Отчёт по лабораторной работе №3

Операционные системы

Самсоноват Мария Ильинична

Содержание

Цель работы 1
Задание 1
Теоретическое введение 1
Базовые сведения о Markdown 1
Выполнение лабораторной работы 2
Вывод 4

Цель работы

Научиться оформлять отчёты с помощью легковесного языка разметки Markdown.

Задание

Составить отчёт по выполнению лабораторной работы №2.

Теоретическое введение

Базовые сведения о Markdown

Чтобы создать заголовок, используйте знак (#) и уберите кавычки, например: “# This is heading 1” “## This is heading 2” “### This is heading 3” “#### This is heading 4”

Чтобы задать для текста полужирное начертание, заключите его в двойные звездочки и уберите пробелы между символами и звездочками: This text is ** bold **.

Чтобы задать для текста курсивное начертание, заключите его в одинарные звездочки и уберите пробелы между символами и звездочками: This text is * italic *.

Чтобы задать для текста полужирное и курсивное начертание, заключите его в тройные звездочки и уберите пробелы между символами и звездочками: 1 This is text is both *** bold and italic ***.

Блоки цитирования создаются с помощью символа > (уберите кавычки):

“>” The drought had lasted now for ten million years, and the reign of the terrible lizards had long since ended. Here on the Equator, in the continent which would one day be known as Africa, the battle for existence had reached a new climax of ferocity, and the victor was not yet in sight. In this barren and desiccated land, only the small or the swift or the fierce could flourish, or even hope to survive.

Неупорядоченный (маркированный) список можно отформатировать с помощью звездочек или тире (поставьте пробел между тире/звездой и заголовком):

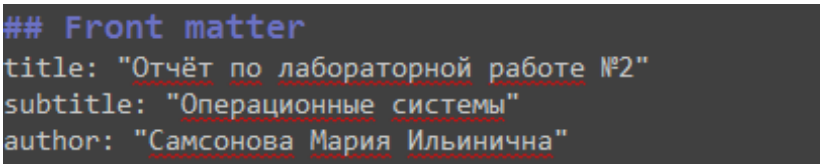
-List item 1

-List item 2

-List item 3

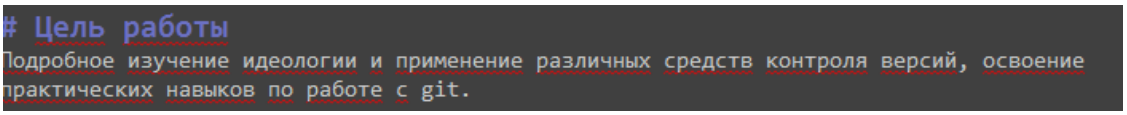
Выполнение лабораторной работы

1. Указание данных о работе (название работы, название дисциплины, по которой выполняется работа, ФИО выполняющего): (рис. [-@fig:001])



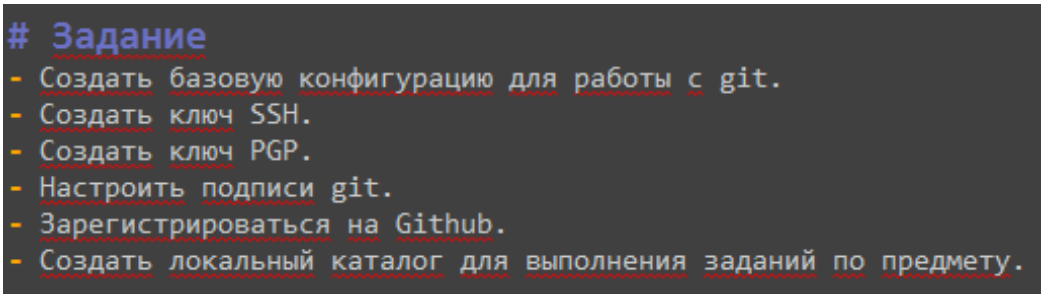
{ #fig:001 width=70% }

2. Указание цели лабораторной работы №2 с помощью заглавия, начинающегося с # [Название_заглавия]: (рис. [-@fig:002])



{ #fig:002 width=70% }

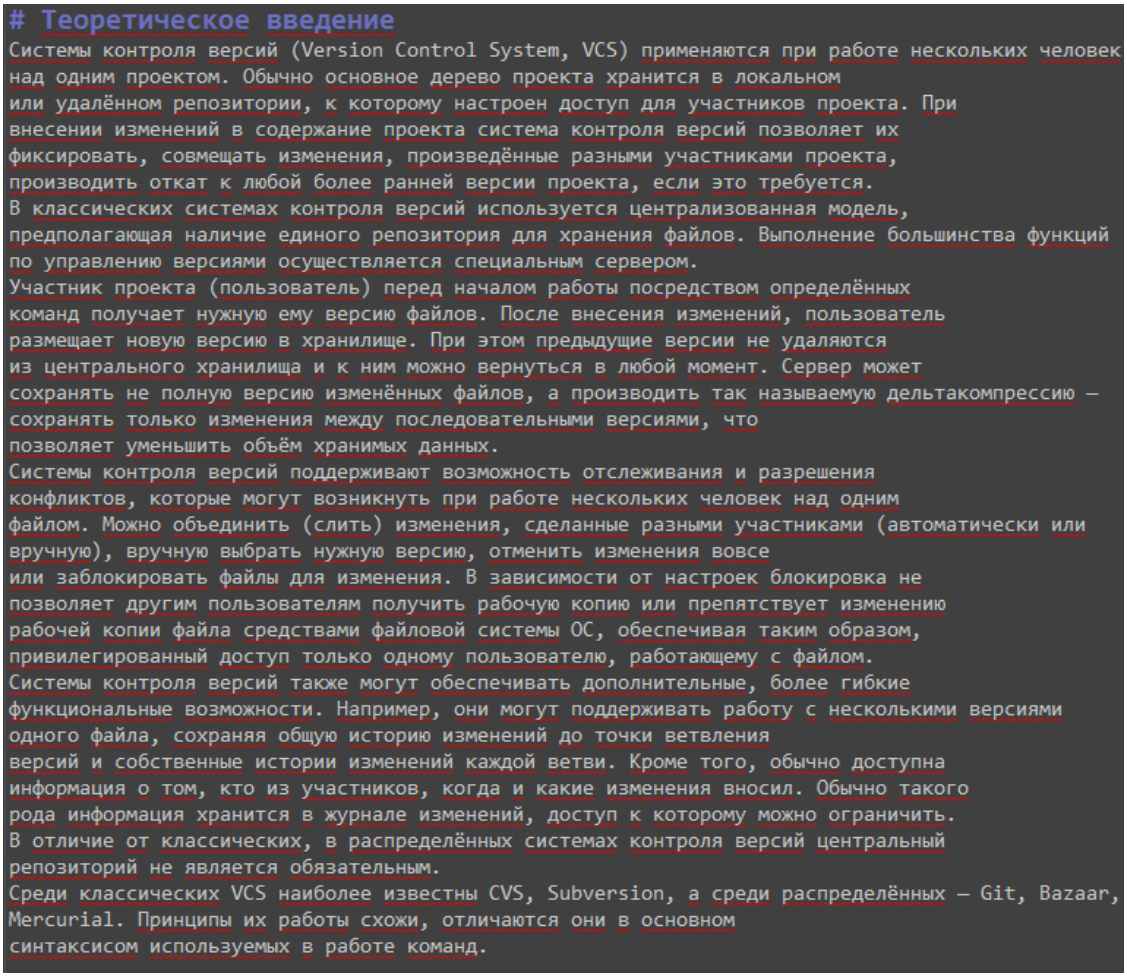
3. Указание заданий, которые требовалось выполнить в лабораторной работе №2, с помощью заглавия, начинающегося с # [Название_заглавия], и тире, которые помогают перечислить определённые пункты заданий: (рис. [-@fig:003])



Указание заданий лабораторной работы

{ #fig:003 width=70% }

4. Указание теоретического введения лабораторной работы №2 с помощью заглавия, начинающегося с # [Название_заглавия]:(рис. [-@fig:004])



```
# 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 дельтакомпрессию —
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 (слить) изменения, сделанные разными участниками (автоматически или
вручную), 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. Кроме того, 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.
```

Указание теоретического введения лабораторной работы

{ #fig:004 width=70% }

5. Прописывание хода выполнения лабораторной работы №2 с помощью заглавия, начинающегося с # [Название_заглавия], нумерации определённой цифрой и точкой и добавления скринов с помощью данного шаблона без кавычек: (рис. [-@fig:005])

Шаблон: **[“(рис. [-@fig:003])[Указание заданий лабораторной работы]”(название_папки/номер_скрина.png)“{ #fig:003 width=70% }”]**

```
# Выполнение лабораторной работы

1. Создаём свою учётную запись на https://github.com: (рис. [-@fig:001])

![Учетная запись на github](images/1.png){ #fig:001 width=70% }

2.  Задаём имя и email владельца репозитория данными командами  git config --global user.name "Name Surname" и git config --global user.email "work@mail", настраиваем utf-8 в выводе сообщений git командой git config --global core.quotepath false, а также задаём имя начальной ветки (будем называть её master):
(рис. [-@fig:002])

![Задание имени и email пользователя, настройка utf-8 в выводе сообщений, задание имени начальной ветки](images/2.png)

{#fig:002 width=70% }

3. Создаём ключей ssh по алгоритму rsa размером 4096 бит и по алгоритму ed25519: (рис. [-@fig:003])

![Создание ключа ssh по алгоритму rsa с ключём размером 4096 бит](images/3.png)

{#fig:003 width=70% }

(рис. [-@fig:004])
```

Прописывание хода выполнения лабораторной работы

{ #fig:005 width=70% }

6. Указание вывода лабораторной работы №2 с помощью заглавия, начинающегося с # [Название_заглавия]: (рис. [-@fig:006])

```
# Вывод

В процессе выполнения данной лабораторной работы мы приобрели практические навыки применения средств контроля версий и научились работать  с git.
```

Указание вывода лабораторной работы

{ #fig:006 width=70% }

Вывод

В процессе выполнения лабораторной работы №3 мы научились оформлять отчёты с помощью легковесного языка разметки Markdown.