

Отчет по лабораторной работе 3

Язык разметки Markdown

Власов Артем Сергеевич

Содержание

1	Цель работы	1
2	Задание	1
3	Выполнение лабораторной работы 2.....	1
4	Ответы на контрольные вопросы.....	4
5	Выполнение лабораторной работы.....	6
6	Выводы.....	8
	Список литературы	8

1 Цель работы

Научиться оформлять отчеты с помощью легковесного языка разметки Markdown.

2 Задание

Сформировать отчет по лабораторной работе №2 с помощью Markdown.

3 Выполнение лабораторной работы 2.

Сформируем отчет лабораторной работы номер 2.

Делаем предварительную конфигурацию git. (рис. fig. 1).

```
[vlasovas@vbox ~]$ git config --global user.name "Artem Vlasov"
[vlasovas@vbox ~]$ git config --global user.email "1132246841@pfur.ru"
```

Рис. 1: Задаем имя и email репозитория

Настраиваем utf-8 в выводе сообщения git. (рис. fig. 2).

```
[vlasovas@vbox ~]$ git config --global core.quotePath false
```

Рис. 2: Настраиваем utf-8

Задаем имя начальной ветки. (рис. fig. 3).

```
[vlasovas@vbox ~]$ git config --global init.defaultBranch master
```

Рис. 3: Задаем имя начальной ветки, как master

```
[vlasovas@vbox ~]$ git config --global core.autocrlf input
```

Рис. 4: Устанавливаем настройку autocrlf

```
[vlasovas@vbox ~]$ git config --global core.safecrlf warn
```

Рис. 5: Устанавливаем параметр safecrlf

Создаем SSH ключ (рис. fig. 6).

```
[vlasovas@vbox ~]$ ssh-keygen -C "vlasovas52 11322468451@pfur.ru"
```

Рис. 6: Генерируем пару ключей командой keygen

```
[vlasovas@vbox ~]$ cat ~/.ssh/id_ed25519.pub
```

Рис. 7: Копируем ключ из локальной консоли в буфер обмена

Заходим в свой аккаунт на сайте github. Переходим в настройки, SSH ключи. (рис. fig. 8).

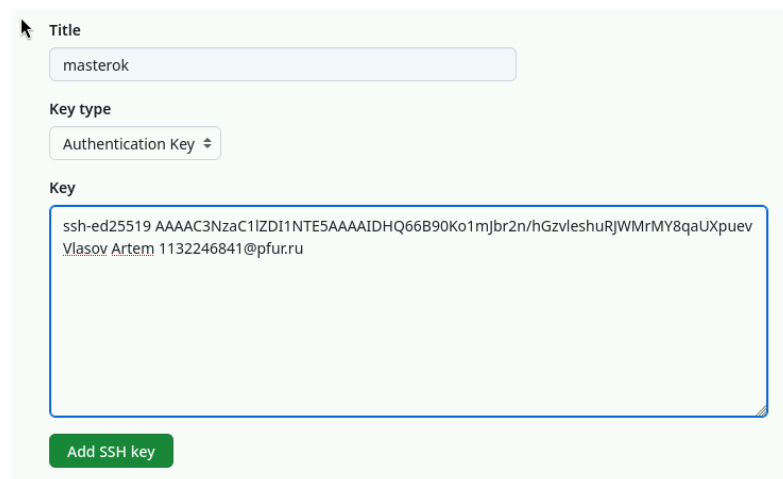


Рис. 8: Вставляем ключ и сохраняем

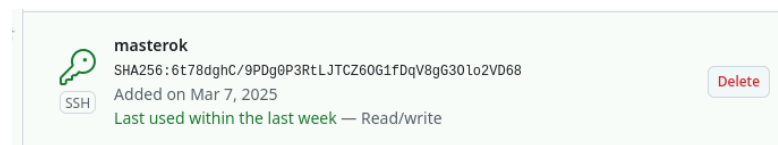


Рис. 9: Проверяем добавление ключа

Открываем терминал и создаем каталоги для предмета “Архитектура компьютера” (рис. fig. 10).

```
[vlasovas@vbox ~]$ mkdir -p ~/work/study/2024-2025/"Операционные системы"
```

Рис. 10: Создаем каталоги последовательно

Переходим на страницу репозитория с шаблоном (рис. fig. 11).

The screenshot shows the GitHub 'Create repository' page with a template selected. The template is 'yamadharma/course-directory-student-template'. Below the template name, it says 'Start your repository with a template repository's contents.' There is a checkbox for 'Include all branches' which is currently unchecked. Below this, it says 'Copy all branches from yamadharma/course-directory-student-template and not just the default branch.' The 'Owner' field is set to 'vlasovas52' and the 'Repository name' field is empty. A hint suggests 'Great repository names are short and memorable. Need inspiration? How about psychic-giggle?'. There is an optional 'Description' text area. Below this, there are two radio button options: 'Public' (selected) and 'Private'. The 'Public' option description says 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option description says 'You choose who can see and commit to this repository.' At the bottom, there is a note: 'You are creating a public repository in your personal account.' and a green 'Create repository' button.

Рис. 11: Создаем репозиторий по шаблону

Переходим в папку с предметом (рис. fig. 12).

```
[vlasovas@vbox ~]$ cd ~/work/study/2024-2025/"Операционные системы"
```

Рис. 12: Переходим в каталог курса

```
[vlasovas@vbox Операционные системы]$ git clone --recursive git@github.com:<owner>/study_2024-2025_os-intro.git os-intro
```

Рис. 13: Клонировем созданный репозиторий

Переходим в каталог arch-rc (рис. fig. 14).

```
[vlasovas@vbox Операционные системы]$ cd ~/work/study/2024-2025/"Операционные системы"/os-intro
```

Рис. 14: Переходим в нужный каталог

```
[vlasovas@vbox os-intro]$ rm package.json
```

Рис. 15: Удаляем лишние файлы

Создаем папки по образцу (рис. fig. 16).

```
[vlasovas@vbox os-intro]$ echo os-intro > COURSE
make prepare
```

Рис. 16: Создаем необходимые каталоги

Отправляем файлы на сервер (рис. fig. 17).

```
[vlasovas@vbox os-intro]$ git add .
```

Рис. 17: Отправляем файлы на git

```
[vlasovas@vbox os-intro]$ git commit -am 'feat(main): make course structure'
```

Рис. 18: Отправляем файлы на git

```
[vlasovas@vbox os-intro]$ git push
```

Рис. 19: Отправляем файлы на git

4 Ответы на контрольные вопросы

1. **Что такое системы контроля версий (VCS) и для решения каких задач они предназначены?**

Системы контроля версий (VCS) — это инструменты для управления изменениями в файлах. Они решают задачи:

- Хранения истории изменений.
 - Совместной работы над проектами.
 - Ветвления и слияния кода.
 - Отслеживания изменений и их авторов.
2. **Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.**
 - **Хранилище (репозиторий):** База данных, где хранятся все версии файлов и их история.
 - **Commit:** Фиксация изменений в репозитории. Каждый коммит сохраняет изменения и имеет уникальный идентификатор.
 - **История:** Последовательность коммитов, показывающая, как изменялись файлы.
 - **Рабочая копия:** Текущие файлы, с которыми работает разработчик, извлечённые из репозитория.
 3. **Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.**
 - **Централизованные VCS:** Один сервер хранит всю историю. Разработчики работают с локальными копиями, но для фиксации

изменений требуется подключение к серверу. Пример: **SVN (Subversion)**.

- **Децентрализованные VCS:** Каждый разработчик имеет полную копию репозитория, включая всю историю. Примеры: **Git, Mercurial**.

4. **Опишите действия с VCS при единоличной работе с хранилищем.**

- Создать репозиторий: `git init`.
- Добавить файлы в индекс: `git add <файл>`.
- Зафиксировать изменения: `git commit -m "Сообщение"`.
- Просматривать историю: `git log`.

5. **Опишите порядок работы с общим хранилищем VCS.**

- Клонировать репозиторий: `git clone`.
- Создать ветку для работы: `git branch <имя_ветки>`.
- Переключиться на ветку: `git checkout <имя_ветки>`.
- Зафиксировать изменения: `git commit -m "Сообщение"`.
- Отправить изменения на сервер: `git push`.
- Получить изменения с сервера: `git pull`.

6. **Каковы основные задачи, решаемые инструментальным средством Git?**

- Управление версиями файлов.
- Ветвление и слияние кода.
- Совместная работа над проектами.
- Отслеживание изменений и их авторов.

7. **Назовите и дайте краткую характеристику командам Git.**

- `git init`: Создать новый репозиторий.
- `git add`: Добавить файлы в индекс для последующего коммита.
- `git commit`: Зафиксировать изменения в репозитории.
- `git push`: Отправить изменения в удалённый репозиторий.
- `git pull`: Получить изменения из удалённого репозитория.
- `git branch`: Управление ветками (создание, удаление, просмотр).
- `git checkout`: Переключение между ветками или коммитами.
- `git merge`: Слияние веток.
- `git log`: Просмотр истории коммитов.

8. **Приведите примеры использования при работе с локальным и удалённым репозиториями.**

- Локальный репозиторий:

```
git init
git add .
git commit -m "Initial commit"
```

- Удалённый репозиторий:

```
git clone <URL>
git push origin main
git pull origin main
```

9. Что такое и зачем могут быть нужны ветви (branches)?

Ветви (branches) — это отдельные линии разработки в репозитории. Они нужны для:

- Параллельной работы над разными задачами.
- Изоляции экспериментальных изменений.
- Упрощения слияния изменений после завершения работы.

10. Как и зачем можно игнорировать некоторые файлы при commit?

Для игнорирования файлов используется файл `.gitignore`. В него добавляются шаблоны файлов или папок, которые не должны отслеживаться Git. Это полезно для исключения временных файлов, бинарных данных или конфиденциальной информации.

11. Что такое и зачем могут быть нужны ветви (branches)?

(Повтор вопроса 9 для полноты.) Ветви (branches) — это отдельные линии разработки в репозитории. Они нужны для:

- Параллельной работы над разными задачами.
- Изоляции экспериментальных изменений.
- Упрощения слияния изменений после завершения работы.

5 Выполнение лабораторной работы

Переходим в каталог, который привязан к репозиторию Git на сайте Github. (рис. fig. 20).

```
[vlasovas@vbox ~]$ cd ~/work/study/2024-2025/Операционные\ системы/os-intro/
[vlasovas@vbox os-intro]$
```

Рис. 20: Переходим в нужный каталог

С помощью команды `git pull` обновляем локальный репозиторий, скачивая изменения. (рис. fig. 21).

```
[vlasovas@vbox os-intro]$ git pull
Уже актуально.
```

Рис. 21: Используем команду `git pull`

Переходим в каталог report 3 лабораторной работы. (рис. fig. 22).

```
[vlasovas@vbox os-intro]$ cd ~/work/study/2024-2025/Операционные\ системы/os-intro/lab03/report/
```

Рис. 22: Переходим в следующий каталог

Используем команду `gedit report.md`, которая открывает редактор данного документа (рис. fig. 23).

```
[vlasovas@vbox report]$ gedit report.md
```

Рис. 23: Используем команду `gedit`

Изучаем открывшийся файл (рис. fig. 24).

```
## Front matter
title: "Шаблон отчёта по лабораторной работе"
subtitle: "Простейший вариант"
author: "Дмитрий Сергеевич Кулябов"

## Generic options
lang: ru-RU
toc-title: "Содержание"

## Bibliography
bibliography: bib/cite.bib
csl: pandoc/csl/gost-r-7-0-5-2008-numeric.csl

## Pdf output format
toc: true # Table of contents
toc-depth: 2
lof: true # List of figures
lot: true # List of tables
fontsize: 12pt
linestretch: 1.5
papersize: a4
documentclass: scrreprt

## I18n polyglossia
polyglossia-lang:
  name: russian
  options:
    - spelling=modern
    - babelshorthands=true
polyglossia-otherlangs:
  name: english

## I18n babel
babel-lang: russian
babel-otherlangs: english
```

Рис. 24: Изучаем документ

Изучив структуру файла, изменяем его (рис. fig. 25).

Сформируем отчет лабораторной работы номер 2.

Делаем предварительную конфигурацию git. (рис. @fig:001).

```
![Задаем имя и email репозитория](image/1.png){#fig:001 width=70%}
```

Настраиваем utf-8 в выводе сообщения git. (рис. @fig:002).

```
![Настраиваем utf-8](image/2.png){#fig:002 width=70%}
```

Задаем имя начальной ветки. (рис. @fig:003).

```
![Задаем имя начальной ветки, как master](image/3.png){#fig:003 width=70%}
```

```
![Устанавливаем настройку autocrlf](image/4.png){#fig:004 width=70%}
```

```
![Устанавливаем параметр safecrlf](image/5.png){#fig:005 width=70%}
```

Создаем SSH ключ(рис. @fig:006).

```
![Генерируем пару ключей командой keygen](image/6.png){#fig:006 width=70%}
```

```
![Копируем ключ из локальной консоли в буфер обмена](image/7.png){#fig:007 width=70%}
```

Заходим в свой аккаунт на сайте github. Переходим в настройки, SSH ключи. (рис. @fig:008).

```
![вставляем ключ и сохраняем](image/8.png){#fig:008 width=70%}
```

```
![Проверяем добавление ключа](image/9.png){#fig:009 width=70%}
```

Открываем терминал и создаем каталоги для предмета "Архитектура компьютера"(рис. @fig:010).

```
![Создаем каталоги последовательно](image/10.png){#fig:010 width=70%}
```

Рис. 25: Изменяем документ

6 Выводы

Мы познакомились с языком разметки Markdown и оформили отчет в ней и загрузили на Github.

Список литературы