**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

# ОТЧЕТ
# ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

*дисциплина: Архитектура компьютера*

Студент: Кесаев Алан Таймуразович

Группа: НКАбд-02-24

**МОСКВА**

2024г.

# Содержание

# Список иллюстраций

# 1. Цель работы

Целью работы является применение средств контроля версий. А также очень важно приобрести практические навыки по работе с системой git.

## 2. Задание

1. Настройка GitHub.
2. Базовая настройка Git.
3. Создание SSH-ключа.
4. Создание рабочего пространства и репозитория курса на основе шаблона.
5. Создание репозитория курса на основе шаблона.
6. Настройка каталога курса.
7. Выполнение заданий для самостоятельной работы.

# 3. Теоретическое введение

Система контроля версий ( Version Control System, VCS ) — это инструмент, используемый разработчиками программного обеспечения для управления изменениями в исходном коде и других файловых ресурсах.

Системы контроля версий разработаны специально для того, чтобы максимально упростить и упорядочить работу над проектом (вне зависимости от того, сколько человек в этом участвуют). СКВ дает возможность видеть, кто, когда и какие изменения вносил; позволяет формировать новые ветви проекта, объединять уже имеющиеся; настраивать контроль доступа к проекту ; осуществлять откат до предыдущих версий.

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 дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом. Демидова А. В. 14 Архитектура ЭВМ Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности.

Например, они могут поддерживать работу с несколькими 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. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным. Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

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.

Ниже показан перечень основных команд git (рис.3.1)

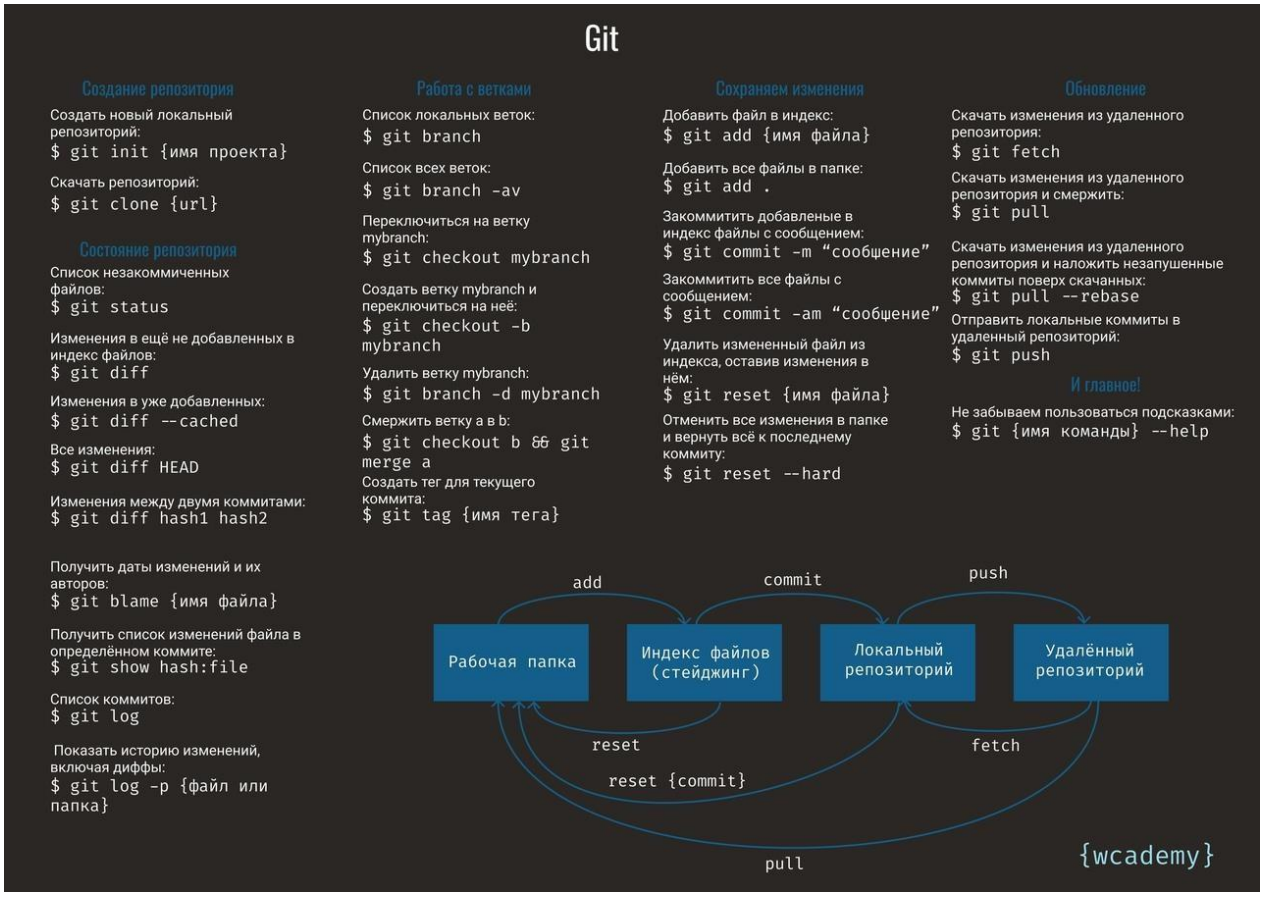



*Рис.3.1 . Перечень основных команд git.*

# 4. Выполнение лабораторной работы

## 4.1. Настройка GitHub.

Для выполнения лабораторной работы создаю учетную запись на https://github.com/ (рис. 4.1.1)

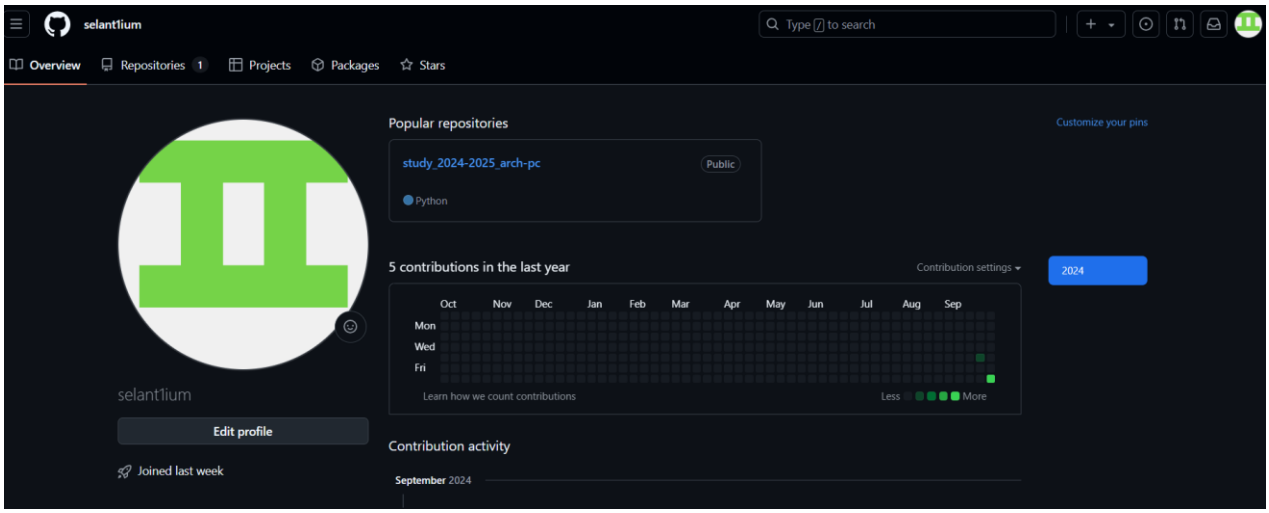



*рис.4.1.1. Учетная запись на GitHub.*

## 4.2. Базовая настройка git.

Делаю предварительную конфигурацию git. Захожу в терминал и ввожу команды, указывая свое имя и email (рис.4.2.1)

```
alan@alan-VirtualBox:~$ git config --global user.name "<Alan Kesaev>"
alan@alan-VirtualBox:~$ git config --global user.email "<1132242975@pfur.ru>"
```

*рис.4.2.1. Предварительная конфигурация в git.*

Настраиваю utf-8 в выходе сообщений git. (рис.4.2.2)

```
alan@alan-VirtualBox:~$ git config --global core.quotepath false
```

*рис.4.2.2. Настраивание utf-8.*

Задаю имя начальной ветки, которую буду называть master (рис. 4.2.3)

```
alan@alan-VirtualBox:~$ git config --global init.defaultBranch master
```

*рис.4.2.3. Имя начальной ветки*

А также ввожу autocrlf и safecrlf (рис. 4.2.4)

```
alan@alan-VirtualBox:~$ git config --global core.autocrlf input
alan@alan-VirtualBox:~$ git config --global core.safecrlf warn
```

*рис.4.2.4. Ввод команд autocrlf и safecrlt.*

## 4.3. Создание SSH ключа

Для последующей идентификации пользователя на сервере репозиториев генерирую пару ключей (приватный и открытый). (рис. 4.3.1)

```
alan@alan-VirtualBox:~$ ssh-keygen -C "Alan Kesaev <1132242975@pfur.ru>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/alan/.ssh/id_rsa):
/home/alan/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/alan/.ssh/id_rsa
Your public key has been saved in /home/alan/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:/rq9PELuiSGA04XY6xo8J0T8YwCRpW2a4VgCKQ8eajY Alan Kesaev <1132242975@pfur.ru>
The key's randomart image is:
+---[RSA 3072]----+
|+=.              |
|O* .             |
|O*B .            |
|=E+o             |
|Oo=+    S        |
|oo...  ..        |
|.+... .o.        |
| o+  . oo=.      |
|.     ..=+=o     |
+----[SHA256]-----+
```

*рис.4.3.1. Генерация ключей.*

Чтобы скопировать из локальной консоли ключ в буфер обмена, устанавливаю команду xclip (рис. 4.3.2).

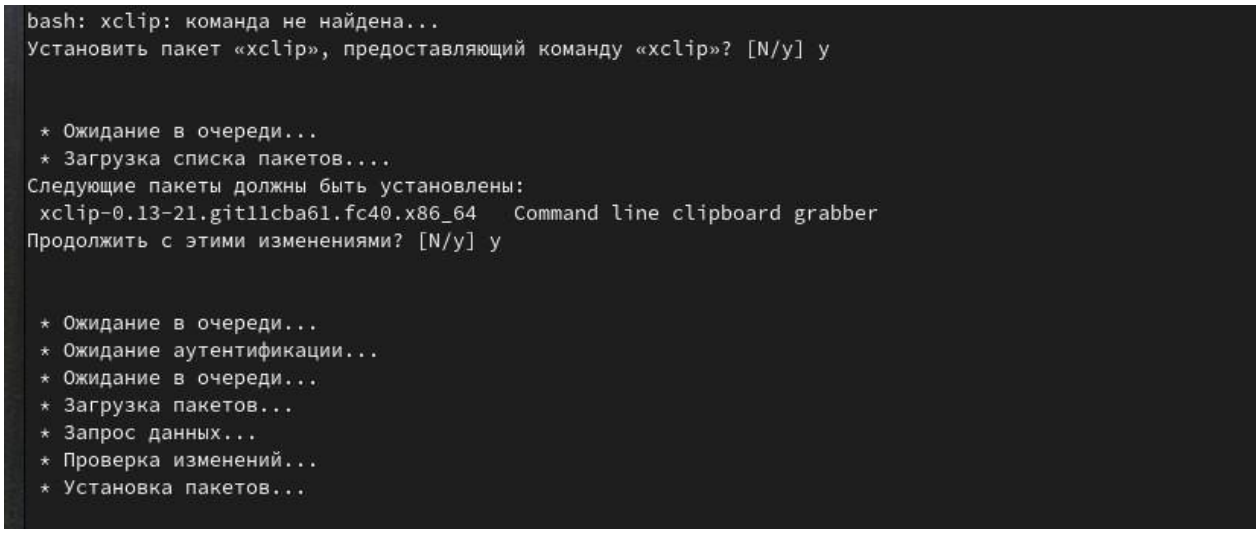



*рис.4.3.2. Скачивание команды xclip.*

Теперь воспользуюсь командой xclip (рис. 4.3.3).


*рис.4.3.3. Копирование ключа.*

Вставляю ключ в появившееся на сайте поле, указываю его имя. (рис. 4.3.4)

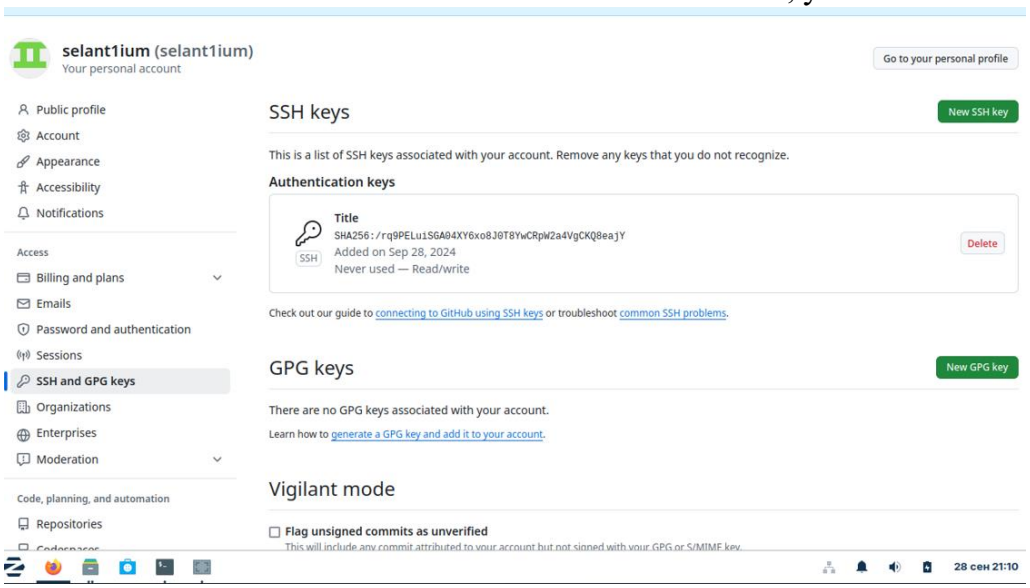



*рис.4.3.4. Вставка ключа.*

## 4.4. Создание рабочего пространства, репозитория курса на основе шаблона.

Открываю терминал и создаю репозиторий для предмета «Архитектура компьютера». (рис. 4.4.1.)

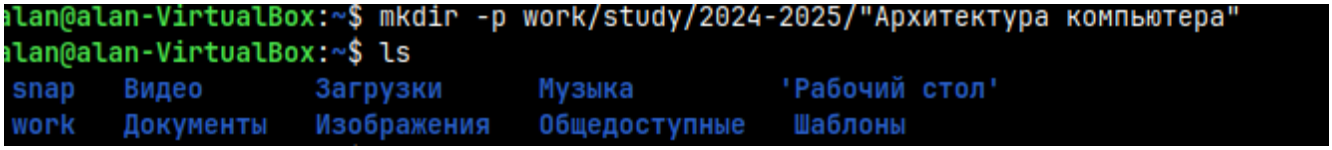



*рис.4.4.1. Создание терминала для предмета «Архитектура компьютера».*

## 4.5. Создание репозитория курса на основе шаблона.

Захожу на страницу репозитория с шаблоном курса, выбираю его в качестве своего нового. (рис. 4.5.1.)

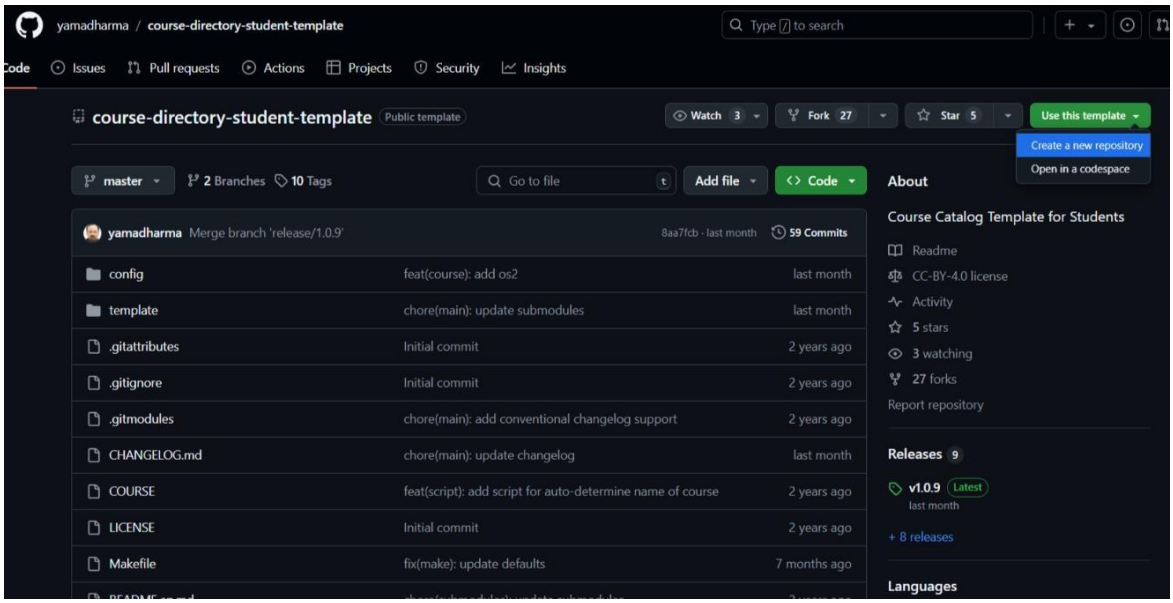



*рис.4.5.1. Выбор шаблона*

Далее создаю его, задав ему имя. (рис. 4.5.2.)

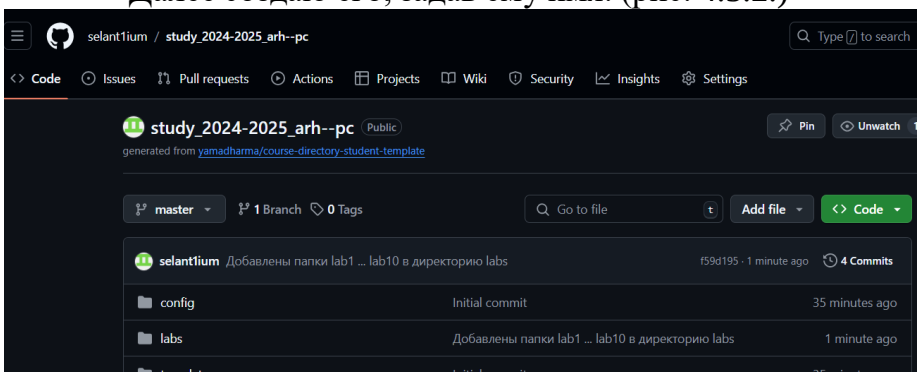



*рис.4.5.2. Создание репозитория.*

Открываю терминал и перехожу в каталог курса. (рис. 4.5.3).


*рис.4.5.2. Переход в каталог курса.*

Клонирую созданный репозиторий (рис. 4.5.4.)

```
alan@alan-VirtualBox:~/work/study/2024-2025/Архитектура компьютера$ git clone --recursive git@github.com:selant1ium/study_2024-2025_arch-pc.git arch-pc
Клонирование в «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)
Получение объектов: 100% (33/33), 18.81 КиБ | 9.41 МиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.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 «/home/alan/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)
Получение объектов: 100% (111/111), 102.17 КиБ | 1.06 МиБ/с, готово.
Определение изменений: 100% (42/42), готово.
Клонирование в «/home/alan/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% (97/97), done.
remote: Total 142 (delta 60), reused 121 (delta 39), pack-reused 0 (from 0)
Получение объектов: 100% (142/142), 341.09 КиБ | 267.00 КиБ/с, готово.
Определение изменений: 100% (60/60), готово.
Submodule path 'template/presentation': checked out 'c9b2712b4b2d431ad5086c9c72a02bd2fca1d4a6'
Submodule path 'template/report': checked out 'c26e22effe7b3e0495707d82ef561ab185f5c748'
```

*рис.4.5.4. Клонирование репозитория.*

## 4.6. Настройка каталога курса.

Перехожу в каталог курса и удаляю лишний файлы. (рис. 4.5.5.)

```
alan@alan-VirtualBox:~/work/study/2024-2025/Архитектура компьютера$ cd ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc
```

*рис.4.5.5. Удаление лишних файлов в каталоге курса.*

```
alan@alan-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ rm package.json
```

Создаю необходимые каталоги, отправляю файлы на сервер. (рис. 4.5.6.)

```
alan@alan-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ echo arch-pc > COURSE
alan@alan-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ make prepare
alan@alan-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git add .
alan@alan-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git commit -am "Feat(main): make course structure"
master 8adfc8d] Feat(main): make course structure
221 files changed, 53680 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/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
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
create mode 100644 labs/lab01/report/report.md
create mode 100644 labs/lab02/presentation/.projectile
create mode 100644 labs/lab02/presentation/.texlabroot
create mode 100644 labs/lab02/presentation/Makefile
create mode 100644 labs/lab02/presentation/image/kulyabov.jpg
create mode 100644 labs/lab02/presentation/presentation.md
create mode 100644 labs/lab02/report/Makefile
create mode 100644 labs/lab02/report/bib/cite.bib
```

*рис.4.5.6. Создание каталогов и их отправка на сервер.*

В локальном репозитории проверяю результат выполненной работы (рис.4.5.7)

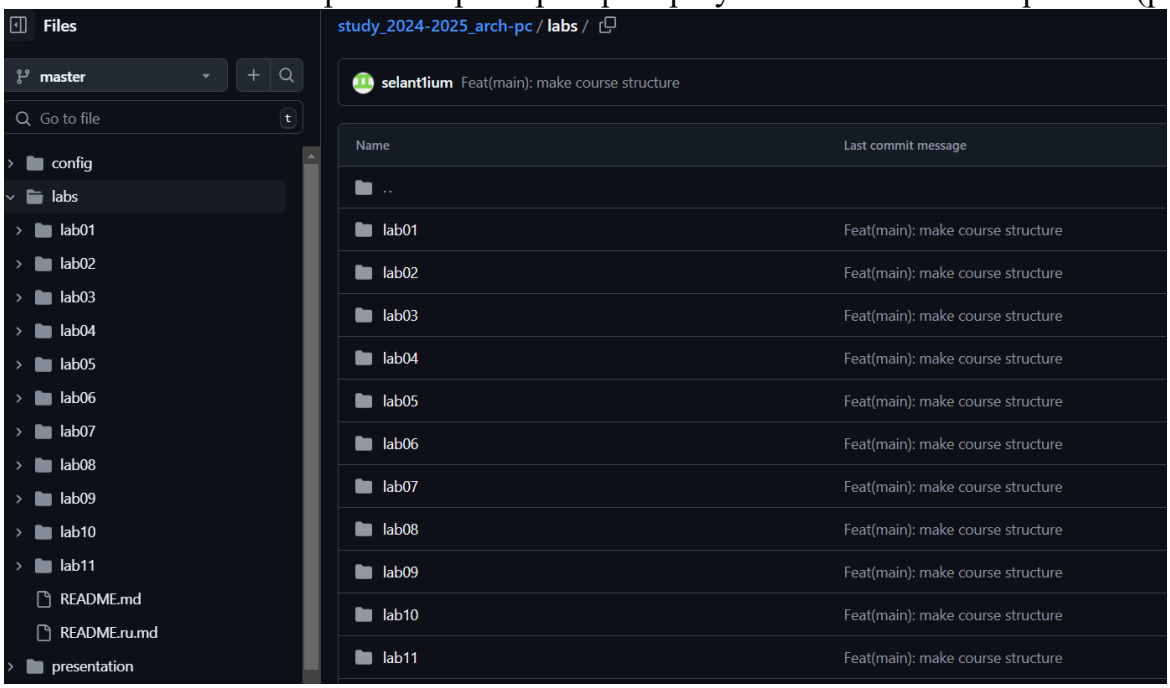



*рис.4.5.7. Проверка выполненной работы.*

## 4.7. Выполнение заданий для самостоятельной работы.

Создаю отчет по выполнению второй лабораторной работы в соответствующем каталоге. (рис. 4.7.1) С помощью команды ls проверяю, создан ли файл.

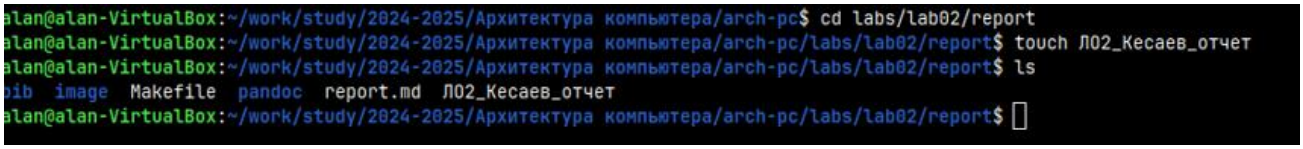
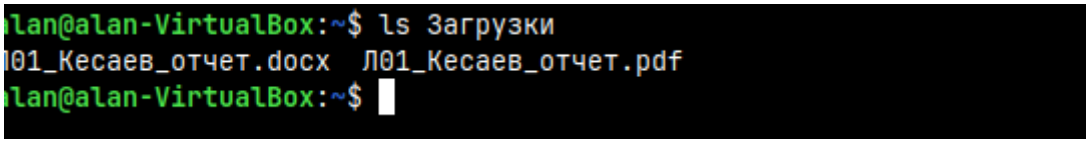



*рис.4.7.1. Создание отчета о выполнении работы.*

Для выполнения второго задания проверяю местонахождение своих лабораторных работ. (рис. 4.7.2)

*рис.4.7.2. Местонахождение лабораторных работ.*




Копирую лабораторную работу с помощью утилиты cp. (рис.4.7.3)

ls

```
alan@alan-VirtualBox:~$ cp ~/Загрузки/Л01_Кесаев_отчет.docx /home/alan/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab01/report
alan@alan-VirtualBox:~$ cp ~/Загрузки/Л01_Кесаев_отчет.pdf /home/alan/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab01/report
alan@alan-VirtualBox:~$
```

*рис.4.7.3. Копирование отчета по лабораторной работе в нужный каталог.*

Для того чтобы загрузить эти файлы на GitHub, в первую очередь я использую команду git add. Так добавленные мной файы станут отслеживаемыми. (рис.4.7.4)

```
alan@alan-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab01/report$ git add Л01_Кесаев_отчет.pdf
alan@alan-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab01/report$ git add Л01_Кесаев_отчет.docx
```

*рис.4.7.4. Добавление файлов с помощью команды git add*

Теперь осуществляю полноценный перенос файлов с помощью команды git commit

(рис. 4.7.5.)

```
alan@alan-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab01/report$ git commit -m "Add lab01/report"
[master 04c7795] Add lab01/report
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 labs/lab01/report/Л01_Кесаев_отчет.docx
 create mode 100644 labs/lab01/report/Л01_Кесаев_отчет.pdf
```

*рис.4.7.5. Поручаю консоли совершить изменения.*

Использую команды: git status и git push, чтобы опубликовать свои локальные коммиты. (рис. 4.7.6.)

```
lan@alan-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab01/report$ git status
екущая ветка: master
аша ветка опережает «origin/master» на 1 коммит.
  (используйте «git push», чтобы опубликовать ваши локальные коммиты)

еотслеживаемые файлы:
  (используйте «git add <файл>...», чтобы добавить в то, что будет включено в коммит)
        ../../lab02/report/Л02_Кесаев_отчет

ндекс пуст, но есть неотслеживаемые файлы
используйте «git add», чтобы проиндексировать их)
lan@alan-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab01/report$ git push
еречисление объектов: 11, готово.
одсчет объектов: 100% (11/11), готово.
ри сжатии изменений используется до 4 потоков
жатие объектов: 100% (7/7), готово.
апись объектов: 100% (7/7), 4.76 МиБ | 3.43 МиБ/с, готово.
сего 7 (изменений 2), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
emote: Resolving deltas: 100% (2/2), completed with 2 local objects.
o github.com:selant1ium/study_2024-2025_arch-pc.git
   8adfc8d..04c7795  master -> master
lan@alan-VirtualBox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab01/report$
```

*рис.4.7.6. Команды git status и git push для завершения копирования.*

Перехожу в каталоги на GitHub, чтобы убедиться в том, что файлы находятся в нужных репозиториях. (рис. 4.7.7)

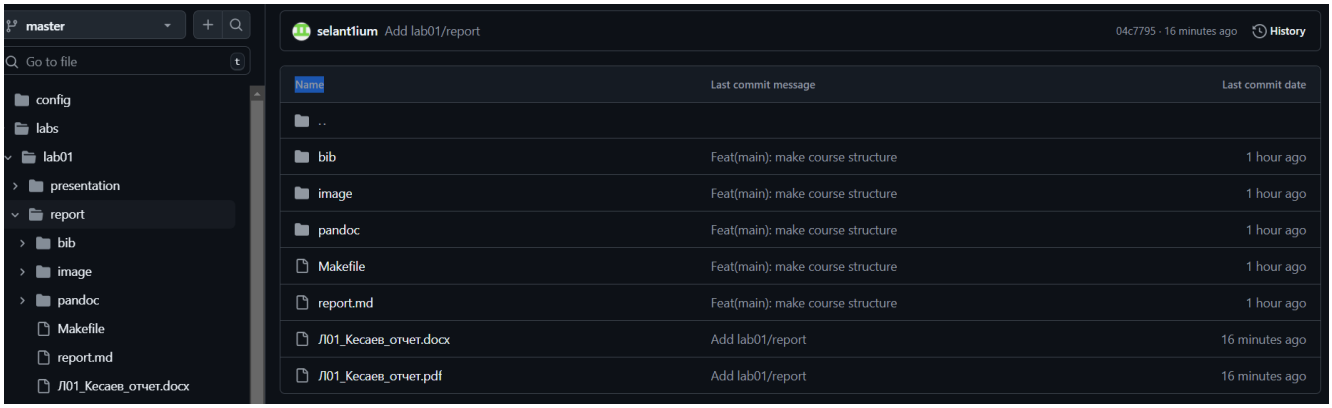



*рис.4.7.7. Проверка проделанных операций.*

# 5. Заключение

В заключение хочется отметить, что данная лабораторная работа позволила мне научиться работать с системой Git. Я практиковал свои навыки в работе с командной строкой, теперь уже связывая выполнимое с директориями GitHub.

# 6. Список используемой литературы


1. Архитектура ЭВМ https://esystem.rudn.ru/pluginfile.php/2089082/mod_resource/content/0/Лабораторная%20работа%20№2.%20Система%20контроля%20версий%20Git.pdf
2. 30 команд Git, необходимых для освоения интерфейса командной строки Git / Хабр https://habr.com/ru/companies/ruvds/articles/599929/
3. Система контроля версий: определение, функции, популярные решения https://gb.ru/blog/sistema-kontrolya-versij/