

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина: *Операционные системы*

Студент:

Калистратова Ксения Евгеньевна

Группа: НПМбд-01-20

МОСКВА

2021 г.

1. **Цель работы:** Изучить идеологию и применение средств контроля версий.

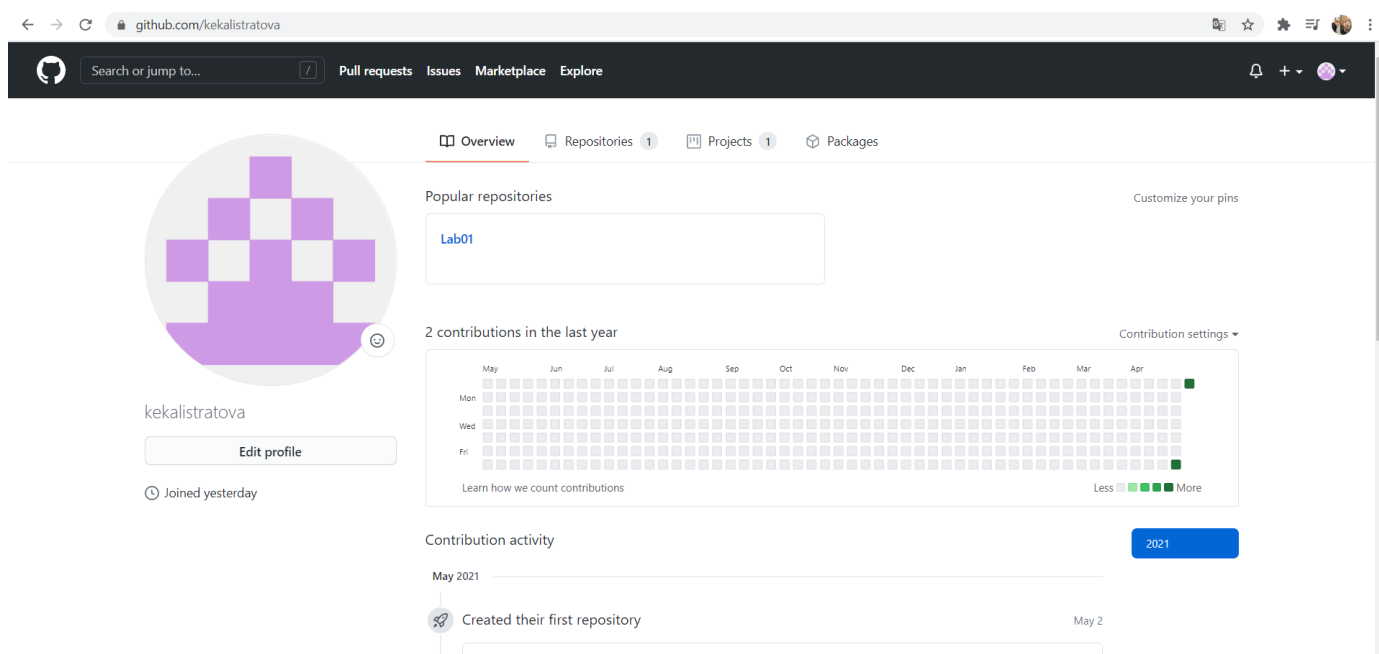
2. **Задание:**

Настроить git, подключить репозиторий к github, изменить первичную конфигурацию и конфигурацию get-flow.

3. **Ход работы:**

1) Для начала я настроила систему контроля версий git, как это описано в указаниях к лабораторной работе с использованием сервера репозитория <http://bitbucket.org/>.

Создаю учетную запись на <https://github.com>.



Настраиваю систему контроля версий git. Синхронизирую учётную запись github с компьютером:

```
git config --global user.name "kekalistratova"
```

```
git config --global user.email "1032201723@pfur.ru"
```

Далее я открыла терминал и сгенерировала SSH ключ, чтобы связать удаленное файловое хранилище с локальными репозиториями, используя команду «ssh-keygen -C "kekalistratova <1032201723@pfur.ru>"»

Или ssh-keygen -t rsa

```
kekalistratova@kekalistratova: ~/Рабочий стол
Файл Действия Правка Вид Справка
(kekalistratova@kekalistratova)-[~/Рабочий стол]
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kekalistratova/.ssh/id_rsa):
Created directory '/home/kekalistratova/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kekalistratova/.ssh/id_rsa
Your public key has been saved in /home/kekalistratova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:iG8CGkLkkUSzAAaw9pBegBvVHuQkjBDzoLsvdUpjWFg kekalistratova@kekalistratova
The key's randomart image is:
+---[RSA 3072]-----+
| ^&+oo
| X*E+o
| o%.o ..
| *+= .. .
| ++ ... . S
| O+= ...
| O+ +. o
| ... o
| ..
+---[SHA256]-----+
```

Скопировала данный ключ в буфер обмена (команда «cat ~/.ssh/id_rsa.pub | xclip -sel clip»), предварительно установив пакет xclip с помощью команд «sudo apt update» и «sudo apt install xclip».

```
kekalistratova@kekalistratova: ~/Рабочий стол
Файл Действия Правка Вид Справка
(kekalistratova@kekalistratova)-[~/Рабочий стол]
$ sudo apt update
[sudo] пароль для kekalistratova:
Пол:1 http://mirror-1.truenetwork.ru/kali kali-rolling InRelease [30,5 kB]
Пол:2 http://mirror-1.truenetwork.ru/kali kali-rolling/main amd64 Packages [17,7 MB]
Пол:3 http://mirror-1.truenetwork.ru/kali kali-rolling/main amd64 Contents (deb) [39,7 MB]
71% [3 Contents-amd64 19,2 MB/39,7 MB 48%] 101 kB/s 3мин 22с ssh-k
Получено 57,5 MB за 6мин 20с (151 kB/s)
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Может быть обновлено 287 пакетов. Запустите «apt list --upgradable» для их показа.
```

```
kekalistratova@kekalistratova: ~/Рабочий стол
Файл Действия Правка Вид Справка

(kekalistratova@kekalistratova)~[~/Рабочий стол]
$ sudo apt install xclip
[sudo] пароль для kekalistratova:
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Следующие НОВЫЕ пакеты будут установлены:
  xclip
Обновлено 0 пакетов, установлено 1 новых пакетов, для удаления отмечено 0 пакетов, и 287 пакетов
не обновлено.
Необходимо скачать 23,3 kB архивов.
После данной операции объем занятого дискового пространства возрастёт на 65,5 kB.
Пол:1 http://mirror-1.truenetwork.ru/kali kali-rolling/main amd64 xclip amd64 0.13-2 [23,3 kB]
Получено 23,3 kB за 1с (20,8 kB/s)
Выбор ранее не выбранного пакета xclip.
(Чтение базы данных ... на данный момент установлено 110318 файлов и каталогов.)
Подготовка к распаковке .../xclip_0.13-2_amd64.deb ...
Распаковывается xclip (0.13-2) ...
Настраивается пакет xclip (0.13-2) ...
Обрабатываются триггеры для man-db (2.9.3-2) ...
Обрабатываются триггеры для kali-menu (2021.1.4) ...
```

```
kekalistratova@kekalistratova: ~/Рабочий стол
Файл Действия Правка Вид Справка

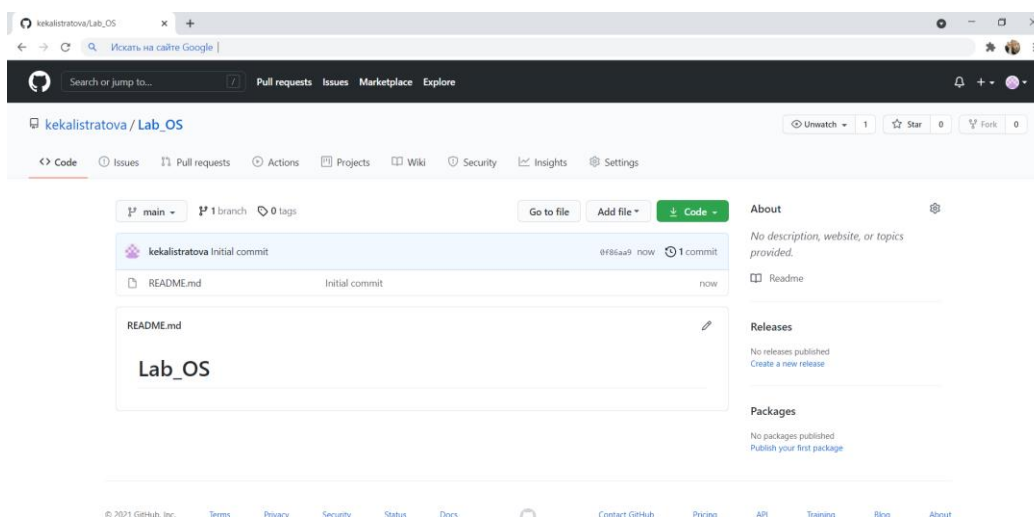
(kekalistratova@kekalistratova)~[~/Рабочий стол]
$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
cat: /home/kekalistratova/.ssh/id_rsa.pub: Нет такого файла или каталога
```

Добавила скопированный SSH ключ в GitHub.

1) Подключение репозитория к github

В GitHub захожу в «repositories» и создаю новый репозиторий (имя «lab_OS», заголовок для файла README). Копируем в консоль ссылку на репозиторий.

Git clone https://github.com/kekalistratova/Lab_OS#lab_os.git



Для создания репозитория я вернулась в терминал и перешла в нужный каталог с помощью команды «cd lab_OS». Выполнила инициализацию репозитория

(создала основное дерево репозитория), используя команду «git init». С помощью команды «ls -a» убедилась в том, что появился скрытый каталог .git

```
(kekalistratova@kekalistratova)-[~/Рабочий стол]
$ git clone https://github.com/kekalistratova/Lab_OS.git
Клонирование в «Lab_OS»...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Получение объектов: 100% (3/3), готово.
```

Работаю с каталогом и папками через консоль. Перед тем, как создавать файлы, захожу в репозиторий:

```
(kekalistratova@kekalistratova)-[~/Рабочий стол]
$ ls
key  key.pub  Lab_OS  lab_prog  README.md

(kekalistratova@kekalistratova)-[~/Рабочий стол]
$ cd Lab_OS

(kekalistratova@kekalistratova)-[~/Рабочий стол/Lab_OS]
$ mkdir 2020-2021

(kekalistratova@kekalistratova)-[~/Рабочий стол/Lab_OS]
$ cd 2020-2021

(kekalistratova@kekalistratova)-[~/Рабочий стол/Lab_OS/2020-2021]
$ mkdir may

(kekalistratova@kekalistratova)-[~/Рабочий стол/Lab_OS/2020-2021]
$ cd may

(kekalistratova@kekalistratova)-[~/Рабочий стол/Lab_OS/2020-2021/may]
$ mkdir lab02

(kekalistratova@kekalistratova)-[~/Рабочий стол/Lab_OS/2020-2021/may]
$ cd lab02

(kekalistratova@kekalistratova)-[~/.../Lab_OS/2020-2021/may/lab02]
$ touch a.txt
```

Добавляю первый коммит и выкладываю на github. Для того, чтобы правильно разместить первый коммит, необходимо добавить команду git add . , далее с помощью команды git commit -m "first commit" выкладываем коммит:

```
(kekalistratova@kekalistratova)-[~/.../Lab_OS/2020-2021/may/lab02]
$ git add .

(kekalistratova@kekalistratova)-[~/.../Lab_OS/2020-2021/may/lab02]
$ git commit -m "first commit"
[main 16cbf18] first commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 2020-2021/may/lab02/a.txt
```

Сохраняю первый коммит (git push):

```
(kekalistratova@kekalistratova)-[~/.../Lab_OS/2020-2021/may/lab02]
$ git push
Username for 'https://github.com': kekalistratova
Password for 'https://kekalistratova@github.com':
Перечисление объектов: 7, готово.
Подсчет объектов: 100% (7/7), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (6/6), 401 bytes | 401.00 KiB/s, готово.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/kekalistratova/Lab_OS.git
0f86aa9..16cbf18 main -> main
```

2) Первичная конфигурация

Добавляю файл лицензии; Добавляю шаблон игнорируемых файлов. Получаю список имеющихся шаблонов (на скрине представлены не все шаблоны):

```
kekalistratova@kekalistratova: ~/Рабочий стол/Lab_OS/2020-2021/may/lab02
Файл Действия Правка Вид Справка
(kekalistratova@kekalistratova)-[~/.../Lab_OS/2020-2021/may/lab02]
$ wget https://creativecommons.org/licenses/by/4.0/legalcode.txt
--2021-05-05 21:47:33-- https://creativecommons.org/licenses/by/4.0/legalcode.txt
Распознаётся creativecommons.org (creativecommons.org)... 172.67.34.140, 104.20.150.16, 104.20.151.16, ...
Подключение к creativecommons.org (creativecommons.org)|172.67.34.140|:443 ... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: нет данных [text/plain]
Сохранение в: «legalcode.txt»

legalcode.txt [ ⇌ ] 18,22K --KB/s за 0,009s
2021-05-05 21:47:34 (2,07 MB/s) - «legalcode.txt» сохранён [18657]

(kekalistratova@kekalistratova)-[~/.../Lab_OS/2020-2021/may/lab02]
$ curl -L -s https://www.gitignore.io/api/list
1c,1c-bitrix,a-frame,actionsript,ada
adobe,advancedinstaller,adventuregamestudio,agda,al
alteraquartusii,altium,amplify,android,androidstudio
angular,anjuta,ansible,ansibletower,apachecordova
apachehadoop,appbuilder,appcelerator,titanium,appcode,appcode+all
appcode+iml,appengine,aptanastudio,arcanist,archive
archives,archlinuxpackages,aspnetcore,assembler,ate
atmelstudio,ats,audio,automationstudio,autotools
autotools+strict,awr,azurefunctions,backup,ballerina
basercms,basic,batch,bazaar,bazel
bitrise,bitrix,bittorrent,blackbox,bloop
bluej,bookdown,bower,bricxcc,buck
c,c++,cake,cakephp,cakephp2
cakephp3,calabash,carthage,certificates,ceylon
cfwheels,chefcookbook,chocolatey,clean,clion
clion+all,clion+iml,clojure,cloud9,cmake
cocoapods,cocos2dx,cocoscreator,code-java,codeblocks
codecomposerstudio,codeigniter,codeio,codekit,codesniffer
coffeescript,commonlisp,compodoc,composer,compressed
compressedarchive,compression,conan,concrete5,coq
cordova,craftcms,crashlytics,crbasic,crossbar
crystal,cs-cart,csharp,cuda,cvs
cypressio,d,dart,darteditor,data
database,datarecovery,dbeaver,defold,delphi
dframe,diff,direnv,diskimage,django
dm,docfx,docpress,docz,dotenv
```

Скачиваю шаблон, например, для C. Также добавляю новые файлы и выполняю коммит; Отправляю на github (git push):


```

(kekalistratova@kekalistratova)-[~/.../Lab_OS/2020-2021/may/lab02]
$ curl -L -s https://www.gitignore.io/api/c >> .gitignore

(kekalistratova@kekalistratova)-[~/.../Lab_OS/2020-2021/may/lab02]
$ git add .

(kekalistratova@kekalistratova)-[~/.../Lab_OS/2020-2021/may/lab02]
$ git commit -a
Отмена коммита из-за пустого сообщения коммита.

(kekalistratova@kekalistratova)-[~/.../Lab_OS/2020-2021/may/lab02]
$ git push
zsh: bad pattern: ^[[200~git
1 ✖

(kekalistratova@kekalistratova)-[~/.../Lab_OS/2020-2021/may/lab02]
$ git commit -am "Create template for C"
1 ✖

[main 47f43fc] Create template for C
2 files changed, 455 insertions(+)
create mode 100644 2020-2021/may/lab02/.gitignore
create mode 100644 2020-2021/may/lab02/legalcode.txt

(kekalistratova@kekalistratova)-[~/.../Lab_OS/2020-2021/may/lab02]
$ git push
Username for 'https://github.com': kekalistratova
Password for 'https://kekalistratova@github.com':
Перечисление объектов: 11, готово.
Подсчет объектов: 100% (11/11), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (5/5), готово.
Запись объектов: 100% (7/7), 6.60 KiB | 6.60 MiB/s, готово.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/kekalistratova/Lab_OS.git
16cbf18..47f43fc main -> main

```

3) Конфигурация git-flow

Инициализирую git-flow, используя команду `git flow init -f` (префикс для ярлыков установлен в v);

Проверяю, что нахожусь на ветке develop (git branch); Создаю релиз с версией 1.0.0;

Записываю версию и добавляю в индекс:

```
echo "1.0.0" >> VERSION
```

```
git add .
```

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

```
kekalistratova@kekalistratova: ~/Рабочий стол
Файл Действия Правка Вид Справка
(kekalistratova@kekalistratova)-[~/Рабочий стол]
$ git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [/home/kekalistratova/Рабочий стол/.git/hooks]

(kekalistratova@kekalistratova)-[~/Рабочий стол]
$ git branch
* develop
main

(kekalistratova@kekalistratova)-[~/Рабочий стол]
$ git flow release start 1.0.
fatal: «release/1.0.» не является действительным именем ветки.
Fatal: Could not create release branch 'release/1.0.'.

(kekalistratova@kekalistratova)-[~/Рабочий стол]
$ git flow release start 1.0.0

Переключено на новую ветку «release/1.0.0»

Summary of actions:
- A new branch 'release/1.0.0' was created, based on 'develop'
- You are now on branch 'release/1.0.0'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

    git flow release finish '1.0.0'

(kekalistratova@kekalistratova)-[~/Рабочий стол]
$ echo "1.0.0" >> VERSION

(kekalistratova@kekalistratova)-[~/Рабочий стол]
$ git add .
```

```
kekalistratova@kekalistratova: ~/Рабочий стол/Lab_OS/2020-2021/may/lab02
Файл Действия Правка Вид Справка
(kekalistratova@kekalistratova)-[~/.../Lab_OS/2020-2021/may/lab02]
$ git commit -am 'chore(main): add version'
[release/1.0.0 d98df70] chore(main): add version
1 file changed, 1 insertion(+)

(kekalistratova@kekalistratova)-[~/.../Lab_OS/2020-2021/may/lab02]
$ git flow release finish 1.0.0
Переключено на ветку «main»
Ваша ветка обновлена в соответствии с «origin/main».
Merge made by the 'recursive' strategy.
2020-2021/may/lab02/VERSION | 2 ++
1 file changed, 2 insertions(+)
create mode 100644 2020-2021/may/lab02/VERSION
Уже на «main»
Ваша ветка опережает «origin/main» на 3 коммита.
(используйте «git push», чтобы опубликовать ваши локальные коммиты)
Переключено на ветку «develop»
Merge made by the 'recursive' strategy.
2020-2021/may/lab02/VERSION | 1 +
1 file changed, 1 insertion(+)
Ветка release/1.0.0 удалена (была d98df70).

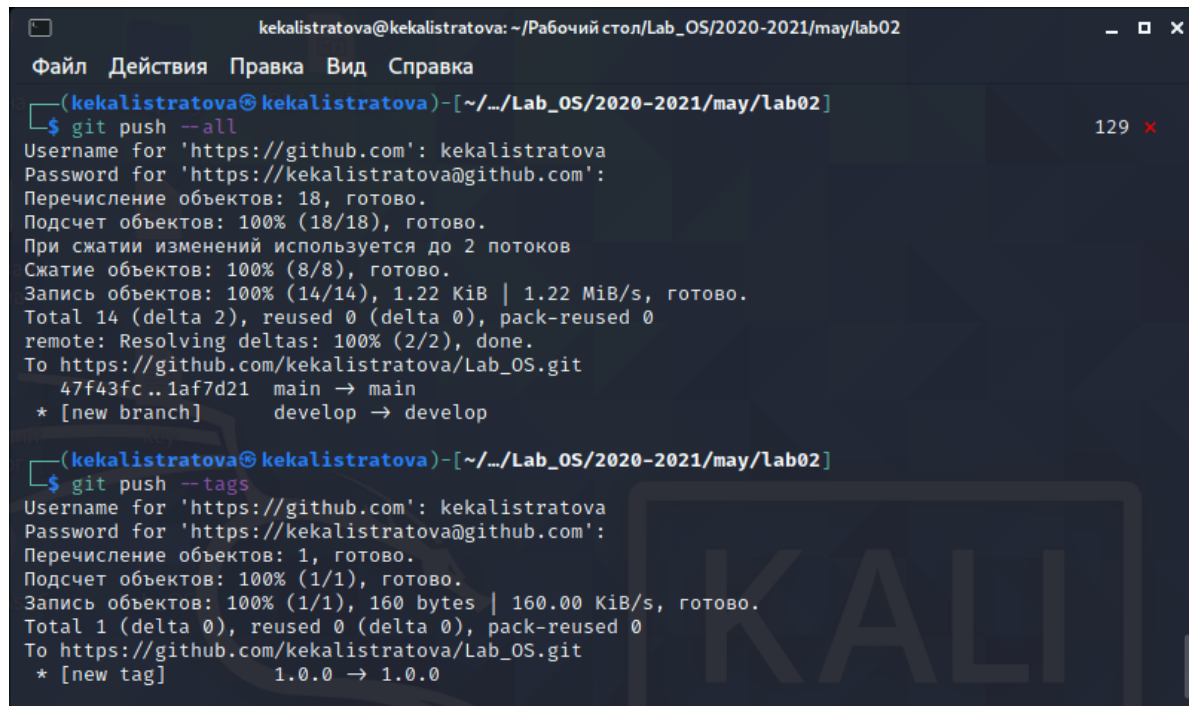
Summary of actions:
- Release branch 'release/1.0.0' has been merged into 'main'
- The release was tagged '1.0.0'
- Release tag '1.0.0' has been back-merged into 'develop'
- Release branch 'release/1.0.0' has been locally deleted
- You are now on branch 'develop'
```


Заливаю релизую ветку в основную ветку (команда `git flow release finish 1.0.0`).

Отправляю данные на github:

`git push --all`

`git push --tags`



```
kekalistratova@kekalistratova: ~/Рабочий стол/Lab_OS/2020-2021/may/lab02
Файл Действия Правка Вид Справка
(kekalistratova@kekalistratova)-[~/../Lab_OS/2020-2021/may/lab02]
$ git push --all
Username for 'https://github.com': kekalistratova
Password for 'https://kekalistratova@github.com':
Перечисление объектов: 18, готово.
Подсчет объектов: 100% (18/18), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (8/8), готово.
Запись объектов: 100% (14/14), 1.22 KiB | 1.22 MiB/s, готово.
Total 14 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/kekalistratova/Lab_OS.git
  47f43fc..1af7d21  main -> main
* [new branch]      develop -> develop

(kekalistratova@kekalistratova)-[~/../Lab_OS/2020-2021/may/lab02]
$ git push --tags
Username for 'https://github.com': kekalistratova
Password for 'https://kekalistratova@github.com':
Перечисление объектов: 1, готово.
Подсчет объектов: 100% (1/1), готово.
Запись объектов: 100% (1/1), 160 bytes | 160.00 KiB/s, готово.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/kekalistratova/Lab_OS.git
* [new tag]         1.0.0 -> 1.0.0
```

Создаю релиз на github. Заходим в «Releases», нажимаю «Создать новый релиз».

Захожу в теги и заполняю все поля (теги для версии 1.0.0). После создания тега автоматически сформируется релиз.

4) 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

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: — создание 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 изменений: 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 с текущим 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) Использование `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 список имеющихся шаблонов: `curl -L -s https://www.gitignore.io/api/list`

Затем скачать шаблон, например, для C и C++

```
curl -L -s https://www.gitignore.io/api/c >> .gitignore
```

```
curl -L -s https://www.gitignore.io/api/c++ >> .gitignore
```

4. Вывод: В ходе выполнения данной лабораторной работы я изучила идеологию и применение средств контроля версий.