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: Белицкий Валентин Александрович

Группа: НПМ-бв-01-18

МОСКВА

2022г.

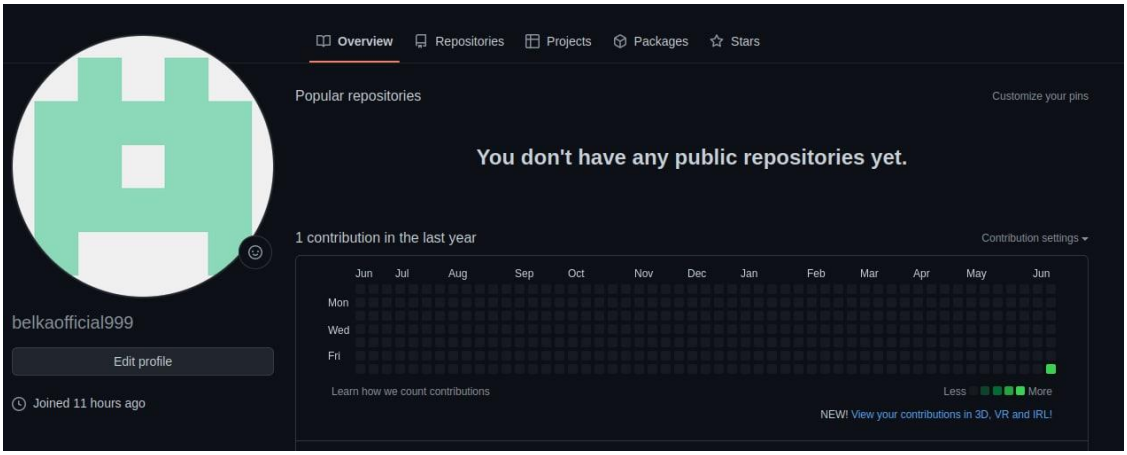
Цель работы

Изучить идеологию и применение средств контроля версий.

Ход работы

Настройка git

1) Создание учётной записи на Github



2) Настройка системы контроля версий git

```
valentin@valentin-VirtualBox:~$ git config --global user.name "Valentin B"
valentin@valentin-VirtualBox:~$ git config --global user.email "Rylum@yandex.ru"
```

Подключение репозитория к GitHub

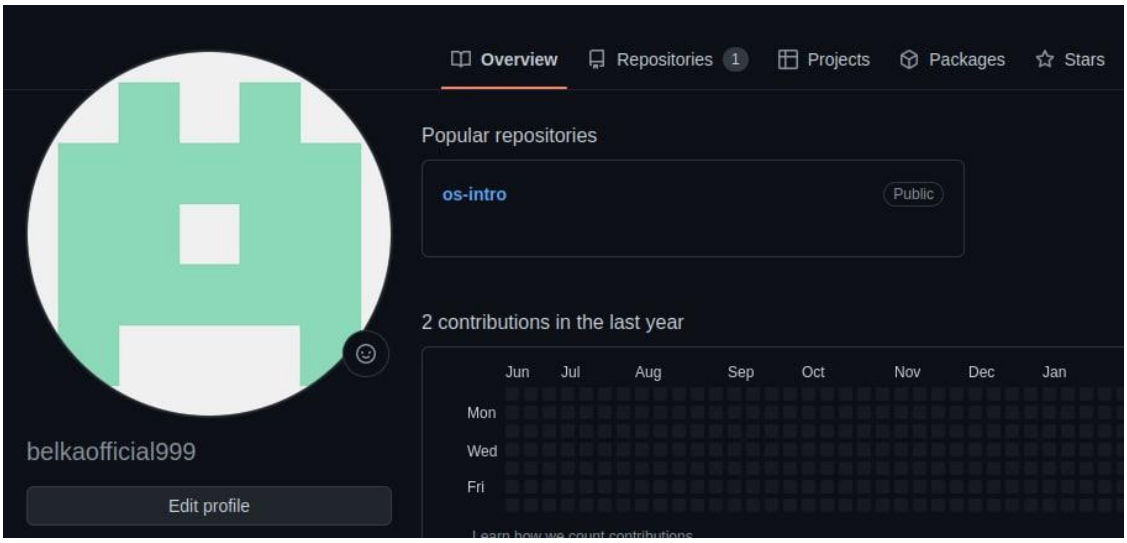
1) генерируем ssh ключ

```
valentin@valentin-VirtualBox:~$ ssh-keygen -C "Valentin B Rylum@yandex.ru"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/valentin/.ssh/id_rsa):
Created directory '/home/valentin/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/valentin/.ssh/id_rsa
Your public key has been saved in /home/valentin/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:p7bHIbUfKCQO8DKMDvFnoBQBLuuO23lucAF6WXj+3f8 Valentin B Rylum@yandex.ru
The key's randomart image is:
+---[RSA 3072]----+
|ooo.             |
|o.= o            |
|oO X             |
|*.B B . . .      |
|+. = = +So.o     |
|... . o +o= .    |
| . o    o+ + .   |
|o. .o   . .o o   |
|ooo+.    ..   .E |
+----[SHA256]-----+
valentin@valentin-VirtualBox:~$
```

2)Копируем полученный ключ и вставляем в гитхаб

```
valentin@valentin-VirtualBox:~$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQC2Ck+A4u62VKUK0eggGM+Ra1y8/cZxm9z
oyEY05eqUMWO6USy/lIlhGlP/Y1ZsmT51DerY8aNaaBReinVWC1nqXUUNPvonvUPd+S0IPH
```

3) Создаём репозиторий на GitHub



4) инициализируем репозиторий

```
valentin@valentin-VirtualBox: $ mkdir laboratory
valentin@valentin-VirtualBox: $ cd laboratory/
valentin@valentin-VirtualBox:~/laboratory$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/valentin/laboratory/.git/
valentin@valentin-VirtualBox:~/laboratory$
```

5) Подключаемся к репозиторию os-intro

```
valentin@valentin-VirtualBox:~/laboratory$ git remote add origin git@github.com:belkaofficial999/os-intro.git
valentin@valentin-VirtualBox:~/laboratory$ git branch -M main
valentin@valentin-VirtualBox:~/laboratory$
```

6) Создаём заготовку для файла README.md и отправляем на Github

```
valentin@valentin-VirtualBox:~/laboratory$ echo "# Лабораторные работы" >> README.md
valentin@valentin-VirtualBox:~/laboratory$ ls
README.md
valentin@valentin-VirtualBox:~/laboratory$
```

```
valentin@valentin-VirtualBox:~/laboratory$ git add README.md
valentin@valentin-VirtualBox:~/laboratory$ git commit -m "first commit"
[main (root-commit) a9dc7fd] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
valentin@valentin-VirtualBox:~/laboratory$ git push -u origin main
The authenticity of host 'github.com (140.82.121.4)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
valentin@valentin-VirtualBox:~/laboratory$ git push -u origin main
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:belkaofficial999/os-intro.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
valentin@valentin-VirtualBox:~/laboratory$
```

Первичная конфигурация

Добавим файл лицензии:

```
valentin@valentin-VirtualBox:~/laboratory$ wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O LICENSE
--2022-06-18 23:30:03--  https://creativecommons.org/licenses/by/4.0/legalcode.txt
Resolving creativecommons.org (creativecommons.org)... 172.67.34.140, 104.20.150.16, 104.20.151.16, ...
Connecting to creativecommons.org (creativecommons.org)|172.67.34.140|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: ‘LICENSE’

LICENSE                                 [ <=>

2022-06-18 23:30:04 (1,34 MB/s) - ‘LICENSE’ saved [18657]
```

Добавим шаблон игнорируемых файлов. Просмотрим список имеющихся шаблонов:

```
valentin@valentin-VirtualBox:~/laboratory$ 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,ansibletower,apachecordova
apachehadoop,appbuilder,appceleratortitanium,appcode,appcode+all
appcode+iml,appengine,aptanastudio,arcanist,archive
archives,archlinuxpackages,aspnetcore,assembler,ate
atmelstudio,ats,audio,automationstudio,autotools
```

Затем скачаем шаблон, например, для C:

```
zsh,zukencr8000valentin@valentin-VirtualBox:~/laboratory$ curl -L -s https://www.gitignore.io/api/c >> .gitignore
valentin@valentin-VirtualBox:~/laboratory$ ls -la
total 40
drwxrwxr-x  3 valentin valentin  4096 июн 18 23:33 .
drwxr-x--- 18 valentin valentin  4096 июн 18 23:30 ..
drwxrwxr-x  8 valentin valentin  4096 июн 18 23:26 .git
-rw-rw-r--  1 valentin valentin   631 июн 18 23:33 .gitignore
-rw-rw-r--  1 valentin valentin 18657 мар 24  2020 LICENSE
-rw-rw-r--  1 valentin valentin    40 июн 18 23:24 README.md
valentin@valentin-VirtualBox:~/laboratory$
```

Добавим новые файлы:

```
valentin@valentin-VirtualBox:~/laboratory$ git add .
valentin@valentin-VirtualBox:~/laboratory$ git commit -a
Aborting commit due to empty commit message.
valentin@valentin-VirtualBox:~/laboratory$ git commit -m "add gitignore C"
[main c88e12e] add gitignore C
 2 files changed, 455 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 LICENSE
valentin@valentin-VirtualBox:~/laboratory$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 6.43 KiB | 3.22 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:belkaofficial999/os-intro.git
   a9dc7fd..c88e12e  main -> main
valentin@valentin-VirtualBox:~/laboratory$
```

Конфигурация git-flow

Инициализируем git-flow

```
valentin@valentin-VirtualBox:~/laboratory$ 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/valentin/laboratory/.git/hooks]
valentin@valentin-VirtualBox:~/laboratory$
```

Проверьте, что Вы на ветке develop:

```
valentin@valentin-VirtualBox:~/laboratory$ git branch
* develop
  main
valentin@valentin-VirtualBox:~/laboratory$
```

Создадим релиз с версией 1.0.0

```
valentin@valentin-VirtualBox:~/laboratory$ 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'

valentin@valentin-VirtualBox:~/laboratory$
```

Запишем версию:

```
valentin@valentin-VirtualBox:~/laboratory$ echo "1.0.0" >> VERSION
valentin@valentin-VirtualBox:~/laboratory$
valentin@valentin-VirtualBox:~/laboratory$
```

Добавим в индекс:

```
valentin@valentin-VirtualBox:~/laboratory$ git add .
valentin@valentin-VirtualBox:~/laboratory$ git commit -am "chore(main): add version"
[release/1.0.0 923298d] chore(main): add version
 1 file changed, 1 insertion(+)
 create mode 100644 VERSION
valentin@valentin-VirtualBox:~/laboratory$ git flow release finish 1.0.0
Switched to branch 'main'
```

Отправим данные на github

```
valentin@valentin-VirtualBox:~/laboratory$ git push --al
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 377 bytes | 377.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To github.com:belkaofficial999/os-intro.git
   c88e12e..b783ab2  main -> main
 * [new branch]      develop -> develop
 * [new branch]      release/1.0.0 -> release/1.0.0
valentin@valentin-VirtualBox:~/laboratory$ git push --tags
```

Вывод

В ходе выполнения лабораторной работы №2 были освоены навыки администрирования и взаимодействия с децентрализованной системой и операционной системой git для параллельного контроля поддержки программного кода.

Контрольные вопросы

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

Системы контроля версий (VCS) – это программное обеспечение для облегчения работы с изменяющейся информацией. Предназначаются для работы нескольких человек над одним проектом, совместная работа путем изменения файлов в одном репозитории.

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

- Хранилище – это общее пространство для хранения файлов
- Commit – это команда для записи индексированных изменений в репозитории
- В истории сохраняются все коммиты, по которым можно отследить автора, сообщение, дату и хэш коммита
- Все файлы кроме .git/ называются рабочей копией, и принадлежат пользователю

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

Централизованные системы контроля версий сохраняют проект и его файлы на один общий сервер, а децентрализованные системы контроля версий при каждом копировании данных удаленного репозитория, происходит полное копирование данных в локальный репозиторий. К примеру, централизованные системы контроля версий – SVN, MS TFS, ClearCase; децентрализованные системы контроля версий – Git, Mercurial, Bazaar.

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

1) Создаем репозиторий и именуем его
2) Добавляем файлы в репозиторий
3) Фиксируем с помощью коммитов
4) Изменяем файлы репозитория и фиксируем изменения

5. Опишите порядок работы с общим хранилищем VCS.

1) Создаем репозиторий, именуем его или присоединяемся к нему в качестве contributor
2) Добавляем файлы в репозиторий
3) Фиксируем с помощью коммитов
4) Изменяем файлы репозитория и фиксируем изменения
5) Ждем проверки коммитов при участии других пользователей в общем репозитории

6. Каковы основные задачи, решаемые инструментальным средством git?

Систематизация, параллельность разработки программного обеспечения, единое месть для хранения файлов проекта

7. Назовите и дайте краткую характеристику командам git.

- Git init – создание репозитория,
- Git clone – клонирование репозитория,
- Git add – добавление изменений в индекс,
- Git reset – удаление изменений из индекса,
- Git commit – коммиты,
- Git rm – удаление файла.

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

К примеру, я использую локальные репозитории для черновых работ по лабораторным работам, а удаленный репозиторий git для их распространения и оценивания преподавателем.

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

Ветви служат для параллельной разработки программного обеспечения, тестирования, отладки и улучшения

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

Игнорирование можно установить для проекта, компьютера и репозитория, цель игнорирования заключается в том, чтобы не отслеживать файлы служебного типа, например временные файлы сборных утилит для проектов или только те файлы, которые полезны при взаимодействии только с очень ограниченным программным обеспечением