

# **Отчёт по лабораторной работе №3**

**дисциплина: Операционные системы**

Подъярова Ксения Витальевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
3.1	Базовые сведения о Markdown . . . . .	7
3.2	Структура отчета по лабораторной работе . . . . .	7
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Ответы на контрольные вопросы</b>	<b>11</b>
<b>6</b>	<b>Выводы</b>	<b>15</b>

## Список иллюстраций

4.1	Создание учётной записи . . . . .	8
4.2	Создание ключа ssh . . . . .	8
4.3	Добавление ключа ssh в Github . . . . .	9
4.4	Создание ключа pgr . . . . .	9
4.5	Добавление ключа pgr в Github . . . . .	9
4.6	Создание и подключение репозитория . . . . .	10
4.7	Настраивание каталога курса . . . . .	10
4.8	Добавление коммита . . . . .	10
4.9	Сохранение коммита . . . . .	10

## Список таблиц

# 1 Цель работы

Научиться оформлять отчёты с помощью легковесного языка разметки Markdown.

## 2 Задание

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

## **3 Теоретическое введение**

### **3.1 Базовые сведения о Markdown**

Чтобы создать заголовок, нужно использовать знак ( # ).

Неупорядоченный (маркированный) список можно отформатировать с помощью звездочек или тире.

Упорядоченный список можно отформатировать с помощью соответствующих цифр.

### **3.2 Структура отчета по лабораторной работе**

Согласно ГОСТ 7.32-2001, любая научно-исследовательская работа должна обязательно содержать следующие элементы:

- титульный лист;
- реферат;
- введение;
- основную часть;
- заключение.

## 4 Выполнение лабораторной работы

1. Создаем учётную запись на <https://github.com>. Учетная запись называется kvpodjhyarova. (Рис. 4.1)

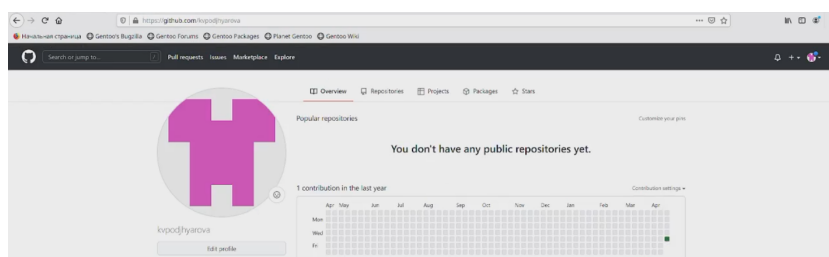


Рис. 4.1: Создание учётной записи

2. Настраиваем систему контроля версий git. Синхронизируем учётную запись github с компьютером: `git config --global user.name "kvpodjhyarova"` `git config --global user.email "ksu.pod.vit@yandex.ru"` Затем создаём новый ключ ssh (Рис. 4.2) и добавляем его в GitHub (Рис.4.3)

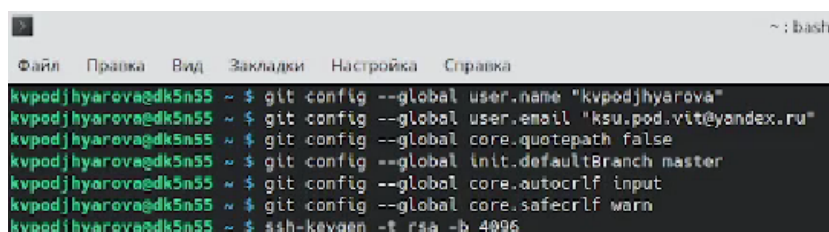



Рис. 4.2: Создание ключа ssh



## SSH keys

[New SSH key](#)

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

 **dk**  
SHA256:yqgq46x8GF8Bs5KhsNrTnK2rDe0BYa4U+D18ZnZQr0w  
Added on 21 Apr 2022  
Last used within the last week — Read/write

Delete

Рис. 4.3: Добавление ключа ssh в Github

3. Создаем ключи `pgp` и добавляем в GitHub. Для этого выводим список ключей, копируем отпечаток приватного ключа в буфер обмена (Рис.4.4), переходим в настройки GitHub, нажимаем на кнопку `New GPG key` и вставляем полученный ключ в поле ввода (Рис.4.5).

```
kvpodjhyarova@dk5n55 ~ $ gpg --full-generate-key
gpg (GnuPG) 2.2.27; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.


Выберите тип ключа:
(1) RSA и RSA (по умолчанию)
(2) DSA и Elgamal
(3) DSA (только для подписи)
(4) RSA (только для подписи)
(14) Имеющийся на карте ключ
Ваш выбор? 1
длина ключей RSA может быть от 1024 до 4096.
Какой размер ключа Вам необходим? (3072) 4096
Запрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
0 - не ограничен
<n> = срок действия ключа - n дней
<n>w = срок действия ключа - n недель
<n>m = срок действия ключа - n месяцев
<n>y = срок действия ключа - n лет
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (y/N) y
```

Рис. 4.4: Создание ключа `pgp`

## GPG keys

[New GPG key](#)

This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.

 **GPG Email address:** ...  
ksu.pod.vit@yandex.ru Key ID: 68CFCE43C2EA3989  
Subkeys: AC04A932FD7C3053  
Added on 21 Apr 2022

Delete

Рис. 4.5: Добавление ключа `pgp` в Github

#### 4. Создаем и подключаем репозиторий к github. (Рис.4.6)

```
kvpodjhyarova@dk5n55 ~$ mkdir -p ~/work/study/2021-2022/Операционные системы
kvpodjhyarova@dk5n55 ~$ git clone --recursive https://github.com/kvpodjhyarova/study_2021-2022_os-intro.git os-intro
Клонирование в os-intro...
remote: Enumerating objects: 28, done.
remote: Counting objects: 100% (28/28), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 28 (delta 2), reused 15 (delta 2), pack-reused 0
Получение объектов: 100% (28/28), 12.49 Kib | 0.25 Mib/s, готово.
Определение изменений: 100% (2/2), готово.
Получаю <template/presentation> (https://github.com/yamadharma/academic-presentation-markdown-template.git) записан по пути <template/presentation>
Получаю <template/report> (https://github.com/yamadharma/academic-laboratory-report-template.git) записан по пути <template/report>
Клонирование в os-intro...
remote: Enumerating objects: 42, done.
remote: Counting objects: 100% (42/42), done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 42 (delta 9), reused 40 (delta 7), pack-reused 0
Получение объектов: 100% (42/42), 31.19 Kib | 1.01 Mib/s, готово.
Определение изменений: 100% (9/9), готово.
```

Рис. 4.6: Создание и подключение репозитория

#### 5. Настраиваем каталог курса: переходим в него, удаляем лишние файлы, создаем необходимые каталоги (Рис.4.7)

```
kvpodjhyarova@dk5n55 ~$ cd ~/work/study/2021-2022/Операционные системы/os-intro
bash: cd: /afs/dk.scl.pfu.edu.ru/home/k/v/kvpodjhyarova/work/study/2021-2022/Операционные системы/os-intro: Нет такого файла или каталога
kvpodjhyarova@dk5n55 ~$ cd ~/work/study/2021-2022/Операционные системы/os-intro
kvpodjhyarova@dk5n55 ~/work/study/2021-2022/Операционные системы/os-intro $ rm package.json
kvpodjhyarova@dk5n55 ~/work/study/2021-2022/Операционные системы/os-intro $ make COURSE=os-intro
kvpodjhyarova@dk5n55 ~/work/study/2021-2022/Операционные системы/os-intro $ git add .
kvpodjhyarova@dk5n55 ~/work/study/2021-2022/Операционные системы/os-intro $ git commit -am "feat(main):make course structure"
```

Рис. 4.7: Настройка каталога курса

#### 6. Добавляем первый коммит и выкладываем на GitHub. Для того, чтобы правильно разместить первый коммит, необходимо добавить команду git add ., после этого с помощью команды git commit -am выкладываем коммит. Сохраняем коммит, используя команду git push. (Рис.4.8) (Рис.4.9)

```
kvpodjhyarova@dk5n55 ~$ cd ~/work/study/2021-2022/Операционные системы/os-intro
bash: cd: /afs/dk.scl.pfu.edu.ru/home/k/v/kvpodjhyarova/work/study/2021-2022/Операционные системы/os-intro: Нет такого файла или каталога
kvpodjhyarova@dk5n55 ~$ cd ~/work/study/2021-2022/Операционные системы/os-intro
kvpodjhyarova@dk5n55 ~/work/study/2021-2022/Операционные системы/os-intro $ rm package.json
kvpodjhyarova@dk5n55 ~/work/study/2021-2022/Операционные системы/os-intro $ make COURSE=os-intro
kvpodjhyarova@dk5n55 ~/work/study/2021-2022/Операционные системы/os-intro $ git add .
kvpodjhyarova@dk5n55 ~/work/study/2021-2022/Операционные системы/os-intro $ git commit -am "feat(main):make course structure"
```

Рис. 4.8: Добавление коммита

```
create mode 100644 project-personal/stage5/presentation/Makefile
create mode 100644 project-personal/stage5/presentation/presentation.md
create mode 100644 project-personal/stage5/report/Makefile
create mode 100644 project-personal/stage5/report/bib/cite.bib
create mode 100644 project-personal/stage5/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage5/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 project-personal/stage5/report/report.md
create mode 100644 project-personal/stage6/presentation/Makefile
create mode 100644 project-personal/stage6/presentation/presentation.md
create mode 100644 project-personal/stage6/report/Makefile
create mode 100644 project-personal/stage6/report/bib/cite.bib
create mode 100644 project-personal/stage6/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage6/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 project-personal/stage6/report/report.md
create mode 100644 structure
kvpodjhyarova@dk5n55 ~/work/study/2021-2022/Операционные системы/os-intro $ git push
```

Рис. 4.9: Сохранение коммита

## 5 Ответы на контрольные вопросы

1. Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` с различными опциями. Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом.
2. 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 неполную версию изменённых файлов, а производить так называемую дельта-компрессию—сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. 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.

3. Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Пример - Wikipedia. В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. Пример — Bitcoin. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером.
4. Создадим локальный репозиторий. Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория: `git config --global user.name "Имя Фамилия"` `git config --global user.email "work@mail"` и настроив utf-8 в выводе сообщений `git: git config --global quotepath false` Для инициализации локального репозитория, расположенного, например, в каталоге `~/tutorial`, необходимо ввести в командной строке: `cd mkdir tutorial cd tutorial git init`
5. Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый): `ssh-keygen -C "Имя Фамилия work@mail"` Ключи сохраняются в каталоге `~/.ssh/`. Скопировав из локальной консоли ключ в буфер обмена `cat ~/.ssh/id_rsa.pub | xclip -sel clip` вставляем ключ в появившееся на сайте поле.
6. У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.
7. Основные команды git: Наиболее часто используемые команды git: – созда-

ние основного дерева репозитория :git init–получение обновлений (изменений) текущего дерева из центрального репозитория: git pull–отправка всех произведённых изменений локального дерева в центральный репозиторий:git push–просмотр списка изменённых файлов в текущей директории: git status–просмотр текущих изменения: git diff–сохранение текущих изменений:–добавить все изменённые и/или созданные файлы и/или каталоги: git add .–добавить конкретные изменённые и/или созданные файлы и/или каталоги: git add имена\_файлов – удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): git rm имена\_файлов – сохранение добавленных изменений: – сохранить все добавленные изменения и все изменённые файлы: git commit -am ‘Описание коммита’–сохранить добавленные изменения с внесением комментария через встроенный редактор: git commit–создание новой ветки, базирующейся на текущей: git checkout -b имя\_ветки–переключение на некоторую ветку: git checkout имя\_ветки (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой) – отправка изменений конкретной ветки в центральный репозиторий: git push origin имя\_ветки–слияние ветки стекущим деревом:git merge –no-ff имя\_ветки–удаление ветки: – удаление локальной уже слитой с основным деревом ветки:git branch -d имя\_ветки–принудительное удаление локальной ветки: git branch -D имя\_ветки–удаление ветки с центрального репозитория: git push origin :имя\_ветки

8. Исползования git при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий): git add hello.txt git commit -am ‘Новый файл’
9. Проблемы, которые решают ветки git: • нужно постоянно создавать архивы с рабочим кодом • сложно “переключаться” между архивами • сложно перетаскивать изменения между архивами • легко что-то напутать или

потерять

10. Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл.gitignore с помощью сервисов. Для этого сначала нужно получить списки меняющихся шаблонов: `curl -L -s https://www.gitignore.io/api/list` Затем скачать шаблон, например, для С и С++ `curl -L -s https://www.gitignore.io/api/c » .gitignore` `curl -L -s https://www.gitignore.io/api/c++ » .gitignore`

## 6 Выводы

Я научилась оформлять отчёты с помощью легковесного языка разметки Markdown.