

Архитектура компьютеров и операционные системы | Операционные системы

Markdown

Акрур Имад НКАбд-01-24

Содержание

1	Цель работы:	5
2	Задания:	6
3	Теоретическое введение:	7
3.1	Markdown:	7
3.2	Зачем это нужно?	7
4	Выполнение лабораторной работы:	8
4.1	Цель второй лабораторной работы:	8
4.2	Теоретическое введение:	8
4.2.1	Примеры использования git:	8
4.2.2	Выполнение второй лабораторной работы:	8
4.3	Контрольные вопросы:	15
4.4	выводы по результатам выполнения заданий:	19
5	Выводы, согласованные с целью работы:	20

Список иллюстраций

4.1	рисунок 1	9
4.2	рисунок 2	9
4.3	рисунок 3	9
4.4	рисунок 4	10
4.5	рисунок 5	11
4.6	рисунок 6	12
4.7	рисунок 7	12
4.8	рисунок 8	13
4.9	рисунок 9	13
4.10	рисунок 10	13
4.11	рисунок 11	14
4.12	рисунок 12	14
4.13	рисунок 13	15
4.14	рисунок 14	15
4.15	рисунок 15	16
4.16	рисунок 16	17
4.17	рисунок 17	17

Список таблиц

1 Цель работы:

- Научиться оформлять отчёты с помощью легковесного языка разметки **Markdown**.

2 Задания:

- Сделать отчёт по предыдущей лабораторной работе в формате **Markdown**.
- В качестве отчёта нужно предоставить отчёты в **3 форматах: pdf, docx и md** (в **архиве**, поскольку он должен содержать **скриншоты, Makefile** и т.д.)

3 Теоретическое введение:

3.1 Markdown:

- **Markdown** — язык разметки текстов. Такие тексты легко писать и читать. Их можно без труда сконвертировать в HTML. Большинство программистов предпочитают Markdown для написания документации, описаний своих проектов, написания блогов и так далее.

3.2 Зачем это нужно?

1. Для добавления разметки туда, где невозможна реальная разметка. Например, в простом текстовом файле или в тех же СМС, где невозможно выделение жирным, создание заголовков, выделение цитат и пр.
2. Для более удобного написания текстов для последующей конвертации в HTML или другие форматы.

4 Выполнение лабораторной работы:

4.1 Цель второй лабораторной работы:

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

4.2 Теоретическое введение:

4.2.1 Примеры использования git:

- Система контроля версий **Git** представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды **git** с различными опциями.
- Благодаря тому, что **Git** является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией.

4.2.2 Выполнение второй лабораторной работы:

4.2.2.1 Установка программного обеспечения :

4.2.2.1.1 Установка git:

- На этом шаге мы должны были установить **git** через консоль (в нашем случае **git** уже был установлен) (рис. fig. 4.1)


```

imadakrour@vbox:~$ dnf install git
The requested operation requires superuser privileges. Please log in as a user with elevated rights, or use the "--assumeno" or "--
downloadonly" options to run the command without modifying the system state.
imadakrour@vbox:~$ sudo -i
[sudo] password for imadakrour:
root@vbox:~# dnf install git
Updating and loading repositories:
Repositories loaded.
Package "git-2.48.1-1.fc41.x86_64" is already installed.

```

Рис. 4.1: рисунок 1

4.2.2.1.2 Установка gh:

- Затем нам пришлось скачать **gh** (рис. fig. 4.2)

```

root@vbox:~# dnf install gh
Updating and loading repositories:
Repositories loaded.
Package "gh-2.65.0-1.fc41.x86_64" is already installed.

Nothing to do.

```

Рис. 4.2: рисунок 2

4.2.2.2 Базовая настройка git:

- на этом шаге я ничего не делал, потому что мой репозиторий уже был настроен. (рис. fig. 4.3)

```

root@vbox:~# git config --global user.name "Имад Акруп"
root@vbox:~# git config --global user.email "serine.imad@gmail.com"
root@vbox:~# git config --global core.quotepath false
root@vbox:~# git config --global init.defaultBranch master
\
> ^C
root@vbox:~# git config --global init.defaultBranch master
root@vbox:~# git config --global core.autocrlf input
root@vbox:~# ^C
root@vbox:~# git config --global core.safecrlf warn

```

Рис. 4.3: рисунок 3

4.2.2.3 Создать ключи ssh:

- Для **ssh-ключа** было то же самое, поэтому мне пришлось пропустить этот шаг

4.2.2.4 Создать gpg ключ:

- На этом шаге мы должны были сгенерировать ** gpg ключ **, введя следующие команды (рис. fig. 4.4)

```
root@vbox:~# gpg --full-generate-key
gpg (GnuPG) 2.4.5; 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.

gpg: directory '/root/.gnupg' created
Please select what kind of key you want:
  (1) RSA and RSA
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (9) ECC (sign and encrypt) *default*
 (10) ECC (sign only)
 (14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 4096
Requested keysize is 4096 bits
Please specify how long the key should be valid.
    0 = key does not expire
    <n> = key expires in n days
    <n>w = key expires in n weeks
    <n>m = key expires in n months
    <n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.
```

Рис. 4.4: рисунок 4

- затем нам пришлось ввести кодовую фразу для защиты нашего ключа (рис. fig. 4.5)

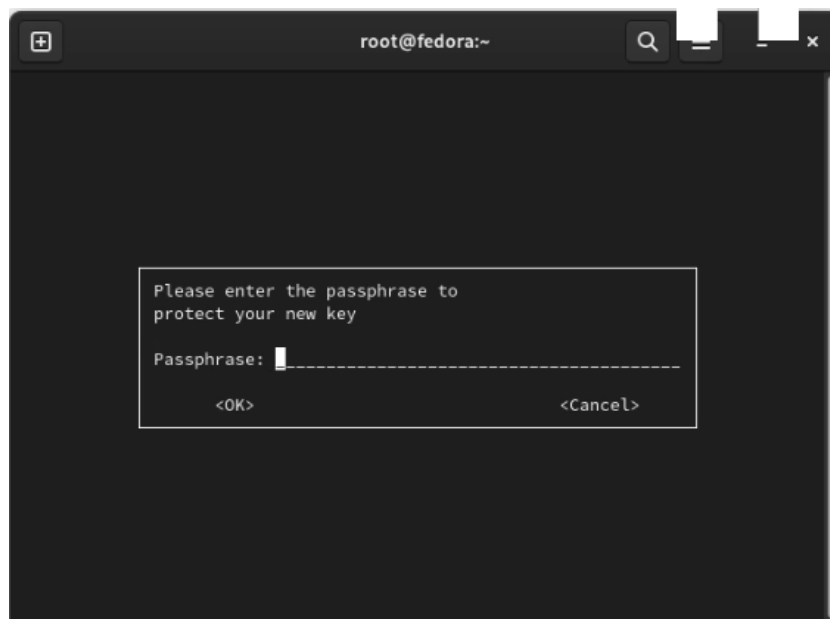


Рис. 4.5: рисунок 5

4.2.2.5 Настройка github:

- Я пропустил этот шаг, потому что я уже создал **учетную запись github** раньше.
- все мои данные были заполнены в учетной записи.(рис. fig. 4.6)

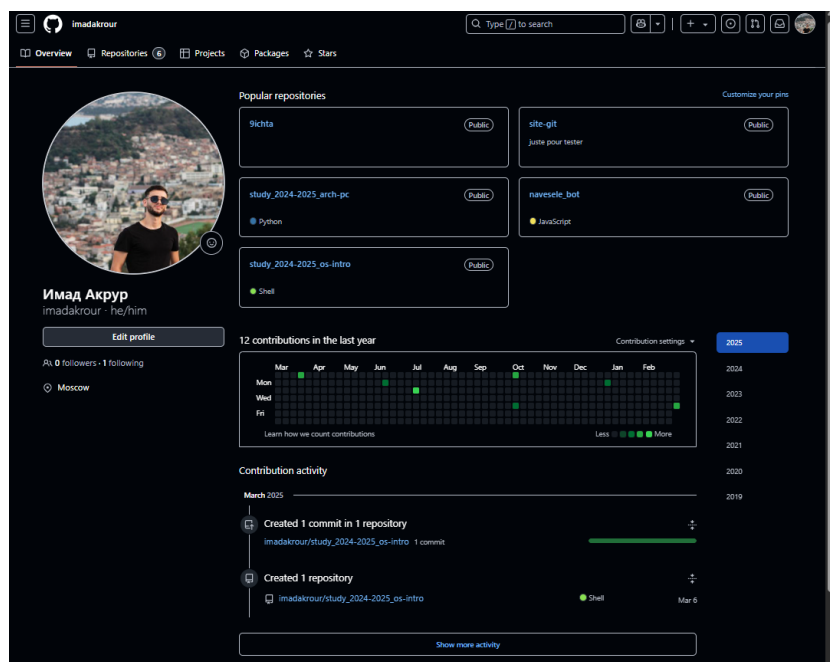


Рис. 4.6: рисунок 6

4.2.2.6 Добавление PGP ключа в GitHub:

- На этом шаге мы хотели добавить **ключ pgr** в нашу учетную запись github, поэтому нам пришлось скопировать отпечаток ключа (рис. fig. 4.7)

```
root@vbox:~# gpg --list-secret-keys --keyid-format LONG
gpg: checking the trustdb
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
[keyboxd]
-----
sec   rsa4096/08AAA695E0B73403 2025-03-07 [SC]
      8426FF125D868F4D369D22B208AAA695E0B73403
uid    [ultimate] imadakrour <serine.imad@gmail.com>
ssb    rsa4096/3177154B121461B7 2025-03-07 [E]
```

Рис. 4.7: рисунок 7

- после этого мы открыли **github** и в **настройках** добавили наш ключ PGP (рис. fig. 4.8)

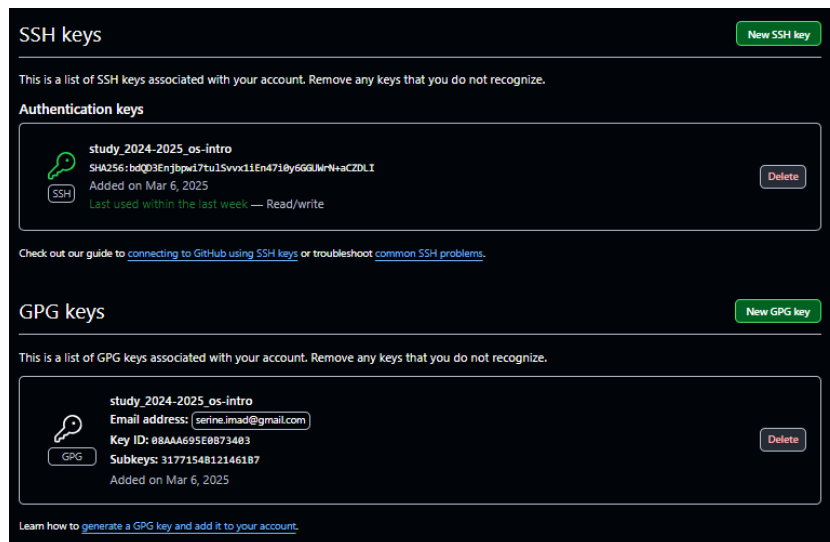


Рис. 4.8: рисунок 8

4.2.2.7 Настройка автоматических подписей коммитов git и gh:

- На этом шаге мы хотели заставить **github** использовать **наш введённый email**, в качестве подписи при отправке коммитов.
- для этого нам пришлось создать токен аутентификации, чтобы иметь доступ к нашей учетной записи git через консоль (рис. fig. 4.9) (рис. fig. 4.10)

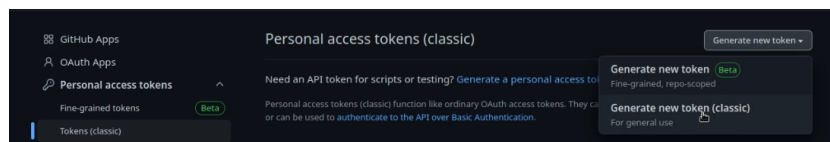


Рис. 4.9: рисунок 9

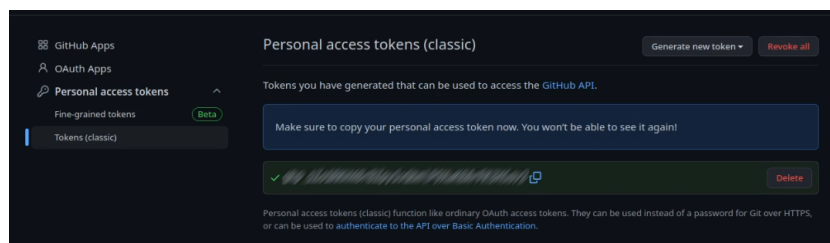


Рис. 4.10: рисунок 10

- и после этого мы вставили токен в консоль, которая дала нам доступ к нашей учетной записи github (рис. fig. 4.11)

```
root@fedora ~]# gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Paste an authentication token
Tip: you can generate a Personal Access Token here https://github.com/settings/tokens
The minimum required scopes are 'repo', 'read:org', 'workflow'.
? Paste your authentication token: *****
```

Рис. 4.11: рисунок 11

4.2.2.8 Сознание репозитория курса на основе шаблона:

- Для меня этот путь `~/work/study/2022-2023/“Операционные системы”` уже был создан, поэтому я сразу ввел вторую команду.
- На этом шаге мне пришлось пройти аутентификацию с помощью **токена**, чтобы получить **доступ**, затем я клонировал глобальный репозиторий в свой локальный (рис. fig. 4.12) (рис. fig. 4.13)

```
root@vbox:~# gh auth login
? Where do you use GitHub? GitHub.com
? What is your preferred protocol for Git operations on this host? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Paste an authentication token
Tip: you can generate a Personal Access Token here https://github.com/settings/tokens
The minimum required scopes are 'repo', 'read:org', 'workflow'.
? Paste your authentication token: *****
- gh config set -h github.com git_protocol https
✓ Configured git protocol
! Authentication credentials saved in plain text
✓ Logged in as imadakrour
```

Рис. 4.12: рисунок 12

```

root@vbox:~/work/study/2024-2025/Операционные системы# git clone --recursive git@github.com:imadkrour/study_2024-2025_os-intro.git
os-intro
Cloning into 'os-intro'...
remote: Enumerating objects: 36, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 36 (delta 1), reused 21 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (36/36), 19.41 KiB | 19.41 MiB/s, done.
Resolving deltas: 100% (1/1), done.
Submodule 'template/presentation' (https://github.com/yamadharma/academic-presentation-markdown-template.git) registered for path 'template/presentation'
Submodule 'template/report' (https://github.com/yamadharma/academic-laboratory-report-template.git) registered for path 'template/report'
Cloning into '/root/work/study/2024-2025/Операционные системы/os-intro/template/presentation'...
remote: Enumerating objects: 111, done.
remote: Counting objects: 100% (111/111), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 111 (delta 42), reused 100 (delta 31), pack-reused 0 (from 0)
Receiving objects: 100% (111/111), 102.17 KiB | 1.11 MiB/s, done.
Resolving deltas: 100% (42/42), done.
Cloning into '/root/work/study/2024-2025/Операционные системы/os-intro/template/report'...
remote: Enumerating objects: 142, done.
remote: Counting objects: 100% (142/142), done.
remote: Compressing objects: 100% (97/97), done.
remote: Total 142 (delta 60), reused 121 (delta 39), pack-reused 0 (from 0)
Receiving objects: 100% (142/142), 341.09 KiB | 2.16 MiB/s, done.
Resolving deltas: 100% (60/60), done.
Submodule path 'template/presentation': checked out 'c9b2712b4b2d431ad5086c9c72a02bd2fca1d4a6'
Submodule path 'template/report': checked out 'c26e22effe7b3e0495707d82ef561ab185f5c748'

```

Рис. 4.13: рисунок 13

4.2.2.9 Настройка каталога курса:

- На этом шаге я попытался удалить файл package.json, но он уже был удален, поэтому я немедленно отправил другие изменения (рис. fig. 4.14)

```

root@vbox:~/work/study/2024-2025/Операционные системы/os-intro# make prepare
root@vbox:~/work/study/2024-2025/Операционные системы/os-intro# ls
CHANGELOG.md  COURSE  LICENSE  prepare  project-personal  README.git-flow.md  template
config        labs   Makefile  presentation  README.en.md      README.md
root@vbox:~/work/study/2024-2025/Операционные системы/os-intro# git add .
root@vbox:~/work/study/2024-2025/Операционные системы/os-intro# git commit -am 'feat(main): make course structure'
[master b022488] feat(main): make course structure
403 files changed, 98412 insertions(+)
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
create mode 100644 labs/lab01/presentation/.projectile
create mode 100644 labs/lab01/presentation/.texlabroot
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/placemg_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
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_secnos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/_init_.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/core.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/main.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/pandocattributes.py

```

Рис. 4.14: рисунок 14

4.3 Контрольные вопросы:

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

2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

- хранилище (repository, сокр. репо), или репозиторий, — место хранения всех версий и служебной информации.
- commit : создание новой версии («сделать коммит», «закоммитить»)
- история журнал : перечень версий можно вернуться к любой.
- Рабочая копия (working copy или working tree) — текущее состояние файлов проекта, основанное на версии из хранилища (обычно на последней).

3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.(рис. fig. 4.15)

Виды систем контроля версий

Централизованные

- Простота использования.
- Вся история — всегда в едином общем хранилище.
- Нужно подключение к сети.
- Резервное копирование нужно только одному хранилищу.
- Удобство разделения прав доступа к хранилищу.
- Почти все изменения навсегда попадают в общее хранилище.

Распределенные

- Двухфазный commit:
 - 1) запись в локальную историю;
 - 2) пересылка изменений другим.
- Подключение к сети не нужно.
- Локальные хранилища могут служить резервными копиями.
- Локальное хранилище контролирует его владелец,
 - но общее — администратор.
- Возможна правка локальной истории перед отправкой на сервер.

Рис. 4.15: рисунок 15

4. Опишите действия с VCS при единоличной работе с хранилищем.(рис. fig. 4.16)

Общее хранилище: централизованная VCS

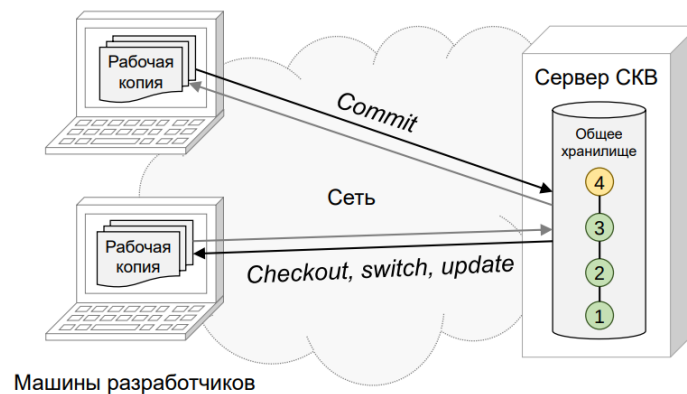


Рис. 4.16: рисунок 16

5. Опишите порядок работы с общим хранилищем VCS.(рис. fig. 4.17)

Отдельные хранилища: распределенная VCS (DVCS)

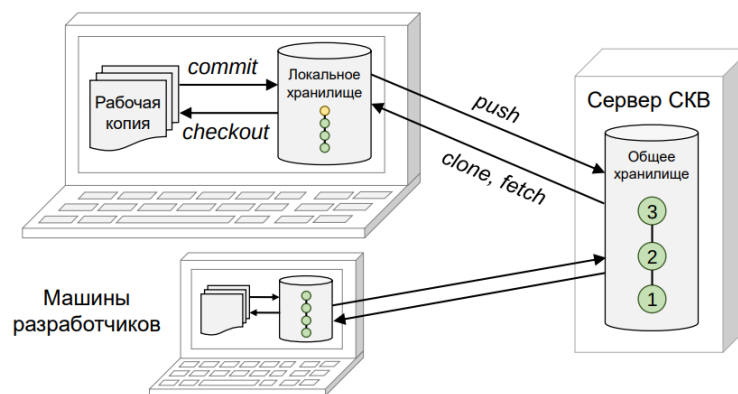


Рис. 4.17: рисунок 17

7. Назовите и дайте краткую характеристику командам git.

- Перечислим 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`

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

- Ветвь (branch) — это линейный участок истории.
- Ветвь по умолчанию называется master. О том, зачем нужны другие ветви и как история может быть нелинейной, см. далее.
- В выводе `git log` отмечен конец ветви.

10. Как и зачем можно игнорировать некоторые файлы при commit?

- Существуют различные типы файлов, которые мы, возможно, захотим, чтобы git проигнорировал перед фиксацией, например, файлы, связанные с нашими пользовательскими настройками или любыми настройками утилиты, личные файлы, такие как пароли и ключи API. Эти файлы никому больше не нужны, и мы не хотим загромождать наш git. Мы можем сделать это с помощью `“.gitignore”`

4.4 выводы по результатам выполнения заданий:

- после выполнения этих упражнений мы смогли применить на практике наши знания, которые мы получили о git и системе контроля версий в целом.

5 Выводы, согласованные с целью работы:

- к концу лабораторной работы этой лабораторной работе мы узнали, как использовать markdown для создания pdf-файлов быстрее и эффективнее.