2025/Операционные системы$ rm package.json
rm: невозможно удалить 'package.json': Нет такого файла или каталога
zvpalina@zvpalina:~/work/study/2024-2025/Операционные системы/os-intro$ echo os-intro > COURSE
zvpalina@zvpalina:~/work/study/2024-2025/Операционные системы/os-intro$ make
Usage:
  make <target>

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

Рис. 4.7: Удаление файлов и создание каталогов

3. Отправка файлов на сервер. Команды git add . и git commit -m (рис. 4.8).

```
zvpalina@zvpalina:~/work/study/2024-2025/Операционные системы/os-intro/labs$ ls
lab01 lab03 lab05 lab07 lab09 lab11 lab13 lab15 README.ru.md
lab02 lab04 lab06 lab08 lab10 lab12 lab14 README.md
zvpalina@zvpalina:~/work/study/2024-2025/Операционные системы/os-intro/labs$ git add .
zvpalina@zvpalina:~/work/study/2024-2025/Операционные системы/os-intro/labs$ git status
Текущая ветка: master
Эта ветка соответствует «origin/master».

Изменения, которые будут включены в коммит:
(используйте «git restore --staged <файл>...», чтобы убрать из индекса)
новый файл: lab01/presentation/image/1.png
новый файл: lab01/presentation/image/10.png
новый файл: lab01/presentation/image/11.png
новый файл: lab01/presentation/image/12.png
новый файл: lab01/presentation/image/13.png
новый файл: lab01/presentation/image/14.png
новый файл: lab01/presentation/image/15.png
новый файл: lab01/presentation/image/2.png
новый файл: lab01/presentation/image/3.png
новый файл: lab01/presentation/image/4.png
новый файл: lab01/presentation/image/5.png
новый файл: lab01/presentation/image/6.png
новый файл: lab01/presentation/image/7.png
новый файл: lab01/presentation/image/8.png
новый файл: lab01/presentation/image/9.png
новый файл: lab01/presentation/presentation.html
новый файл: lab01/presentation/presentation.pdf
новый файл: lab01/presentation/панель_01_презентации.html
новый файл: lab01/presentation/панель_01_презентации.md
новый файл: lab01/presentation/панель_01_презентации.pdf
zvpalina@zvpalina:~/work/study/2024-2025/Операционные системы/os-intro/labs$ git commit -m "add files for lab01"
```

Рис. 4.8: Отправка файлов на сервер

4. Команда git push (рис. 4.9).

```
zvpalina@zvpalina:~/work/study/2024-2025/Операционные системы/os-intro/labs$
create mode 100644 labs/lab01/presentation/image/12.png
create mode 100644 labs/lab01/presentation/image/13.png
create mode 100644 labs/lab01/presentation/image/14.png
create mode 100644 labs/lab01/presentation/image/15.png
create mode 100644 labs/lab01/presentation/image/2.png
create mode 100644 labs/lab01/presentation/image/3.png
create mode 100644 labs/lab01/presentation/image/4.png
create mode 100644 labs/lab01/presentation/image/5.png
create mode 100644 labs/lab01/presentation/image/6.png
create mode 100644 labs/lab01/presentation/image/7.png
create mode 100644 labs/lab01/presentation/image/8.png
create mode 100644 labs/lab01/presentation/image/9.png
create mode 100644 labs/lab01/presentation/presentation.html
create mode 100644 labs/lab01/presentation/presentation.pdf
create mode 100644 labs/lab01/presentation/Панна_01_презентация.html
create mode 100644 labs/lab01/presentation/Панна_01_презентация.md
create mode 100644 labs/lab01/presentation/Панна_01_презентация.pdf
zvpalina@zvpalina:~/work/study/2024-2025/Операционные системы/os-intro/labs$ git push
Перечисление объектов: 36, готово.
Подсчет объектов: 100% (25/25), готово.
Сжатие объектов: 100% (25/25), готово.
Запись объектов: 100% (25/25), 14.86 МБ | 5.67 МБ/с, готово.
total 25 (delta 2), reused 14 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:zvpalina/study_2024-2025_os-intro.git
13a9fae..ecf85a4 master -> master
```

Рис. 4.9: Отправка файлов на сервер

5 Ответы на контрольные вопросы

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

6 Выводы

Во время выполнения работы я изучила идеологию и применения средств контроля версий, освоить умения по работе с git.