

Отчет по лабораторной работе №3

Дисциплина: Операционные системы

Волчок Кристина Александровна

Оглавление

1 Цель работы	4
2 Выполнение лабораторной работы	5
3 Вывод	12
4 Контрольные вопросы	13

List of Figures

2.1	создаем учётную запись	5
2.2	имя и email пользователя	5
2.3	Настройка utf-8	6
2.4	Настройка верификации и подписания коммитов git	6
2.5	Параметр autocrlf и параметр safecrlf	6
2.6	ключи ssh	7
2.7	Генерирование ключа	8
2.8	Вывод списка ключей и копирование отпечатка приватного ключа	9
2.9	Копирование нашего ключа и переход в настройки GitHub	10
2.10	Настройка автоматических подписей коммитов git	10
2.11	переходим в каталог курса (cd os-intro)	10
2.12	Удаление лишних файлов	11
2.13	Создание необходимых каталогов	11
2.14	отправление каталогов на сервер	11

1 Цель работы

В ходе выполнения лабораторной работы я должна изучить идеологию и применение средств контроля версий, а также освоить умения по работе с git.

2 Выполнение лабораторной работы

1. Заходим и создаем учётную запись на <https://github.com>. Далее заполняем основные данные на <https://github.com>. (рис. 2.1)

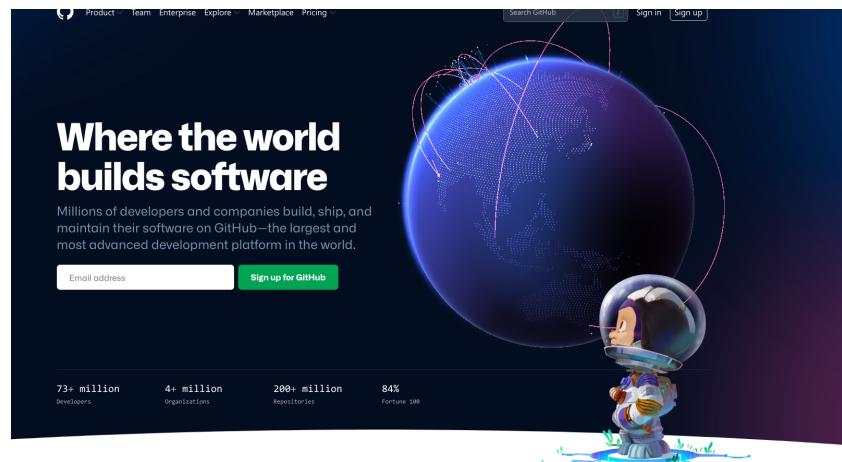


Figure 2.1: создаем учётную запись

2. Далее начинаем базовую настройку git.(рис. 2.2) Зададим имя и email владельца репозитория (Kristina Volchok):

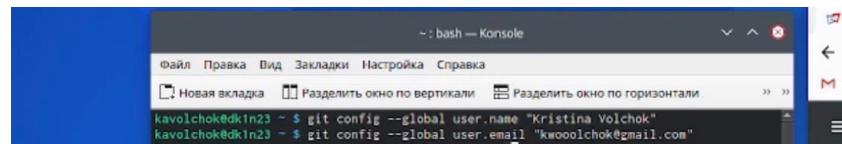
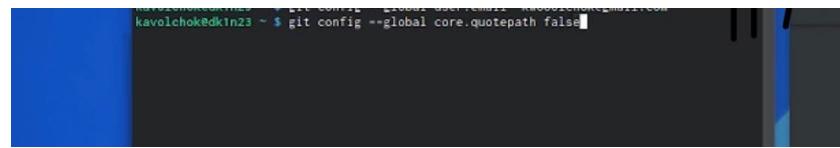


Figure 2.2: имя и email пользователя

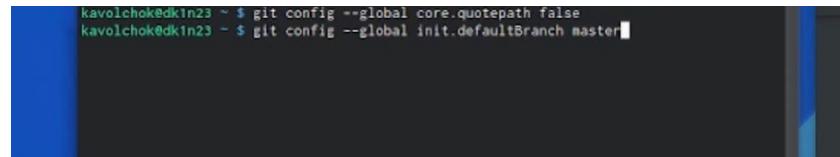
3. Настроим utf-8 в выводе сообщений git(рис.2.3):



```
kavolchok@dk1n23 ~ $ git config --global core.quotepath false
```

Figure 2.3: Настройка utf-8

4. Настроим верификацию и подписание коммитов git и зададим имя начальной ветки (будем называть её master):(рис.2.4)



```
kavolchok@dk1n23 ~ $ git config --global core.quotepath false  
kavolchok@dk1n23 ~ $ git config --global init.defaultBranch master
```

Figure 2.4: Настройка верификации и подписания коммитов git

5. Параметр autocrlf и параметр safecrlf делали вручную т. к. консоль не грузилась (рис.2.5):

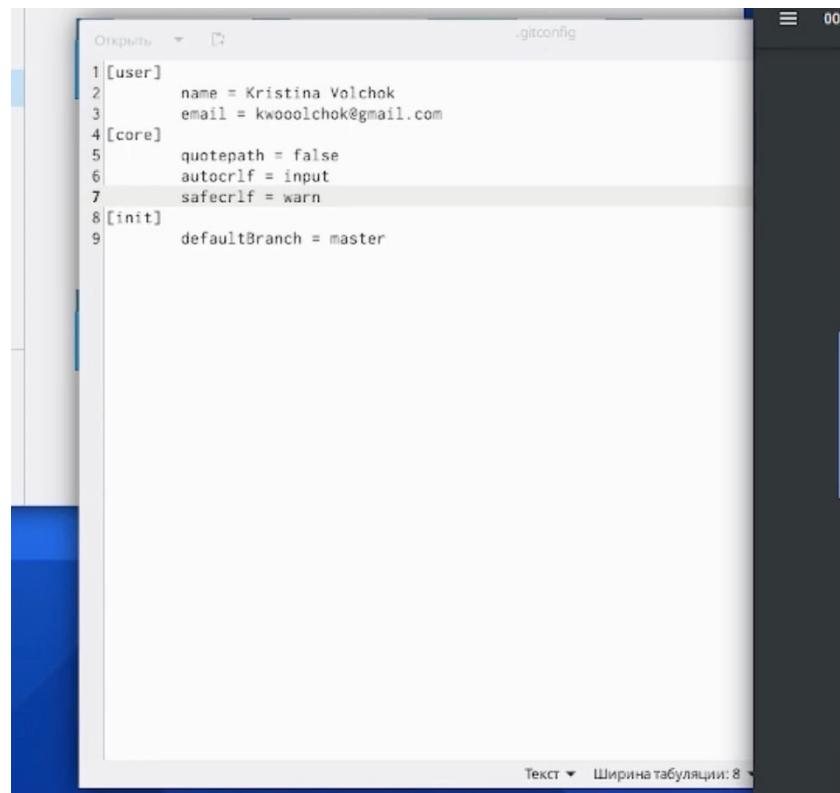
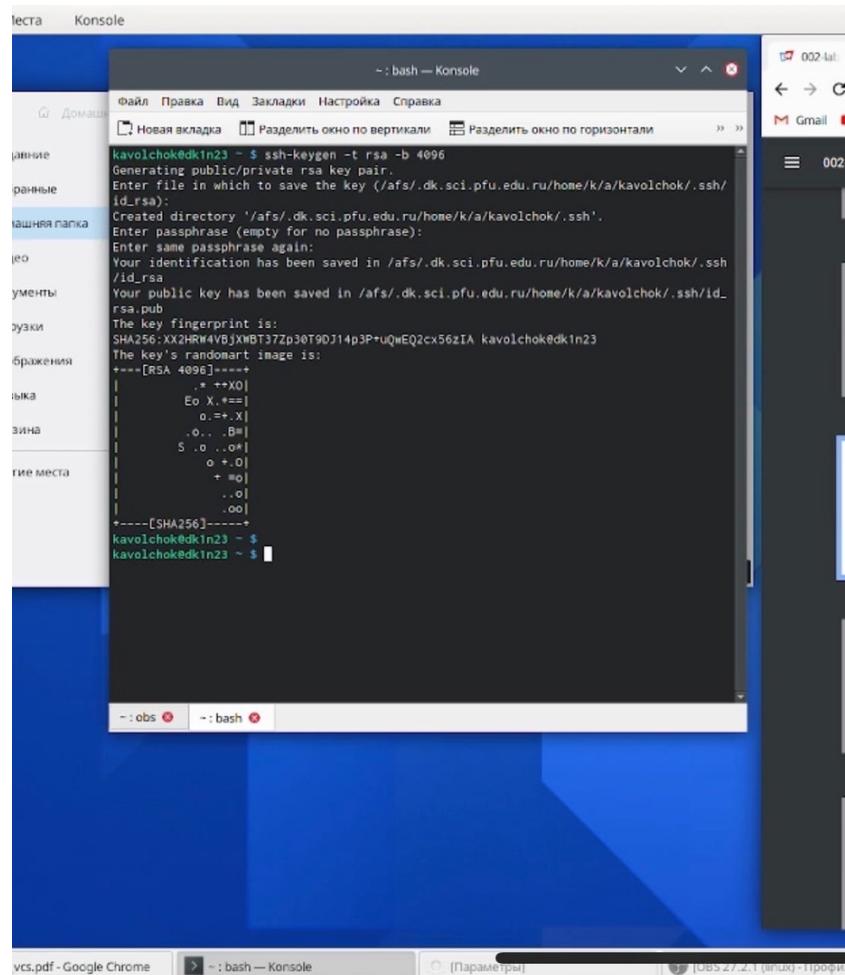


Figure 2.5: Параметр autocrlf и параметр safecrlf

6. После создаем ключи ssh по алгоритму rsa с ключём размером 4096 бит(рис.2.6):



The screenshot shows a terminal window titled 'Konsole' running on a desktop environment. The terminal displays the command 'ssh-keygen -t rsa -b 4096' being executed. The output shows the creation of a public key ('id_rsa') and a private key ('id_rsa'). It also displays the SHA256 fingerprint of the key. The terminal window is part of a desktop interface with other windows like 'Gmail' and '002' visible.

```
kavolchok@dk1n23 ~ $ 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/k/a/kavolchok/.ssh/
id_rsa):
Created directory '/afs/.dk.sci.pfu.edu.ru/home/k/a/kavolchok/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/k/a/kavolchok/.ssh/
id_rsa.
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/k/a/kavolchok/.ssh/
id_rsa.pub.
The key fingerprint is:
SHA256:XX2HRW4V8jXMBT37Zp30T9Dj14p3P+uQwEQ2cx56zIA kavolchok@dk1n23
The key's randomart image is:
+---[RSA 4096]----+
|          * ++X0|
|         Eo X.==|
|        o.=+.X|
|       .o.. B=|
|      5 ..o..o*|
|     o +.0|
|    + =o|
|   ..o|
|  .oo|
+---[SHA256]----+
kavolchok@dk1n23 ~ $
```

Figure 2.6: ключи ssh

7. Генерируем ключ(рис.2.7):

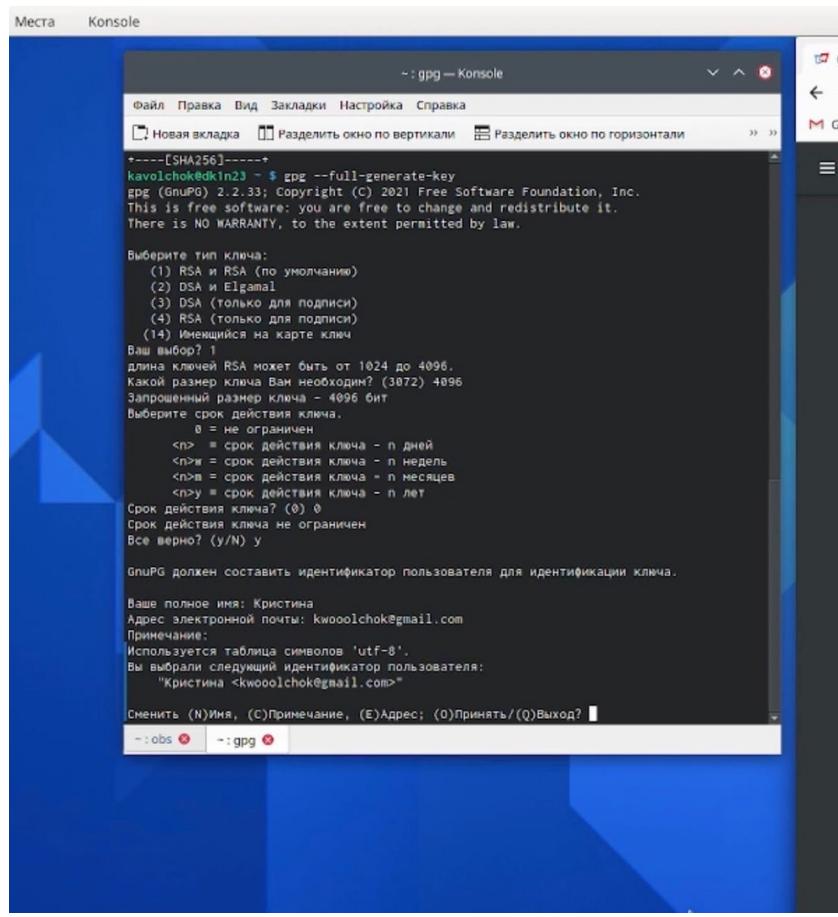
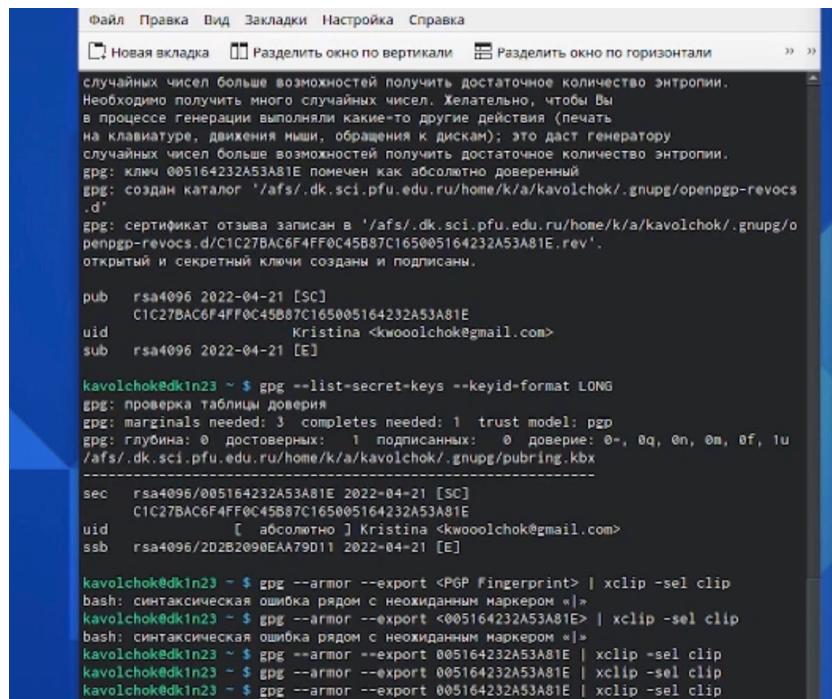


Figure 2.7: Генерирование ключа

8. Выводим список ключей и копируем отпечаток приватного ключа (Отпечаток ключа — это последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа.): Выводим список ключей и копируем отпечаток приватного ключа (Отпечаток ключа — это последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа.): (рис.2.8)



```
Файл Правка Вид Закладки Настройка Справка
Новая вкладка Разделить окно по вертикали Разделить окно по горизонтали > >
случайных чисел больше возможностей получить достаточное количество энтропии.
Необходимо получить много случайных чисел. Хорошо, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движение мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
gpg: ключ 005164232A53A81E помечен как абсолютно доверенный
gpg: создан каталог '/afs/.dk.sci.pfu.edu.ru/home/k/a/kavolchok/.gnupg/openpgp-revocs
.d'
gpg: сертификат отзыва записан в '/afs/.dk.sci.pfu.edu.ru/home/k/a/kavolchok/.gnupg/o
penpgp-revocs.d/C1C27BAC6F4FF0C45B87C165005164232A53A81E.rev'.
открытый и секретный ключи созданы и подписаны.

pub rsa4096 2022-04-21 [SC]
C1C27BAC6F4FF0C45B87C165005164232A53A81E
uid Kristina <kwooolchok@gmail.com>
sub rsa4096 2022-04-21 [E]

kavolchok@dkIn23 ~ $ 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 подписаных: 0 доверие: 0=, 0q, 0n, 0m, 0f, 1u
/afs/.dk.sci.pfu.edu.ru/home/k/a/kavolchok/.gnupg/pubring.kbx

sec rsa4096/005164232A53A81E 2022-04-21 [SC]
C1C27BAC6F4FF0C45B87C165005164232A53A81E
uid [ абсолютно ] Kristina <kwooolchok@gmail.com>
ssb rsa4096/2D2B2090EAA79D11 2022-04-21 [E]

kavolchok@dkIn23 ~ $ gpg --armor --export <PGP Fingerprint> | xclip -sel clip
bash: синтаксическая ошибка рядом с неожиданным маркером «|»
kavolchok@dkIn23 ~ $ gpg --armor --export <005164232A53A81E> | xclip -sel clip
bash: синтаксическая ошибка рядом с неожиданным маркером «|»
kavolchok@dkIn23 ~ $ gpg --armor --export 005164232A53A81E | xclip -sel clip
kavolchok@dkIn23 ~ $ gpg --armor --export 005164232A53A81E | xclip -sel clip
kavolchok@dkIn23 ~ $ gpg --armor --export 005164232A53A81E | xclip -sel clip
```

Figure 2.8: Вывод списка ключей и копирование отпечатка приватного ключа

9. Копируем наш ключ и переходим в настройки GitHub (<https://github.com/settings/keys>),
нажмите на кнопку New GPG key и вставляем полученный ключ в поле
ввода. На этом же этапе добавляем SSH ключ из буфера обмена(рис.2.9).

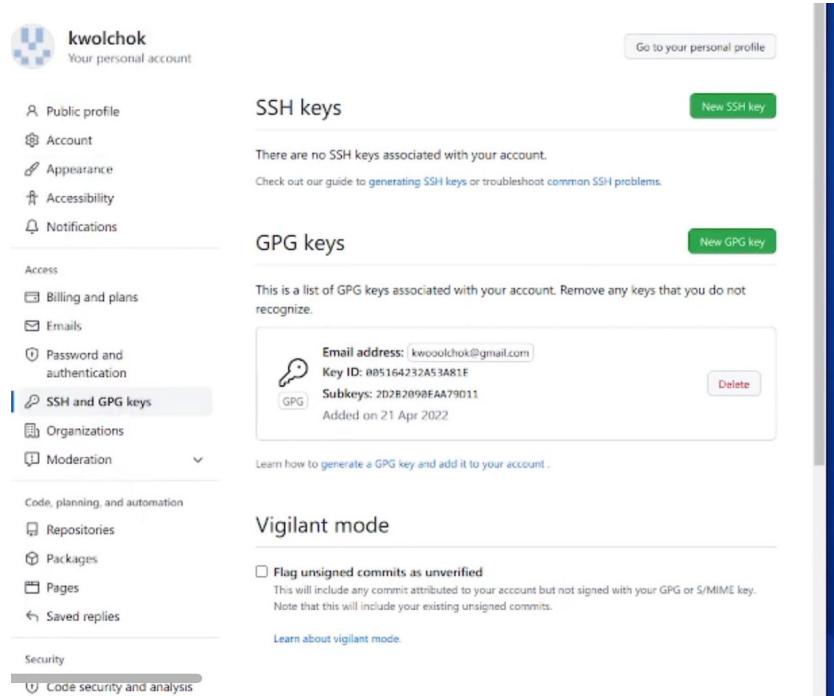


Figure 2.9: Копирование нашего ключа и переход в настройки GitHub

10. Настройка автоматических подписей коммитов git Используя введённый email, указываем Git применяем его при подписи коммитов(рис.2.10):

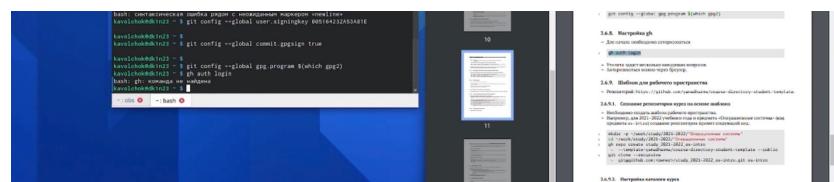


Figure 2.10: Настройка автоматических подписей коммитов git

11. Позже для настройка gh мы авторизовываемся через браузер и переходим в каталог курса (cd os-intro)(рис.??).



Figure 2.11: переходим в каталог курса (cd os-intro)

12. Удаляем лишние файлы(rm package.json)(рис.2.12):



Figure 2.12: Удаление лишних файлов

13. Создаем необходимые каталоги (make COURSE=os-intro)(рис.2.13):



Figure 2.13: Создание необходимых каталогов

14. И после отправляем каталоги на сервер(-git add-git commit -am 'feat(main): make course structure -git push)(рис.2.14):



Figure 2.14: отправление каталогов на сервер

```
create mode 100644 project-personal/stage6/report/report.md
create mode 100644 structure
kavolchok@kdk1n23 ~os-intro $ git push
Перечисление объектов: 20, готово.
Подсчет объектов: 100% (20/20), готово.
При сжатии изменений используется 2 потока
Сжатие объектов: 100% (16/16), готово.
Готово 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:kavolchok/os-intro.git
   0a65e50..b9dca96  master -> master
```

3 Вывод

В ходе выполнения лабораторной работы я изучила идеологию и применение средств контроля версий, а также освоила умения по работе с git.

4 Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?
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. Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями. 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 нескольки-ми 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.

14. 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 дельтакомпрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.
15. Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общув ходе выполнения лабораторной работы я должна изучить идеологию и применение средств контроля версий, а также освоить умения по работе с git.ю 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.
16. У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение 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 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.
18. Использования 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 перетас-

кивать изменев ходе выполнения лабораторной работы я должна изучить идеологию и применение средств контроля версий, а также освоить умения по работе с git.ния 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