РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

дисциплина: Операционные системы

Студент: Карвецкий Всеволод Анатольевич

Группа: НКНбд-01-20

МОСКВА

2021 г.

Цель работы

Изучить идеологию и применение средств контроля версий. Научиться использовать систему контроля версии в своих проектах

Задание

Настроить git на своем компьютере, инициализировать локальный репозиторий в папке с лабораторными работами, добавить все нужные файлы в репозиторий, подключить удаленный репозиторий с github, связать локальный и удаленные репозитории

Выполнение лабораторной работы

0. На моем личном компьютере система Git уже была настроена

1. Подключение локального репозитория к GitHub

   i. Инициализируем локальный репозиторий

```
vsevolod@DESKTOP-CNQSN0P MINGW64 ~/Desktop/os
$ git init
Initialized empty Git repository in C:/Users/vsevolod/Desktop/os/.git/
```

   ii. Добавляем файлы в репозиторий

```
vsevolod@DESKTOP-CNQSN0P MINGW64 ~/Desktop/os (master)
$ git add .
```

   iii. Делаем коммит

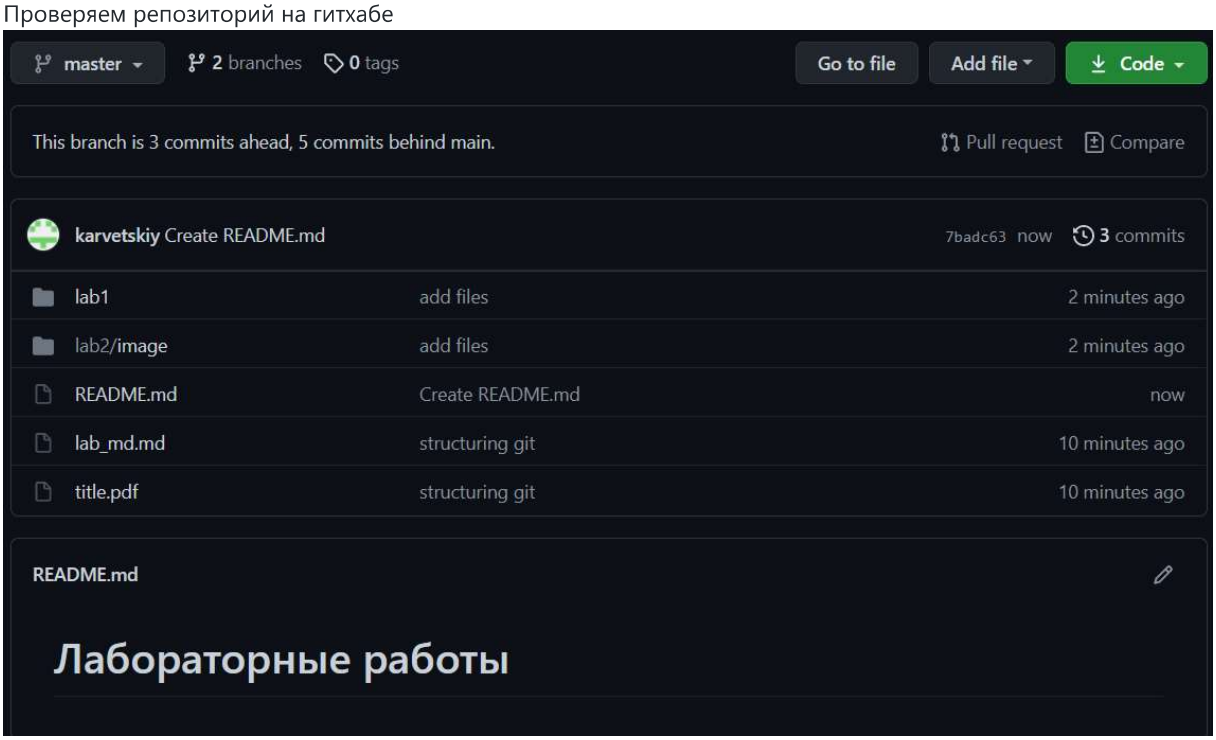
```
vsevolod@DESKTOP-CNQSN0P MINGW64 ~/Desktop/os (master)
$ git commit -m "structuring git"
[master (root-commit) 6fedb9f] structuring git
 45 files changed, 136 insertions(+)
 create mode 100644 lab1/image.zip
 create mode 100644 lab1/image/067c0a96c103ee7017cde087b246bab207852d2c.png
 create mode 100644 lab1/image/1.1.png
 create mode 100644 lab1/image/2.1.png
 create mode 100644 lab1/image/2.2.png
 create mode 100644 lab1/image/2.3.png
 create mode 100644 lab1/image/2.4.png
 create mode 100644 lab1/image/23e8b83c18eef9317bfc5071a51e38531373bdde.png
 create mode 100644 lab1/image/3.1.png
 create mode 100644 lab1/image/3.2.png
 create mode 100644 lab1/image/3.3.png
 create mode 100644 lab1/image/3.4.png
 create mode 100644 lab1/image/3.5.png
 create mode 100644 lab1/image/39eab489f1d11ec6a27efe98355bf6c1ade6368c.png
 create mode 100644 lab1/image/4.1.png
```

   iv. пуш на гитхаб

```
vsevolod@DESKTOP-CNQSN0P MINGW64 ~/Desktop/os (master)
$ git remote add origin https://github.com/karvetskiy/git-os.git
```

```
vsevolod@DESKTOP-CNQSN0P MINGW64 ~/Desktop/os (master)
$ git push -u origin master
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 47.16 KiB | 9.43 MiB/s, done.
Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/karvetskiy/git-os.git
   6fedb9f..1caf8b1  master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

v. Проверяем репозиторий на гитхабе



2. Конфигурация Git

i. Добавил файл лицензии с помощью `wget`

```
vsevolod@DESKTOP-CNQSNOP MINGW64 ~/Desktop/os (master)
$ wget https://creativecommons.org/licenses/by/4.0/legalcode.txt
--2021-04-22 11:52:56--  https://creativecommons.org/licenses/by/4.0/legalcode.t
xt
Resolving creativecommons.org (creativecommons.org)... 2606:4700:10::6814:9710,
2606:4700:10::ac43:228c, 2606:4700:10::6814:9610, ...
Connecting to creativecommons.org (creativecommons.org)|2606:4700:10::6814:9710|
:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: 'legalcode.txt'

     OK ..........  ........                                  65,7K=0,3s

2021-04-22 11:52:56 (65,7 KB/s) - 'legalcode.txt' saved [18657]
```

ii. Подгрузил список доступных шаблонов игнорируемых файлов

```
vsevolod@DESKTOP-CNQSNOP MINGW64 ~/Desktop/os (master)
$ curl -L -s https://www.gitignore.io/api/list
1c,1c-bitrix,a-frame,actionscript,ada
adobe,advancedinstaller,adventuregamestudio,agda,al
alteraquartusii,altium,amplify,android,androidstudio
angular,anjuta,ansible,apachecordova,apachehadoop
appbuilder,appceleratortitanium,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,code-java,codeblocks
codecomposerstudio,codeigniter,codeio,codekit,codesniffer
coffeescript,commonlisp,compodoc,composer,compressed
compressedarchive,compression,conan,concrete5,coq
```

iii. Выбрал и загрузил шаблон для C++

```
vsevolod@DESKTOP-CNQSNOP MINGW64 ~/Desktop/os (master)
$ curl -L -s https://www.gitignore.io/api/c++ >> .gitignore
```

iv. Пуш на GitHub

```
vsevolod@DESKTOP-CNQSNOP MINGW64 ~/Desktop/os (master)
$ git push
Enumerating objects: 20, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 8 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (14/14), 154.44 KiB | 15.44 MiB/s, done.
Total 14 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/karvetskiy/git-os.git
   7badc63..006c843  master -> master
```

3. Конфигурация GitFlow

i. Инициализирую GitFlow

```
vsevolod@DESKTOP-CNQSNOP MINGW64 ~/Desktop/os (master)
$ git flow init

Which branch should be used for bringing forth production releases?
   - master
Branch name for production releases: [master] master
Branch name for "next release" development: [develop] 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? [] v
Hooks and filters directory? [C:/Users/vsevolod/Desktop/os/.git/hooks]

vsevolod@DESKTOP-CNQSNOP MINGW64 ~/Desktop/os (develop)
```

ii. Переключаюсь на ветку `develop`

```
vsevolod@DESKTOP-CNQSNOP MINGW64 ~/Desktop/os (develop)
$ git branch
* develop
  master
```

iii. Создаю релизную версию `1.0.0`

```
vsevolod@DESKTOP-CNQSNOP MINGW64 ~/Desktop/os (develop)
$ git flow release start 1.0.0
Switched to a new branch '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'


vsevolod@DESKTOP-CNQSNOP MINGW64 ~/Desktop/os (release/1.0.0)
$ echo "1.0.0" >> VERSION
```

iv. Добавил туда нужные файлы и закрыл релизную ветку

```
vsevolod@DESKTOP-CNQSNOP MINGW64 ~/Desktop/os (release/1.0.0)
$ git flow release finish 1.0.0
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
hint: Waiting for your editor to close the file...
Attempting to call a function in a renderer window that has been closed or relea
sed.
Function provided here: Object.<anonymous> (C:\Users\vsevolod\AppData\Local\atom
\app-1.56.0\resources\app.asar\node_modules\github\lib\worker.js:79:22
Remote event names: crashed, destroyed
Merge made by the 'recursive' strategy.
 VERSION            |   1 +
 lab2/image/2.4.png | Bin 0 -> 23307 bytes
 lab2/image/3.1.png | Bin 0 -> 27683 bytes
 lab2/image/3.2.png | Bin 0 -> 6513 bytes
 lab2/image/3.3.png | Bin 0 -> 25734 bytes
 5 files changed, 1 insertion(+)
 create mode 100644 VERSION
 create mode 100644 lab2/image/2.4.png
 create mode 100644 lab2/image/3.1.png
 create mode 100644 lab2/image/3.2.png
 create mode 100644 lab2/image/3.3.png
Already on 'master'
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)
hint: Waiting for your editor to close the file...
fatal: no tag message?
Fatal: Tagging failed. Please run finish again to retry.
```

v. Пушим все ветки на гит

```
vsevolod@DESKTOP-CNQSNOP MINGW64 ~/Desktop/os (master)
$ git push --all
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 8 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (10/10), 78.29 KiB | 15.66 MiB/s, done.
Total 10 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To https://github.com/karvetskiy/git-os.git
   006c843..710366f  master -> master
 * [new branch]      develop -> develop
 * [new branch]      release/1.0.0 -> release/1.0.0

vsevolod@DESKTOP-CNQSNOP MINGW64 ~/Desktop/os (master)
$ git push --tags
Everything up-to-date
```

Выводы

Выполняя данную лабораторную работу, я научился пользоваться `Git` , а именно: инициализировать локальный репозиторий, подключать удаленные репозитории, добавлять и удалять нужные файлы, синхронизировать данные. Также я научился использовать `Git Flow` , который очень сильно упрощает разработку проектов и навигацию между ветками

Контрольные вопросы

1. Системы контроля версий - это программное обеспечение, которое используется для облегчения работы с изменяющейся информацией, обычно - с проектами. Чаще всего применяется в разработке, когда над одним проектом работает большое количество людей
2. - Хранилище в системе контроля версий - это удаленный репозиторий, где хранятся все файлы проекта
   - commit - это фиксация изменений перед загрузкой файлов в систему контроля версий
   - история хранит все изменения проекта, и в случае необходимости позволяет откатиться к нужному месту
   - рабочая копия - это копия проекта на компьютере разработчика. Если другой член команды изменил проект, необходимо загрузить новую версию проекта себе на компьютер
3. Централизованные системы контроля версий хранят данные о проекте на едином сервере, и в случае его отключения, доступ к данным будет утерян (Perforce) В децентрализовынных системах у каждого из участников проекта на компьютере хранитсяя полная копия проекта, что позволяет меньше зависеть от сервера (Git)

4. Сначала надо создать и подключить удаленный репозиторий. Затем, т.к. никто кроме тебя не изменяет проект, по мере изменения проекта пушить изменения на сервер, и нет необходимости загружать изменения
5. Каждый раз перед разработкой необходимо загрузить актуальную версию проекта на свой компьютер, а уже потом раюотать над ним. После работы необходимо закоммитить изменения и запушить на сервер
6. Упрощение обмена информацией, ускорение разработки, устранение ошибок и недоработок во время разработки.
7. - `git init` - инициализирует локальный репозиторий
   - `git add` - добавляет файлы в репозиторий
   - `git commit` - коммит версии
   - `git pull` - загружает актуальную версию проекта
   - `git push` - отправляет измененный проект на сервер
   - `git checkout` - позволяет переключаться между ветками
   - `git status` - текущий статус проекта
   - `git branch` - просмотр доступных веток
   - `git remote add` - добавление удаленного репозитория
8.

Если я забыл, в какой ветке нахожусь, то с помощью `git branch` могу посмотреть это. Если мне нужно подключить систему контроля версий к уже существующему проекту, то я инициализирую локальный репозиторий `git init` и подключаю удаленный `git remote add` , затем добавляю все файлы `git add` и коммичу их `git commit` , затем пушу на удаленный репозиторий `git push` . Теперь к моему проекту подключена система контроля версий 9. Ветки нужны для разделения разработки. Например, когда разрабатывается новая фича, не нужно, чтобы она присутствовала в основном проекте, поэтому для нее создают отдельную ветку. В случае успешной разработки фичи, эту ветку сливают с основной. Так убираются риски багов, ошибок, а также утечки данных 10. Есть временные и системные файлы, которые засоряют проект и не нужны. путь к ним можно добавить в файл `.gitignore` , тогда они не будут добавляться в проект