

Отчёт по лабораторной работе №3

Дисциплина 'Операционные системы'

Батова Ирина Сергеевна, НММбд-01-22

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	8

Список иллюстраций

2.1	Формирования титульного листа	6
2.2	Цель работы	6
2.3	Выполнение лабораторной работы	6
2.4	Выводы	7
2.5	Контрольные вопросы	7
2.6	Неупорядоченный список	7

Список таблиц

1 Цель работы

Целью данной лабораторной работы является научиться оформлять отчёты с помощью легковесного языка разметки Markdown.

2 Выполнение лабораторной работы

Для создания отчета по лабораторной работе №2 переходим в соответствующий каталог и открываем файл 'report.md'.

В начале файла вносим номер лабораторной работы, по которой мы делаем отчет, в следующей строке указываем дисциплину “Операционные системы, в третьей строке свои ФИО с номером группы (рис. 2.1).

```
-----  
title: "Отчёт по лабораторной работе №2"  
subtitle: "Дисциплина: Операционные системы"  
author: "Батова Ирина Сергеевна, НММбд-01-22"
```

Рис. 2.1: Формирования титульного листа

Далее под заголовком “Цель работы” обозначаем нашу цель (рис. 2.2).

```
# Цель работы  
Целью данной лабораторной работы является изучение идеологии и применение средств контроля версий, а также освоение умения по работе с git.
```

Рис. 2.2: Цель работы

Далее под заголовком “Выполнение лабораторной работы” пошагово описываем наши действия, делая ссылки на изображения и вставляя их в файл (рис. 2.3).

```
Первым делом устанавливаем git (рис. @fig:001).  
![[Установка git](image/1.jpg){*fig:001 width=70%}]
```

Рис. 2.3: Выполнение лабораторной работы

Важно проверить, что все необходимые изображения лежат в соответствующей папке ‘image’.

После описания выполнения лабораторной работы под заголовком “Выводы” пишем вывод по данной лабораторной работе, который согласуется с целью работы (рис. 2.4).

Выводы

В данной лабораторной работе мной были изучены идеологии и применение средств контроля версий, а также освоены умения по работе с [git](#).

Рис. 2.4: Выводы

Последним шагом создаем заголовок “Контрольные вопросы” и, нумеруя их, отвечаем на поставленные в лабораторной работе вопросы (рис. 2.5).

Контрольные вопросы

1. Система контроля версий ([VCS](#)) – это система, регистрирующая изменения в файлах. [VCS](#) может хранить несколько версий одного документа, позволяет возвращаться к более ранним версиям, показывает, кто и какие конкретно изменения внес в документ. Система контроля версий обычно применяется в случае, если над одним проектом работает группа людей.
2. При выполнении участником проекта своей части работы он получает нужную ему версию файлов из хранилища, а затем сохраняет новую версию файлов в хранилище. То есть централизованное хранилище сохраняет все файлы – и до правки участником, и после. Следующий пользователь перед работой начнет работу с новой, измененной версией файла. Команда [commit](#) как раз осуществляет сохранение изменений (но при этом на сервер они уходят с помощью другой команды). История проекта – это история изменения файлов, то есть кто изменял, какие файлы, какие изменения были внесены. Рабочую версию участник проекта всегда извлекает перед началом работы – то есть, рабочей версией называется та, в которой сохранены все необходимые для работы конкретного пользователя изменения.
3. Централизованные [VCS](#) созданы для [бэкапирования](#), отслеживания и синхронизации файлов. Все изменения происходят через центральный сервер. А в децентрализованных [VCS](#) у каждого пользователя есть свой полноценный [репозиторий](#) и нет жестко заданной структуры [репозитория](#) с центральным сервером. Децентрализованные [VCS](#) были созданы для обмена изменениями, например, [git](#). Пример централизованной [VCS](#) – [Subversion \(SVN\)](#).

Рис. 2.5: Контрольные вопросы

Для создания неупорядоченного списка используем символ ‘*’ (рис. 2.6).

7. Основные команды [git](#):

- [git init](#) – создание основного дерева [репозитория](#)
- [git pull](#) – получение обновлений (изменений) текущего дерева из центрального [репозитория](#)
- [git push](#) – отправка всех произведённых изменений локального дерева в центральный [репозиторий](#)
- [git status](#) – просмотр списка изменённых файлов в текущей директории
- [git diff](#) – просмотр текущих изменений
- [git add](#) . - добавление всех изменённых и/или созданных файлов и/или каталогов
- [git add](#) имена_файлов - добавление конкретных изменённых и/или созданных файлов и/или каталогов
- [git rm](#) имена_файлов - удаление файла и/или каталога из индекса [репозитория](#) (при этом файл и/или каталог остаётся в локальной директории)
- [git commit -m 'Описание коммита'](#) - сохранение всех добавленных изменений и всех изменённых файлов
- [git commit](#) - сохранение добавленных изменений с внесением комментария через встроенный редактор

Рис. 2.6: Неупорядоченный список

3 Выводы

В данной лабораторной работе я научилась оформлять отчёты с помощью легковесного языка разметки Markdown.