

# **Отчет по лабораторной работе №3**

*Дисциплина: Операционные системы*

Кабанова Варвара Дмитриевна

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	14
4	Контрольные вопросы	15

# Список иллюстраций

2.1	рис.1	. . . . .	6
2.2	рис.2	. . . . .	7
2.3	рис.3	. . . . .	7
2.4	рис.4	. . . . .	7
2.5	рис.5	. . . . .	7
2.6	рис.6	. . . . .	8
2.7	рис.7	. . . . .	8
2.8	рис.8	. . . . .	9
2.9	рис.9	. . . . .	9
2.10	рис.10	. . . . .	10
2.11	рис.11	. . . . .	10
2.12	рис.12	. . . . .	10
2.13	рис.13	. . . . .	10
2.14	рис.14	. . . . .	11
2.15	рис.15	. . . . .	11
2.16	рис.16	. . . . .	12
2.17	рис.17	. . . . .	12
2.18	рис.18	. . . . .	12
2.19	рис.19	. . . . .	13
2.20	рис.20	. . . . .	13
2.21	рис.21	. . . . .	13
2.22	рис.22	. . . . .	13

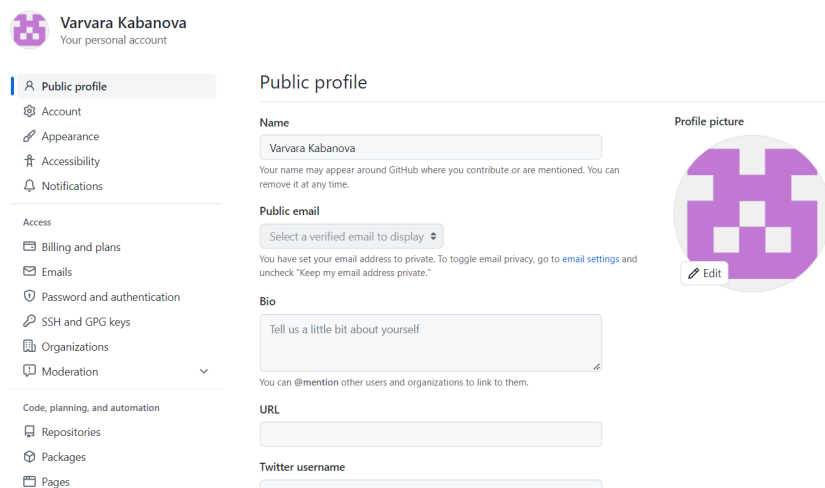
## **Список таблиц**

# 1 Цель работы

Целью данной работы является изучение идеологии и применения средств контроля версий, а также освоение умения по работе с git.

## 2 Выполнение лабораторной работы

Создаю учётную запись на <https://github.com>. Заполняю основные данные (рис.1). На рис. 2 видно, что логин совпадает с именем пользователя в дисплейном классе.



The screenshot shows the GitHub account creation profile page for 'Varvara Kabanova'. On the left is a sidebar with navigation links: 'Public profile' (selected), 'Account', 'Appearance', 'Accessibility', 'Notifications', 'Access', 'Billing and plans', 'Emails', 'Password and authentication', 'SSH and GPG keys', 'Organizations', 'Moderation', 'Code, planning, and automation', 'Repositories', 'Packages', and 'Pages'. The main content area is titled 'Public profile' and contains several input fields: 'Name' (filled with 'Varvara Kabanova'), 'Public email' (with a dropdown menu), 'Bio' (with a text area), 'URL', and 'Twitter username'. A 'Profile picture' section on the right shows a placeholder image with an 'Edit' button. A note below the name field states: 'Your name may appear around GitHub where you contribute or are mentioned. You can remove it at any time.' A note below the email field states: 'You have set your email address to private. To toggle email privacy, go to [email settings](#) and uncheck "Keep my email address private."' A note below the bio field states: 'You can @mention other users and organizations to link to them.'

Рис. 2.1: рис.1

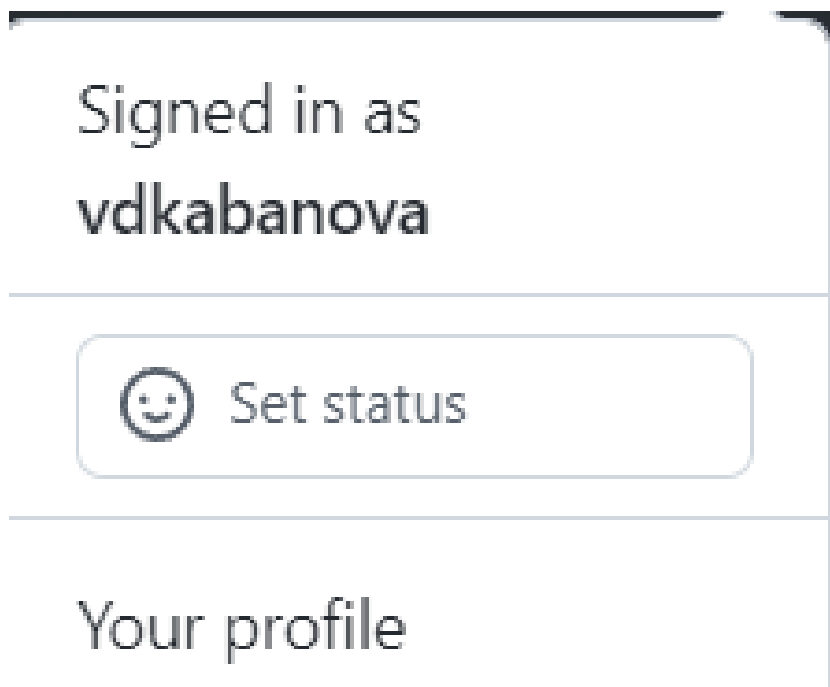


Рис. 2.2: рис.2

Начинаю базовую настройку git. Задаю имя и email владельца репозитория (рис.3).

```
vdkabanova@dk6n51 ~ $ git config --global user.name "Varvara Kabanova"
vdkabanova@dk6n51 ~ $ git config --global user.email "kabanova.varya@yandex.ru"
```

Рис. 2.3: рис.3

Настраиваю utf-8 в выводе сообщений git (рис.4).

```
vdkabanova@dk6n51 ~ $ git config --global user.email "kabanova.varya@yandex.ru"
vdkabanova@dk6n51 ~ $ git config --global core.quotePath false
vdkabanova@dk6n51 ~ $ git config --global init.defaultBranch master
```

Рис. 2.4: рис.4

Настраиваю верификацию и подписание коммитов git. Задаю имя начальной ветки (допустим master) (рис.5). А также параметр autocrlf и параметр safecrlf (рис.6).

```
vdkabanova@dk6n51 ~ $ git config --global core.quotePath false
vdkabanova@dk6n51 ~ $ git config --global init.defaultBranch master
vdkabanova@dk6n51 ~ $ git config --global core.autocrlf input
```

Рис. 2.5: рис.5

```
vdkabanova@dk6n51 ~ $ git config --global core.autocrlf input
vdkabanova@dk6n51 ~ $ git config --global core.safecrlf warn
```

Рис. 2.6: рис.6

Создаю ключи ssh – по алгоритму rsa с ключом размером 4096 бит и по алгоритму ed25519 (рис. 7).

```
vdkabanova@dk6n51 ~ $ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/afs/.dk.sci.pfu.edu.ru/home/v/d/vdkabanova/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/v/d/vdkabanova/.ssh/id_rsa
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/v/d/vdkabanova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:YT850yB10hN931iKMLKcPaD8QjwSsoAtmAmTkFCD65o vdkabanova@dk6n51
The key's randomart image is:
+---[RSA 4096]-----+
|@Oo ..+o |
|E.o.. o+.. . |
|+ o * o+..* o =|
|. . . *.+o+. o o|
|. . o oS *.. |
|. . . . * |
|. . . . |
|E |
+---[SHA256]-----+
vdkabanova@dk6n51 ~ $ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/afs/.dk.sci.pfu.edu.ru/home/v/d/vdkabanova/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/v/d/vdkabanova/.ssh/id_ed25519
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/v/d/vdkabanova/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:8exLrwzwl0UqgRwDgn/wr1gtggc3o0t/ipB6r0B/p6Y vdkabanova@dk6n51
The key's randomart image is:
+---[ED25519 256]---+
|. |
|. o . . |
|. . + . = o |
|. = o + S + . |
|. + + + * o o |
|*o o = o = |
|*o+ + . = o |
|*oEB.o +.. |
+---[SHA256]-----+
```

Рис. 2.7: рис.7

Создаю ключи gpg. Генерирую ключ. Из предложенных опций выбираю: – тип RSA and RSA; – размер 4096; срок действия: значение по умолчанию— 0 (срок действия не истекает никогда). – GPG запросил личную информацию, которая сохранится в ключе: – Имя. – Адрес электронной почты. – Ввожу email, используемый на GitHub. – Комментарий. Нажимаю клавишу ввода, чтобы оставить это поле пустым (рис. 8).



```

vdkabanova@dk6n51 ~ $ gpg --full-generate-key
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
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 и RSA (по умолчанию)
  (2) DSA и Elgamal
  (3) DSA (только для подписи)
  (4) RSA (только для подписи)
  (14) Именющийся на карте ключ
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: Varvara
Адрес электронной почты: kabanova.varya@yandex.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "Varvara <kabanova.varya@yandex.ru>"

```

Рис. 2.8: рис.8

Добавляю GPG ключ в GitHub. Вывожу список ключей и копирую отпечаток приватного ключа. Копирую сгенерированный GPG ключ в буфер обмена (рис. 9). Перехожу в настройки GitHub (<https://github.com/settings/keys>), нажимаю на кнопку New GPG key и вставляю полученный ключ в поле ввода (рис. 10).

```

vdkabanova@dk6n51 ~ $ 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
/afs/.dk.sci.pfu.edu.ru/home/v/d/vdkabanova/.gnupg/pubring.kbx
-----
sec  rsa4096/ED7B882BA67FFED7 2022-04-21 [SC]
      9B9FAAFF9C76C4EAF921E3E3ED7B882BA67FFED7
uid          [ абсолютно ] Varvara <kabanova.varya@yandex.ru>
ssb  rsa4096/6C6A16BFFD651627 2022-04-21 [E]

vdkabanova@dk6n51 ~ $ gpg --armor --export ED7B882BA67FFED7 | xclip -sel clip


```

Рис. 2.9: рис.9

## GPG keys

[New GPG key](#)

This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.



Email address: `kabanova.varya@yandex.ru`

Key ID: `ED7B882BA67FFED7`

Subkeys: `6C6A16BFFD651627`

Added on 21 Apr 2022

[Delete](#)

Learn how to [generate a GPG key and add it to your account](#).

Рис. 2.10: рис.10

Настраиваю автоматические подписи коммитов git. Используя введённый email, указываю Git применять его при подписи коммитов (рис.11-12).

```
vdkabanova@dk6n51 ~ $ git config --global user.signingkey ED7B882BA67FFED7
vdkabanova@dk6n51 ~ $ git config --global commit.gpgsign true
```

Рис. 2.11: рис.11

```
vdkabanova@dk6n51 ~ $ git config --global gpg.program $(which gpg2)
vdkabanova@dk6n51 ~ $ git config --global commit.gpgsign true
```

Рис. 2.12: рис.12

Используя шаблон для рабочего пространства (<https://github.com/yamadharma/course-directory-student-template>), создаю репозиторий os-intro (рис. 13-14).

master	1 branch	0 tags	Go to file	Add file	Code	Use this template
yamadharma	chore(submodules): update submodules	274a002	5 days ago	3 commits		
config	chore(main): add conventional changelog support		8 days ago			
template	chore(submodules): update submodules		5 days ago			
.gitattributes	Initial commit		8 days ago			
.gitignore	Initial commit		8 days ago			
.gitmodules	chore(main): add conventional changelog support		8 days ago			
LICENSE	Initial commit		8 days ago			
Makefile	chore(main): add conventional changelog support		8 days ago			
README.en.md	chore(submodules): update submodules		5 days ago			
README.git-flow.md	Initial commit		8 days ago			
README.md	chore(main): add conventional changelog support		8 days ago			
package.json	chore(main): add conventional changelog support		8 days ago			

Рис. 2.13: рис.13


## Create a new repository from course-directory-student-template


The new repository will start with the same files and folders as [yamadharm/course-directory-student-template](#).

Owner \* vdkabanova / Repository name \* os-intros ✓

Great repository names are: os-intros is available.. Need inspiration? How about [symmetrical-system](#)?

Description (optional)

☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

☐ **Include all branches**  
Copy all branches from yamadharm/course-directory-student-template and not just master.

[Create repository from template](#)

Рис. 2.14: рис.14

Создаю репозиторий курса на основе шаблона. Для этого я создаю каталог «Операционные системы» и перехожу в него (рис. 15). Далее на github я копирую ссылку на свой репозиторий (рис.16) и заполняю репозиторий по шаблону, добавляя ссылку в команду `git clone –recursive` (рис.17).

```
vdkabanova@dk6n51 ~ $ mkdir -p ~/work/study/2021-2022/"Операционные системы"
vdkabanova@dk6n51 ~ $ cd ~/work/study/2021-2022/"Операционные системы"
```

Рис. 2.15: рис.15

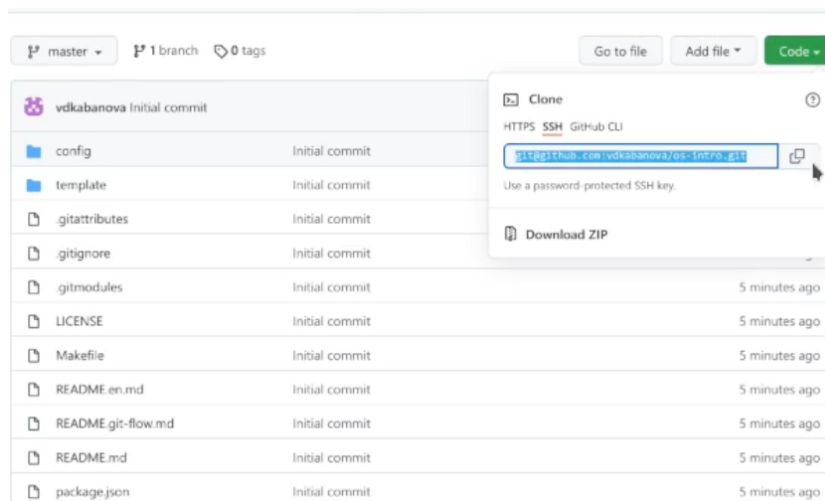


Рис. 2.16: рис.16

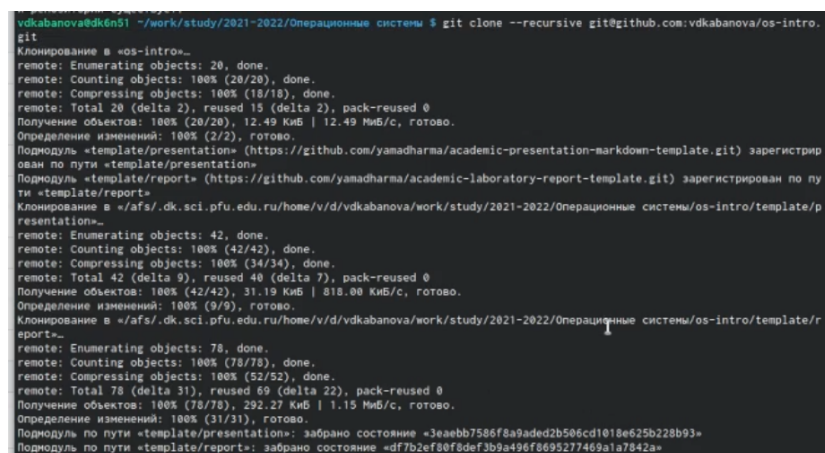


Рис. 2.17: рис.17

Настраиваю каталог курса. Перехожу в каталог курса (рис.18). Удаляю лишние файлы (package.json), но для начала проверяю его наличие в каталоге с помощью команды ls (рис.19). Создаю необходимые каталоги (рис.20). Отправляю файлы на сервер (рис.21-22).



Рис. 2.18: рис.18

```

vdkabanova@dk6n51 ~/work/study/2021-2022/Операционные системы/os-intro $ ls
config LICENSE Makefile package.json README.en.md README.git-flow.md README.md template
vdkabanova@dk6n51 ~/work/study/2021-2022/Операционные системы/os-intro $ rm package.json

```

Рис. 2.19: рис.19

```

vdkabanova@dk6n51 ~/work/study/2021-2022/Операционные системы/os-intro $ make COURSE=os-intro

```

Рис. 2.20: рис.20

```

vdkabanova@dk6n51 ~/work/study/2021-2022/Операционные системы/os-intro $ git add .
vdkabanova@dk6n51 ~/work/study/2021-2022/Операционные системы/os-intro $ git commit -am'feat(main): make course structure'
[master 6553ff0] feat(main): make course structure
149 files changed, 16590 insertions(+), 14 deletions(-)
create mode 100644 labs/lab01/presentation/Makefile
create mode 100644 labs/lab01/presentation/presentation.md
create mode 100644 labs/lab01/report/Makefile
create mode 100644 labs/lab01/report/bib/cite.bib
create mode 100644 labs/lab01/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab01/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab01/report/report.md
create mode 100644 labs/lab02/presentation/Makefile
create mode 100644 labs/lab02/presentation/presentation.md
create mode 100644 labs/lab02/report/Makefile
create mode 100644 labs/lab02/report/bib/cite.bib
create mode 100644 labs/lab02/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab02/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab02/report/report.md
create mode 100644 labs/lab03/presentation/Makefile
create mode 100644 labs/lab03/presentation/presentation.md
create mode 100644 labs/lab03/report/Makefile
create mode 100644 labs/lab03/report/bib/cite.bib
create mode 100644 labs/lab03/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab03/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 labs/lab03/report/report.md
create mode 100644 labs/lab04/presentation/Makefile
create mode 100644 labs/lab04/presentation/presentation.md
create mode 100644 labs/lab04/report/Makefile
create mode 100644 labs/lab04/report/bib/cite.bib

```

Рис. 2.21: рис.21

```

create mode 100644 project-personal/stage5/report/bib/cite.bib
create mode 100644 project-personal/stage5/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage5/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 project-personal/stage5/report/report.md
create mode 100644 project-personal/stage6/presentation/Makefile
create mode 100644 project-personal/stage6/presentation/presentation.md
create mode 100644 project-personal/stage6/report/Makefile
create mode 100644 project-personal/stage6/report/bib/cite.bib
create mode 100644 project-personal/stage6/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage6/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 project-personal/stage6/report/report.md
create mode 100644 structure
vdkabanova@dk6n51 ~/work/study/2021-2022/Операционные системы/os-intro $ git push
Перечисление объектов: 20, готово.
Подсчет объектов: 100% (20/20), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (16/16), готово.
Запись объектов: 100% (19/19), 266.53 КБ | 2.17 МБ/с, готово.
Всего 19 (изменений 2), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To github.com:vdkabanova/os-intro.git
 31c2797..6553ff0 master -> master

```

Рис. 2.22: рис.22

## 3 Выводы

В ходе выполнения данной лабораторной работы я приобрела практические навыки работы с github, научилась создавать репозитории и размещать файлы в них, что упростит работу со следующими лабораторными работами и поможет структурировать информацию, а также изучила идеологию применения средств контроля версий

## 4 Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначены?
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.
4. Опишите действия с VCS при единоличной работе с хранилищем.
5. Опишите порядок работы с общим хранилищем VCS.
6. Каковы основные задачи, решаемые инструментальным средством git?
7. Назовите и дайте краткую характеристику командам git.
8. Приведите примеры использования при работе с локальным и удалённым репозиториями.
9. Что такое и зачем могут быть нужны ветви (branches)?
10. Как и зачем можно игнорировать некоторые файлы при commit?
11. Version Control System — программное обеспечение для облегчения работы с изменяющейся информацией. 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

12. 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 неполную версию изменённых файлов, а производить так называемую дельта-компрессию—сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. 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.
13. Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Пример - Wikipedia. В децентрализованных системах каждый узел принимает свое собственное решение.



Конечное поведение системы является совокупностью решений отдельных узлов. Пример — Bitcoin. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером.

14. Создадим локальный репозиторий. Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория: `git config --global user.name "Имя Фамилия"` `git config --global user.email "work@mail"` и настроив utf-8 в выводе сообщений: `git config --global quotePath false` Для инициализации локального репозитория, расположенного, например, в каталоге `~/tutorial`, необходимо ввести в командной строке: `cd` `mkdir tutorial` `cd tutorial` `git init`
15. Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый): `ssh-keygen -C "Имя Фамилия work@mail"` Ключи сохраняются в каталоге `~/.ssh/`. Скопировав из локальной консоли ключ в буфер обмена `cat ~/.ssh/id_rsa.pub | xclip -sel clip` вставляем ключ в появившееся на сайте поле.
16. У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечивать удобства командной работы над кодом.
17. Основные команды 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 стекущим деревом: `git merge --no-ff имя_ветки` – удаление ветки: – удаление локальной уже слиятой 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`

18. Использование `git` при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий): `git add hello.txt git commit -am 'Новый файл'`
19. Проблемы, которые решают ветки `git`: • нужно постоянно создавать архивы с рабочим кодом • сложно “переключаться” между архивами • сложно перетаскивать изменения между архивами • легко что-то напутать или потерять
20. 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. Для этого сначала нужно получить списки меняющихся шаблонов:

curl -L -s <https://www.gitignore.io/api/list> Затем скачать шаблон, например,  
для С и С++ curl -L -s <https://www.gitignore.io/api/c> » .gitignore curl -L -s  
<https://www.gitignore.io/api/c++> » .gitignore