

Отчет по лабораторной работе №2

Дисциплина: Архитектура компьютера

Максимова Дарья Валерьевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	14

Список иллюстраций

4.1	рис. 1 Создание учетной записи на github	9
4.2	рис. 2 Предварительная конфигурация	10
4.3	рис. 3 Настройка utf-8	10
4.4	рис. 4	11
4.5	рис. 5	11
4.6	рис. 6	12
4.7	рис. 7	12
4.8	рис. 8	13
4.9	рис. 9	13
4.10	рис. 10	13
4.11	рис. 11	13

Список таблиц

1 Цель работы

Целью данной работы является изучение идеологии и применение средств контроля версий, а также приобрести практические навыки работы 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.

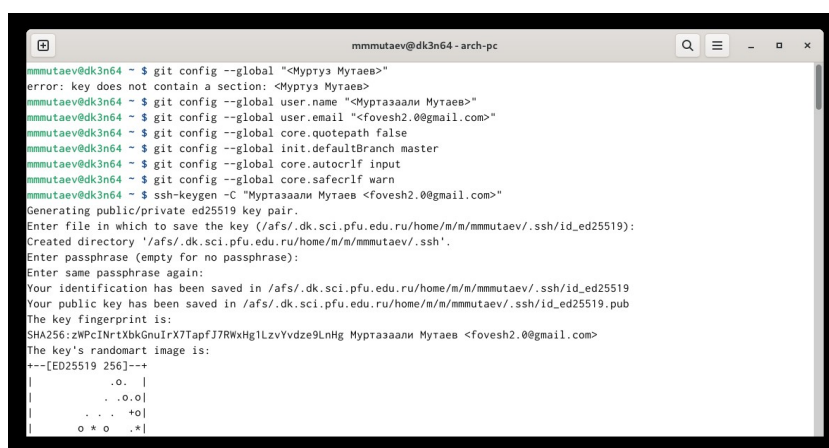
3 Теоретическое введение

Системы управления версией (Система управления версией, 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. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций управления версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы с опреем определенных 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 измененных файлов, а производить так называемое дельта-сжатие — сохранять только изменения между последовательными версиями, что позволяет уменьшить объем 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 или заблокировать файлы для изменения. В зависимости от настро-

ек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла с средствами файловой системы ОС, обеспечивая таким образом привилегированный доступ только одному пользователю, работающему с файлом. Системы управления версиями 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 вносили. Обычно такая информация хранится в журнале изменений, доступ к которому можно ограничить. В отличие от классических, в распределённых системах управления версиями центральный репозиторий не является обязательным. Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы похожи, они отличаются в основном синтаксисом используемых в работе команд. Система управления версиями Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала, введя `git`-команды 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 быть внесено изменений). Затем можно внести изменения в локальное дерево и/или ветку. После внесения каких-либо изменений в файлы и/или каталоги проекта необходимо разместить их в центральном репозитории.

4 Выполнение лабораторной работы

Создаю учетную запись на сайте <https://github.com/> и заполняю основные данные. (рис. [-fig:001])



```
mmmutaev@dk3n64 ~ $ git config --global "<Муртаза Муртаев>"
error: key does not contain a section: <Муртаза Муртаев>
mmmutaev@dk3n64 ~ $ git config --global user.name "<Муртазааали Муртаев>"
mmmutaev@dk3n64 ~ $ git config --global user.email "<fovesh2.0@gmail.com>"
mmmutaev@dk3n64 ~ $ git config --global core.quotepath false
mmmutaev@dk3n64 ~ $ git config --global init.defaultBranch master
mmmutaev@dk3n64 ~ $ git config --global core.autocrlf input
mmmutaev@dk3n64 ~ $ git config --global core.safecrlf warn
mmmutaev@dk3n64 ~ $ ssh-keygen -C "Муртазааали Муртаев <fovesh2.0@gmail.com>"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/afs/.dk.sci.pfu.edu.ru/home/m/mmmutaev/.ssh/id_ed25519):
Created directory '/afs/.dk.sci.pfu.edu.ru/home/m/mmmutaev/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/m/mmmutaev/.ssh/id_ed25519
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/m/mmmutaev/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:zWpCINrtXbkGnuIrX7TapfJ7RWxHg1LzvYvdze9LnHg Муртазааали Муртаев <fovesh2.0@gmail.com>
The key's randomart image is:
+--[ED25519 256]--+
|      .o.  |
|      .o.o |
|      . . +o|
|      o * o .*|
```

Рис. 4.1: рис. 1 Создание учетной записи на github

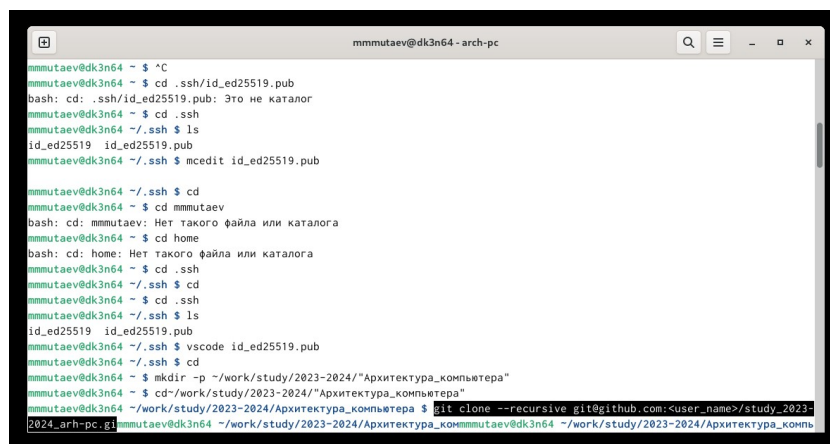
Делаю предварительную конфигурацию git, указав имя и email владельца репозитория (рис. [-fig:002])



```
mmmutaev@dk3n64 ~ $ git config --global "<Мутря Мутаев>"
error: key does not contain a section: <Мутря Мутаев>
mmmutaev@dk3n64 ~ $ git config --global user.name "<Мутря Мутаев>"
mmmutaev@dk3n64 ~ $ git config --global user.email "<fovesh2.0@gmail.com>"
mmmutaev@dk3n64 ~ $ git config --global core.quotepath false
mmmutaev@dk3n64 ~ $ git config --global init.defaultBranch master
mmmutaev@dk3n64 ~ $ git config --global core.autocrlf input
mmmutaev@dk3n64 ~ $ git config --global core.safecrlf warn
mmmutaev@dk3n64 ~ $ ssh-keygen -C "Мутря Мутаев <fovesh2.0@gmail.com>"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/afs/.dk.sci.pfu.edu.ru/home/m/mmmutaev/.ssh/id_ed25519):
Created directory '/afs/.dk.sci.pfu.edu.ru/home/m/mmmutaev/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/m/mmmutaev/.ssh/id_ed25519
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/m/mmmutaev/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:zWpCINrtXbkGnuIrX7TapfJ7RWxHglZvYvde9LnHg Мутря Мутаев <fovesh2.0@gmail.com>
The key's randomart image is:
+--[ED25519 256]--+
|      .o. |
|      .o.o|
|      . . +o|
|      o * o .*|
```

Рис. 4.2: рис. 2 Предварительная конфигурация

Настроим utf-8 в выводе сообщений git, зададим имя начальной ветке(будем называть её master), укажем значение параметров autocrlf и safecrlf (рис. [-fig:003])



```
mmmutaev@dk3n64 ~ $ ^C
mmmutaev@dk3n64 ~ $ cd .ssh/id_ed25519.pub
bash: cd: .ssh/id_ed25519.pub: Это не каталог
mmmutaev@dk3n64 ~ $ cd .ssh
mmmutaev@dk3n64 ~/.ssh $ ls
id_ed25519 id_ed25519.pub
mmmutaev@dk3n64 ~/.ssh $ mcedit id_ed25519.pub

mmmutaev@dk3n64 ~/.ssh $ cd
mmmutaev@dk3n64 ~ $ cd mmmutaev
bash: cd: mmmutaev: Нет такого файла или каталога
mmmutaev@dk3n64 ~ $ cd home
bash: cd: home: Нет такого файла или каталога
mmmutaev@dk3n64 ~ $ cd .ssh
mmmutaev@dk3n64 ~/.ssh $ cd
mmmutaev@dk3n64 ~/.ssh $ cd
mmmutaev@dk3n64 ~/.ssh $ ls
id_ed25519 id_ed25519.pub
mmmutaev@dk3n64 ~/.ssh $ vscode id_ed25519.pub
mmmutaev@dk3n64 ~/.ssh $ cd
mmmutaev@dk3n64 ~ $ mkdir -p ~/work/study/2023-2024/"Архитектура_компьютера"
mmmutaev@dk3n64 ~ $ cd ~/work/study/2023-2024/"Архитектура_компьютера"
mmmutaev@dk3n64 ~/work/study/2023-2024/Архитектура_компьютера $ git clone --recursive git@github.com:<user_name>/study_2023-2024_arh-pc.git
Cloning into 'study_2023-2024_arh-pc'...
```

Рис. 4.3: рис. 3 Настройка utf-8

Для последующей идентификации пользователя на сервере репозитория сгенерируем пару ключей (приватный и открытый) (рис. [-fig:004])

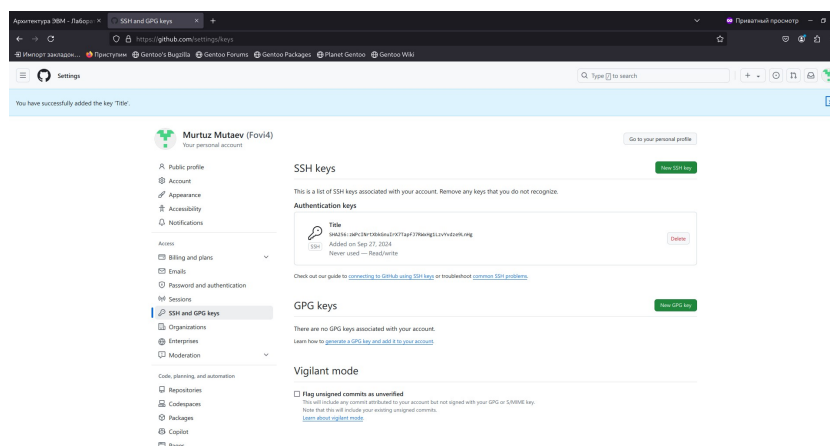


Рис. 4.4: рис. 4

(у меня уже был сгенерирован код) Имеющийся ключ я загрузила на github, загрузив его в буфер обмена. Вставляю скопированный ключ в поле «Ключ». В поле Название указываю имя для ключа. Нажимаю «Добавить SSH-ключ», чтобы завершить добавление ключа. (рис. [-fig:005])

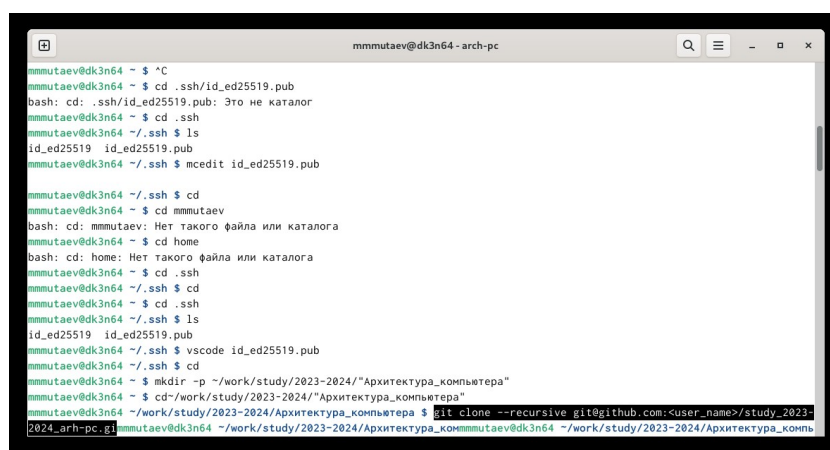


Рис. 4.5: рис. 5

Создадим каталог для предмета «Архитектура компьютера» для последующего создания рабочего пространства. (рис. [-fig:006])

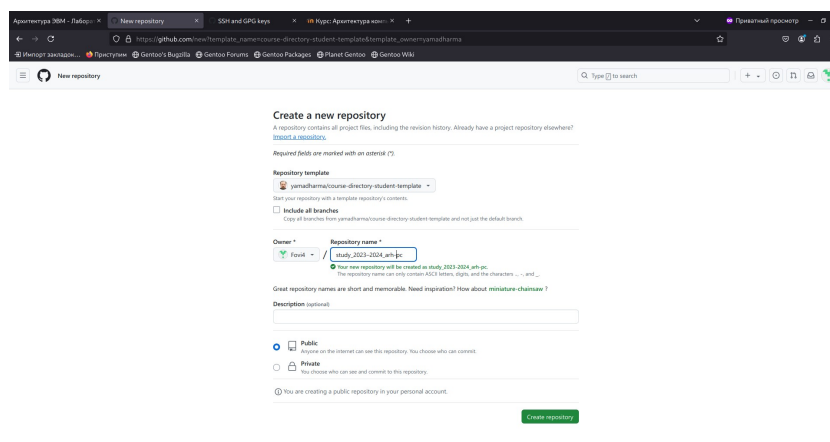


Рис. 4.6: рис. 6

Через web-интерфейс github создадим репозиторий на основе шаблона, указав имя study_2024–2025_arh-pc (рис. [-fig:007])

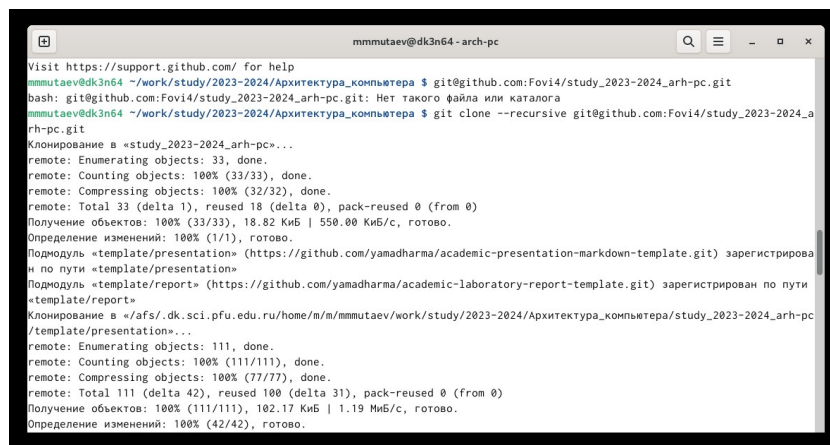


Рис. 4.7: рис. 7

Перейдем в каталог курса и скопируем в него созданный репозиторий с помощью ссылки для клонирования: (рис. [-fig:008])

```
mmmutaev@dk3n64 - arch-pc
mmmutaev@dk3n64 ~/work/study/2023-2024/Архитектура_компьютера $ cd ~/work/study/2023-2024/Архитектура_компьютера/study_2023-2024_arh-pc
mmmutaev@dk3n64 ~/work/study/2023-2024/Архитектура_компьютера/study_2023-2024_arh-pc $ ls
CHANGELOG.md config COURSE LICENSE Makefile package.json README.en.md README.git-flow.md README.md template
mmmutaev@dk3n64 ~/work/study/2023-2024/Архитектура_компьютера/study_2023-2024_arh-pc $ rm package.json
mmmutaev@dk3n64 ~/work/study/2023-2024/Архитектура_компьютера/study_2023-2024_arh-pc arch-pc $ mv study_2023-2024_arh-pc arch-pc
mmmutaev@dk3n64 ~/work/study/2023-2024/Архитектура_компьютера/study_2023-2024_arh-pc $ cd
mmmutaev@dk3n64 ~/work/study/2023-2024/Архитектура_компьютера $ mv study_2023-2024_arh-pc arch-pc
mmmutaev@dk3n64 ~/work/study/2023-2024/Архитектура_компьютера $ ls
arch-pc
mmmutaev@dk3n64 ~/work/study/2023-2024/Архитектура_компьютера $ cd arch-pc
mmmutaev@dk3n64 ~/work/study/2023-2024/Архитектура_компьютера/arch-pc $ echo arch-pc > COURSE
mmmutaev@dk3n64 ~/work/study/2023-2024/Архитектура_компьютера/arch-pc $ make
Usage:
  make <target>

Targets:
  list           List of courses
  prepare       Generate directories structure
  submodule     Update submodules
mmmutaev@dk3n64 ~/work/study/2023-2024/Архитектура_компьютера/arch-pc $ git add .
```

Рис. 4.8: рис. 8

Перейдём в каталог курса, удалим лишние файлы, создадим нужные каталоги и загрузим файлы на сервер: (рис. [-fig:009])

```
mmmutaev@dk3n64 - arch-pc
Usage:
  make <target>

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

mmmutaev@dk3n64 ~/work/study/2023-2024/Архитектура_компьютера/arch-pc $ git add .
mmmutaev@dk3n64 ~/work/study/2023-2024/Архитектура_компьютера/arch-pc $ git commit -am 'feat(main): make course structure'
[master 5c47716] feat(main): make course structure
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
mmmutaev@dk3n64 ~/work/study/2023-2024/Архитектура_компьютера/arch-pc $ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 316 байтов | 316.00 Киб/с, готово.
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:Fov14/study_2023-2024_arh-pc.git
70971e3..5c47716 master -> master
mmmutaev@dk3n64 ~/work/study/2023-2024/Архитектура_компьютера/arch-pc $
```

Рис. 4.9: рис. 9

Создание нужных каталогов: (рис. [-fig:010])

рис. 10

Рис. 4.10: рис. 10

Проверим правильность введенных команд: (рис. [-fig:011])

рис. 11

Рис. 4.11: рис. 11

5 Выводы

В ходе выполнения этой я исследовала концепции и познакомилась с использованием систем контроля версий, а также приобрела практические навыки работы с git.