

**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**  
**ФАКУЛЬТЕТ ФИЗИКО-МАТЕМАТИЧЕСКИХ И ЕСТЕСТВЕННЫХ НАУК**

## **Лабораторная работа №2.**

**Система контроля версий Git.**

Левищева Анастасия Петровна, НКАбд-02-25

# **Содержание**

<b>1. Цель работы.....</b>
<b>2. Задание.....</b>
<b>3. Теоретическое введение.....</b>
<b>3.1. Системы контроля версий. Общие понятия.....</b>
<b>3.2. Система контроля версий Git.....</b>
<b>3.3. Основные команды git.....</b>
<b>3.4. Стандартные процедуры работы при наличии центрального репозитория.....</b>
<b>4. Выполнение лабораторной работы.....</b>
<b>5. Выводы.....</b>

## **Список литературы**

# **Список иллюстраций**

<b>4.1.</b> Создание учетной записи.....	10
<b>4.2.</b> Имя и e-mail.....	10
<b>4.3.</b> Настройка utf-8.....	10
<b>4.4.</b> Имя ветки.....	10
<b>4.5.</b> Параметр autocrlf.....	10
<b>4.6.</b> Параметр safecrlf.....	11
<b>4.7.</b> Генерация ключей.....	11
<b>4.8.</b> Setting.....	11
<b>4.9.</b> New SSH key.....	12
<b>4.10.</b> Копируем ключ.....	12
<b>4.11.</b> Ключ.....	12
<b>4.12.</b> Создание каталога для предмета “Архитектура компьютера”.....	13
<b>4.13.</b> Создание репозитория.....	13
<b>4.14.</b> Переход в каталог курса.....	14
<b>4.15.</b> Клонируем репозиторий.....	14
<b>4.16.</b> Переход в каталог курса.....	14
<b>4.17.</b> Создание каталогов.....	14
<b>4.18.</b> Отправка файлов на сервер.....	14
<b>4.19.</b> Создание отчета.....	15
<b>4.20.</b> Копирование отчетов.....	15
<b>4.21.</b> Загрузка файлов на github.....	15

## **Список таблиц**

3.3.1. Основные команды git.....	8
----------------------------------	---

## **1. Цель работы**

Целью работы является изучение идеологии и применения средств контроля версий, приобретение практических навыков по работе с системой контроля версий git.

## **2. Задание**

1. Создайте отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs/lab02/report).
2. Скопируйте отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства.
3. Загрузите файлы на github.

### **3. Теоретическое введение**

#### **3.1. Системы контроля версий. Общие понятия**

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удаленном репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведенные разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.

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.

Среди классических VCS наиболее известны CVS, Subversion, а среди распределенных — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

#### **3.2. Система контроля версий 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 распределенной системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией.

#### **3.3. Основные команды git**

Наиболее часто используемые команды `git` представлены в табл. 3.3.1.

Команда	Описание
git init	создание основного дерева репозитория
git pull	получение обновлений (изменений) текущего дерева из центрального репозитория
git push	отправка всех произведённых изменений локального дерева в центральный репозиторий
git status	просмотр списка изменённых файлов в текущей директории
git diff	просмотр текущих изменений
git add .	добавить все изменённые и/или созданные файлы и/или каталоги
git add имена_файлов	добавить конкретные изменённые и/или созданные файлы и/или каталоги
git rm имена_файлов	удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории)
git commit -am 'Описание коммита'	сохранить все добавленные изменения и все изменённые файлы
git checkout -b имя_ветки	создание новой ветки, базирующейся на текущей
git checkout имя_ветки	переключение на некоторую ветку (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)
git push origin имя_ветки	отправка изменений конкретной ветки в центральный репозиторий
git merge --no-ff имя_ветки	слияние ветки с текущим деревом
git branch -d имя_ветки	удаление локальной уже слитой с основным деревом ветки
git branch -D имя_ветки	принудительное удаление локальной ветки
git push origin :имя_ветки	удаление ветки с центрального репозитория

Табл. 3.3.1. Основные команды git

### 3.4. Стандартные процедуры работы при наличии центрального репозитория

Работа пользователя со своей веткой начинается с проверки и получения изменений из центрального репозитория (при этом в локальное дерево до начала этой процедуры не должно было вноситься изменений):

```
git checkout master
```

```
git pull
```

```
git checkout -b имя_ветки
```

Затем можно вносить изменения в локальном дереве и/или ветке. После завершения внесения какого-то изменения в файлы и/или каталоги проекта необходимо разместить их в центральном репозитории. Для этого необходимо проверить, какие файлы изменились к текущему моменту:

```
git status
```

При необходимости удаляем лишние файлы, которые не хотим отправлять в центральный репозиторий. Затем полезно просмотреть текст изменений на предмет соответствия правилам ведения чистых коммитов:

```
git diff
```

Если какие-либо файлы не должны попасть в коммит, то помечаем только те файлы, изменения которых нужно сохранить. Для этого используем команды добавления и/или удаления с нужными опциями:

```
git add имени_файлов
```

```
git rm имени_файлов
```

Если нужно сохранить все изменения в текущем каталоге, то используем:

```
git add
```

Затем сохраняем изменения, поясняя, что было сделано:

```
git commit -am "Some commit message"
```

и отправляем в центральный репозиторий:

```
git push origin имя_ветки
```

или

```
git push
```

## 4. Выполнение лабораторной работы

### 4.1. Настройка github

Создадим учетную запись на сайте <https://github.com/> и заполним основные данные.

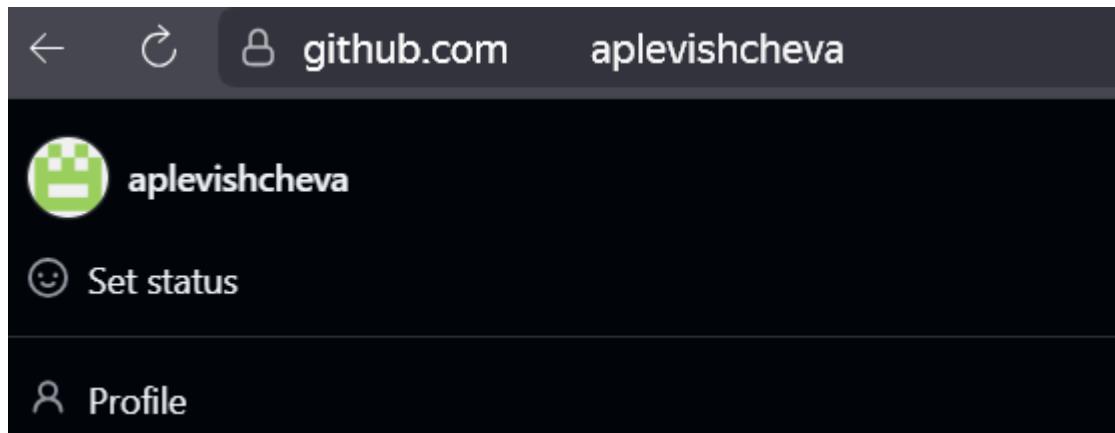


Рис.4.1. Создание учетной записи

### 4.2. Базовая настройка git

Откроем терминал и введем следующие команды, указав имя и e-mail:

```
aplevishcheva@debian:~$ git config --global user.name "aplevishcheva"
aplevishcheva@debian:~$ git config --global user.email "1032252643@pfur.ru"
```

Рис.4.2. Имя и e-mail

Настроим utf-8 в выводе сообщений git:

```
aplevishcheva@debian:~$ git config --global core.quotepath false
```

Рис.4.3. Настройка utf-8

Зададим имя начальной ветки (будем называть ее master):

```
aplevishcheva@debian:~$ git config --global init.defaultBranch master
```

Рис.4.4. Имя ветки

Параметр autocrlf:

```
aplevishcheva@debian:~$ git config --global core.autocrlf input
```

Рис.4.5. Параметр autocrlf

Параметр safecrlf:

```
aplevishcheva@debian:~$ git config --global core.safecrlf warn
```

Рис.4.6. Параметр safecrlf

### 4.3. Создание SSH-ключа

Сгенерируем пару ключей (приватный и открытый) для последующей идентификации пользователя на сервере репозиториев:

```
alevishcheva@debian:~$ ssh-keygen -C "Anastasiya Levishcheva 1032252643@pfur.ru"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/alevishcheva/.ssh/id_ed25519):
/home/alevishcheva/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y
Enter passphrase for "/home/alevishcheva/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/alevishcheva/.ssh/id_ed25519
Your public key has been saved in /home/alevishcheva/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:xo9A7BR776N+YlcEcEaYcsinJtFkbBRJ4sYUTFmkbFk Anastasiya Levishcheva 1032252643@pfur.ru
The key's randomart image is:
+-- [ED25519 256] --+
| o=%E+.+=
| =+B0o=o.
| Oo==. .
| o.+oo . .
| oo S ..
| o + .
| . +.
| o.o.
| oo+
+--- [SHA256] ---+
```

Рис.4.7. Генерация ключей

Ключи сохраняются в каталоге `~/.ssh/`.

Далее загрузим сгенерированный открытый ключ. Для этого зайдем на сайт <http://github.org/> под своей учетной записью и перейдем в меню Setting:

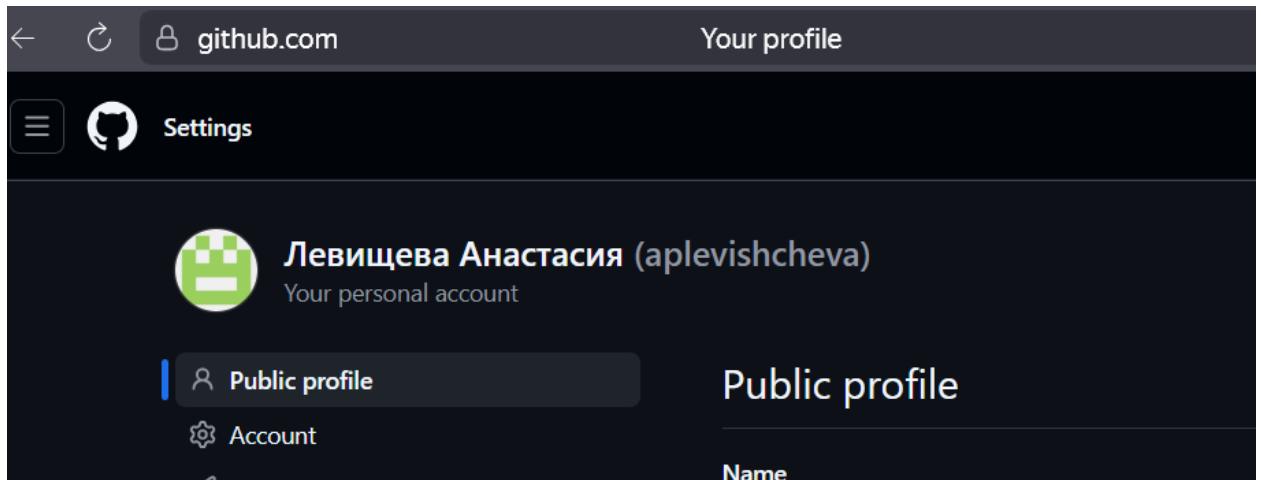
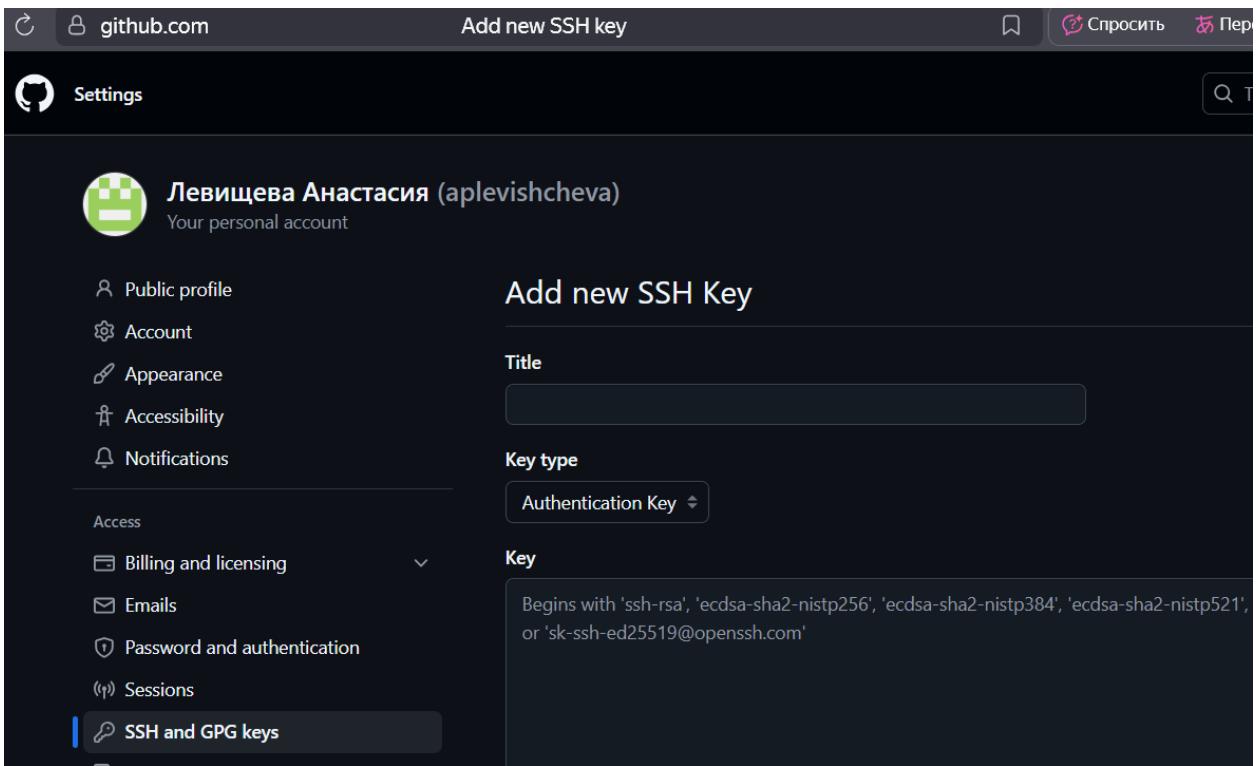


Рис.4.8. Setting

После этого выберем в боковом меню SSH and GPG keys и перейдем к New SSH key:



The screenshot shows the GitHub 'Add new SSH key' interface. At the top, there's a header with the GitHub logo, the URL 'github.com', and a search bar. Below the header, the user's profile picture and name 'Левищева Анастасия (aplevishcheva)' are displayed, along with the text 'Your personal account'. On the left, a sidebar lists various account settings: 'Public profile', 'Account', 'Appearance', 'Accessibility', 'Notifications', 'Access' (with 'Billing and licensing' expanded), 'Emails', 'Password and authentication', 'Sessions', and 'SSH and GPG keys' (which is selected and highlighted in blue). The main right-hand area is titled 'Add new SSH Key' and contains fields for 'Title' (an empty input field) and 'Key type' (set to 'Authentication Key'). A note below the key type dropdown says: 'Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', or 'sk-ssh-ed25519@openssh.com''. There's also a note about the key starting with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', or 'sk-ssh-ed25519@openssh.com'.

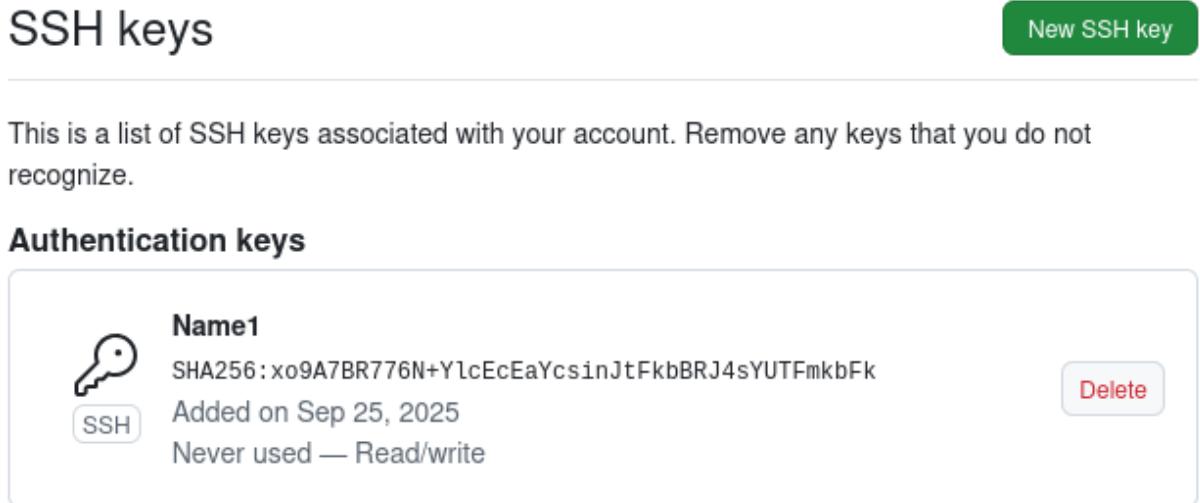
Рис.4.9. New SSH key

Копируем из локальной консоли ключ в буфер обмена cat:

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

Рис.4.10. Копируем ключ

Вставляем ключ в появившееся на сайте поле и указываем для ключа имя (Title).



The screenshot shows the GitHub 'SSH keys' page. At the top, there's a green button labeled 'New SSH key'. Below it, a message reads: 'This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.' Underneath, there's a section titled 'Authentication keys' containing a single key entry. The entry includes a key icon, the title 'Name1', the SHA256 fingerprint 'SHA256: xo9A7BR776N+YlcEcEaYcsinJtFkbBRJ4sYUTFmkbFk', the date 'Added on Sep 25, 2025', and the status 'Never used — Read/write'. To the right of the key details is a red 'Delete' button.

Check out our guide to [connecting to GitHub using SSH keys](#) or troubleshoot [common SSH problems](#).

Рис.4.11. Ключ

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

При выполнении лабораторной работы будем придерживаться следующей иерархии рабочего пространства:

```
~/work/study/
└─ 2025-2026/
    └─ Архитектура компьютера/
        └─ arch-pc/
            └─ labs/
                └─ lab01/
                └─ lab02/
                └─ lab03/
                ...
            ...
```

Откроем терминал и создадим каталог предмета “Архитектура компьютера”:

```
aplevishcheva@debian:~$ mkdir -p ~/work/study/2025-2026/"Архитектура компьютера"
```

Рис.4.12. Создание каталога для предмета “Архитектура компьютера”

## 4.5. Создание репозитория курса на основе шаблона

Перейдем на страницу репозитория с шаблоном курса

<https://github.com/yamadharma/course-directory-student-template> и создадим свой репозиторий study\_2025–2026\_arh-pc:

The screenshot shows the GitHub 'Create a new repository' interface. It's a dark-themed form with two main sections: 'General' and 'Configuration'.  
**General Section:**

- Start with a template:** A dropdown menu shows 'yamadharma/course-directory-student-template'.
- Include all branches:** A toggle switch is set to 'Off'.
- Owner:** Set to 'aplevishcheva'.
- Repository name \***: The input field contains 'study\_2025–2026\_arh-pc'. A note below says: 'Your new repository will be created as study\_2025–2026\_arh-pc. The repository name can only contain ASCII letters, digits, and the characters ., -, and \_.'
- Description:** A text area with placeholder 'Great repository names are short and memorable. How about reimaged-octo-spork?' and character count '0 / 350 characters'.

**Configuration Section:**

- Choose visibility \***: Set to 'Public'.

At the bottom right is a large green button labeled 'Create repository'.

Рис.4.13. Создание репозитория

Откроем терминал и перейдем в каталог курса:

```
alevishcheva@debian:~/work/study/2025-2026/"Архитектура компьютера"  
alevishcheva@debian:~/work/study/2025-2026/Архитектура компьютера$
```

Рис.4.14. Переход в каталог курса

Клонируем созданный репозиторий:

```
alevishcheva@debian:~/work/study/2025-2026/Архитектура компьютера$ git clone --recursive https://github.com/alevishcheva/study_2025-2026_arch-pc.git arch-pc  
Клонирование в «arch-pc»...  
remote: Enumerating objects: 38, done.  
remote: Counting objects: 100% (38/38), done.  
remote: Compressing objects: 100% (36/36), done.  
remote: Total 38 (delta 1), reused 26 (delta 1), pack-reused 0 (from 0)  
Получение объектов: 100% (38/38), 23.49 КиБ | 224.00 КиБ/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/alevishcheva/work/study/2025-2026/Архитектура компьютера/arch-pc/template/presentation»...  
remote: Enumerating objects: 161, done.  
remote: Counting objects: 100% (161/161), done.  
remote: Compressing objects: 100% (111/111), done.  
remote: Total 161 (delta 60), reused 142 (delta 41), pack-reused 0 (from 0)  
Получение объектов: 100% (161/161), 2.65 КиБ | 823.00 КиБ/с, готово.  
Определение изменений: 100% (60/60), готово.  
Клонирование в «/home/alevishcheva/work/study/2025-2026/Архитектура компьютера/arch-pc/template/report»...  
remote: Enumerating objects: 221, done.  
remote: Counting objects: 100% (221/221), done.  
remote: Compressing objects: 100% (152/152), done.  
remote: Total 221 (delta 98), reused 180 (delta 57), pack-reused 0 (from 0)  
Получение объектов: 100% (221/221), 765.46 КиБ | 438.00 КиБ/с, готово.  
Определение изменений: 100% (98/98), готово.  
Submodule path «template/presentation»: checked out '6ef5d5c4ee78e4456caff3dc7062cfcad26058ca6'  
Submodule path «template/report»: checked out '89a96e22199b4df88227b9b3fa3d4714c85f68dd2'  
alevishcheva@debian:~/work/study/2025-2026/Архитектура компьютера$
```

Рис.4.15. Клонируем репозиторий

## 4.6. Настройка каталога курса

Перейдем в каталог курса:

```
alevishcheva@debian:~/work/study/2025-2026/Архитектура компьютера$ cd arch-pc  
alevishcheva@debian:~/work/study/2025-2026/Архитектура компьютера/arch-pc$
```

Рис.4.16. Переход в каталог курса

Создадим необходимые каталоги:

```
alevishcheva@debian:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ echo arch-pc > COURSE  
alevishcheva@debian:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ make prepare
```

Рис.4.17. Создание каталогов

Отправим файлы на сервер:

```
alevishcheva@debian:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git add .  
  
alevishcheva@debian:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git commit -am 'feat(main): make course structure'  
  
alevishcheva@debian:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git push
```

Рис.4.18. Отправка файлов на сервер

Проверим правильность создания иерархии рабочего пространства в локальном репозитории и на странице github.

## Задание для самостоятельной работы.

1. Создадим отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs/lab02/report).

```
alevishcheva@debian:~$ cd /home/alevishcheva/work/study/2025-2026/"Архитектура компьютера"/arch-pc/labs/lab01/report  
alevishcheva@debian:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab01/report  
$ git status
```

Рис.4.19. Создание отчета

2. Скопирем отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства.

```
alevishcheva@debian:~$ cp ~/Загрузки/lab01_report.pdf labs/lab01/report/report.md
```

Рис.4.20. Копирование отчетов

3. Загрузим файлы на github.

```
alevishcheva@debian:~/work/study/2025-2026/Архитектура компьютера/arch-pc$ git add .
```

```
alevishcheva@debian:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab01/report$ git commit -m "feat(labs): add report for lab02 and copy lab01"
```

```
alevishcheva@debian:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab01/report$ git push
```

Рис.4.21. Загрузка файлов на github

## **5. Выводы**

В ходе выполнения данной лабораторной работы я изучила теоретические основы и получила практические навыки работы с системой контроля версий Git. Я научилась выполнять базовую настройку Git, создавать и настраивать репозитории на GitHub, использовать SSH-ключи для безопасного соединения, а также освоила основной рабочий цикл: добавление изменений (add), их фиксацию (commit) и отправку на удаленный сервер (push). Цель работы была полностью достигнута.

## Список литературы

1. Введение - О системе контроля версий/ <https://git-scm.com/book/ru/v2/%D0%92%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5%D0%9E-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B5-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8F-%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D0%B5%D0%9E>
  2. What is Git version control? / <https://about.gitlab.com/topics/version-control/what-is-git-version-control/>
  3. Основные команды GIT / <https://habr.com/ru/articles/918386/>