

Архитектура компьютеров и операционные системы | Операционные системы

Лабораторная работа № 3. Markdown

Акрур Имад НКАбд-06-24

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение :	7
	3.0.1 Контрольные вопросы для самопроверки	7
4	Выполнение лабораторной работы	9
4.1	Цель второй лабораторной работы:	9
4.2	Теоретическое введение:	9
	4.2.1 Контрольные вопросы:	9
	4.2.2 Выполнение лабораторной работы:	11
4.3	1. Написание отчёта в формате Markdown	18
4.4	2. Генерация форматов PDF, DOCX и MD	19
4.5	3. Отправка файлов в репозиторий	20
4.6	Описание результатов выполнения заданий для самостоя- тельной работы	20
	4.6.1 Создание отчета в соответствующем каталоге рабочего пространства :	20
4.7	выводы по результатам выполнения заданий: :	22
5	Выводы	23

Список иллюстраций

4.1 рисунок 1	11
4.2 рисунок 2	12
4.3 рисунок 3	13
4.4 рисунок 4	13
4.5 рисунок 5	14
4.6 рисунок 6	14
4.7 рисунок 7	15
4.8 рисунок 8	15
4.9 рисунок 9	15
4.10рисунок 10	16
4.11рисунок 11	16
4.12рисунок 12	17
4.13рисунок 13	17
4.14рисунок 14	17
4.15рисунок 15	17
4.16рисунок 16	18
4.17рисунок 17	18
4.18рисунок 18	18
4.19рисунок 23	19
4.20рисунок 24	19
4.21рисунок 25	20
4.22рисунок 19	21
4.23рисунок 20	21
4.24рисунок 21	21
4.25рисунок 22	22

Список таблиц

1 Цель работы

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

2 Задание

- 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 :

3.0.1 Контрольные вопросы для самопроверки

1. Что такое Markdown?

Markdown — это легкий язык разметки, который позволяет форматировать текст с помощью простых символов, делая его легко читаемым и редактируемым.

2. Как в Markdown задается начертание шрифтов?

- *Курсив*: обрaмление текста в одиночные звездочки или подчеркивания, например, *курсив* или _курсив_.
- **Жирный**: обрaмление текста в двойные звездочки или подчеркивания, например, **жирный** или __жирный__.
- ***Жирный курсив***: обрaмление текста в три звездочки, например, ***жирный курсив***.

3. Как в Markdown оформляются списки?

- **Ненумерованный список**: используется звездочка, плюс или дефис:
 - Пункт 1
 - Пункт 2
- **Нумерованный список**: используются числа с точками:

1. Первый пункт

2. Второй пункт

4. Как в Markdown оформляются изображения и ссылки на них?

- **Изображение:** `![Alt text](url_изображения)`

- **Ссылка:** `[Текст ссылки](url_ссылки)`

5. Как в Markdown оформляются математические формулы и ссылки на них?

- Для отображения формул используйте знаки доллара:

- Внутри строки: `$E=mc^2$`

- На отдельной строке:

- `$$E=mc^2$$`

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 Контрольные вопросы:

1. Что такое системы контроля версий (VCS) и для чего они предназначены?

Системы контроля версий (VCS) — это инструменты для управления изменениями в коде и файлах проекта, позволяющие отслеживать историю изменений, работать с разными версиями и сотрудничать в команде.

2. Объясните понятия VCS: хранилище, commit, история, рабочая копия

- Хранилище `repository` место, где хранятся все версии проекта.
- `Commit` зафиксированные изменения с комментарием, представляющие версию проекта.
- История последовательность всех `commit`
- Рабочая копия `working` локальная версия проекта, с которой работает разработчик.

3. Чем отличаются централизованные и децентрализованные VCS?

Примеры.

- Централизованные VCS (например, SVN): одно центральное хранилище, доступное для всех.
- Децентрализованные VCS (например, Git): каждый пользователь имеет своё полное хранилище, с возможностью синхронизации.

4. Действия с VCS при единоличной работе с хранилищем.

- Инициализация репозитория.
- Добавление файлов и создание commit
- Проверка состояния (git status), просмотр истории (git log).

5. Порядок работы с общим хранилищем VCS.

- Клонирование репозитория.
- Создание ветки, внесение изменений.
- Commit изменений.
- Слияние с основной веткой через pull request.

6. Основные задачи, решаемые Git.

Управление версиями, создание веток, слияние изменений, разрешение конфликтов, работа с удалёнными репозиториями.

7. Команды Git: краткая характеристика.

- 'git init': создание репозитория.
- 'git clone': клонирование удалённого репозитория.
- 'git add': добавление файлов для отслеживания.
- 'git commit': фиксация изменений.
- 'git push': отправка изменений в удалённый репозиторий.
- 'git pull': получение изменений из удалённого репозитория.

8. Примеры работы с локальными и удалёнными репозиториями.

- Локальные : инициализация репозитория, создание commit , проверка состояния.
- Удалённые: клонирование, отправка изменений (git push), получение

ние обновлений (git pull).

4.2.2 Выполнение лабораторной работы:

4.2.2.1 Создание учетной записи на GitHub

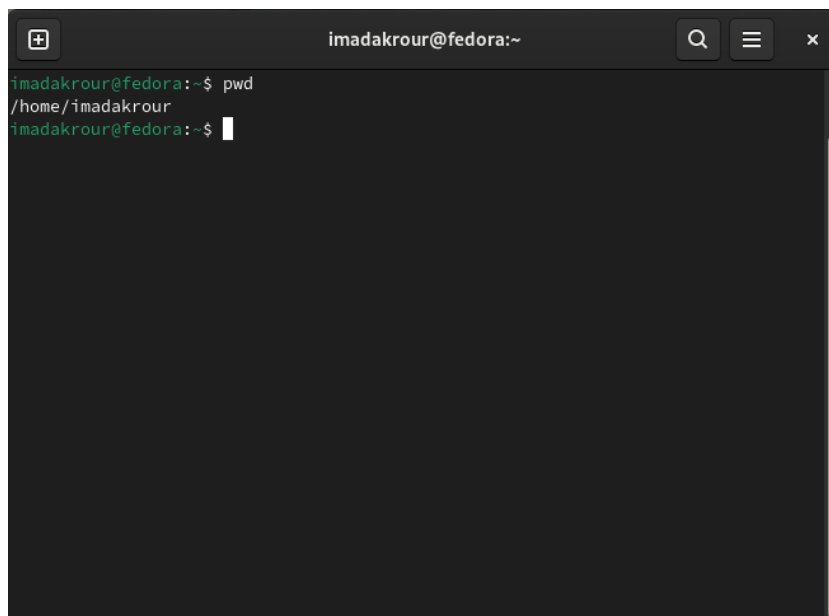


Рис. 4.1: рисунок 1

в моем случае учетная запись уже существует, поэтому этот шаг был пропущен.

4.2.2.2 Базовая настройка Git:

1. **Указание имени пользователя и email для создания коммитов в репозиториях. Эти данные будут добавляться к каждому коммиту, чтобы идентифицировать автора изменений.:**

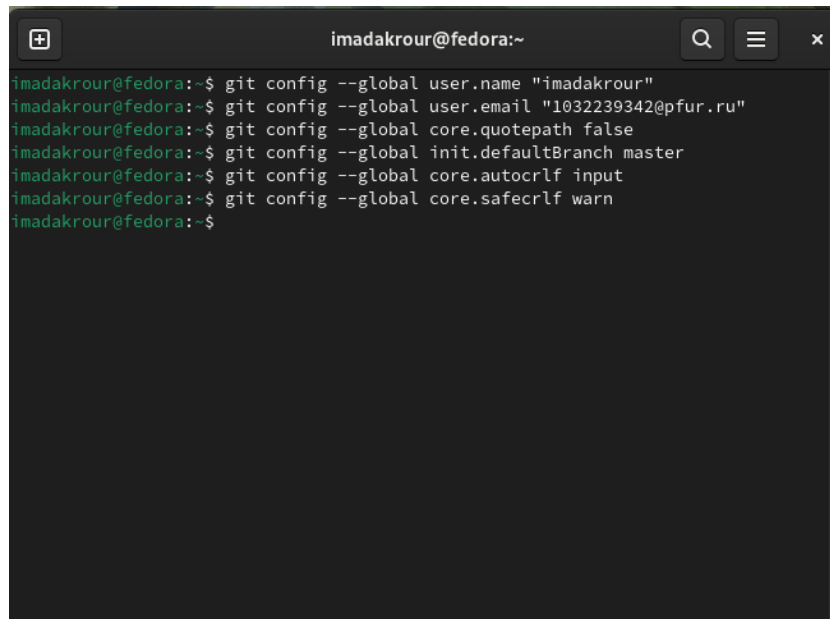
A terminal window titled 'imadakrour@fedora:~' with search, menu, and close buttons. It contains a sequence of six 'git config' commands to set global Git settings. The commands are: 'git config --global user.name "imadakrour"', 'git config --global user.email "1032239342@pfur.ru"', 'git config --global core.quotepath false', 'git config --global init.defaultBranch master', 'git config --global core.autocrlf input', and 'git config --global core.safecrlf warn'. Each command is followed by a prompt 'imadakrour@fedora:~\$'.

Рис. 4.2: рисунок 2

Эта последовательность команд настраивает имя пользователя, email, кодировку UTF-8, задает начальную ветку master, устанавливает обработку концов строк (autocrlf), и включает предупреждения о несоответствиях концов строк (safecrlf). Выполнение всех команд в одной строке позволяет быстрее закончить настройку и сразу перейти к работе с Git. Все параметры будут применены глобально для всех будущих репозиториев.

4.2.2.3 Создание SSH-ключа

Описание выполняемого задания:

Для безопасного подключения к репозиториям на GitHub, нужно сгенерировать SSH-ключи (публичный и приватный). Это позволит вам работать с репозиториями, не вводя каждый раз логин и пароль.

1. Генерация SSH-ключа:

```
imadakrour@fedora:~  
imadakrour@fedora:~$ ssh-keygen -C "Akrour Imad 1032239342@pfur.ru"  
Generating public/private ed25519 key pair.  
Enter file in which to save the key (/home/imadakrour/.ssh/id_ed25519):  
/home/imadakrour/.ssh/id_ed25519 already exists.  
Overwrite (y/n)? y  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/imadakrour/.ssh/id_ed25519  
Your public key has been saved in /home/imadakrour/.ssh/id_ed25519.pub  
The key fingerprint is:  
SHA256:Smt2I3J4Wlmx6Rb2KnzqxqJf6DqBS4+qPkbRmgDacWxU Akrour Imad 1032239342@pfur.ru  
The key's randomart image is:  
+--[ED25519 256]--+  
|      E          |  
|      .          |  
|      .          |  
|o . . . +       |  
|o+o . . S       |  
|+o+o + B.o      |  
|.Bo o.% ==.     |  
|ooo Oo==o.      |  
|oo... o0+       |  
+-----[SHA256]-----+  
imadakrour@fedora:~$
```

Рис. 4.3: рисунок 3

Каталог ~/.ssh/ — это стандартное место для хранения SSH-ключей. Не изменяйте путь, если вы не хотите использовать другое место для хранения.

2. Копирование публичного ключа в буфер обмена:

```
imadakrour@fedora:~  
imadakrour@fedora:~$ cat ~/.ssh/id_ed25519.pub | xclip -sel clip  
imadakrour@fedora:~$
```

Рис. 4.4: рисунок 4

Команда `cat ~/.ssh/id_rsa.pub` выводит содержимое публичного ключа, а команда `xclip -sel clip` копирует это содержимое в буфер обмена

Add new SSH Key

Title
SSH for Архитектура лабы

Key type
Authentication Key

Key
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAISafS3NkKc0DwZqNnoGTlnScQE9LOOdYTW+rmJjM Akrou: Imad 1032239342@pfur.ru

Add SSH key

Рис. 4.5: рисунок 5

Имад Акруп (imadakrou)
Your personal account

Add new SSH

Title

Key type
Authentication Key

Key
begins with 'ssh-rsa', or 'sk-ssh-ed25519@openssh.com'

Add SSH key

Copy Ctrl+C
Paste Ctrl+V
Paste as plain text Ctrl+Shift+V
Select all Ctrl+A
Open in sidebar
Writing direction
Block element...
Inspect

Рис. 4.6: рисунок 6

вставить ключ на сайт *GitHub*.

4.2.2.4 Создание рабочего пространства и репозитория курса на основе шаблона

Для правильной организации рабочих файлов и проектов в рамках курса необходимо создать рабочее пространство по определенной структуре. В этом шаге будет выполнено создание директории для предмета «Архитектура компьютера», а также будет продемонстрировано, как структурировать папки для лабораторных работ.

Описание выполняемого задания:

Создание структуры рабочего пространства :

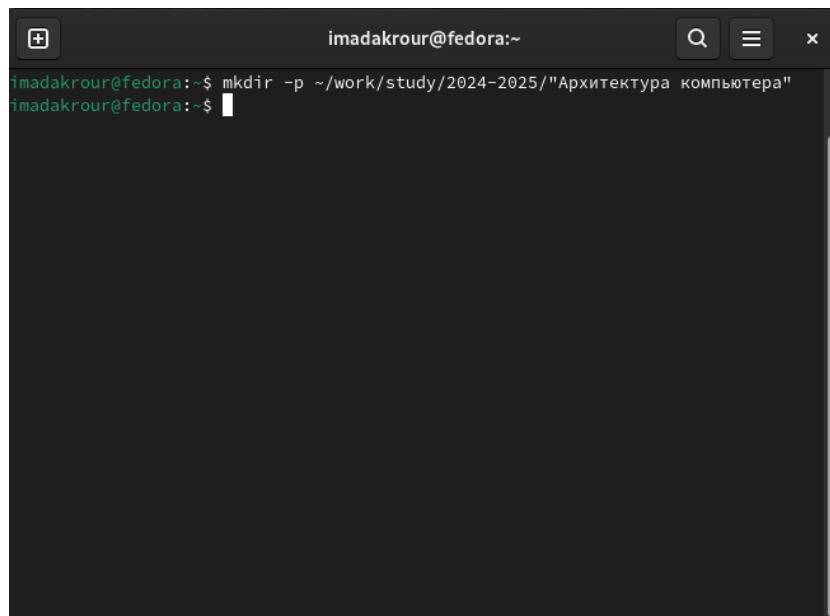


Рис. 4.7: рисунок 7

Эта команда создаст нужные каталоги по иерархии. Опция -p создает промежуточные каталоги, если они еще не существуют

4.2.2.5 Создание репозитория курса на основе шаблона



Рис. 4.8: рисунок 8

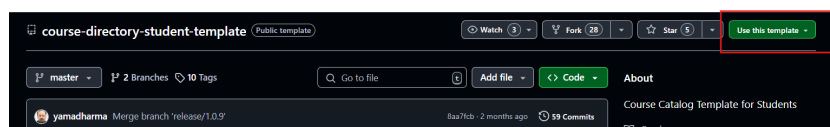


Рис. 4.9: рисунок 9

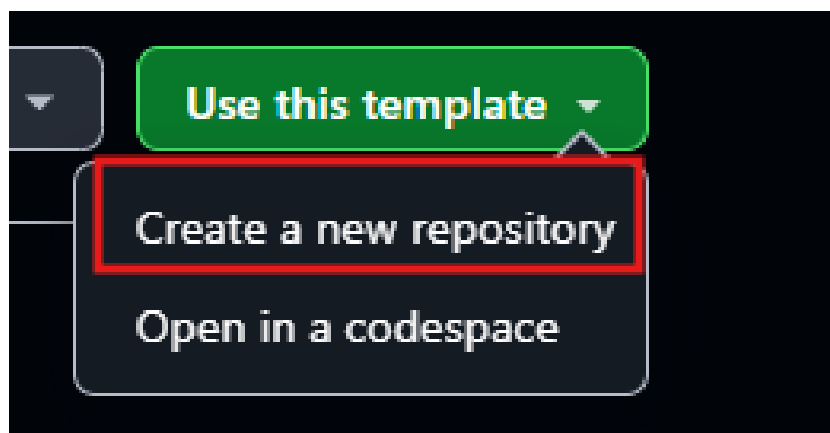


Рис. 4.10: рисунок 10

Repository template

yamadharm/course-directory-student-template

Start your repository with a template repository's contents.

☐ Include all branches
Copy all branches from yamadharm/course-directory-student-template and not just the default branch.

Owner * / Repository name *

imadakrour / study_2024-2025_arch-pc

✔ study_2024-2025_arch-pc is available.

Great repository names are short and memorable. Need inspiration? How about [jubilant-enigma](#) ?

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.

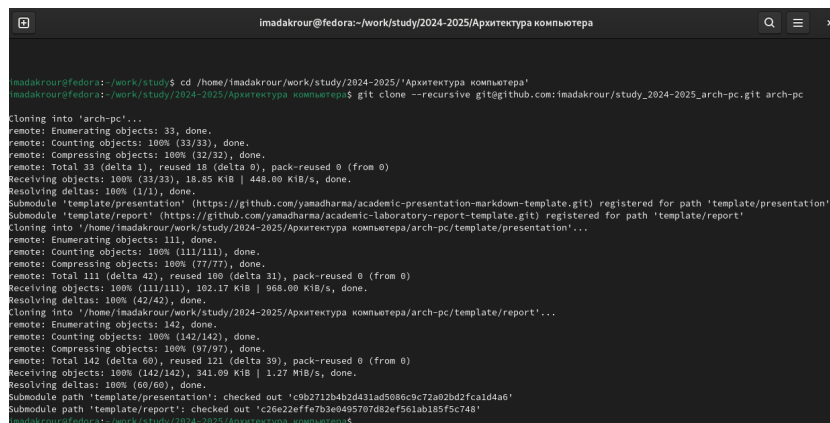
ⓘ You are creating a public repository in your personal account.

Create repository

Рис. 4.11: рисунок 11

Эти действия создадут новый репозиторий на основе предоставленного шаблона, который содержит структуру и необходимые файлы для работы по курсу.

Клонирование репозитория на локальный компьютер :



```
imadakraur@fedora: ~/work/study/2024-2025/Архитектура компьютера
imadakraur@fedora: ~/work/study$ cd /home/imadakraur/work/study/2024-2025/Архитектура компьютера
imadakraur@fedora: ~/work/study/2024-2025/Архитектура компьютера$ git clone --recursive git@github.com:imadakraur/study_2024-2025_arch-pc.git arch-pc
Cloning into 'arch-pc'...
remote: Enumerating objects: 33, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (32/32), done.
remote: Total 33 (delta 1), reused 18 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (33/33), 18.85 KiB | 448.00 KiB/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 '/home/imadakraur/work/study/2024-2025/Архитектура компьютера/arch-pc/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 | 968.00 KiB/s, done.
Resolving deltas: 100% (42/42), done.
Cloning into '/home/imadakraur/work/study/2024-2025/Архитектура компьютера/arch-pc/template/report'...
remote: Enumerating objects: 142, done.
remote: Counting objects: 100% (142/142), done.
remote: Compressing objects: 100% (91/91), done.
remote: Total 142 (delta 60), reused 121 (delta 39), pack-reused 0 (from 0)
Receiving objects: 100% (142/142), 341.09 KiB | 1.27 MiB/s, done.
Resolving deltas: 100% (60/60), done.
Submodule path 'template/presentation': checked out 'c9b2712b4b2d431ad5086c9c72a02bd2fca1d4a6'
Submodule path 'template/report': checked out 'c26e22effe7b3e0495707d82ef561ab185f5c748'
imadakraur@fedora: ~/work/study/2024-2025/Архитектура компьютера$
```

Рис. 4.12: рисунок 12

Команда *git clone --recursive* позволяет загрузить все файлы из удаленного репозитория в папку *arch-pc*.

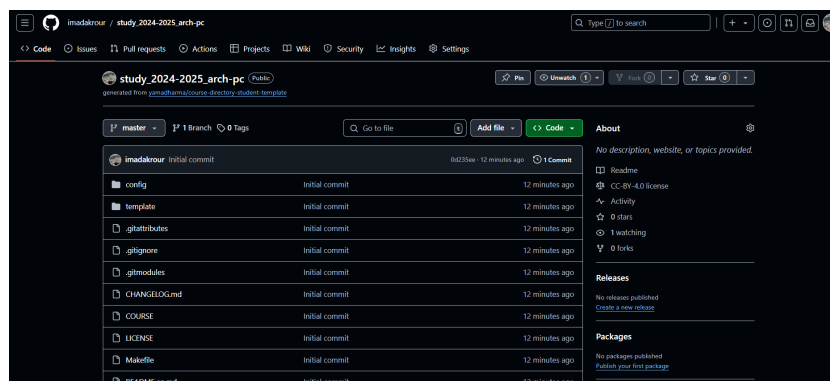
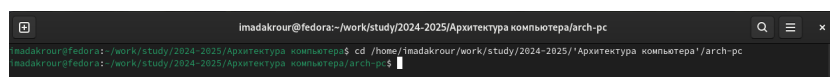


Рис. 4.13: рисунок 13

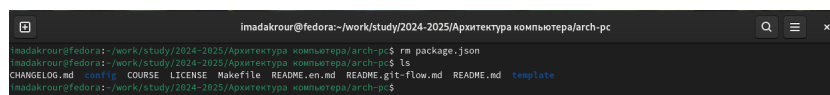
Настройка каталога курса :



```
imadakraur@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc
imadakraur@fedora: ~/work/study/2024-2025/Архитектура компьютера$ cd /home/imadakraur/work/study/2024-2025/Архитектура компьютера/arch-pc
imadakraur@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc$
```

Рис. 4.14: рисунок 14

Удаление ненужного файла `package.json` :

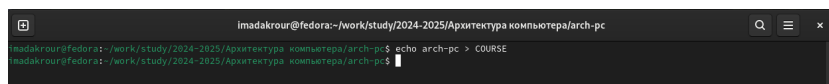


```
imadakraur@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc
imadakraur@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc$ rm package.json
imadakraur@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc$ ls
CHANGELOG.md  config  COURSE  LICENSE  Makefile  README.en.md  README.git-Flow.md  README.md  tempListe
imadakraur@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc$
```

Рис. 4.15: рисунок 15

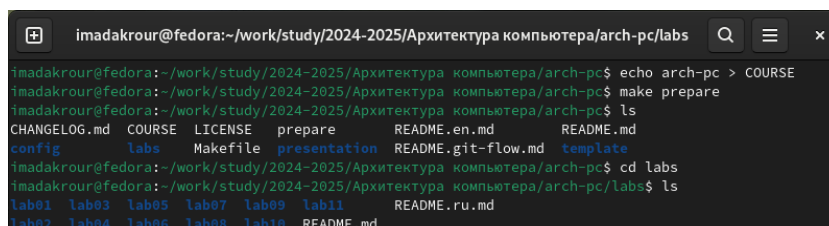
Удаление лишних файлов помогает избежать путаницы и оставить только необходимые для курса файлы.

Создание файла COURSE с названием курса:



```
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ echo arch-pc > COURSE
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$
```

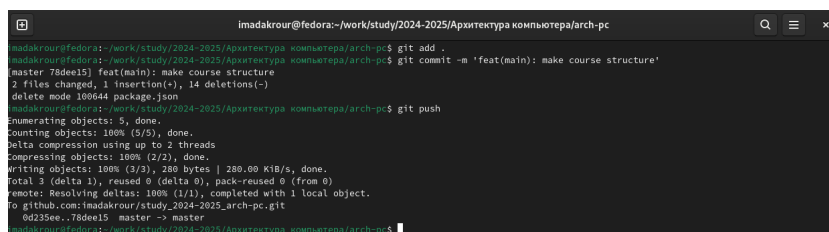
Рис. 4.16: рисунок 16



```
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ echo arch-pc > COURSE
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ make prepare
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ ls
CHANGELOG.md  COURSE  LICENSE  prepare  README.en.md  README.md
config  labs  Makefile  presentation  README.git-flow.md  template
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ cd labs
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs$ ls
lab01  lab03  lab05  lab07  lab09  lab11  README.ru.md
lab02  lab04  lab06  lab08  lab10  README.md
```

Рис. 4.17: рисунок 17

Отправка изменений на сервер :



```
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git add .
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git commit -m 'feat(main): make course structure'
[master 78dee15] feat(main): make course structure
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 280 bytes | 280.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:imadakrour/study_2024-2025_arch-pc.git
0d235ee..78dee15 master -> master
imadakrour@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$
```

Рис. 4.18: рисунок 18

Эти команды добавляют изменения в локальный репозиторий, создают коммит с описанием и отправляют изменения в удаленный репозиторий на *GitHub*.

4.3 1. Написание отчёта в формате Markdown

Сначала я создал файл под названием `heroit.md` в текстовом редакторе. Этот файл содержит весь материал для лабораторной работы №2, оформленный в формате Markdown

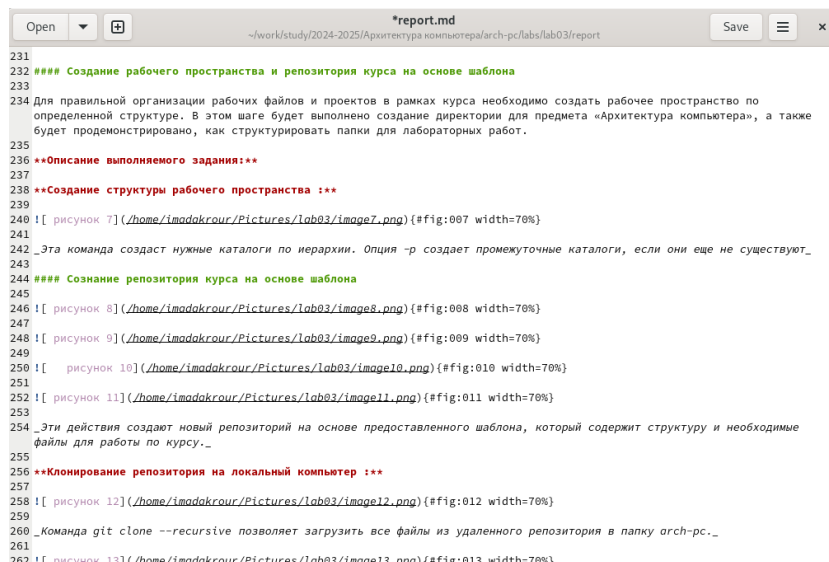


Рис. 4.19: рисунок 23

4.4 2. Генерация форматов PDF, DOCX и MD

Когда файл Markdown был готов, я с помощью команды `make report` сгенерировал отчёт в трёх форматах: Markdown (.md), PDF (.pdf) и Word (.docx).

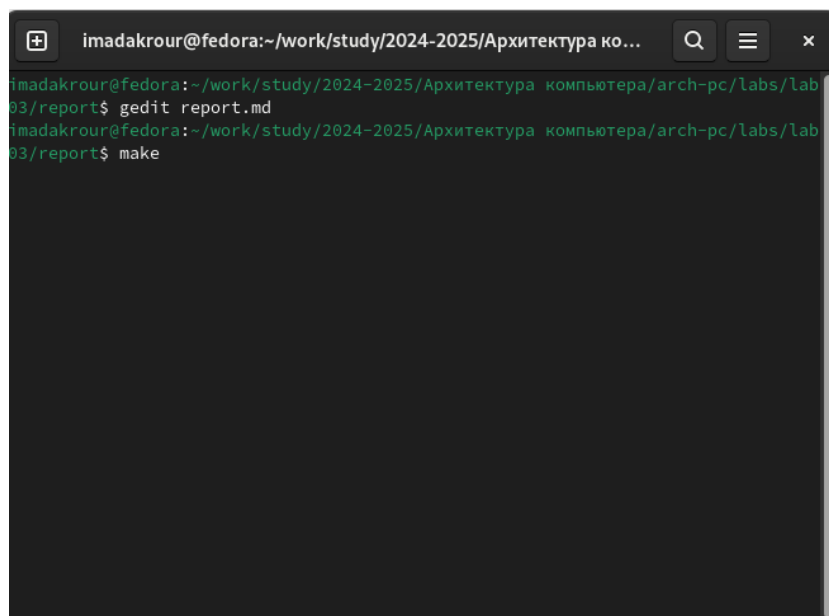


Рис. 4.20: рисунок 24

4.5 3. Отправка файлов в репозиторий

После создания файлов я зафиксировал изменения и отправил их в свой Git-репозиторий.

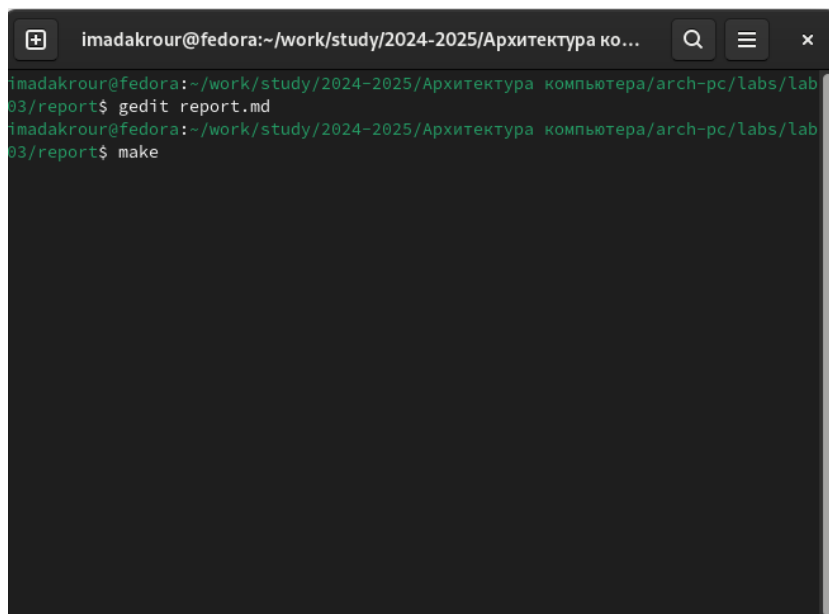


Рис. 4.21: рисунок 25

4.6 Описание результатов выполнения заданий для самостоятельной работы

4.6.1 Создание отчета в соответствующем каталоге рабочего пространства :

Описание задания:

Данное задание включает в себя создание отчета о выполнении лабораторной работы, копирование предыдущих отчетов и загрузку файлов на GitHub.

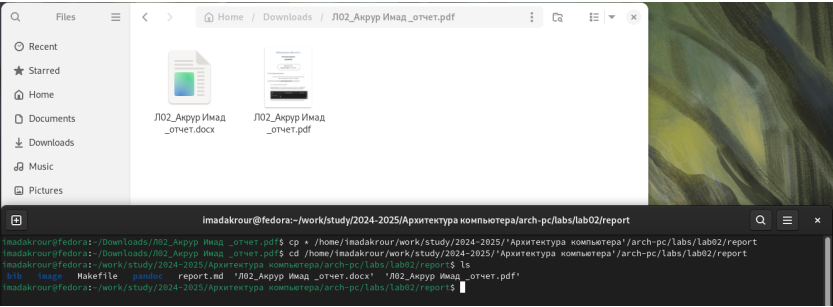


Рис. 4.22: рисунок 19

Я скопировал файл отчета из lab01 в lab01/report, а файл отчета из lab02 в lab02/report, используя команду cp. Сначала я перешел в каталог с файлами отчета с помощью команды cd, а затем выполнил команду для копирования всех файлов

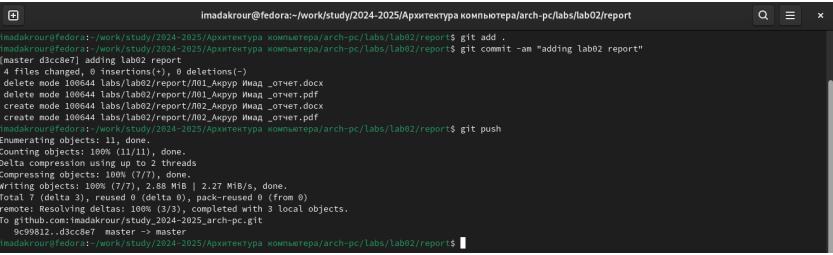


Рис. 4.23: рисунок 20

После копирования файлов я выполнил команды для отправки их на **GitHub**



Рис. 4.24: рисунок 21

Name	Last commit message	Last commit date
..		
bib	feat(main): make course structure	10 hours ago
image	feat(main): make course structure	10 hours ago
pandoc	feat(main): make course structure	10 hours ago
Makefile	feat(main): make course structure	10 hours ago
report.md	feat(main): make course structure	10 hours ago
Л01_Акрур Имам_отчет.docx	Add files via upload	10 hours ago
Л01_Акрур Имам_отчет.pdf	Add files via upload	10 hours ago

Рис. 4.25: рисунок 22

Затем я сделал скриншот в GitHub, чтобы показать, что файлы были обновлены и теперь отображаются в репозито

4.7 выводы по результатам выполнения заданий: :

Выполнение задания помогло закрепить навыки работы с системами контроля версий, организацией файлов в рабочем пространстве, а также загрузкой проектов на GitHub. Эти навыки важны для поддержания эффективной командной работы, обеспечения безопасности данных и удобства доступа к прошлым версиям проектов.

5 Выводы

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