

# Отчет по лабораторной работе №2

Ким Илья Владиславович

# Содержание

Цель работы	4
Задание	5
Выполнение лабораторной работы	6
Ответы на вопросы	13
Выводы	17

# Список иллюстраций

0.1	Настройка VCS git . . . . .	6
0.2	SSH ключ . . . . .	7
0.3	Структура каталога . . . . .	7
0.4	Учебный репозиторий . . . . .	8
0.5	Репозиторий на GitHub . . . . .	9
0.6	Файл LECENSE . . . . .	9
0.7	Списки шаблонов . . . . .	9
0.8	Шаблон . . . . .	10
0.9	Отправка файлов на GitHub . . . . .	10
0.10	Репозиторий на GitHub . . . . .	11
0.11	Измененный файл на GitHub . . . . .	12

## Цель работы

Изучить идеологию и применение средств контроля версий.

# 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.)

# Выполнение лабораторной работы

1. Установка необходимых программ (Chocolatey, git Bash).
2. Настроил систему контроля версий git.

```
15062021@DESKTOP-SJP6EQH MINGW64 ~  
$ git config --global user.name "Kim Ilya"  
  
15062021@DESKTOP-SJP6EQH MINGW64 ~  
$ git config --global user.email "work@mail"  
  
15062021@DESKTOP-SJP6EQH MINGW64 ~  
$ git config --global user.email "work@mail"  
  
15062021@DESKTOP-SJP6EQH MINGW64 ~  
$ git config --global user.email "ksudzuki@yandex.ru"  
  
15062021@DESKTOP-SJP6EQH MINGW64 ~  
$ git config --global quotepath false  
error: key does not contain a section: quotepath  
  
15062021@DESKTOP-SJP6EQH MINGW64 ~  
$ git config --global core.autocrlf true  
  
15062021@DESKTOP-SJP6EQH MINGW64 ~  
$ git config --global core.safecrlf true  
  
15062021@DESKTOP-SJP6EQH MINGW64 ~  
$ git config --global core.autocrlf true  
  
15062021@DESKTOP-SJP6EQH MINGW64 ~  
$ git config --global core.safecrlf true  
  
15062021@DESKTOP-SJP6EQH MINGW64 ~  
$ git config --global core.quotepath off
```

Рис. 0.1: Настройка VCS git

3. Создал учетную запись на github и подключил к ней ssh ключ.

```
15062021@DESKTOP-SJP6EQH MINGW64 ~
$ ssh-keygen -C "Kim Ilya <ksudzuki@yandex.ru>"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/15062021/.ssh/id_rsa):
Created directory '/c/Users/15062021/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/15062021/.ssh/id_rsa
Your public key has been saved in /c/Users/15062021/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:7uwGqGHHQc9Qb805wA8dS7kXy2vQ58RIw3bgkGxf20U Kim Ilya <ksudzuki@yandex.ru>
The key's randomart image is:
+---[RSA 3072]-----+
|O+.=+.+ ..|
|.O=**B +.|
|.OB+B.O E|
|oo=.B +|
|.ooo= S|
| O +O.O|
|. +. ..|
|. O.|
| O+|
+---[SHA256]-----+
```

Рис. 0.2: SSH ключ

4. Создал структуру каталога лабораторных работ.

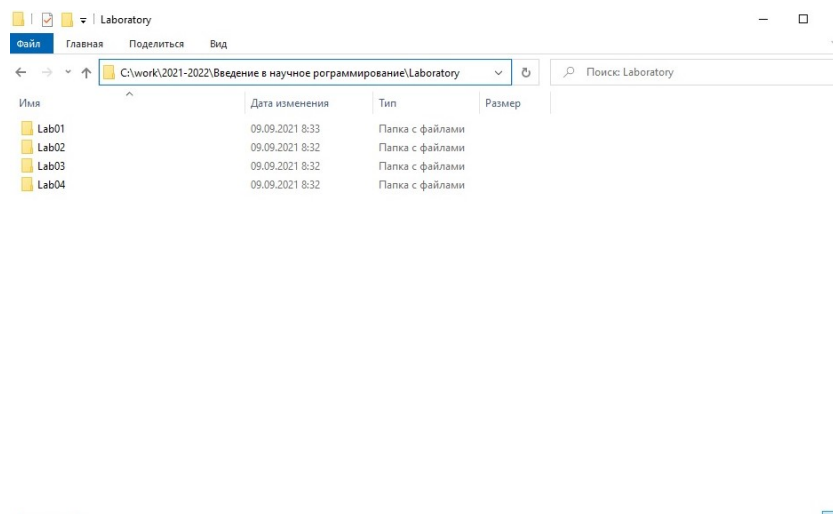


Рис. 0.3: Структура каталога

5. Создал репозиторий на github и назвал его “sciproc-intro”

- Рабочий каталог обозначил как “Laboratory”

- В Git Bash перешёл в него командой “cd laboratory”
- Инициализировал системы git командой “git init”
- Создал заготовку для файла README.md

```
git add README.md
```

- Сделал первый коммит и выложил его на github

```
git commit -m “first commit”
```

```
git remote add origin git@github.com:KSudzuki/sciproc-intro.git
```

```
git push -u origin master
```

```
15062021@DESKTOP-SJP6EQH MINGW64 ~
$ cd Laboratory

15062021@DESKTOP-SJP6EQH MINGW64 ~/Laboratory
$ git init
Initialized empty Git repository in C:/Users/15062021/Laboratory/.git/

15062021@DESKTOP-SJP6EQH MINGW64 ~/Laboratory (master)
$ echo "#Лабораторные работы" >> README.md

15062021@DESKTOP-SJP6EQH MINGW64 ~/Laboratory (master)
$ git add README.md
fatal: LF would be replaced by CRLF in README.md

15062021@DESKTOP-SJP6EQH MINGW64 ~/Laboratory (master)
$ git add README.md

15062021@DESKTOP-SJP6EQH MINGW64 ~/Laboratory (master)
$ git commit -m "first commit"
[master (root-commit) aa56ac1] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md

15062021@DESKTOP-SJP6EQH MINGW64 ~/Laboratory (master)
$ git remote add origin git@github.com:<ksudzuki>/sciproc-intro.git
bash: ksudzuki: No such file or directory

15062021@DESKTOP-SJP6EQH MINGW64 ~/Laboratory (master)
$ git@github.com:KSudzuki/sciproc-intro.git
bash: git@github.com:KSudzuki/sciproc-intro.git: No such file or directory

15062021@DESKTOP-SJP6EQH MINGW64 ~/Laboratory (master)
$ git remote add origin git@github.com:KSudzuki/sciproc-intro.git

15062021@DESKTOP-SJP6EQH MINGW64 ~/Laboratory (master)
$ git push -u origin master
The authenticity of host 'github.com (140.82.121.3)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWG17E1IGOCspRomTxdCARLviKw6E5SY8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'github.com' (RSA) to the list of known hosts.
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 245 bytes | 245.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:KSudzuki/sciproc-intro.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Рис. 0.4: Учебный репозиторий



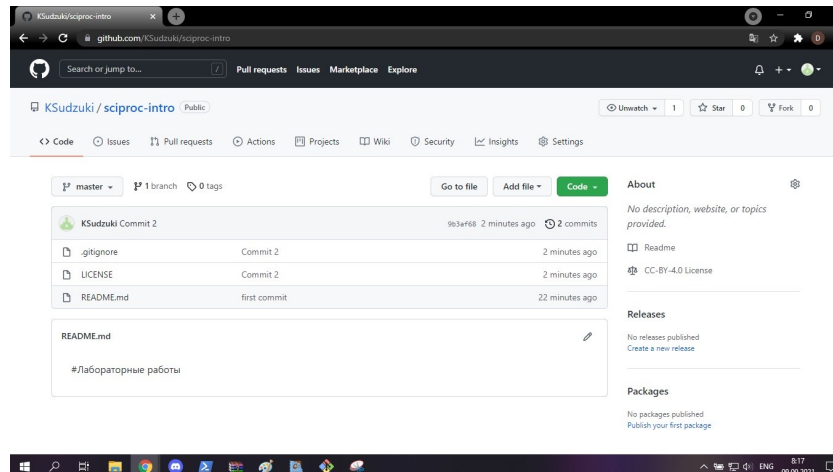


Рис. 0.5: Репозиторий на GitHub

## 6. Первичная конфигурация

- Добавил файл лицензии командой wget (Установил ее через Chocolatey)

```
15062021@DESKTOP-SJP6EQH MINGW64 ~/Laboratory (master)
$ wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O LICENSE
--2021-09-09 08:13:49-- https://creativecommons.org/licenses/by/4.0/legalcode.txt
Resolving creativecommons.org (creativecommons.org)... 104.20.151.16, 172.67.34.140, 104.20.150.16
Connecting to creativecommons.org (creativecommons.org)[104.20.151.16]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: 'LICENSE'

OK ..... 9,11M=0,002s

2021-09-09 08:13:49 (9,11 MB/s) - 'LICENSE' saved [18657]
```

Рис. 0.6: Файл LICENSE

- Посмотрел список шаблонов игнорируемых файлов

```
15062021@DESKTOP-SJP6EQH MINGW64 ~/Laboratory (master)
$ curl -L -s https://www.gitignore.io/api/list
```

Рис. 0.7: Списки шаблонов

- Скачал шаблон

```
15062021@DESKTOP-SJP6EQH MINGW64 ~/Laboratory (master)
$ curl -L -s https://www.gitignore.io/api/c >> .gitignore
```

Рис. 0.8: Шаблон

- Добавил новые файлы

`git add .`

- Выполнил коммит

`git commit -a`

- Отправил на github

`git push`

```
15062021@DESKTOP-SJP6EQH MINGW64 ~/laboratory (master)
$ git add .

15062021@DESKTOP-SJP6EQH MINGW64 ~/laboratory (master)
$ git commit -a
[master 9b3af68] Commit 2
 2 files changed, 514 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 LICENSE

15062021@DESKTOP-SJP6EQH MINGW64 ~/laboratory (master)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 6.44 KiB | 3.22 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:KSudzuki/sciproc-intro.git
 aa56ac1..9b3af68  master -> master
```

Рис. 0.9: Отправка файлов на GitHub

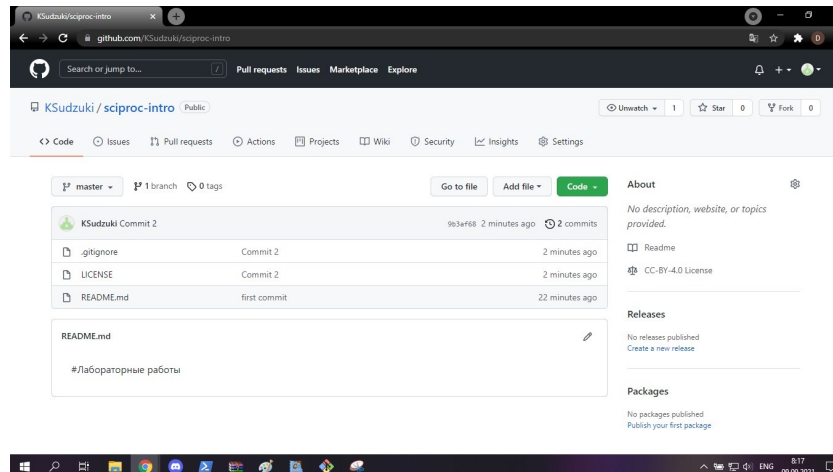


Рис. 0.10: Репозиторий на GitHub

## 7. Конфигурация git-flow

- Инициализировал git-flow

`git flow init`

Префикс для ярлыков установил v.

- Проверил что я на ветке develop (горит зеленым)

`git branch`

- Создал релиз с версией 1.0.0

`git flow release start 1.0.0`

- Записал версию:

`Echo "1.0.0" » VERSION`

- Добавил в индекс:

`git add .`

`git commit -am 'chore(main): add version'`

- Залил релизную ветку в основную ветку

`git flow release finish 1.0.0`

- Отправил данные на github

`git push -all`

`git push -tags`

Рис. 0.11: Измененный файл на GitHub

# Ответы на вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначены?

Система контроля версий – это система, которая позволяет группе людей работать над одним проектом. Его основное дерево, обычно, находится на локальном или удаленном репозитории, к которому имеют доступ все участники проекта.

2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

Хранилище (repository) – место, где хранятся наши файлы на удаленном или локальном VCS.

Commit – команда, которая сохраняет сделанные изменения с внесением комментария, без отправки на репозиторий.

История – история изменений, которые были сделаны в хранилище.

Рабочая копия – место работы разработчика, до отправки в репозиторий (commit вносит изменения в рабочую копию)

3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

Централизованные – это VCS с одним основным хранилищем для всего проекта. Каждый пользователь может скопировать себе необходимые ему файлы из этого репозитория, поменять, а затем добавить обратно

(Примеры: Subversion, CVS, TFS)

Децентрализованные – это VCS, где у каждого пользователя свой вариант(возможно несколько) репозитория. Есть возможность добавлять и забирать изменения из любого хранилища.

(Примеры: Git, Mercurial, Bazaar)

4. Опишите действия с VCS при единоличной работе с хранилищем.

???

5. Опишите порядок работы с общим хранилищем VCS.

Общее хранилище VCS служит для хранения текущих(актуальных) версий того, или иного проекта, над которым работает команда. Изначально мы добавляем туда файлы, а далее может брать их, менять, и добавлять новые версии.

6. Каковы основные задачи, решаемые инструментальным средством git?

Упрощение работы над групповым и индивидуальным проектом, отслеживание изменений, возврат к предыдущим версиям.

7. Назовите и дайте краткую характеристику командам 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 изменения

git add . – добавить все изменённые и/или созданные файлы и/или каталоги

`git add имена_файлов` – добавить конкретные изменённые и/или созданные файлы и/или каталоги

`git rm имена_файлов` – удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории)

`git commit -am 'Описание коммита'` – сохранить все добавленные изменения и все изменённые файлы

`git commit` – сохранить добавленные изменения с внесением комментария через встроенный редактор

`git checkout -b имя_ветки` – создание новой ветки, базирующейся на текущей

`git checkout имя_ветки` – переключение на некоторую ветку (при переключении на ветку, которой ещё нет в локальной репозитории, она будет создана и связана с удалённой)

`git push origin имя_ветки` – отправка изменений конкретной ветки в центральный репозиторий

`git merge --no-ff имя_ветки` – слияние ветки с текущим деревом

`git branch -d имя_ветки` – удаление локальной уже слитой с основным деревом ветки

`git branch -D имя_ветки` – принудительное удаление локальной ветки

`git push origin :имя_ветки` – удаление ветки с центрального репозитория

8. Приведите примеры использования при работе с локальным и удалённым репозиториями.

???

9. Что такое и зачем могут быть нужны ветви (branches)?

Ветви нужны для того, чтобы люди, работающие над проектом могли вести совместную работу, не мешая друг другу. В ветвях можно тестировать код, а потом, после всех проверок, отправлять обратно в основную(master) ветку.

10. Как и зачем можно игнорировать некоторые файлы при commit?

Игнорирование используется для того, чтобы исключить не нужные для определенных случаев файлы (ос, языка программирования или среды разработки)



# Выводы

Изучили идеологию и применение средств контроля версий git.