

# **Лабораторная работа №3**

**Markdown**

Заболотная Кристина Александровна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Контрольные вопросы</b>	<b>13</b>
<b>6</b>	<b>Выводы</b>	<b>20</b>
	<b>Список литературы</b>	<b>21</b>

## Список иллюстраций

4.1	Аккаунт github . . . . .	8
4.2	key . . . . .	9
4.3	Создаём шаблон . . . . .	9
4.4	Создаём шаблон . . . . .	10
4.5	Создаём ключ . . . . .	10
4.6	Создаём ключ . . . . .	11
4.7	Экспортируем ключ . . . . .	11
4.8	Каталог . . . . .	12
4.9	push . . . . .	12

## Список таблиц

# 1 Цель работы

Научиться оформлять отчёты с помощью легковесного языка разметки Markdown.

## 2 Задание

1. Сделать отчёт по предыдущей лабораторной работе - №2 в формате Markdown.
2. В качестве отчёта предоставить отчёты в 3 форматах: pdf, docx и md (в архиве, поскольку он должен содержать скриншоты, Makefile и т.д.)

### 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.

## 4 Выполнение лабораторной работы

### 0. Аккаунт github и ключи ssh, gpg.

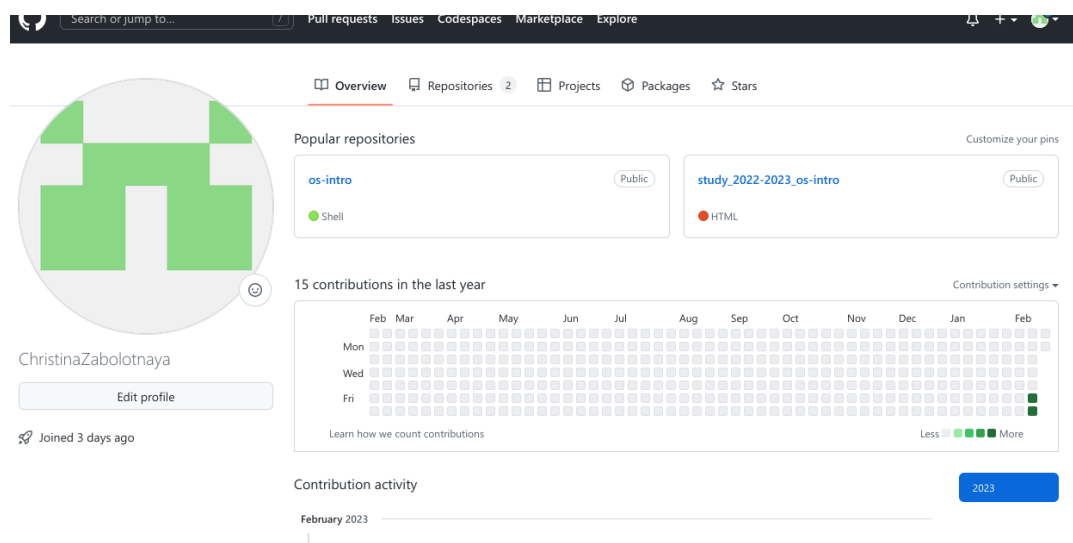


Рис. 4.1: Аккаунт github



The screenshot shows the GitHub account settings for ChristinaZabolotnaya. The left sidebar contains navigation links: Public profile, Account, Appearance, Accessibility, Notifications, Access, Billing and plans, Emails, Password and authentication, Sessions, SSH and GPG keys (highlighted), Organizations, and Moderation. The main content area is divided into two sections: SSH keys and GPG keys. The SSH keys section has a 'New SSH key' button and a list of authentication keys. One key is shown with a title, SHA256 fingerprint, addition date (Feb 17, 2023), last used status, and permissions (Read/write). The GPG keys section has a 'New GPG key' button and a list of GPG keys. One key is shown with an email address, key ID, subkeys, and addition date. Both sections include a 'Delete' button for each key.

Рис. 4.2: key

1. Создадим шаблон рабочего пространства.

```
kazabolotnaya@dk6n57 ~ $ mkdir -p ~/work/study/2022-2023/"Операционные системы"
kazabolotnaya@dk6n57 ~ $ cd ~/work/study/2022-2023/"Операционные системы"
kazabolotnaya@dk6n57 ~/work/study/2022-2023/Операционные системы $ gh repo create os-intro --template=yamadharma/course-directory-student-template --public
GraphQL: Could not clone: Name already exists on this account (cloneTemplateRepository)
kazabolotnaya@dk6n57 ~/work/study/2022-2023/Операционные системы $ ^C
kazabolotnaya@dk6n57 ~/work/study/2022-2023/Операционные системы $ gh repo create study_2022-2023_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository ChristinaZabolotnaya/study_2022-2023_os-intro on GitHub
```

Рис. 4.3: Создаём шаблон

2. Ссылку - git@github.com: - берем с нашего gitub.

```
kazabolotnaya@dk6n57 ~/work/study/2022-2023/Операционные системы $ gh repo create study_2022-2023_
s-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository ChristinaZabolotnaya/study_2022-2023_os-intro on GitHub
kazabolotnaya@dk6n57 ~/work/study/2022-2023/Операционные системы $ git clone --recursive git@github
.com:ChristinaZabolotnaya/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), 16.94 КиБ | 16.94 МиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-te
plate.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git)
зарегистрирован по пути «template/report»
Клонирование в «/afs/.dk.sci.pfu.edu.ru/home/k/a/kazabolotnaya/work/study/2022-2023/Операционные с
истемы/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 | 1.05 МиБ/с, готово.
Определение изменений: 100% (28/28), готово.
```

Рис. 4.4: Создаём шаблон

3. Создадим ключ `pgp`. Генерируем ключ. Выбираем опции: тип RSA and RSA; размер 4096; выберите срок действия; значение по умолчанию — 0 (срок действия не истекает никогда).

```
kazabolotnaya@dk6n57 ~/work/study/2022-2023/Операционные системы $ gpg --full-generate-key
gpg (GnuPG) 2.2.40; Copyright (C) 2022 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 и RSA (по умолчанию)
  (2) DSA и Elgamal
  (3) DSA (только для подписи)
  (4) RSA (только для подписи)
  (14) Имеющийся 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? (у/Н) у

GnuPG должен составить идентификатор пользователя для идентификации ключа.
```

Рис. 4.5: Создаём ключ

4. GPG запросит личную информацию, которая сохранится в ключе: Имя (не менее 5 символов), Адрес электронной почты. Комментарий, оставляю это поле пустым.

```
Ваше полное имя: ChristinaZabolotnaya03
Адрес электронной почты: christinazabolotnaya@mail.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
"ChristinaZabolotnaya03 <christinazabolotnaya@mail.ru>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? O
Необходимо получить много случайных чисел. Хелательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
Необходимо получить много случайных чисел. Хелательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
gpg: создан каталог '/afs/.dk.sci.pfu.edu.ru/home/k/a/kazabolotnaya/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/afs/.dk.sci.pfu.edu.ru/home/k/a/kazabolotnaya/.gnupg/openpgp-revocs.d/9BC2FB9574B6F64FB1812D1C27A93B2EC779A66A.rev'.
Открытый и секретный ключи созданы и подписаны.

pub  rsa4096 2023-02-16 [SC]
     9BC2FB9574B6F64FB1812D1C27A93B2EC779A66A
uid          ChristinaZabolotnaya03 <christinazabolotnaya@mail.ru>
sub  rsa4096 2023-02-16 [E]

kazabolotnaya@dk6n57 ~/work/study/2022-2023/Операционные системы $
```

Рис. 4.6: Создаём ключ

5. Экспортируем ключ в формате ASCII по его отпечатку.

```
kazabolotnaya@dk6n57 ~/work/study/2022-2023/Операционные системы $ gpg --armor --export
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGPuULUBEADGvY5Lc8a3jpuW/FAGotD9UJuf++S/HDnqm3zS+D4RqFbEds7m
OEZv07NdMKc37/+Mau201hGsWHBuNb/d2dz4SMmh7bXdZ7CceZ9jMnON/XoVDuvT
5IovTRAG8YAP5+bZQMNEhD/A8Gprmpw5xZ4HZ06LV9r6ABz1Yo0sq3y9CVjQYfg
iH0Y5J0JcyMPJngZtTvJNKAcLsivWp5CmwZHT9cVhECM2MrPvhLrraHjpNqj+Ngd
xrWbpALc+jNayf5zn2RI/cWWmEBoJo4pZIwp2xp0oAZiss6KQFQGrjit7U9CX2cx
X4ihhqA/9UZBhIWMR6UoeVoG66Nsanv6FSyBAHeHq9qasGTYD5vWy138Pr+Qzs/5
jU7hTx/k8HNN74pCovQND4K4NC3i7d2E/s512NbqgUWm9yx+eeIeA45nm0sezYBN
VElqMnCdQrNQAhwa00kMs37EtD2v5DRRAis3Hh5u3wHbjGzJ83WvBsAYgurjsFS
p3Jm1rjiXs51a+B1b9vRpGvGbTYPGsU4CAaCKqwfncDuNhp6XAt0z0V7pjfdkYiW
YcVEtTghrKs0XvGBsz03z1jQ32x+b0/NjosvFvwbgY1bgY7ZSxyiRWn1D1H6enZv
GzBJ86sXqEljA6/nycHu/fyFWYDQvZkize8IQofA75GNbW45Zvv/AHqvlwARAQAB
tDVBdAHnc3RpbmF5YWJybG90bmF5YTAzIDxiAHnc3RpbmF5YWJybG90bmF5YURt
-----END PGP PUBLIC KEY BLOCK-----
```

Рис. 4.7: Экспортируем ключ

6. Перейдём в каталог курса, создадим необходимые каталоги, отправим файлы на сервер.

```

Получение объектов: 100% (101/101), 327.25 КиБ | 2.07 МиБ/с, готово.
Определение изменений: 100% (40/40), готово.
Submodule path 'template/presentation': checked out 'b1be3800ee91f5809264cb755d316174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b82e3aef11a33b1e3b2'
kazaboltnaya@dk6n57 ~/work/study/2022-2023/Операционные системы $ cd ~/work/study/2022-2023/"Операционные системы"/os-intro
kazaboltnaya@dk6n57 ~/work/study/2022-2023/Операционные системы/os-intro $ echo os-intro > COURSE
kazaboltnaya@dk6n57 ~/work/study/2022-2023/Операционные системы/os-intro $ make
kazaboltnaya@dk6n57 ~/work/study/2022-2023/Операционные системы/os-intro $ git add .
kazaboltnaya@dk6n57 ~/work/study/2022-2023/Операционные системы/os-intro $ git commit -am 'feat(main): make course structure'
[master 311c55b] feat(main): make course structure
360 files changed, 100327 insertions(+)
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
create mode 100644 labs/lab01/presentation/Makefile
create mode 100644 labs/lab01/presentation/image/kulyabov.jpg
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 100755 labs/lab01/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_fignos.py

```

Рис. 4.8: Каталог

## 7. Отправим файлы на сервер.

```

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 100755 project-personal/stage6/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_fignos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_secnos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/core.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/main.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 project-personal/stage6/report/report.md
kazaboltnaya@dk6n57 ~/work/study/2022-2023/Операционные системы/os-intro $ git push
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 343.06 КиБ | 2.38 МиБ/с, готово.
Всего 38 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:ChristinaZaboltnaya/study_2022-2023_os-intro.git
   b973c2e..311c55b  master -> master
kazaboltnaya@dk6n57 ~/work/study/2022-2023/Операционные системы/os-intro $

```

Рис. 4.9: push

## 5 Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Система контроля версий (VCS) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов. Для примеров в этой книге мы будем использовать исходные коды программ, но на самом деле под версионный контроль можно поместить файлы практически любого типа. Если вы графический или вебдизайнер и хотели бы хранить каждую версию изображения или макета — а этого вам наверняка хочется — то пользоваться системой контроля версий будет очень мудрым решением. даёт возможность возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения, определять, кто последним вносил изменения во внезапно переставший работать модуль, кто и когда внёс в код какую-то ошибку, и многое другое. Вообще, если, пользуясь, вы всё испортите или потеряете файлы, всё можно будет легко восстановить. Вдобавок, накладные расходы за всё, что вы получаете, будут очень маленькими.
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Хранилище-система, которая обеспечивает хранение всех существовавших вариантов файлов Commit-фиксация изменений История-список предыдущих ревизий Рабочая копия-копия другой ветки Команде commit можно передать сообщение, описывающее изменения в

реvisions. Она также записывает идентификатор пользователя, текущее время и временную зону, плюс список измененных файлов и их содержимого. Сообщение, описывающее изменения, определяется через опцию -m, или -message. Можно также вводить сообщения, состоящие из нескольких строк; в большинстве оболочек вы можете сделать это оставив открытую кавычку в конце строки. `commit -m` “добавлен первый файл.

3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Системы контроля версий. Централизованная система контроля версий Subversion и децентрализованная система контроля версий Mercurial. Существуют СКВ централизованные, в которых имеется один репозиторий, в который собираются изменения со всех рабочих копий разработчиков, и децентрализованные, когда репозиториев много, и они могут обмениваться изменениями между собой. Централизованные СКВ - репозиторий один. У каждого разработчика своя рабочая копия. Время от времени разработчик может затягивать к себе в рабочую копию новые изменения из репозитория, или проталкивать свои изменения из своей рабочей копии в репозиторий. Прочие особенности централизованных СКВ зависят от реализации.
4. Опишите действия с VCS при единоличной работе с хранилищем. Традиционные системы управления версиями используют централизованную модель, когда имеется единое хранилище документов, управляемое специальным сервером, который и выполняет большую часть функций по управлению версиями. Пользователь, работающий с документами, должен сначала получить нужную ему версию документа из хранилища; обычно создаётся локальная копия документа, т. н. «рабочая копия». Может быть получена последняя версия или любая из предыдущих, которая может быть выбрана по номеру версии или дате создания, иногда и по другим признакам. После того, как в документ внесены нужные изменения, но-

вая версия помещается в хранилище. В отличие от простого сохранения файла, предыдущая версия не стирается, а тоже остаётся в хранилище и может быть оттуда получена в любое время. Сервер может использовать т. н. дельта-компрессию — такой способ хранения документов, при котором сохраняются только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Поскольку обычно наиболее востребованной является последняя версия файла, система может при сохранении новой версии сохранять её целиком, заменяя в хранилище последнюю ранее сохранённую версию на разницу между этой и последней версией. Некоторые системы (например, ClearCase) поддерживают сохранение версий обоих видов: большинство версий сохраняется в виде дельт, но периодически (по специальной команде администратора) выполняется сохранение версий всех файлов в полном виде; такой подход обеспечивает максимально полное восстановление истории в случае повреждения репозитория.

5. Опишите порядок работы с общим хранилищем VCS. Традиционные системы управления версиями используют централизованную модель, когда имеется единое хранилище документов, управляемое специальным сервером, который и выполняет большую часть функций по управлению версиями. Пользователь, работающий с документами, должен сначала получить нужную ему версию документа из хранилища; обычно создаётся локальная копия документа, т. н. «рабочая копия». Может быть получена последняя версия или любая из предыдущих, которая может быть выбрана по номеру версии или дате создания, иногда и по другим признакам. После того, как в документ внесены нужные изменения, новая версия помещается в хранилище. В отличие от простого сохранения файла, предыдущая версия не стирается, а тоже остаётся в хранилище и может быть оттуда получена в любое время. Сервер может использовать т. н. дельт компрессию — такой способ хранения документов, при котором сохраняются только измене-

ния между последовательными версиями, что позволяет уменьшить объём хранимых данных. Поскольку обычно наиболее востребованной является последняя версия файла, система может при сохранении новой версии сохранять её целиком, заменяя в хранилище последнюю ранее сохранённую версию на разницу между этой и последней версией. Некоторые системы (например, ClearCase) поддерживают сохранение версий обоих видов: большинство версий сохраняется в виде дельт, но периодически (по специальной команде администратора) выполняется сохранение версий всех файлов в полном виде; такой подход обеспечивает максимально полное восстановление истории в случае повреждения репозитория.

6. Каковы основные задачи, решаемые инструментальным средством git? Устанавливает единственную новую команду, git. Все возможности предоставляются через подкоманды этой команды. Вы можете просмотреть краткую справку командой help. Некоторые идеи группируются по темам, используйте help topics для списка доступных тем. Одна из функций системы контроля версий — отслеживать кто сделал изменения. В распределенных системах для этого требуется идентифицировать каждого автора уникально в глобальном плане. Большинство людей уже имеют такой идентификатор: email адрес. Bazaar достаточно умен, чтобы автоматически создавать email адрес из текущего имени и адреса хоста. Основные задачи: создание ветки, размещение веток, просмотр изменений, фиксация изменений, сообщение из текстового редактора, выборочная фиксация, удаление зафиксированных изменений, игнорирование файлов, просмотр истории, статистика ветки, контроль файлов и каталогов, ветвление, объединение веток, публикация ветки.
7. Назовите и дайте краткую характеристику командам git. Обновление рабочей копии По мере внесения изменений в проект рабочая копия на компьютере разработчика стареет, расхождение её с основной версией проекта



увеличивается. Это повышает риск возникновения конфликтных изменений (см. ниже). Поэтому удобно поддерживать рабочую копию в состоянии, максимально близком к текущей основной версии, для чего разработчик выполняет операцию обновления рабочей копии (update) насколько возможно часто (реальная частота обновлений определяется частотой внесения изменений, зависящей от активности разработки и числа разработчиков, а также временем, затрачиваемым на каждое обновление — если оно велико, разработчик вынужден ограничивать частоту обновлений, чтобы не терять время). Модификация проекта Разработчик модифицирует проект, изменяя входящие в него файлы в рабочей копии в соответствии с проектным заданием. Эта работа производится локально и не требует обращений к серверу VCS. Фиксация изменений Завершив очередной этап работы над заданием, разработчик фиксирует (commit) свои изменения, передавая их на сервер (либо в основную ветвь, если работа над заданием полностью завершена, либо в отдельную ветвь разработки данного задания). VCS может требовать от разработчика перед фиксацией обязательно выполнить обновление рабочей копии. При наличии в системе поддержки отложенных изменений (shelving) изменения могут быть переданы на сервер без фиксации. Если утверждённая политика работы в VCS это позволяет, то фиксация изменений может проводиться не ежедневно, а только по завершении работы над заданием; в этом случае до завершения работы все связанные с заданием изменения сохраняются только в локальной рабочей копии разработчика.

8. Приведите примеры использования при работе с локальным и удалённым репозиториями. Мы создаем новую ветку выполнив `git init` в уже созданном каталоге: `% mkdir tutorial % cd tutorial % ls -a ./ ../ % pwd /home/mbp/work/bzr.test/tutorial % % git init % ls -aF ./ ../ .git/ %` Мы обычно обращаемся к веткам на нашем компьютере просто передав имя каталога содержащего ветку. bzr также поддерживает доступ к веткам

через http и sftp, например: `git log http://bazaar-vcs.org git // git.dev/ git log sftp://bazaarvcs.org/bzr/bzr.dev/` Установив для git плагины можно также осуществлять доступ к веткам с использованием rsync. Команда `status` показывает какие изменения были сделаны в рабочем каталоге с момента последней ревизии: `% git status modified: foo bzr status` скрывает неинтересные файлы, которые либо не менялись, либо игнорируются. Также команде `status` могут быть переданы необязательные имена файлов, или каталогов для проверки.

9. Что такое и зачем могут быть нужны ветви (branches)? Часто вместо того что бы начинать свой собственный проект, вы хотите предложить изменения для уже готового проекта. Что бы сделать это вам нужно получить копию готовой ветки. Так как эта копия может быть потенциальной новой веткой эта команда называется `branch`: Управление версиями `git branch cd git.dev` Эта команда копирует полную историю ветки и после этого вы можете делать все операции с ней локально: просматривать журнал, создавать и объединять другие ветки.
10. Как и зачем можно игнорировать некоторые файлы при `commit`? Нет проблем если шаблон для игнорирования подходит для файла под контролем версий, или вы добавили файл, который игнорируется. Шаблоны не имеют никакого эффекта на файлы под контролем версий, они только определяют показываються неизвестные файлы, или просто игнорируются. Файл `git.rignore` обычно должен быть под контролем версий, что бы новые копии ветки видели такие же шаблоны: `git add . gitignore git commit -m "Добавлены шаблоны для игнорирования"`. Многие деревья с исходным кодом содержат файлы, которые не нужно хранить под контролем версий, например, резервные файлы текстового редактора, объектные файлы и собранные программы. Вы можете просто не добавлять их, но они всегда будут обнаруживаться как неизвестные. Вы также можете сказать `bzr` игнорировать

их добавив их в файл в корне рабочего дерева. Этот файл содержит список шаблонов файлов, по одному в каждой строке. Обычное содержимое может быть таким: `.o ~ .tmp .py [ со ]` Если шаблон содержит слеш, то он будет сопоставлен с полным путем начиная от корня рабочего дерева; иначе он сопоставляется только с именем файла. Таким образом пример выше игнорирует файлы с расширением `.o` во всех подкаталогах, но пример ниже игнорирует только `config.h` в корне рабочего дерева и HTML файлы в каталоге `doc/`: `./config.h doc/.html` Для получения списка файлов которые игнорируются и соответствующих им шаблонов используйте команду `git ignored` :

```
$ git ignored config.h ./config.h configure.in~ ~ $.
```

## 6 Выводы

В ходе выполнения данной лабораторной работы я научилась оформлять отчёты с помощью легковесного языка разметки Markdown.

## **Список литературы**