Российский университет дружбы народов
Факультет Физико-математический и естественных наук

Операционные системы

Лабораторная работа №2

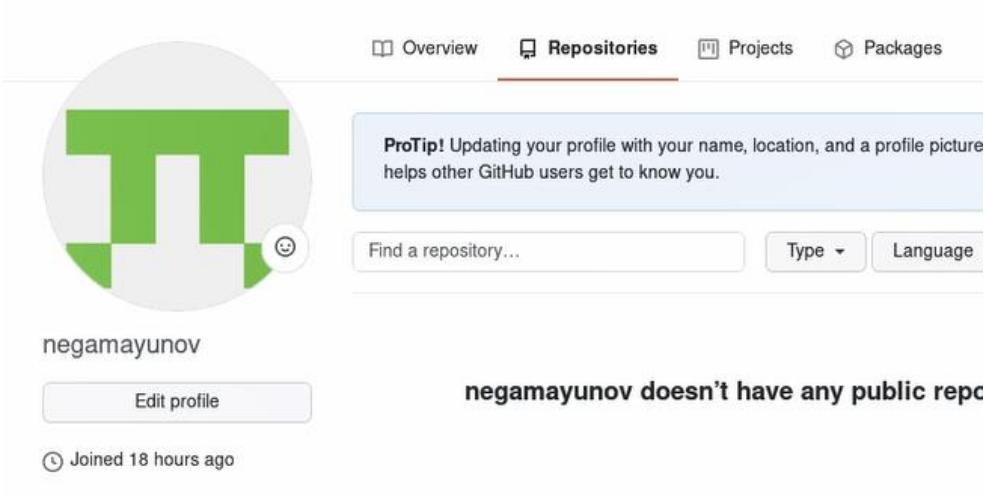
Выполнил Гамаюнов Никита Ефимович,
1032201719, НПМбд-01-20

Москва, 2021 год

Цель работы: изучить идеологию и применение средств контроля версий

Результаты выполнения задания (последовательность выполнения):

1. Создал учетную запись на GitHub:



2. Настроил систему контроля версий git:
   1) Сгенерировал ssh-ключ:

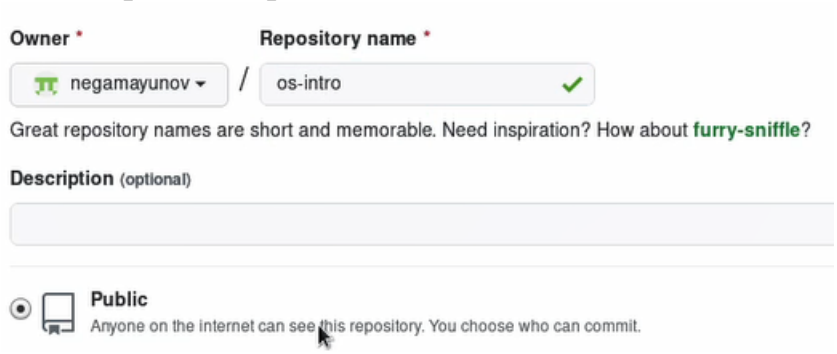
```
[negamayunov@negamayunov ~]$ ssh-keygen -C "negamayunov 1032201719@pfur.ru"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/negamayunov/.ssh/id_rsa):
Created directory '/home/negamayunov/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/negamayunov/.ssh/id_rsa.
Your public key has been saved in /home/negamayunov/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:njjS1zickuZ84A8BujnS1a++dNtoZiLaPKBj/LeIHiI negamayunov 1032201719@pfur.r
u
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|                 |
|    .            |
|   . o           |
|  . . o S        |
| ..+ ..B =       |
|Eo=...X.@ .      |
|=+o++BoB==       |
|o+o++=BO+ .      |
+----[SHA256]-----+
[negamayunov@negamayunov ~]$
```

   2) Добавил его в настройках учетной записи на GitHub:

3. Создал структуру каталога лабораторных работ с помощью mkdir:

```
[negamayunov@negamayunov ~]$ mkdir -p ~/work/2020-20201/OperatingSystems/laboratory
```

4. Создал репозиторий на GitHub. Назвал его os-intro:



5. Перешёл в каталог laboratory:

```
[negamayunov@negamayunov ~]$ cd ~/work/2020-2021/OperatingSystems/laboratory
```

6. Инициализировал системы git:

```
[negamayunov@negamayunov laboratory]$ git init
```

7. Создал заготовку для файла README.md:

```
[negamayunov@negamayunov laboratory]$ echo "# Лабораторные работы" >> README.md
[negamayunov@negamayunov laboratory]$ git add README.md
```

8. Создал первый коммит и отправил его на GitHub:

```
[negamayunov@negamayunov laboratory]$ git commit -m "first commit"
[negamayunov@negamayunov laboratory]$ git remote add origin git@github.com:negamayunov/os-intro.git
[negamayunov@negamayunov laboratory]$ git push -u origin master
```

9. Добавил файл лицензии:

```
[negamayunov@negamayunov laboratory]$ wget https://creativecommons.org/licenses/by/4.0/legalcode.txt
-O LICENSE
--2021-04-24 12:13:53--  https://creativecommons.org/licenses/by/4.0/legalcode.txt
Распознаётся creativecommons.org (creativecommons.org)... 104.20.151.16, 104.20.150.16, 172.67.34.140
, ...
Подключение к creativecommons.org (creativecommons.org)|104.20.151.16|:443... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: нет данных [text/plain]
Сохранение в: «LICENSE»
```

10. Просмотрел список имеющихся шаблонов игнорируемых файлов:

```
[negamayunov@negamayunov laboratory]$ curl -L -s https://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
```

11. Загрузил шаблон для C:

```
zsh,zukencr8000[negamayunov@negamayunov laboratory]$ curl -L -s https://gitignore.io/api/c >> .gitino
re
```

12. Добавил новые файлы, отправил на GutHub:

```
[negamayunov@negamayunov laboratory]$ git add .
[negamayunov@negamayunov laboratory]$ git commit -a
[master e99e192] GNORE
 2 files changed, 455 insertions(+)
 create mode 100644 .gitinore
 create mode 100644 LICENSE
[negamayunov@negamayunov laboratory]$ git push
```

13. Инициализировал git-flow:

```
[negamayunov@negamayunov laboratory]$ git flow init
```

14. Префикс для ярлыков установил в v:

```
Version tag prefix? [] v
```

15. Находясь на ветке develop, создал релиз с версией 1.0.0:

```
[negamayunov@negamayunov laboratory]$ git flow release start 1.0.0
Switched to a new branch 'release/1.0.0'
```

16. Записал версию:

```
[negamayunov@negamayunov laboratory]$ echo "1.0.0" >> VERSION
```

17. Добавил в индекс:

```
[negamayunov@negamayunov laboratory]$ git add .
[negamayunov@negamayunov laboratory]$ git commit -am 'chore(main): add version'
[release/1.0.0 7356bb6] chore(main): add version
 2 files changed, 2 insertions(+)
 create mode 100644 VERSION
 create mode 160000 gitflow
```

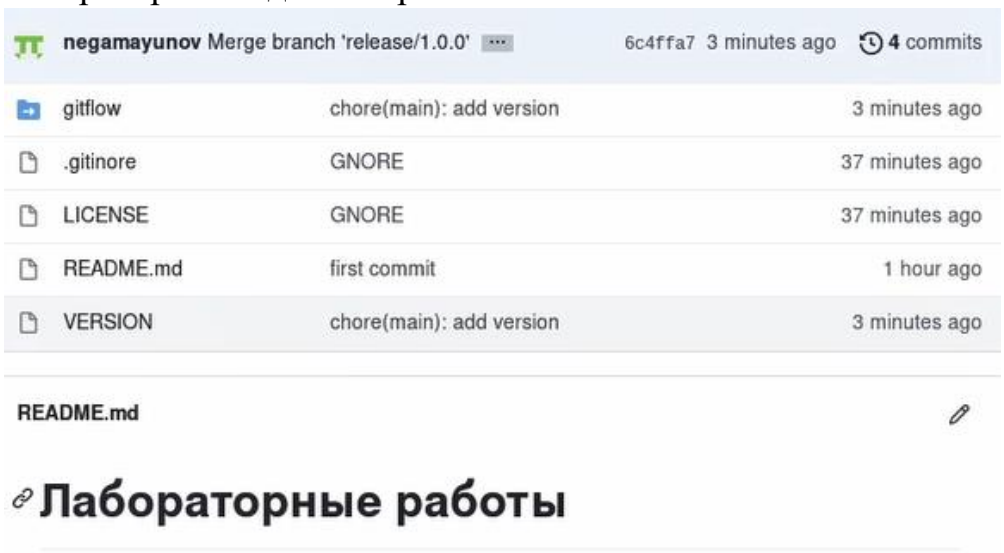
18. Залил релизную ветку в основную ветку:

```
[negamayunov@negamayunov laboratory]$ git flow release finish 1.0.0
```

19.Отправил данные на GitHub:

```
[negamayunov@negamayunov laboratory]$ git push --all
Counting objects: 6, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 602 bytes | 0 bytes/s, done.
Total 5 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To git@github.com:negamayunov/os-intro.git
   e99e192..6c4ffa7  master -> master
 * [new branch]      develop -> develop
[negamayunov@negamayunov laboratory]$ git push --tags
```

20. Проверил созданный релиз на GitHub:



Вывод: я изучил идеологию и применение средств контроля версий.

Ответы на контрольные вопросы:

1. Система контроля версий (VCS) – ПО, облегчающее реализацию удалённой работы нескольких пользователей над одним проектом.
2. Хранилище (репозиторий) – это буквально хранилище для всей информации, которая используется в проекте (код, файлы, изменения в версиях и т.д.)
   Commit – команда, позволяющая сохранить все добавленные изменения.
   История – сортированный по времени список всех когда-либо внесённых изменений.
   Рабочая копия – копия удалённого репозитория, расположенная на устройстве пользователя. Именно её можно изменять и синхронизировать с репозиторием через commit.
3. Централизованная (классическая) VCS представляет собой один репозиторий для хранения файлов. Самые известные примеры: CVS, Subversion.
   Децентрализованная (распределённая) VCS не предполагает обязательного наличия центрального репозитория, её суть в том, что файлы хранятся не на одном сервере, а на устройствах всех участников проекта. Примеры: Git, Bazaar, Mercurial.
4. При единоличной работе с хранилищем на устройстве создаётся локальный репозиторий, в файлы вносятся необходимые изменения, которые через commit отправляются уже в удалённый репозиторий.
5. При работе с общим хранилищем обычно файлы скачиваются из удалённого репозитория, создаётся новая ветка, на которой в файлы так же вносятся и сохраняются с помощью commit нужные изменения. После этого ветка отправляется в удалённый репозиторий, где сливается с основной веткой.
6. Основные задачи git – обеспечение удобства работы с версиями и хранения информации (кода, файлов, etc.)
7. – 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 –
   – 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 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 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. С локальным репозиторием git можно использовать для работы с личными проектами, - система контроля версий позволяет удобно систематизировать файлы и экспериментировать, ведь в случае неудачи можно откатиться на более раннюю версию.
   С удалённым репозиторием удобно осуществлять групповые работы, - помимо возможного отката на ранние версии здесь можно следить за работой каждого из участников, а значит, эффективнее распределять задачи и корректировать проект.
9. Ветвление – это способ хранения проекта. В удалённом репозитории имеется главная ветка – master branch, и каждые его участник, начиная работу, создаёт на ней свою, отдельную ветку. После окончания работы ветку можно слить с главной, перенеся все изменения на неё, или оставить код на проверку другими участниками или администраторами проекта.
   Ветвление удобно, потому что каждый участник проекта может работать обособленно и при этом, если что, сразу загружать себе данные от других пользователей удалённого репозитория.
10. Некоторые файлы не нужно добавлять в проект, - например, файлы редакторов, остаточные файлы и прочий мусор. С помощью gitignore можно удобно выбрать шаблон игнорирования и автоматически избавить себя от необходимости «чистить» репозиторий от ненужный файлов. Список шаблонов можно просмотреть по команде curl -L -s https://www.gitignore.io/api/list.
    После этого можно загрузить нужный шаблон, например, для C: curl -L -s https://www.gitignore.io/api/c >> .gitignore. После этого ненужные файлы, оставляемые редакторами языка C, объектные файлы и т.д. не будут попадать в конечную версию проекта.