

# 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: Шабакowa Карина Баировна

с\б:1132242469

Группа:НКАбд-05-24

МОСКВА

2024г.

# **Содержание**

## **1 Цель работы**

## **2 Задание**

## **3 Теоретические введение**

## **4 Выполнение лабораторной работы**

4.1 Настройка GitHub

4.2 Базовая настройка Git

4.3 Создание SSH-ключа

4.4 Создание рабочего пространства и репозитория курса на основе шаблона

4.5 Создание репозитория курса на основе шаблона

4.6 Настройка каталога курса

4.7 Выполнение заданий для самостоятельной работы

## **5 Выводы**

## **6 Список литературы**

## **1 Цель работы**

Целью работы является изучить идеологию и применение средств контроля версий.

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**

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.

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.

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.

## 4 Выполнение лабораторной работы

### 4.1 Настройка GitHub

Создаю учетную запись на сайте GitHub и заполнила основные данные.

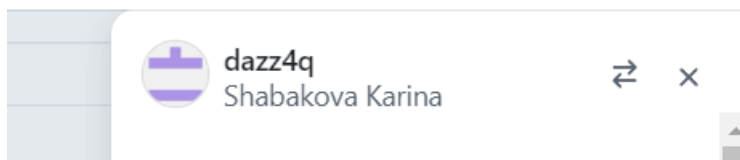


рис 1 Аккаунт GitHub

### 4.2 Базовая настройка Git

Открываю виртуальную машину, затем открываю терминал и делаю

предварительную конфигурацию git. Ввожу команду `git config -global user.name ""`, указывая свое имя и команду `git config -global user.email "work@mail"`, указывая в ней электронную почту владельца, то есть мою.

```
liveuser@localhost-live:~$ git config --global user.name "<Karina Shabakova>"
liveuser@localhost-live:~$
```

```
liveuser@localhost-live:~$ git config --global user.name "<Karina Shabakova>"
liveuser@localhost-live:~$ git config --global user.email "<1132242469@pfur.ru>"
liveuser@localhost-live:~$
```

*Рис. 2: Предварительная конфигурация git*

Настраиваю utf-8 в выводе сообщений git для корректного отображения символов.

```
liveuser@localhost-live:~$ git config --global core.quotePath false
liveuser@localhost-live:~$
```

*Рис. 3: Настройка кодировки*

Задаю имя «master» для начальной ветки

```
liveuser@localhost-live:~$ git config --global init.defaultBranch master
liveuser@localhost-live:~$
```

*Рис. 4: Создание имени для начальной ветки*

Задаю параметр `autocrlf` со значением `input`, так как я работаю в системе Linux, чтобы конвертировать CRLF в LF только при коммитах. CR и LF – это символы, которые можно использовать для обозначения разрыва строки в текстовых файлах.

```
liveuser@localhost-live:~$ git config --global init.defaultBranch mas
liveuser@localhost-live:~$ git config --global core.autocrlf input
liveuser@localhost-live:~$
```

*Рис. 5: Параметр autocrlf*

Задаю параметр `safecrlf` со значением `warn`, так Git будет проверять преобразование на обратимость. При значении `warn` Git только выведет предупреждение, но будет принимать необратимые конвертации.

```
liveuser@localhost-live:~$ git config --global core.safecrlf warn
liveuser@localhost-live:~$
```

*Рис. 6: Параметр safecrlf*

## 4.3 Создание SSH-ключа

Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый). Для этого ввожу команду `ssh-keygen -C "Имя Фамилия, work@email"`, указывая имя владельца и электронную почту владельца. Ключ автоматически сохранится в каталоге `~/.ssh/`.

```
liveuser@localhost-live:~$ ssh-keygen -C "<Karina Shabakova <karinashabakova@gmail.com>"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/liveuser/.ssh/id_ed25519):
Created directory '/home/liveuser/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/liveuser/.ssh/id_ed25519
Your public key has been saved in /home/liveuser/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:+wBHR6Fxxp5uNA4cVoDZmTdsJRpodfUf8krxIlGKfr4 <Karina Shabakova <karinashabakova@gmail.com>
The key's randomart image is:
+--[ED25519 256]--+
|      ++B++o      |
|      = 0+%. .    |
|      . +.% * ..   |
|      . + B = o    |
|      So 0 + o.    |
|      ..= B o      |
|      .o + .       |
|      .. .         |
|      ..E          |
+-----[SHA256]-----+
```

Рис. 7: Генерация SSH-ключа

Через команду cat выводит ключ, копирую

```
liveuser@localhost-live:~$ cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGibUX9zKNThavHa1Hiu046JJf1Ysl1QE3KhJE9u0DY
Karina Shabakova <karinashabakova@gmail.com>
```

Рис. 8: Копирование содержимого файла

Открываю браузер, захожу на сайт GitHub. Открываю свой профиль и выбираю страницу «SSH and GPG keys». Нажимаю кнопку «New SSH key»

Вставляю скопированный ключ в поле «Key». В поле Title указываю имя для ключа. Нажимаю «Add SSH-key», чтобы завершить добавление ключа

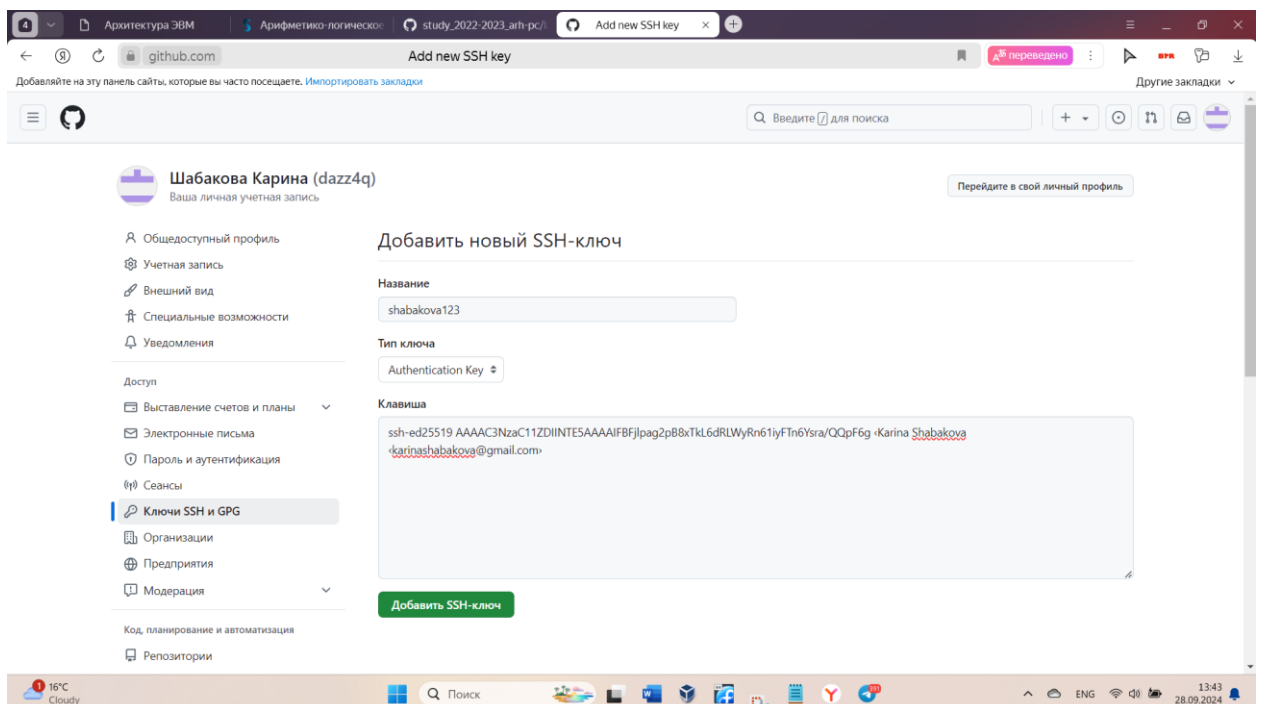


Рис. 9: Добавление ключа

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

Закрываю браузер, открываю терминал. Создаю директорию, рабочее пространство, с помощью утилиты `mkdir`, благодаря ключу `-p` создаю все директории после домашней `~/work/study/2022-2023/“Архитектура компьютера”` рекурсивно. Далее проверяю с помощью `ls`, действительно ли были созданы необходимые мне каталоги

```
livehost-live:~$ mkdir -p work/study/2023-2024/"Architecture Computer"/arch-pc/labs/lab1/lab2/lab3/
liveuser@localhost-live:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos  work
liveuser@localhost-live:~$
```

Рис. 10: Создание рабочего пространства

## 4.5. Создание репозитория курса на основе шаблона

В браузере перехожу на страницу репозитория с шаблоном курса по адресу <https://github.com/yamadharm/course-directory-student-template>. Далее выбираю «Use this template», чтобы использовать этот шаблон для своего репозитория

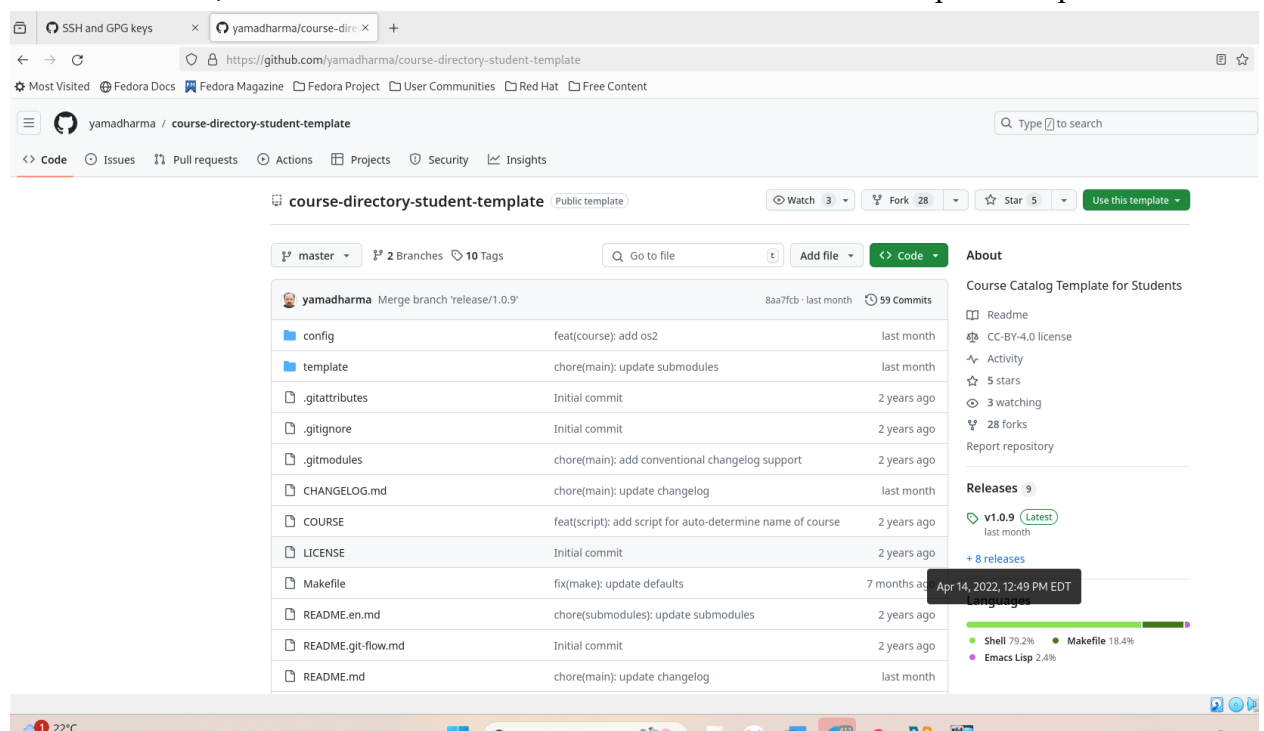


Рис. 11: Страница шаблона для репозитория

В открывшемся окне задаю имя репозитория (Repository name): study\_2022–2023\_arh-pc и создаю репозиторий, нажимаю на кнопку «Create repository from template»


### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

---

*Required fields are marked with an asterisk (\*).*

**Repository template**


 yamadharma/course-directory-student-template ▾

Start your repository with a template repository's contents.

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

---

**Owner \***

 daz4q ▾

**Repository name \***

study\_2023-2024\_arh-pc

✔ study\_2023-2024\_arh-pc is available.

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

**Description (optional)**

Рис. 12: Окно создания репозитория

Репозиторий создан











 daz4q Initial commit	f7ed835 · now	🕒 1 Commit
 config	Initial commit	now
 template	Initial commit	now
 .gitattributes	Initial commit	now
 .gitignore	Initial commit	now
 .gitmodules	Initial commit	now
 CHANGELOG.md	Initial commit	now
 COURSE	Initial commit	now
 LICENSE	Initial commit	now
 Makefile	Initial commit	now

Рис. 13: Созданный репозиторий

Через терминал перехожу в созданный каталог курса с помощью утилиты cd

```
liveuser@localhost-live:~$ cd ~/work/study/2023-2024/"Architecture Computer"
liveuser@localhost-live:~/work/study/2023-2024/Architecture Computer$
```


 COURSE Initial commit 2 minutes ago

Рис. 14: Перемещение между директориями

Клонирую созданный репозиторий с помощью команды `git clone --recursive git@github.com:dazz4q/study_2023-2024_arh-pc.git arch-pc`

```
liveuser@localhost-live:~/work/study/2023-2024/Architecture Computer$ git clone --recursive git@github.com:dazz4q/study_2023-2024_arh-pc.git arch-pc
Cloning into 'arch-pc'...
The authenticity of host 'github.com (140.82.121.4)' can't be established.
```

Рис. 15: Клонирование репозитория

Копирую ссылку для клонирования на странице созданного репозитория, сначала перейдя в окно «code», далее выбрав в окне вкладку «SSH»

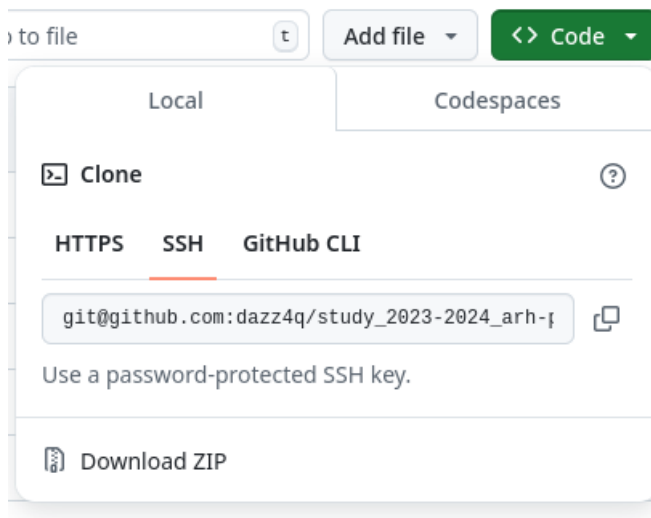


Рис. 16: Окно с ссылкой для копирования репозитория

## 4.6. Настройка каталога курса

Перехожу в каталог `arch-pc` с помощью утилиты `cd`

```
liveuser@localhost-live:~/work/study/2023-2024/Architecture Computer$ cd ~/work/study/2023-2024/Architecture Computer/arch-pc
liveuser@localhost-live:~/work/study/2023-2024/Architecture Computer/arch-pc$
```

Рис. 17: Перемещение между директориями

Удаляю лишние файлы с помощью утилиты `rm`

```
liveuser@localhost-live:~/work/study/2023-2024/Architecture Computer/arch-pc$ rm package.json
liveuser@localhost-live:~/work/study/2023-2024/Architecture Computer/arch-pc$
```

Рис. 18: Удаление файлов

Устанавливаю команду `make`

```
liveuser@localhost-live:~/work/study/2023-2024/Architecture Computer/arch-pc$ sudo yum groupinstall "Development Tools"
Last metadata expiration check: 0:09:55 ago on Sat 28 Sep 2024 10:53:10 AM EDT.
Dependencies resolved.
=====
```

Рис. 19: установка make

Создаю необходимые каталоги, проверила командой `ls` созданные каталоги

```
liveuser@localhost-live:~/work/study/2023-2024/Architecture Computer/arch-pc$ echo arch-pc > COURSE
liveuser@localhost-live:~/work/study/2023-2024/Architecture Computer/arch-pc$ make prepare
liveuser@localhost-live:~/work/study/2023-2024/Architecture Computer/arch-pc$ ls
CHANGELOG.md  config  COURSE  labs  LICENSE  Makefile  prepare  presentation  README.en.md  README.git-flow.md  README.md  template
liveuser@localhost-live:~/work/study/2023-2024/Architecture Computer/arch-pc$ cd labs
liveuser@localhost-live:~/work/study/2023-2024/Architecture Computer/arch-pc/labs$ ls
lab01  lab02  lab03  lab04  lab05  lab06  lab07  lab08  lab09  lab10  lab11  README.md  README.ru.md
liveuser@localhost-live:~/work/study/2023-2024/Architecture Computer/arch-pc/labs$
```



Рис. 20: Создание каталогов

Отправляю созданные каталоги с локального репозитория на сервер: добавляю все созданные каталоги с помощью `git add`, комментирую и сохраняю изменения на сервере как добавление курса с помощью `git commit`

```
liveuser@localhost-live: ~/work/study/2023-2024/Arhetucture Computer/arch-pc/labs/lab01$ git add .
liveuser@localhost-live:~/work/study/2023-2024/Arhetucture Computer/arch-pc/labs/lab01$ git commit -am 'feat(main): make course structure'
[master eefb80b] feat(main): make course structure
225 files changed, 53682 insertions(+), 14 deletions(-)
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
create mode 100644 labs/lab01/.~lock.lb01_Shabakova_otchet#
create mode 100644 labs/lab01/lb01_Shabakova_otchet
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
```


Рис. 21: Добавление и сохранение изменений на сервере

Отправляю все на сервер с помощью `push`

```
liveuser@localhost-live:~/work/study/2023-2024/Arhetucture Computer/arch-pc/labs/lab01$ git push
Enumerating objects: 39, done.
Counting objects: 100% (39/39), done.
Compressing objects: 100% (31/31), done.
Writing objects: 100% (37/37), 341.67 KiB | 2.29 MiB/s, done.
Total 37 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:dazz4q/study_2023-2024_arh-pc.git
 23cc295..eefb80b master -> master
liveuser@localhost-live:~/work/study/2023-2024/Arhetucture Computer/arch-pc/labs/lab01$
```

Рис. 22: Выгрузка изменений на сервер

Проверила правильность создания иерархии рабочего пространства в локальном репозитории на странице *github*

 **dazz4q** feat(main): make course structure

Name	Last commit message
..	
lab01	feat(main): make course struc
lab02	feat(main): make course struc
lab03	feat(main): make course struc
lab04	feat(main): make course struc
lab05	feat(main): make course struc
lab06	feat(main): make course struc
lab07	feat(main): make course struc
lab08	feat(main): make course struc
lab09	feat(main): make course struc
lab10	feat(main): make course struc
lab11	feat(main): make course struc

Рис 23: выполнение лабораторной

**4.7    Выполнение заданий для самостоятельной работы**

Создаю отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs>lab02>report).

```
liveuser@localhost-live:~/work/study/2023-2024/Arhetucture Computer/arch-pc$ cd labs/lab01
liveuser@localhost-live:~/work/study/2023-2024/Arhetucture Computer/arch-pc/labs/lab01$ touch lb01_Shabakova_otchet
liveuser@localhost-live:~/work/study/2023-2024/Arhetucture Computer/arch-pc/labs/lab02$ touch lb02_Shabakova_otchet
liveuser@localhost-live:~/work/study/2023-2024/Arhetucture Computer/arch-pc/labs/lab02$ cd ~/work/study/2023-2024/"Arhetucture Computer"/arch-pc
```

## Загрузила на github файлы лабораторных работ

study\_2023-2024\_arh-pc / лаборатории / lab01 /

dazz4q Добавление файлов с помощью upload			Добавить файл	...
			+ Создать новый файл	
			Загрузить файлы	
Имя	Сообщение о последней фиксации	Дата последней...		
..				
презентация	подвиг (основной): создание структуры курса	18 минут назад		
Сообщить	подвиг (основной): создание структуры курса	18 минут назад		
~lock.lb01_Shabakova_otchet#	подвиг (основной): создание структуры курса	18 минут назад		
lb01_Shabakova_otchet	подвиг (основной): создание структуры курса	18 минут назад		
lb01_Shabakova_otchet.pdf	Добавление файлов с помощью upload	сейчас		

## 5 Выводы

При выполнении данной лабораторной работы я изучила идеологию и применение средств контроля версий, а также приобрела практические навыки по работе с системой git.

## 6 Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: [http://www.stolyarov.info/books/asm\\_unix](http://www.stolyarov.info/books/asm_unix).
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).

