

# Лабораторная работа №3

Операционные системы

---

Мазурский А. Д.

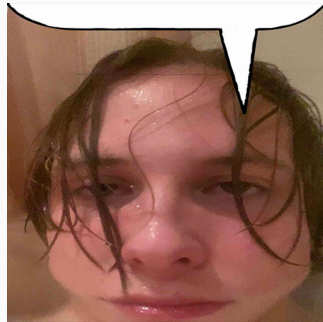
06 марта 2025

Российский университет дружбы народов, Москва, Россия

## Информация

---

- Мазурксий Александр Дмитриевич
- Студент НКАбд-02-24
- я саша
- Российский университет дружбы народов
- 1132242468@pfur.ru



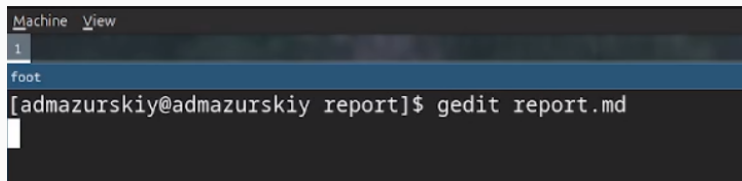
Научиться оформлять отчёты с помощью легковесного языка разметки Markdown.

1. Сделайте отчёт по предыдущей лабораторной работе в формате Markdown.
2. В качестве отчёта просьба предоставить отчёты в 3 форматах: pdf, docx и md (в архиве, поскольку он должен содержать скриншоты, Makefile и т.д.)

Чтобы создать заголовок, используйте знак ( # ). Чтобы задать для текста полужирное начертание, заключите его в двойные звездочки. Чтобы задать для текста курсивное начертание, заключите его в одинарные звездочки. Чтобы задать для текста полужирное и курсивное начертание, заключите его в тройные звездочки. Блоки цитирования создаются с помощью символа >. Неупорядоченный (маркированный) список можно отформатировать с помощью звездочек или тире. Чтобы вложить один список в другой, добавьте отступ для элементов дочернего списка. Упорядоченный список можно отформатировать с помощью соответствующих цифр. Чтобы вложить один список в другой, добавьте отступ для элементов дочернего списка. Синтаксис Markdown для встроенной ссылки состоит из части [link text] , представляющей текст гиперссылки, и части (file-name.md) – URL-адреса или имени файла, на который дается ссылка.

Markdown поддерживает как встраивание фрагментов кода в предложение, так и их размещение между предложениями в виде отдельных огражденных блоков. Огражденные блоки кода — это простой способ выделить синтаксис для фрагментов кода. Внутритекстовые формулы делаются аналогично формулам LaTeX. Для обработки файлов в формате Markdown будем использовать Pandoc. Конкретно, нам понадобится программа `pandoc`, `pandoc-citeproc` <https://github.com/jgm/pandoc/releases>, `pandoc-crossref` <https://github.com/lierdakil/pandoc-crossref/releases>. Преобразовать файл README.md можно следующим образом: `1 pandoc README.md -o README.pdf` или так `1 pandoc README.md -o README.docx` Можно использовать следующий Makefile `1 FILES = $(patsubst %.md, %.docx, $(wildcard *.md)) 2 FILES += $(patsubst %.md, %.pdf, $(wildcard *.md))`

В рабочей директории курса открываю через текстовый редактор файл с шаблоном отчета.



```
Machine View
1
foot
[admazurskiy@admazurskiy report]$ gedit report.md
```

Рис. 1: Редактирование отчета



Указываю основную информацию о лабораторной работе.

```
1 ---  
2 ## Front matter  
3 title: "Лабораторная работа №2"  
4 subtitle: "дисциплина: Архитектура компьютера"  
5 author: "Мазурский Александр Дмитриевич"  
6
```

Рис. 2: Заполнение основной информации

Формирую цель работы, задание и заполняю теоретическое введение.

```
111
112
113 Цель работы
114
115 Целью данной работы является изучение принципов и применения средств контроля версий и повышение уровня по работе с git.
116
117 Задачи
118
119 - изучить конфигурации для работы с git.
120 - изучить SSH.
121 - изучить PaaS.
122 - изучить git.
123 - настроить git.
124 - локальный каталог для выполнения заданий по проекту.
125
126 Теоретическое введение
127
128 Системы контроля версий (Version Control System, VCS) применяются при работе несколькими людьми над одним проектом. Обычно основное действие проекта хранится в локальном или удаленном репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий сохраняет их как файлы, создавая коммиты, позволяющие различать изменения проекта, позволяя откат к любой более ранней версии проекта, если это требуется.
129
130 В классической системе контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Внесение большинства функций по управлению версиями осуществляется специальным сервером. Участники проекта (используя) перед началом работы посредством терминальной команды получают копию нужных файлов. После внесения изменений, пользователи отправят свою версию в репозиторий. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не только версии изменений файлов, а также создавать так называемые дельта-коммиты – сохранять только изменения между последовательными версиями, что позволяет уменьшить объем хранимых данных.
```

Рис. 3: Цель работы, задание и теоретическое введение

Описываю процесс выполнения лабораторной работы.

```
# Выполнение лабораторной работы

Произвожу базовую настройку git. (рис. -@fig:881)

! [Пример конфигурации git] (image/1.png) {#fig:881 width=78%}

Создаю ssh и gpg ключи. (рис. -@fig:882)

! [Генерация ключей] (image/2.png) {#fig:882 width=78%}

Экспортирую gpg ключ для авторизации на github. (рис. -@fig:883)

! [Экспорт ключа] (image/3.png) {#fig:883 width=78%}

Настраиваю автоматические подписи для коммитов. (рис. -@fig:884)

! [Конфигурация подписей для коммитов] (image/4.png) {#fig:884 width=78%}

Авторизуюсь на github для работы через терминал. (рис. -@fig:885)

! [Авторизация на github] (image/5.png) {#fig:885 width=78%}

Создаю директорию курса по шаблону (рис. -@fig:886)

! [Создание директории курса] (image/6.png) {#fig:886 width=78%}

Настраиваю рабочую директорию (рис. -@fig:887)

! [Настройка директории] (image/7.png) {#fig:887 width=78%}
```

Рис. 4: Выполнение отчета по лабораторной работе

В ходе выполнения данной лабораторной работы я научился оформлять отчёты с помощью легковесного языка разметки Markdown.