

# **Отчёт по лабораторной работе №2**

**дисциплина: Архитектура компьютеров**

Терещенкова Маргарита Владимировна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
4.1	Настройка github . . . . .	9
4.2	Базовая настройка git . . . . .	9
4.3	Создание SSH-ключа . . . . .	10
4.4	Создание рабочего пространства и репозитория курса на основе шаблона . . . . .	12
4.5	Создание репозитория курса на основе шаблона . . . . .	12
4.6	Настройка каталога курса . . . . .	14
<b>5</b>	<b>Задание для самостоятельной работы</b>	<b>15</b>
<b>6</b>	<b>Выводы</b>	<b>16</b>
	<b>Список литературы</b>	<b>17</b>

# Список иллюстраций

4.1	Аккаунт GitHub . . . . .	9
4.2	Предварительная конфигурация git . . . . .	9
4.3	Настройка кодировки . . . . .	9
4.4	Создание имени для начальной ветки . . . . .	10
4.5	Параметр autocr . . . . .	10
4.6	Параметр safecrlf . . . . .	10
4.7	Генерация SSH-ключа . . . . .	11
4.8	Копирование SSH-ключа . . . . .	11
4.9	Добавление SSH-ключа . . . . .	11
4.10	Создание директории . . . . .	12
4.11	Окно Github . . . . .	12
4.12	Создание репозитория . . . . .	13
4.13	Клонирование репозитория . . . . .	13
4.14	Перемещение между директориями . . . . .	14
4.15	Создание каталогов . . . . .	14
4.16	Добавление и сохранение изменений на сервере . . . . .	14
5.1	Добавление отчёта . . . . .	15

## **Список таблиц**

# 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 изменения, сделанные разными участниками, 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. 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. 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). Затем можно вносить изменения в локальное дерево и/или ветке. После завершения внесения какого-то изменения в файлы и/или каталоги проекта необходимо разместить их в центральном репозитории.



## 4 Выполнение лабораторной работы

### 4.1 Настройка github

Создала учётную запись на сайте <https://github.com/> и заполнила основные данные.



Рис. 4.1: Аккаунт GitHub

### 4.2 Базовая настройка git

Сначала сделала предварительную конфигурацию git.

```
margo@margo-pc:~$ git config --global user.name "<ritondriy>"  
margo@margo-pc:~$ git config --global user.email "<ritkasuper18@gmail.com>"
```

Рис. 4.2: Предварительная конфигурация git

Настроила utf в выводе сообщений git.

```
margo@margo-pc:~$ git config --global core.quotePath false
```

Рис. 4.3: Настройка кодировки

Задала имя «master» для начальной ветки.

```
margo@margo-pc:~$ git config --global init.defaultBranch master
```

Рис. 4.4: Создание имени для начальной ветки

Задала параметр autocrlf со значением input.

```
margo@margo-pc:~$ git config --global core.autocrlf input
```

Рис. 4.5: Параметр autocr

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

```
margo@margo-pc:~$ git config --global core.safecrlf warn
```

Рис. 4.6: Параметр safecrlf

## 4.3 Создание SSH-ключа

Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый). Для этого вводила команду `ssh-keygen -C "Имя Фамилия, work@email"`, указывая имя владельца и электронную почту владельца (рис. 4.8). Ключ автоматически сохранится в каталоге `~/.ssh/`.

```

margo@margo-pc:~$ ssh-keygen -C "Маргарита Терещенкова <ritkasuper18@gmail.com>"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/margo/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/margo/.ssh/id_ed25519
Your public key has been saved in /home/margo/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:/p1/GmXopxafzUekllYhj5uCY2rlVt5WnIWIIFj+sQo Маргарита Терещенкова <ritkasuper18@gmail.com>
The key's randomart image is:
+--[ED25519 256]--+
|    o.            |
|   ... .         |
|  .... . ....   |
|   . o. ....+   |
|  E  S      .==+ |
|   . o . . o*B. |
|   . + + ..*=+  |
|   . B + *.=*   |
|   ..+ o =o+o.  |
+-----[SHA256]-----+

```

Рис. 4.7: Генерация SSH-ключа

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.

```

margo@margo-pc:~$ cat ~/.ssh/id_ed25519.pub | xclip -sel clip

```

Рис. 4.8: Копирование SSH-ключа

**Add new SSH Key**

**Title**

**Key type**  
 Authentication Key

**Key**  
 ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIAEEOpT7xaZR6uGaihBmp44aqvHL7tUF72jf5YnYsDr8 Маргарита Терещенкова <ritkasuper18@gmail.com>

**Add SSH key**

Рис. 4.9: Добавление SSH-ключа

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

Создаю директорию, рабочее пространство, с помощью утилиты `mkdir`, с помощью ключа `-p` создаю все директории после домашней `~/work/study/2024-2025/“Архитектура компьютера”`

```
margo@margo-pc:~$ mkdir -p ~/work/study/2024-2025/"Архитектура компьютера"
```

Рис. 4.10: Создание директории

## 4.5 Создание репозитория курса на основе шаблона

Перешла на страницу репозитория с шаблоном курса <https://github.com/yamadharma/course-directory-student-template>. Далее выбираю `Use this template`.

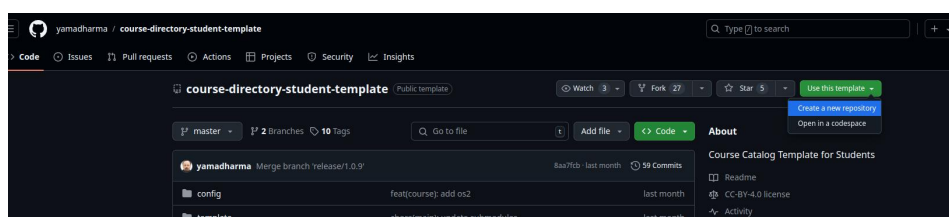


Рис. 4.11: Окно Github

В открывшемся окне задаю имя репозитория (Repository name): `study_2024-2025_arhpc` 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**


 yamadharna/course-directory-student-template ▾

Start your repository with a template repository's contents.

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

---

**Owner \*** **Repository name \***

 ritondriy ▾ /

✔ study\_2024-2025\_arhpc is available.

Great repository names are short and memorable. Need inspiration? How about **silver-spork** ?


**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. 4.12: Создание репозитория

Клонирую созданный репозиторий с помощью команды `git clone --recursive git@github.com:/study_2024-2025_arh-pc.git arch-pc`.

```
margo@margo-pc: ~/work/study/2024-2025/Архитектура компьютера$ git clone --recursive https://github.com/ritondriy/study_2024-2025_arhpc
Клонирование в «study_2024-2025_arhpc»...
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 19 (delta 0), pack-reused 0 (from 0)
Получение объектов: 100% (33/33), 18.82 KiB | 6.27 MiB/c, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharna/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharna/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/margo/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/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 108 (delta 31), pack-reused 0 (from 0)
Получение объектов: 100% (111/111), 162.17 KiB | 495.00 KiB/c, готово.
Определение изменений: 100% (42/42), готово.
Клонирование в «/home/margo/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/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 KiB | 992.00 KiB/c, готово.
Определение изменений: 100% (60/60), готово.
Submodule path 'template/presentation': checked out 'c9b2712b4b2d431ad5086c9c72a02bd2fca1dd46'
Submodule path 'template/report': checked out 'c26e22effe7b3e0495707d82ef561ab185f5c748'
```

Рис. 4.13: Клонирование репозитория

## 4.6 Настройка каталога курса

Перехожу в каталог arch-pc с помощью утилиты `cd`, удаляю лишние файлы с помощью утилиты `rm`.

```
margo@margo-pc: ~/work/study/2024-2025/Архитектура компьютера$ cd ~/work/study/2024-2025/Архитектура\ компью
тера/study_2024-2025_arhpc/
margo@margo-pc: ~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc$ rm package.json
margo@margo-pc: ~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc$
```

Рис. 4.14: Перемещение между директориями

Создаю каталог, отправляю каталоги с локального репозитория.

```
margo@margo-pc: ~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc$ echo study_2024-2025_arhpc > COURSE
margo@margo-pc: ~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc$ make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare       Generate directories structure
  submodule     Update submodules
```

Рис. 4.15: Создание каталогов

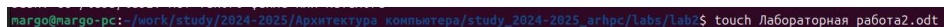
Отправляю созданные каталоги с локального репозитория на сервер: добавляю все созданные каталоги с помощью `git add`, комментирую и сохраняю изменения на сервере как добавление курса с помощью `git commit`.

```
margo@margo-pc: ~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc$ git add .
margo@margo-pc: ~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc$ git commit -am 'feat(main): make cou
rse structure'
[master 5ac4410] feat(main): make course structure
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
margo@margo-pc: ~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc$ git push
```

Рис. 4.16: Добавление и сохранение изменений на сервере

## 5 Задание для самостоятельной работы

1. Перехожу в директорию labs/lab2/report с помощью утилиты cd. Создаю в каталоге файл для отчета по второй лабораторной работе с помощью утилиты touch.



```
margo@margo-pc: /work/study/2024-2025/Архитектура компьютера/study_2024-2025_arhpc/labs/lab2$ touch Лабораторная работа2.odt
```

Рис. 5.1: Добавление отчёта

2. Скопировала отчёт за первую лабораторную работу в папку labs>lab1 аналогично.
3. Загрузили файлы на github.

## **6 Выводы**

Благодаря данной лабораторной работе, я изучила идеологию и применение средств контроля версий. Приобрела практические навыки по работе с системой git.



## 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://midnightcommander.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/Learning-bash-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., Lupin 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. — БХВ- Петербург, 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).