

Лабораторная работа №3

Markdown

Беличева Д.М.

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
5	Выводы	11
	Список литературы	12

Список иллюстраций

4.1	Изменение некоторых данных в отчете	7
4.2	Описание цели, теоретического введения и задания	8
4.3	Добавление картинок и ссылок на них	9
4.4	Список литературы	10
4.5	Пример ссылки на источник	10

1 Цель работы

Научиться оформлять отчёты с помощью легковесного языка разметки Markdown.

2 Задание

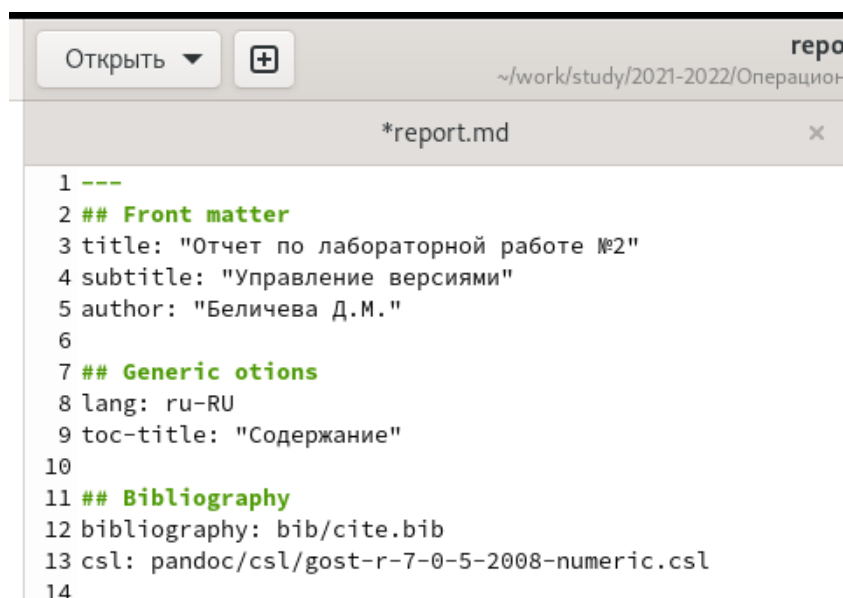
Сделант отчёт по предыдущей лабораторной работе в формате Markdown.

3 Теоретическое введение

Markdown — это облегченный язык разметки с синтаксисом форматирования обычного текста. созданный Джоном Грубером и Аароном Шварцем в 2004 году, сегодня это один из самых популярных языков среди программистов. Для записи Markdown можно использовать любой текстовый редактор. Смысл маркдауна в том, что вы делаете разметку своего документа минимальными усилиями, а уже какой-то другой плагин или программа превращает вашу разметку в итоговый документ — например в HTML. Но можно и не в HTML, а в PDF или что-нибудь ещё. [1]

4 Выполнение лабораторной работы

№1 Открыли шаблон лабораторной работы. Изменили необходимые данные в соответствии с нашей второй лабораторной работой. Изменили название и автора.(рис. 4.1)



```
1 ---
2 ## Front matter
3 title: "Отчет по лабораторной работе №2"
4 subtitle: "Управление версиями"
5 author: "Беличева Д.М."
6
7 ## Generic options
8 lang: ru-RU
9 toc-title: "Содержание"
10
11 ## Bibliography
12 bibliography: bib/cite.bib
13 csl: pandoc/csl/gost-r-7-0-5-2008-numeric.csl
14
```

Рис. 4.1: Изменение некоторых данных в отчете

Поменяли цель, теоретическое введение и задания на нужные. (рис. 4.2)

```

68
69 # Цель работы
70
71 - Изучить идеологию и применение средств контроля версий.
72 - Освоить умения по работе с git.
73
74 # Задание
75
76 1. Зарегистрироваться на GitHub;
77 2. Создать базовую конфигурацию для работы с git;
78 3. Создать ключ SSH;
79 4. Создать ключ PGP;
80 5. Настроить подписи git;
81 6. Создать локальный каталог для выполнения заданий по предмету.
82
83 # Теоретическое введение
84
85 В этой лабораторной работе мы познакомимся с системами контроля версий. 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, производённые
разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется. Существуют
классические и распределённые системы контроля версий (РСКВ). Сегодня мы будем работать с распределённой VSC – Git
86 В РСКВ (таких как Git, Mercurial, Bazaar или Darcs) клиенты не просто скачивают снимок всех файлов – они
полностью копируют репозиторий. В этом случае, если один из серверов, через который разработчики обменивались
данными, умрёт, любой клиентский репозиторий может быть скопирован на другой сервер для продолжения работы. Каждая
копия репозитория является полным бэкапом всех данных.
87 Более того, многие РСКВ могут одновременно взаимодействовать с несколькими удалёнными репозиториями,
благодаря этому вы можете работать с различными группами людей, применяя различные подходы одновременно в рамках
одного проекта. Это позволяет применять сразу несколько подходов в разработке, например, иерархические модели, что
совершенно невозможно в централизованных системах. [gnu-doc:bash]
88
89 # Выполнение лабораторной работы
90
91 1.
92 Создаем учетную запись на Github и заполняем основные данные. (рис. [-@fig:001])
93

```

Рис. 4.2: Описание цели, теоретического введения и задания

№2 Поместили в папку “image” картинки, которые будем использовать. Сделали ссылки на картинки. (рис. 4.3)


```

-
} # Выполнение лабораторной работы
}
1.
? Создаем учетную запись на Github и заполняем основные данные. (рис. [-@fig:001])
}
! [Создание учетной записи на GitHub](image/рис.1.png) { #fig:001 width=70% }
}
2.
? Далее установим программное обеспечение git-flow в Fedora Linux (сделаем это вручн:
}
! [Установка git-flow в Fedora Linux](image/рис.2.png) { #fig:002 width=70% }
}
? Установим gh в Fedora Linux. (рис. [-@fig:003])
}
! [Установка gh в Fedora Linux](image/рис.3.png) { #fig:003 width=70% }
}
? Перейдем к базовой настройке Git: зададим имя и почту владельца репозитория; настроим верификацию и подписание коммитов git (Зададим имя начальной ветки (будем autocrlf, параметр safecrlf). (рис. [-@fig:004])
}
! [Базовая настройка git](image/рис.4.png) { #fig:004 width=70% }
}
3.
? Создаем ключ ssh: по алгоритму rsa с ключём размером 4096 бит: (рис. [-@fig:005])
}
! [Создания ключа ssh по алгоритму rsa с ключем размером 4096](image/рис.5.png) { #fig:005 width=70% }
}
? по алгоритму ed25519: (рис. [-@fig:006])
}
! [Создания ключа ssh по алгоритму ed25519](image/рис.6.png) { #fig:006 width=70% }
}
4.
? Создаем ключ gpg. Генерируем ключ и из предложенных опций выбираем:
}
• Тип RSA and RSA;
}
• Размер 4096;
}
• Выберите срок действия; значение по умолчанию– 0 (срок действия не истекает )
}
• GPG запросит личную информацию, которая сохранится в ключе:
}

```

Рис. 4.3: Добавление картинок и ссылок на них

№3 Создадим список литературы. Для этого сначала в папке “bib” откроем файл “cite.bib” и в нем создадим по шаблону необходимые нам литературные источники. (рис. 4.4)

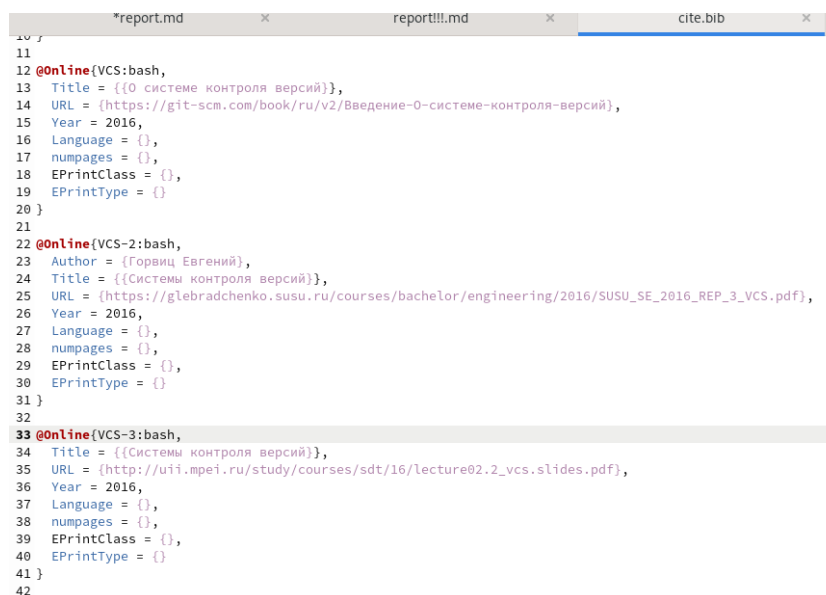


Рис. 4.4: Список литературы

Добавим ссылки на источники. (рис. 4.5)

```

/
8      8. Приведите примеры использования при работе с
9 git push -all (push origin master/любой branch)
9
1     9. Что такое и зачем могут быть нужны ветви (branch)
2 Ветвление («ветка», branch) – один из параллельных путей
   одной версии (точки ветвления). [@VCS-3:bash]
3 • Обычно есть главная ветка (master), или ствол (trunk).
4 • Между ветками, то есть их концами, возможно слияние.
5 Используются для разработки новых функций.
5
7     10. Как и зачем можно игнорировать некоторые файлы
8 Во время работы над проектом так или иначе могут создаваться
   последствия в репозиторий. Например, временные файлы,
   создаваемые компилятором. Можно проигнорировать следующие:

```

Рис. 4.5: Пример ссылки на источник

5 Выводы

В процессе выполнения этой лабораторной работы я научилась работать с языком разметки Markdown. Познакомилась с базовым синтаксисом Markdown.

Список литературы

1. Справочник по Docs Markdown [Электронный ресурс]. Free Software Foundation. URL: <https://docs.microsoft.com/ru-ru/contribute/markdown-reference>.