

# **Шаблон отчёта по лабораторной работе**

**Дисциплина : архитектура компьютера**

Горбунова Яна Сергеевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
3.1	Создание SSH-ключа . . . . .	10
3.2	Создание рабочего пространства и репозитория курса на основе шаблона . . . . .	10
3.3	Настройка каталога курса . . . . .	12
<b>4</b>	<b>Выводы</b>	<b>13</b>
<b>5</b>	<b>Список литературы</b>	<b>14</b>

## Список иллюстраций

3.1	Установка утилиты xclip . . . . .	10
3.2	Окно создания репозитория . . . . .	11
3.3	Созданный репозиторий . . . . .	11
3.4	Клонирование репозитория . . . . .	11
3.5	Перемещение между директориями . . . . .	12
3.6	Создание каталогов . . . . .	12
3.7	Добавление и сохранение изменений на сервере . . . . .	12

## **Список таблиц**

# 1 Цель работы

Здесь приводится формулировка цели лабораторной работы. Формулировки цели для каждой лабораторной работы приведены в методических указаниях.

Цель данного шаблона — максимально упростить подготовку отчётов по лабораторным работам. Модифицируя данный шаблон, студенты смогут без труда подготовить отчёт по лабораторным работам, а также познакомиться с основными возможностями разметки Markdown.

## 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. # 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 локальном дереве и/или ветке. После завершения внесения какого-то изменения в файлы и/или каталоги проекта необходимо разместить их в центральном репозитории.



### 3 Выполнение лабораторной работы

Создаю учетную запись на сайте GitHub . Далее я заполнила основные данные учетной записи. Аккаунт создан . Аккаунт GitHub. ?? ## 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 .

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

```
![Настройка кодировки]
```

Задаю имя «master» для начальной ветки

```
![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.

```
![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.

```
![Параметр safecrlf]
```

```
(image/2.png){ #fig:002} width=70%
```

## 3.1 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 (рис. 3.3). Ключ автоматически сохранится в каталоге `~/.ssh/`.

Генерация SSH-ключа width=70%

Xclip – утилита, позволяющая скопировать любой текст через терминал. Оказывается, в дистрибутиве Linux Kali ее сначала надо установить. Устанавливаю xclip с помощью команды `apt-get install` с ключом `-u` от имени суперпользователя, введя в начале команды `sudo`.

Установка утилиты xclip

Рис. 3.1: Установка утилиты xclip

Копирую открытый ключ из директории, в которой он был сохранен, с помощью утилиты xclip (рис. 3.1).

![Копирование содержимого файла]

Открываю браузер, захожу на сайт GitHub. Открываю свой профиль и выбираю страницу «SSH and GPG keys». Нажимаю кнопку «New SSH key» Создаю новый ключ. (рис. ??). (image/5.png){ #fig:005 width=70% }

## 3.2 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 =

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 ![Страница шаблона для репозитория]

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

Окно создания репозитория

Рис. 3.2: Окно создания репозитория

Репозиторий создан .

Созданный репозиторий

Рис. 3.3: Созданный репозиторий

Через терминал перехожу в созданный каталог курса с помощью утилиты cd

Клонирую созданный репозиторий с помощью команды git clone –recursive git@github.com:/study\_2024–2025\_arh-pc.git arch-pc (рис. 3.4). Undefined cross-reference: fig:016 Undefined cross-reference: fig:006 Undefined cross-reference: fig:006 Undefined cross-reference: fig:005

Клонирование репозитория

Рис. 3.4: Клонирование репозитория

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

### 3.3 Настройка каталога курса

Перехожу в каталог arch-рс с помощью утилиты `cd`.

Перемещение между директориями

Рис. 3.5: Перемещение между директориями

Удаляю лишние файлы с помощью утилиты `rm`

Создаю необходимые каталоги (рис. 3.5).

Создание каталогов

Рис. 3.6: 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` (рис. 3.6).

Добавление и сохранение изменений на сервере

Рис. 3.7: Добавление и сохранение изменений на сервере

Отправляю все на сервер с помощью `push`

Проверяю правильность выполнения работы сначала на самом сайте GitHub (рис. 3.3).

Страница репозитория ## Выполнение заданий для самостоятельной работы

Проверяю правильность работы на сайте Github (image/7.PNG){ #fig:013 width=70% }

## 4 Выводы

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

## 5 Список литературы

1. Архитектура ЭВМ
2. Git - gitattributes Документация