**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

*Дисциплина : Архитектура компьютера*

МОСКВА

2023 г.

Студент : Зоригоо Номун

Группа : НКАбд-04-23

**Содержание**

1. [Цель работы](#_bookmark0) 5
2. [Задание](#_bookmark1) 6
3. [Теоретическое введение](#_bookmark2) 7
4. [Выполнение лабораторной работы](#_bookmark3) 9

[4.1 Настройка GitHub](#_bookmark4) . . . . . . . . . . . . . . . . . . . . . . . . . . 9

* 1. [Базовая настройка Git](#_bookmark7) 10
  2. [Создание SSH-ключа](#_bookmark13) 11
  3. [Создание рабочего пространства и репозитория курса на основе](#_bookmark19) [шаблона](#_bookmark19) 14
  4. [Создание репозитория курса на основе шаблона](#_bookmark21) 15
  5. [Настройка каталога курса](#_bookmark28) 17
  6. [Выполнение заданий для самостоятельной работы](#_bookmark35) 20

1. [Выводы](#_bookmark54) 27
2. [Список литературы](#_bookmark55) 28

# Цель работы

Целью данной работы является изучить идеологию и применение средств контроля версий, а также приобрести практические навыки по работе с системой git.

# 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.

# Теоретическое введение

Системы контроля версий (Version Control System, VCS) применяются при ра- боте нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен до- ступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, про- изведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется. В классических системах контроля версий используется централизованная модель, предполагающая наличие еди- ного репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник про- екта (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 уда- ляются 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 по- следовательными версиями, что позволяет уменьшить объём хранимых данных. Системы контроля версий поддерживают возможность отслеживания и разреше- ния 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 заблокиро- вать 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 измене- ний 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 распреде- лённых — 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 локальном дереве и/или ветке. После завершения внесения какого-то изменения в файлы и/или каталоги проекта необходимо разместить их в центральном репозитории.

# Выполнение лабораторной работы

## Настройка GitHub

Создаю учетную запись на сайте GitHub (рис 1.1)

A screenshot of a computer

Description automatically generated

Рис 1.1: GitHub account

## Базовая настройка Git

Открываю виртуальную машину, затем открываю терминал и делаю пред- варительную 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. (рис [1.2).](#_bookmark8)



Рис1.2: Предварительная конфигурация git

Настраиваю utf-8 в выводе сообщений git для корректного отображения сим- волов (рис. [1.3).](#_bookmark9)

Рис. 1.3: Настройка кодировки

Задаю имя «master» для начальной ветки (рис 1.4)

Рис. 1.4: Создание имени для начальной ветки

Задаю параметр autocrlf с значением input, так как я работаю в системе Linux, чтобы конвертировать CRLF в LF только при коммитах (рис. [1.5).](#_bookmark11) CR и LF – это сим волы, которые можно использовать для обозначения разрыва строки в текстовых файлах



Рис 1.5

Задаю параметр safecrlf со значением warn, так Git будет проверять преоб- разование на обратимость (рис. [1.6).](#_bookmark12) При значении warn Git только выведет предупреждение, но будет принимать необратимые конвертации.



Рис 1.6

## Создание SSH-ключа

Для последующей идентификации пользователя на сервере репозиториев необ- ходимо сгенерировать пару ключей (приватный и открытый). Для этого ввожу

команду ssh-keygen -C “Имя Фамилия, work@email”, указывая имя владельца и электронную почту владельца (рис. [1.7).](#_bookmark14) Ключ автоматически сохранится в каталоге ~/.ssh/

A computer screen shot of a computer

Description automatically generated

Рис 1.7

Xclip – утилита, позволяющая скопировать любой текст через терминал. Ока- зывается, в дистрибутиве Linux Kali ее сначала надо установить. Устанавливаю xclip с помощью команды apt-get install с ключом -y отимени суперпользователя, введя в начале команды sudo (рис. [1.8).](#_bookmark15)

A screenshot of a computer program

Description automatically generated

Рис. 1.8: Установка утилиты xclip

Копирую открытый ключ из директории, в которой он был сохранен, с помо- щью утилиты xclip (рис. [1.9).](#_bookmark16)



Рис 1.9 Копирование содержимого файла

Открываю браузер, захожу на сайт GitHub. Открываю свой профиль и выбираю страницу «SSH and GPG keys». Нажимаю кнопку «New SSH key» (рис. [1.10).](#_bookmark17)

A screenshot of a computer

Description automatically generated

Рис. 1.10: Окно SSH

Вставляю скопированный ключ в поле «Key». В поле Title указываю имя для ключа. (рис 1.11)

A screenshot of a computer

Description automatically generatedРис. 1.11: Добавление ключа

## Создание рабочего пространства и репозитория курса на основе шаблона

Создаю директорию, рабочее про- странство, с помощью утилиты mkdir, блягодаря ключу -p создаю все директории после домашней ~/work/study/2022-2023/“Архитектура компьютера” рекурсивно. Далее проверяю с помощью ls, действительно ли были созданы каталоги (рис. [1.12).](#_bookmark20)

A screenshot of a computer

Description automatically generated

Рис. 1.12: Создание рабочего пространства

## Создание репозитория курса на основе шаблона

В браузере перехожу на страницу репозитория с шаблоном курса по адресу https://github.com/yamadharma/course-directory-student-template. Далее выбираю

«Use this template», чтобы использовать этот шаблон для своего репозитория

A screenshot of a computer

Description automatically generated

Рис 1.13

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

(рис. [1.14).](#_bookmark23)

A screenshot of a computer

Description automatically generated

Рис 1.14

Репозиторий создан (рис.1.15 )(забыла сделать скриншот)

A screenshot of a computer

Description automatically generated

Рис 1.15

Через терминал перехожу в созданный каталог курса с помощью утилиты cd (рис. [1.16).](#_bookmark25)



Рис 1.16

Клонирую созданный репозиторий с помощью команды git clone –recursive git@github.com:/study\_2023–2024\_arh-pc.git arch-pc (рис. [1.17).](#_bookmark26)

A computer screen with white text

Description automatically generated

Рис 1.17

Копирую ссылку для клонирования на странице созданного репозитория, сна- чала перейдя в окно «code», далее выбрав в окне вкладку «SSH» (рис. [1.18).](#_bookmark27)

A screenshot of a computer

Description automatically generated

Рис 1.18

## Настройка каталога курса

Перехожу в каталог arch-pc с помощью утилиты cd (рис. [1.18).](#_bookmark29)



Рис 1.18

Удаляю лишние файлы с помощью утилиты rm (рис. [1.19).](#_bookmark30)



Рис 1.19

Создаю необходимые каталоги (рис. [1.20).](#_bookmark31)

A computer screen with white text

Description automatically generated

Рис 1.20

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 (рис. [1.21).](#_bookmark32)

A screenshot of a computer program

Description automatically generated

Рис 1.21

Отправляю все на сервер с помощью push (рис. [1.22).](#_bookmark33)

A computer screen with white text

Description automatically generated

Рис 1.22

Проверяю правильность выполнения работы сайте GitHub (рис. [1.23).](#_bookmark34)

A screenshot of a computer

Description automatically generated

Рис 1.23

# Выводы

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

# Список литературы

* 1. [Архитектура ЭВМ](https://esystem.rudn.ru/pluginfile.php/1584622/mod_resource/content/1/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%963.pdf)
  2. [Git - gitattributes Документация](https://git-scm.com/docs/gitattributes)