РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

дисциплина: Операционные системы

Выполнила:

Егорова Александра

Группа: НПМбд-02-20

МОСКВА

2021 г.

Цель работы

Изучить идеологию и применение средств контроля версий.

Ход работы

1.Настройка git

1) Создаем учётную запись на https://github.com. (Рис.1)

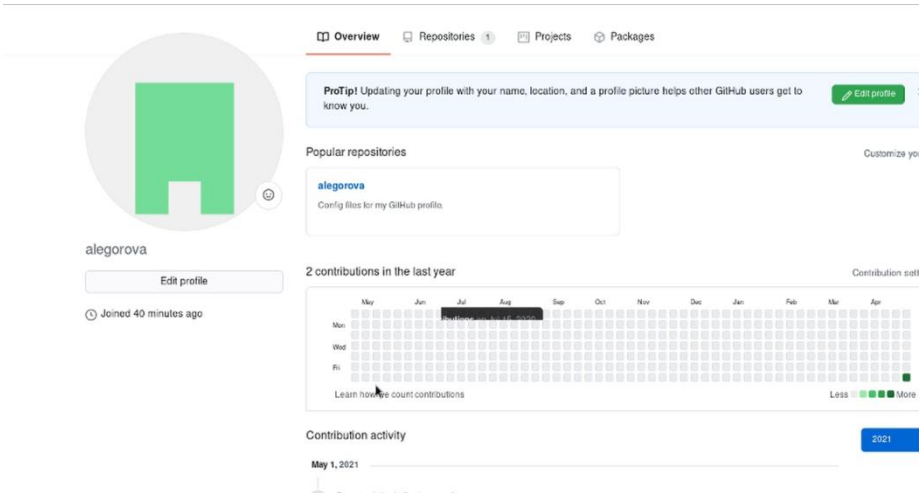

Рис.1

2) Настраиваем систему контроля версий git. Синхранизируем учётную запись github с компьютером (Рис.2):

- git config --global user.name "Имя Фамилия"
- git config --global user.email "work@mail"
-

```
Файл  Правка  Вид  Поиск  Терминал  Справка
[aegorova@aegorova ~]$ git config --global user.name "aegorova"
[aegorova@aegorova ~]$ git config --global user.email "e.alexandra2002@gmail.com"
```

Рис.2

Создаем новый ключ на github (команда ssh-keygen -C "aegorova <e.alexandra2002@gmail.com>") и привязываю его к компьютеру через консоль. (Рис.3, 4, 5, 6)

Рис.3

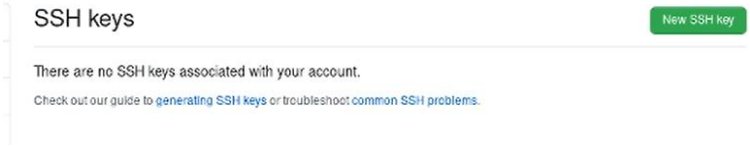

Рис.4

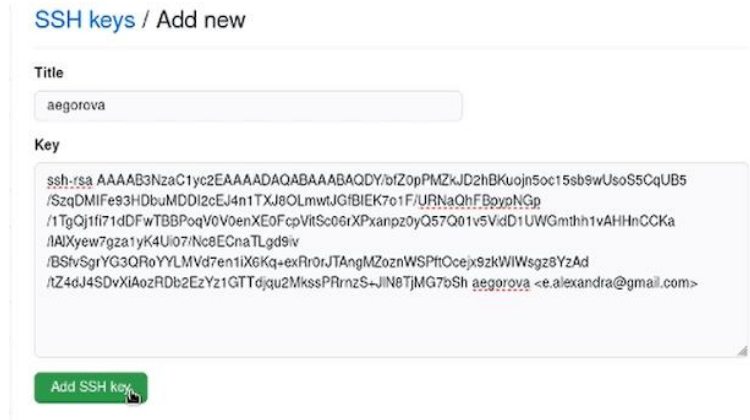

Рис.5

Рис.6

2. Подключение репозитория к github.

- Заходим в «repositories» и создаем новый репозиторий (имя «labor2»). Копируем в консоль ссылку на репозиторий. (Рис.7, 8)

Рис.7

```
[aegorova@aegorova ~]$ git clone https://github.com/alegorova/labor2.git
Cloning into 'labor2'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
```

Рис.8

- Работаем с каталогом и папками через консоль. Создаем файлы. (Рис.9, 10)

```
[aegorova@aegorova ~]$ cd labor2
[aegorova@aegorova labor2]$ mkdir 2020-2021
[aegorova@aegorova labor2]$ cd 2020-2021
[aegorova@aegorova 2020-2021]$ mkdir OS
[aegorova@aegorova 2020-2021]$ cd OS
[aegorova@aegorova OS]$ mkdir laboratory
[aegorova@aegorova OS]$ cd laboratory
[aegorova@aegorova laboratory]$
```

Рис.9

```
[aegorova@aegorova laboratory]$ touch h.txt
```

Рис.10

- Добавляем первый коммит и выкладываем на githup. Чтобы правильно разместить первый коммит, необходимо добавить команду git add . , далее с помощью команды git commit -am "first commit" выкладываем коммит. (Рис.11)

```
[aegorova@aegorova laboratory]$ git add .
[aegorova@aegorova laboratory]$ git commit -am "first commit"
[main 98f3c66] first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 2020-2021/OS/laboratory/h.txt
```

Рис.11

- Сохраняем первый коммит (git push). (Рис.12)

```
[aegorova@aegorova laboratory]$ git push
warning: push.default is unset; its implicit value is changing in
Git 2.0 from 'matching' to 'simple'. To squelch this message
and maintain the current behavior after the default changes, use:

  git config --global push.default matching

To squelch this message and adopt the new behavior now, use:

  git config --global push.default simple

See 'git help config' and search for 'push.default' for further information.
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of Git)

Username for 'https://github.com': alegorova
Password for 'https://alegorova@github.com':
Counting objects: 7, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 406 bytes | 0 bytes/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To https://github.com/alegorova/labor2.git
   5332bfc..98f3c66  main -> main
```

Рис.12

3. Первичная конфигурация

- Добавим файл лицензии. (Рис.13)

```
[aegorova@aegorova laboratory]$ wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O LICENSE
--2021-05-01 23:21:33--  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: «LICENSE»

    [ <=>                                                                   ] 18 657      --.-K/s   за 0s

2021-05-01 23:21:35 (71,3 MB/s) - «LICENSE» сохранён [18657]
```

Рис.13

- Добавим шаблон игнорируемых файлов. Получим список имеющихся шаблонов (на скрине представлены не все шаблоны). (Рис.14)

```
[aegorova@aegorova 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,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
```

Рис.14

- Затем скачаем шаблон, например, для C. Также добавим новые файлы и выполним коммит. (Рис.15)

```
[aegorova@aegorova laboratory]$ curl -L -s https://www.gitignore.io/api/c >> .gitignore
[aegorova@aegorova laboratory]$ git add .
[aegorova@aegorova laboratory]$ git commit -am "first commit"
[main 46ed2b4] first commit
 2 files changed, 455 insertions(+)
 create mode 100644 2020-2021/OS/laboratory/.gitignore
 create mode 100644 2020-2021/OS/laboratory/LICENSE
```

Рис.15

- Отправляем на github (git push). (Рис.16)

```
[aegorova@aegorova laboratory]$ git push
warning: push.default is unset; its implicit value is changing in
Git 2.0 from 'matching' to 'simple'. To squelch this message
and maintain the current behavior after the default changes, use:

  git config --global push.default matching

To squelch this message and adopt the new behavior now, use:

  git config --global push.default simple

See 'git help config' and search for 'push.default' for further information.
(the 'simple' mode was introduced in Git 1.7.11. Use the similar mode
'current' instead of 'simple' if you sometimes use older versions of Git)

Username for 'https://github.com': alegorova
Password for 'https://alegorova@github.com':
Counting objects: 11, done.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (7/7), 6.59 KiB | 0 bytes/s, done.
Total 7 (delta 0), reused 0 (delta 0)
To https://github.com/alegorova/labor2.git
   98f3c66..46ed2b4  main -> main
```

Рис.16

4. Конфигурация git-flow

- Инициализируем git-flow с помощью команды git flow init -f. Префикс для ярлыков установим в v. (Рис.17)

```
[aegorova@aegorova laboratory]$ git flow init -f

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/] v
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? [] v
```

Рис.17

- Проверяем, что находимся на ветке develop (git branch). (Рис.18)

```
[aegorova@aegorova laboratory]$ git branch
* develop
  main
```

Рис.18

- Создадим релиз с версией 1.0.0. (Рис.19)

```
[aegorova@aegorova 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'
```

Рис.19

- Запишем версию и добавим в индекс. (Рис.20)

```
[aegorova@aegorova laboratory]$ echo "1.0.0">> VERSION
[aegorova@aegorova laboratory]$ git add .
[aegorova@aegorova laboratory]$ git commit -am 'chore(main): add version'
[release/1.0.0 6fbbf60] chore(main): add version
 1 file changed, 2 insertions(+)
 create mode 100644 2020-2021/OS/laboratory/VERSION
```

Рис.20

- Зальём релизную ветку в основную ветку. (Рис.21)

```
[aegorova@aegorova laboratory]$ git flow release finish 1.0.0
Switched to branch 'main'
Merge made by the 'recursive' strategy.
 2020-2021/OS/laboratory/VERSION | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 2020-2021/OS/laboratory/VERSION
fatal: no tag message?
Tagging failed. Please run finish again to retry.
```

Рис.21

- Отправим данные на github. (Рис.22)

```
[aegorova@aegorova laboratory]$ git push --all
Username for 'https://github.com': alegorova
Password for 'https://alegorova@github.com':
Counting objects: 11, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 623 bytes | 0 bytes/s, done.
Total 7 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/alegorova/labor2.git
   46ed2b4..787aa68  main -> main
 * [new branch]      develop -> develop
 * [new branch]      release/1.0.0 -> release/1.0.0
[aegorova@aegorova laboratory]$ git push --tags
Username for 'https://github.com': alegorova
Password for 'https://alegorova@github.com':
Everything up-to-date
```

Рис.22

- Создаем релиз на github. Заходим в «Releases», нажимаем «Создать новый релиз». Заходим в теги и заполняем все поля. После создания тега, автоматически сформируется релиз. (Рис.23, 24)

Рис.23

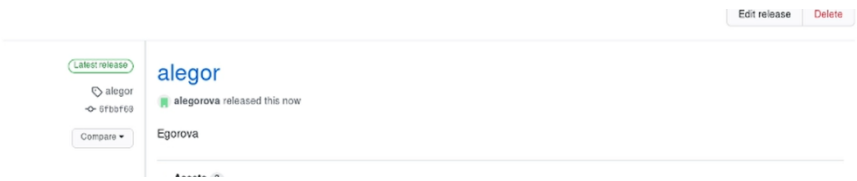

Рис.24

Контрольные вопросы

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

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

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

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) Опишите действия с VCS при единоличной работе с хранилищем.

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) Опишите порядок работы с общим хранилищем VCS.

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый):

ssh-keygen -C"Имя Фамилия <work@mail>"

Ключи сохраняться 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) Каковы основные задачи, решаемые инструментальным средством git?

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) Назовите и дайте краткую характеристику командам git.

Основные 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 изменения: git diff;
- 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;
- 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 натекущей: 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 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) Приведите примеры использования при работе с локальным и удалённым репозиториями.

Использования git при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий):

git add hello.txt

git commit -am ‘Новый файл’

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

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) Как и зачем можно игнорировать некоторые файлы при commit?

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

Вывод:

В ходе выполнения лабораторной работы я изучила идеологию и применение средств контроля версий