

Отчет по лабораторной работе № 2

Российский университет дружбы народов

Абдулгалимов Мурад Арсенович

Содержание

1 Цель работы	5
2 Задание	6
3 Теоретическое введение	7
4 Выполнение лабораторной работы	8
4.1 1. Создал учетную запись и заполнил основные данные	8
4.2 2. Установил git-flow в Fedora Linux(рис. 4.3, 4.4):	9
4.3 3. Провел базовую настройку Git(рис. 4.5):	10
4.4 4. Создал ключи SSH и PGP(рис. 4.6, 4.7):	11
4.5 5. Добавление PGP в GitHub(рис. 4.8, 4.9):	11
4.6 6. Настроил автоматические подписи для комитов Git(рис. 4.10, 4.11):	12
4.7 7. Создание репозитория курса на основе шаблона(рис. 4.12): . . .	13
4.8 8. Настроил каталог курса и отправил на сервер(рис. 4.13): . . .	14
4.9 Контрольные вопросы:	15
4.9.1 1. Системы контроля версий — это программные инструменты,	15
4.9.2 2. Объясните следующие понятия VCS и их отношения: . .	15
4.9.3 3. Что представляют собой и чем отличаются централизо- ванные	16
4.9.4 4. Опишите действия с VCS при единоличной работе с . . .	16
4.9.5 5. Опишите порядок работы с общим хранилищем VCS. . .	16
4.9.6 6. Каковы основные задачи, решаемые инструментальным	16
4.9.7 7. Назовите и дайте краткую характеристику командам git.	16
4.9.8 8. Приведите примеры использования при работе с локаль- ным и удалённым репозиториями.	17
4.9.9 9. Что такое и зачем могут быть нужны ветви (branches)? .	17
4.9.10 10. Как и зачем можно игнорировать некоторые файлы при	17
5 Выводы	19
Список литературы	20

Список иллюстраций

4.1	Создание учетной записи	8
4.2	Ввод основной информации	9
4.3	Установка git-flow в Fedora Linux	9
4.4	Установка git-flow в Fedora Linux	10
4.5	Проведение базовой настройки Git	10
4.6	Генерация SSH	11
4.7	Генерация PGP	11
4.8	Добавление PGP в GitHub	12
4.9	Добавление PGP в GitHub	12
4.10	Настроил автоматические подписи для комитов Git	13
4.11	Настроил gh	13
4.12	Создание репозитория на основе шаблона	14
4.13	Настройка каталога курса	15

Список таблиц

3.1 Описание некоторых каталогов файловой системы GNU Linux . . . 7

1 Цель работы

- Изучить идеологию и применение средств контроля версий.
- Освоить умения по работе с git.

2 Задание

Составить отчет по прошлой лабораторной работе на языке разметки markdown.

3 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы.

Например, в табл. 3.1 приведено краткое описание стандартных каталогов Unix.

Таблица 3.1: Описание некоторых каталогов файловой системы GNU Linux

Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно об Unix см. в [1–6].

4 Выполнение лабораторной работы

4.1 1. Создал учетную запись и заполнил основные данные

(рис. 4.1, 4.2):

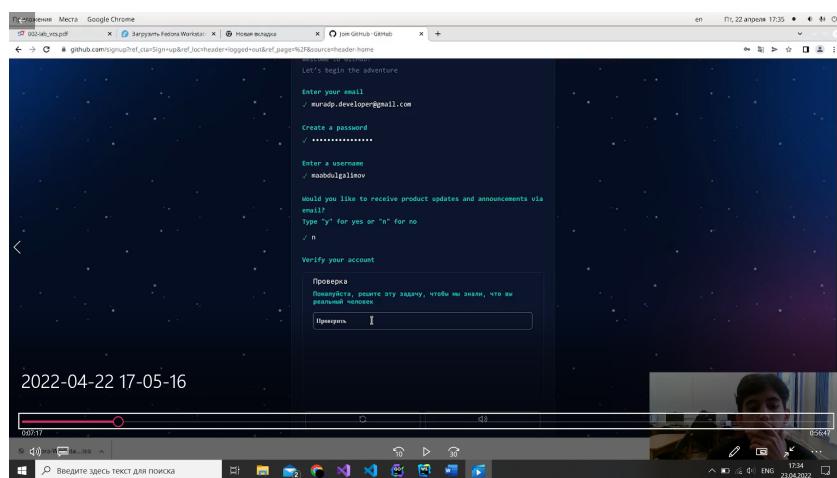


Рис. 4.1: Создание учетной записи

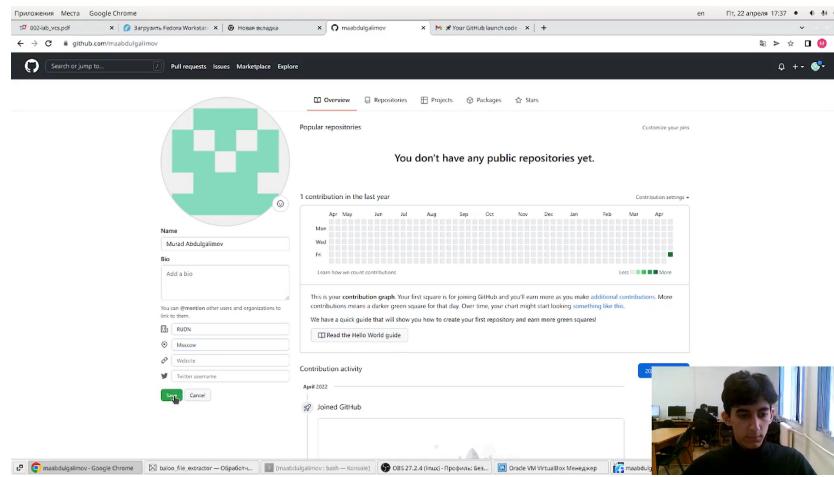


Рис. 4.2: Ввод основной информации

4.2 2. Установил git-flow в Fedora Linux(рис. 4.3, 4.4):

Команды:

```
cd /tmp
wget --no-check-certificate -q https://raw.github.com/petervanderdoes/gitflow/develop/installer.sh
chmod +x gitflow-installer.sh
sudo ./gitflow-installer.sh install stable
```

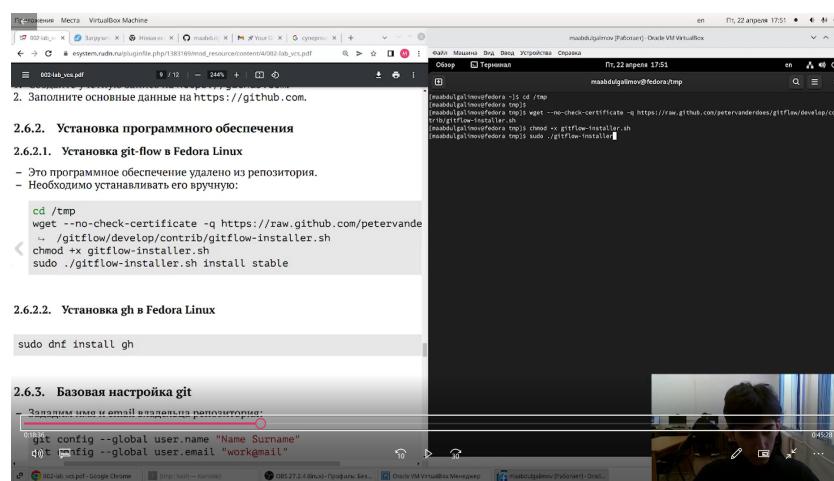


Рис. 4.3: Установка git-flow в Fedora Linux

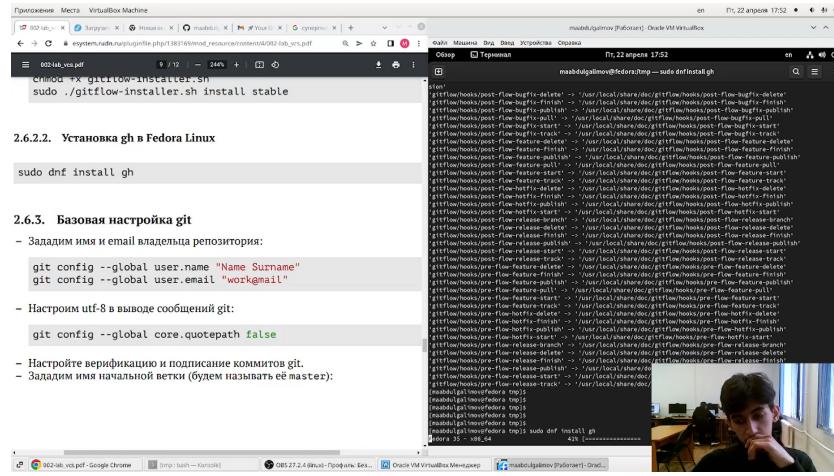


Рис. 4.4: Установка git-flow в Fedora Linux

4.3 3. Провел базовую настройку Git(рис. 4.5):

Команды:

```
git config --global user.name "Name Surname"
git config --global user.email "work@mail"
git config --global core.quotepath false
```

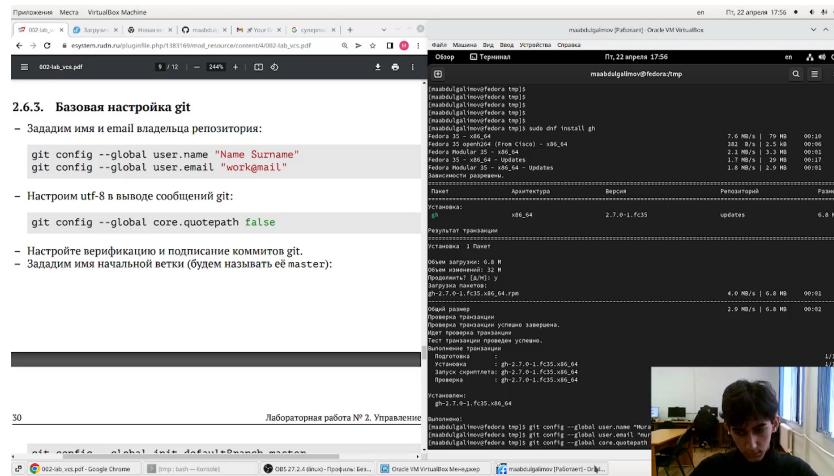


Рис. 4.5: Проведение базовой настройки Git

4.4 4. Создал ключи SSH и PGP(рис. 4.6, 4.7):

Команды:

```
ssh-keygen -t rsa -b 4096
```

```
ssh-keygen -t ed25519
```

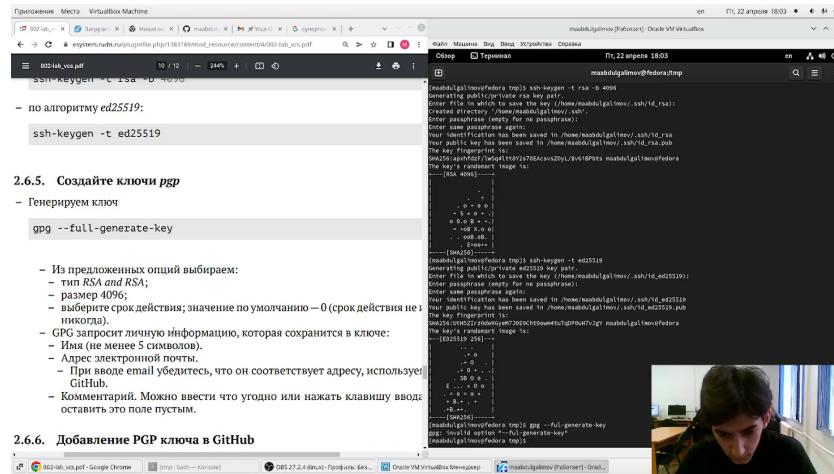


Рис. 4.6: Генерация SSH

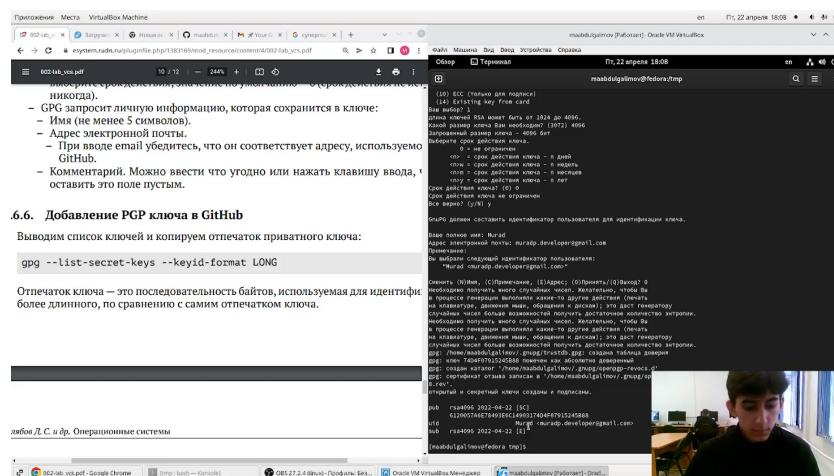


Рис. 4.7: Генерация PGP

4.5 5. Добавление PGP в GitHub(рис. 4.8, 4.9):

Команды:

```

gpg --list-secret-keys --keyid-format LONG
gpg --armor --export <PGP Fingerprint> | xclip

```

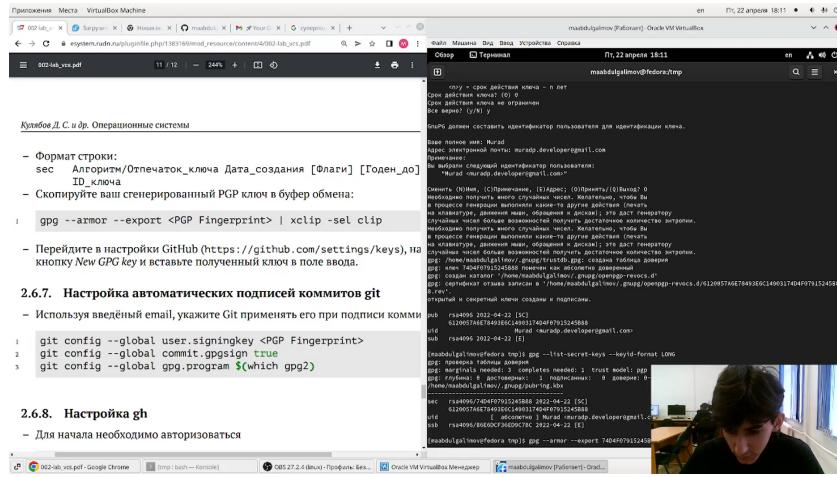


Рис. 4.8: Добавление PGP в GitHub

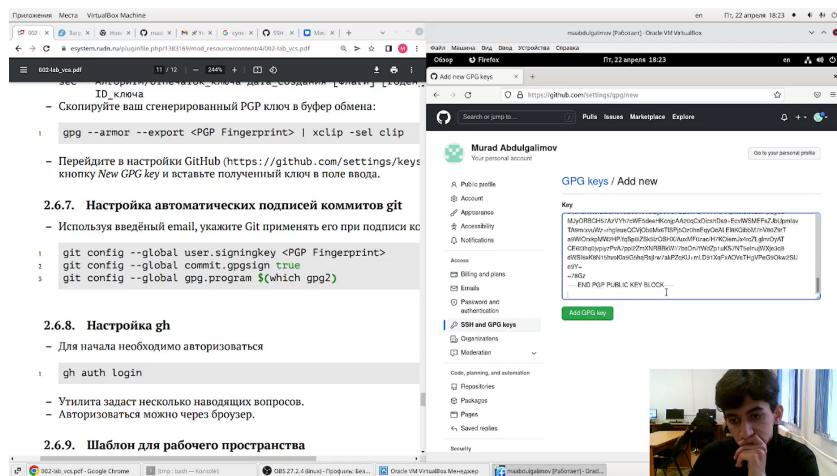


Рис. 4.9: Добавление PGP в GitHub

4.6 6. Настроил автоматические подписи для комитов

Git(рис. 4.10, 4.11):

Команды:

```

git config --global user.signingkey <PGP Fingerprint>
git config --global commit.gpgsign true
git config --global gpg.program $(which gpg2)

```

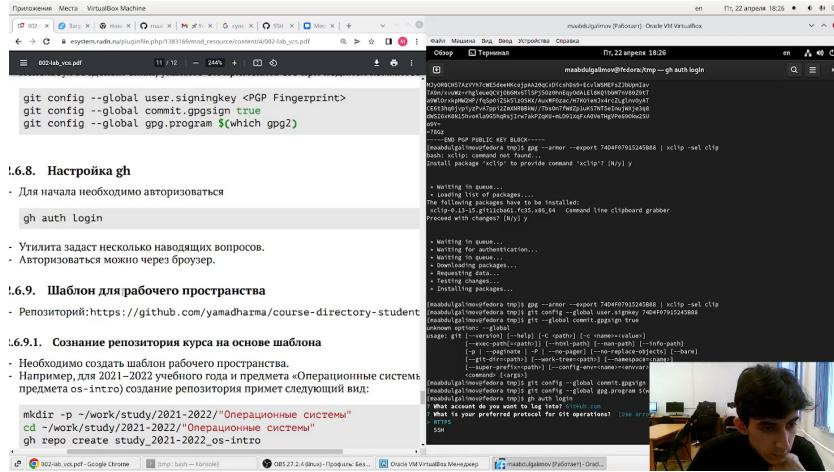


Рис. 4.10: Настроил автоматические подписи для комитов Git

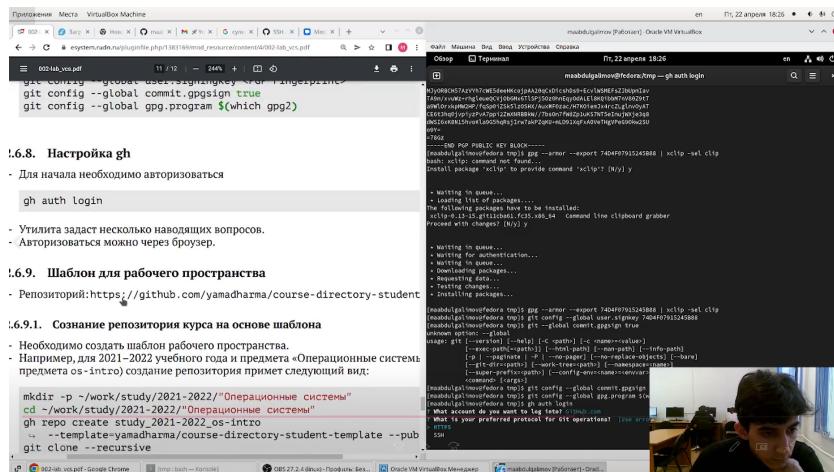


Рис. 4.11: Настроил gh

4.7 7. Создание репозитория курса на основе шаблона(рис. 4.12):

Команды:

```

mkdir -p ~/work/study/2021-2022/"Операционные системы"
cd ~/work/study/2021-2022/"Операционные системы"
gh repo create study_2021-2022_os-intro--template=yamadharma/course-
directory-student-template --public
git clone --recursive git@github.com:<owner>/study_2021-2022_os-
intro.git os-intro

```

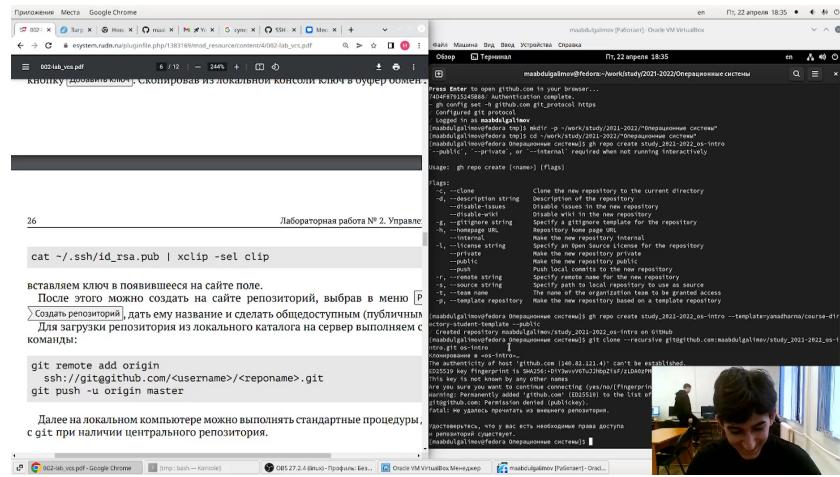


Рис. 4.12: Создание репозитория на основе шаблона

4.8 8. Настроил каталог курса и отправил на сервер(рис.

4.13):

Команды:

```

cd ~/work/study/2021-2022/"Операционные системы"/os-intro
rm package.json
make COURSE=os-intro
git add .
git commit -am 'feat(main): make course structure'
git push

```

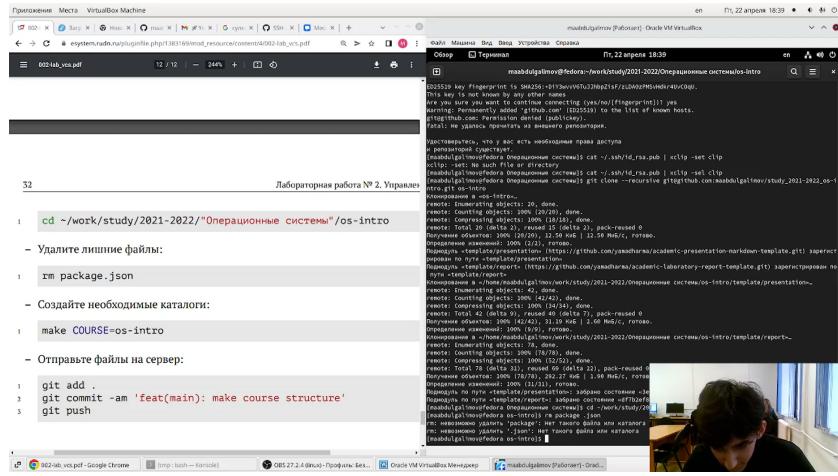


Рис. 4.13: Настройка каталога курса

4.9 Контрольные вопросы:

4.9.1 1. Системы контроля версий – это программные инструменты,

помогающие командам разработчиков управлять изменениями в исходном коде с течением времени. Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом.

4.9.2 2. Объясните следующие понятия VCS и их отношения:

хранилище, commit, история, рабочая копия. Хранилище – репозиторий - место хранения всех версий и служебной информации. Commit - это команда для записи индексированных изменений в репозиторий. История – место, где сохраняются все коммиты, по которым можно посмотреть данные о коммитах. Рабочая копия – текущее состояние файлов проекта, основанное на версии, загруженной из хранилища.

4.9.3 3. Что представляют собой и чем отличаются централизованные

и децентрализованные VCS? Приведите примеры VCS каждого вида Централизованные системы – это системы, в которых одно основное хранилище всего проекта, и каждый пользователь копирует необходимые ему файлы, изменяет и вставляет обратно. Пример – Subversion. Децентрализованные системы – система, в которой каждый пользователь имеет свой вариант репозитория и есть возможность добавлять и забирать изменения из репозиториев. Пример – Git.

4.9.4 4. Опишите действия с VCS при единоличной работе с

хранилищем.

4.9.5 5. Опишите порядок работы с общим хранилищем VCS.

4.9.6 6. Каковы основные задачи, решаемые инструментальным

средством git? У Git две основных задачи: первая – хранить информацию обо всех изменениях в вашем коде, начиная с самой первой строчки, а вторая – обеспечение удобства командной работы над кодом.

4.9.7 7. Назовите и дайте краткую характеристику командам 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 изменения: 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
- удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): git rm имена_файлов – 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 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`

4.9.8 8. Приведите примеры использования при работе с локальным и удалённым репозиториями.

4.9.9 9. Что такое и зачем могут быть нужны ветви (branches)?

Ветка (англ. `branch`) – это последовательность коммитов, в которой ведётся параллельная разработка какого-либо функционала. Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка специально для экспериментов.

4.9.10 10. Как и зачем можно игнорировать некоторые файлы при commit? Игнорируемые файлы – это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты. В Git нет специальной команды для указания игнорируемых файлов: вместо этого

необходимо вручную отредактировать файл . Временно игнорировать изменения в файле можно командой `git update-index assumeunchanged`

5 Выводы

- Изучил идеологию и применение средств контроля версий.
- Освоил умения по работе с git.

Список литературы

1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016.
URL: <https://www.gnu.org/software/bash/manual/>.
2. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.
6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.