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: Юрченко Артём Алексеевич

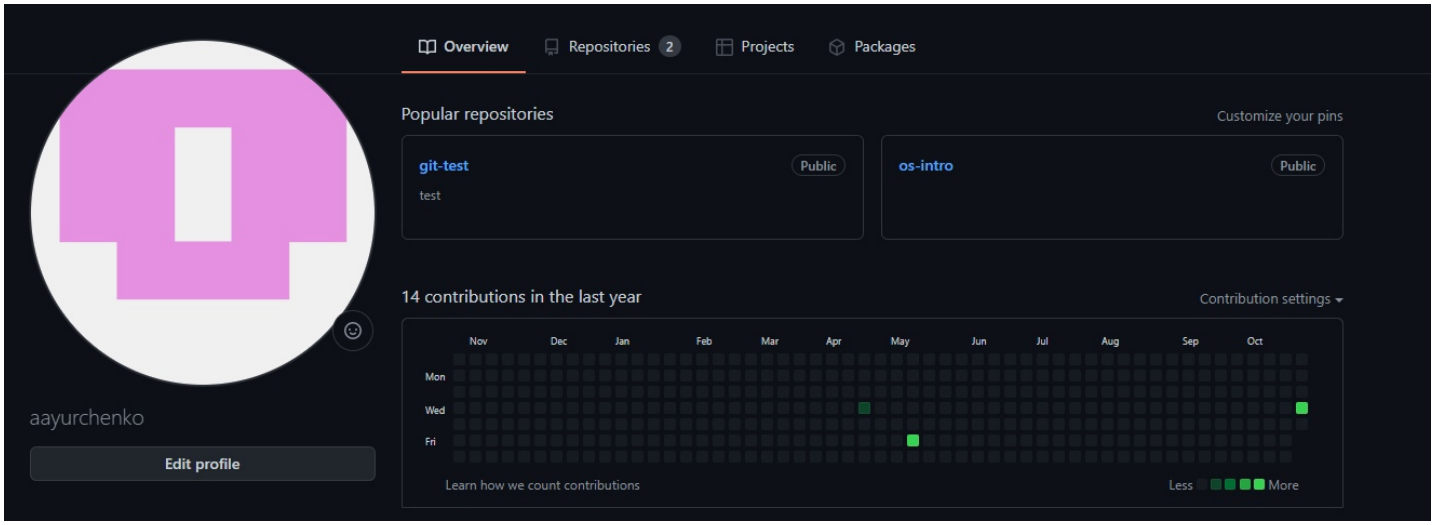
Группа: НФИбд 02-20

Москва

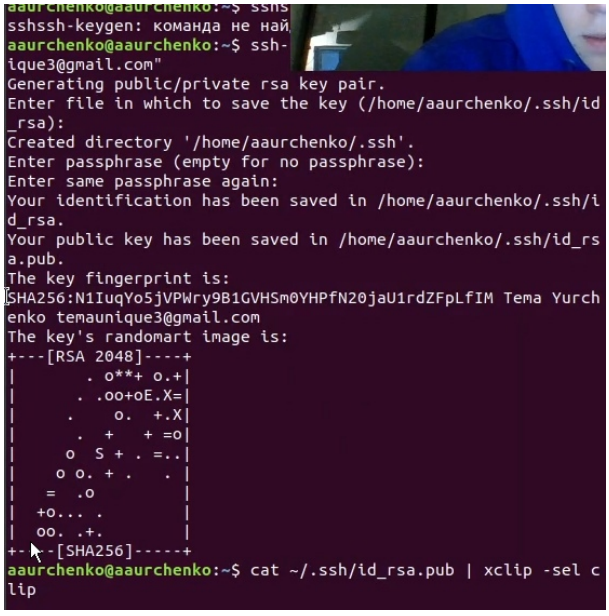
2021

Цель работы: Изучить идеологию и применение средств контроля версий

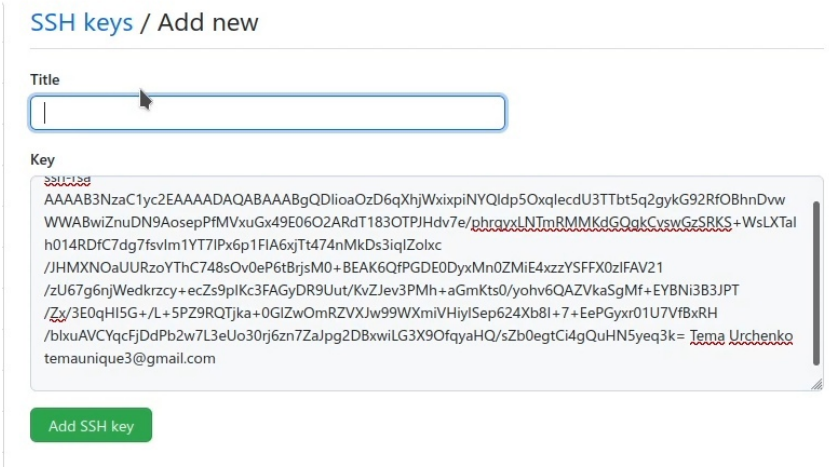
1. Создал учтеную запись на github

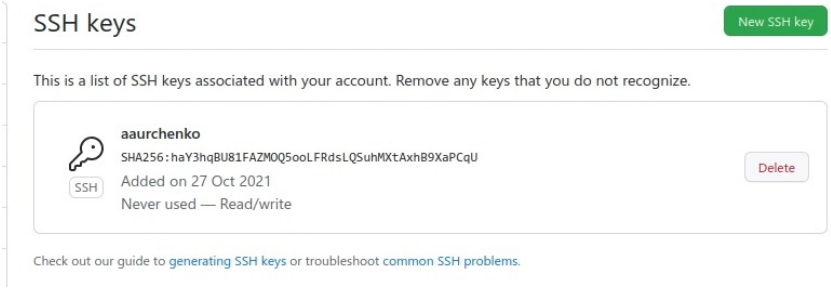


2. Настроил систему контроля версий git, сздал структуру каталога лабораторных работ согласно п. М.2

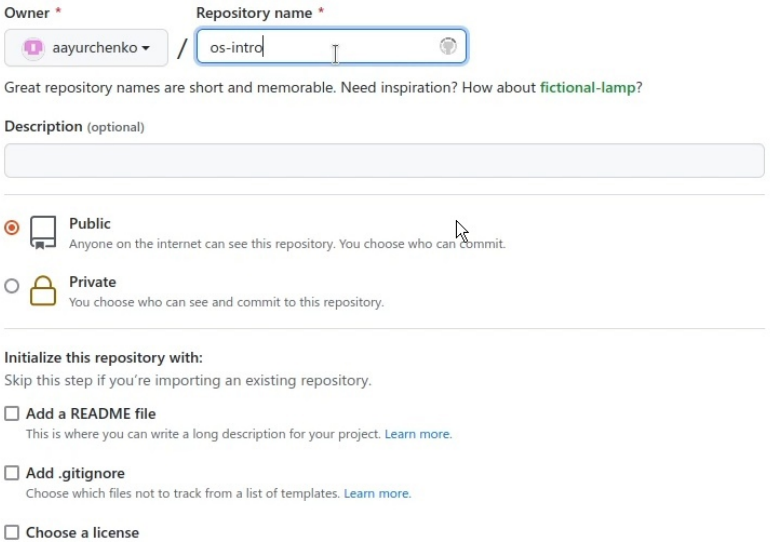


aaurchenko@aaurchenko:~$ cat ~/.ssh/id_rsa.pub | xclip -sel c
lip

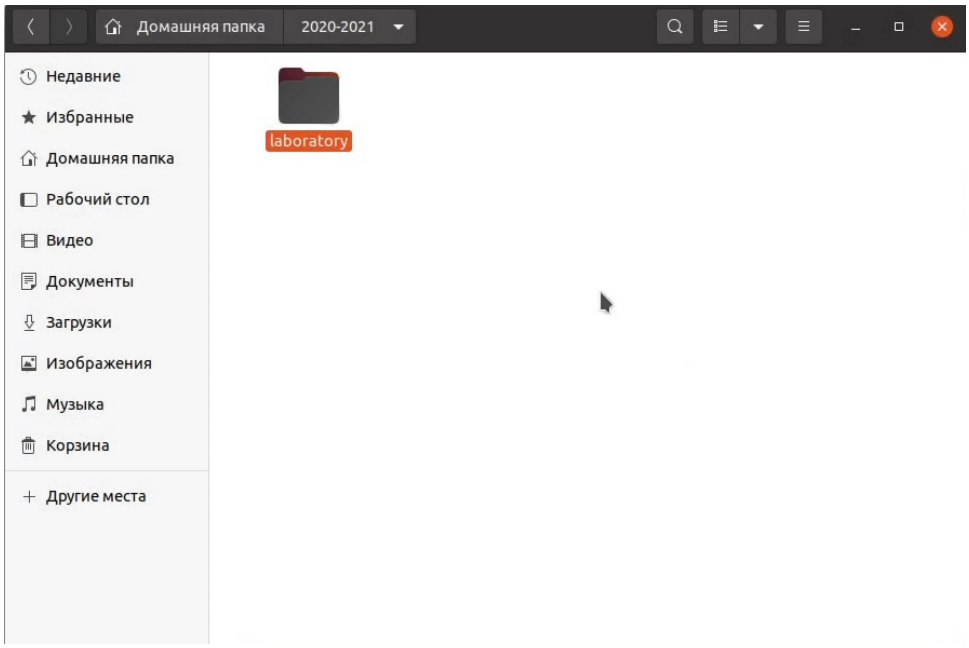




3. Создал репозиторий на github, назвал его os-intro



4. Перешел в каталог laboratory, инициализировал систему git, сделал первый коммит и выложил его на github




```
aaurchenko@aau:~/2020-2021/laboratory$ git init
Инициализирован пустой репозиторий Git в /home/aaurchenko/2020-2021/laboratory/.
git/
aaurchenko@aau:~/2020-2021/laboratory$ echo echo "# Лабораторные работы" >> READ
ME.md
aaurchenko@aau:~/2020-2021/laboratory$ echo "# Лабораторные работы" >> README.md
aaurchenko@aau:~/2020-2021/laboratory$ git add README.md
aaurchenko@aau:~/2020-2021/laboratory$ git commit -m "first commit"
```

5. Создал первичную конфигурацию: добавил файл лицензии, шаблон игнорируемых файлов, новые файлы; выполнил коммит и отправил на github

```
aaurchenko@aau:~/2020-2021/laboratory$ wget https://creativecommons.org/licenses
/by/4.0/legalcode.txt -O LICENSE
--2021-10-27 21:52:28--  https://creativecommons.org/licenses/by/4.0/legalcode.t
xt
Распознаётся creativecommons.org (creativecommons.org)... 104.20.150.16, 104.20.
151.16, 172.67.34.140, ...
Подключение к creativecommons.org (creativecommons.org)|104.20.150.16|:443... со
единение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: нет данных [text/plain]
Сохранение в каталог: ««LICENSE»».

LICENSE                 [ <=>                ]  18,22K  --.-KB/s    за 0s

2021-10-27 21:52:28 (50,8 MB/s) - «LICENSE» сохранён [18657]
```

```
scala,scheme,scons,scrivener,sdcc
seamgen,senchatouch,serverless,shopware,silverstripe
sketchup,slickedit,smalltalk,snap,snapcraft
solidity,soliditytruffle,sonar,sonarqube,sourcepawn
spark,splunk,spreadsheet,ssh,standardml
stata,stdlib,stella,stellar,storybookjs
strapi,stylus,sublimetext,sugarcrm,svn
swift,swiftpackagemanager,swiftpm,symfony,symphonycms
synology,synopsysvcs,tags,tarmainstallmate,terraform
terragrunt,test,testcomplete,testinfra,tex
text,textmate,textpattern,theos-tweak,thinkphp
tla+,tortoisegit,tower,turbogears2,twincat3
tye,typings,typo3,typo3-composer,umbraco
unity,unrealengine,vaadin,vagrant,valgrind
vapor,venv,vertx,video,vim
virtualenv,virtuoso,visualstudio,visualstudiocode,vivado
vlab,vs,vue,vuejs,vvvv
waf,wakanda,web,webmethods,webstorm
webstorm+all,webstorm+iml,werckercli,windows,wintersmith
wordpress,wyam,xamarinstudio,xcode,xcodeinjection
xilinx,xilinxise,xilinxvivado,xill,xojo
xtext,y86,yalc,yarn,yeoman
yii,yii2,zendframework,zephir,zig
zsh,zukencr8000aaurchenko@aau:~/2020-2021/laboratory$
```

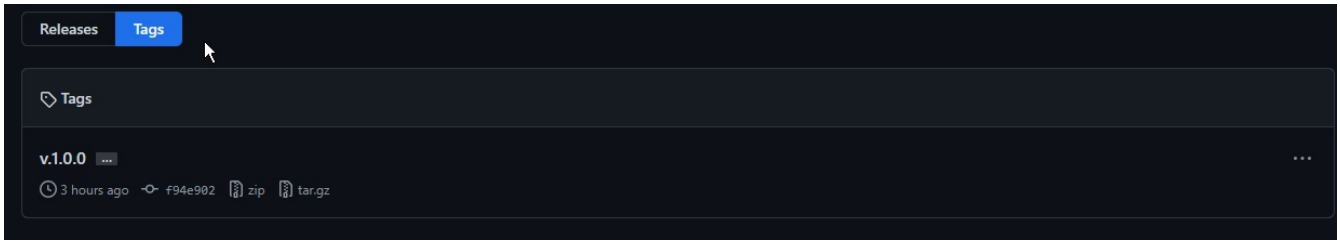
```
zsh,zukencr8000aaurchenko@aau:~/2020-2021/laboratory$ curl -L -s https://www.git
ignore.io/api/c++ >> .gitignore
aaurchenko@aau:~/2020-2021/laboratory$ git add .
aaurchenko@aau:~/2020-2021/laboratory$ git commit -a
```

6. Конфигурация git flow: инициализировал git-flow (установил префикс для ярлыков), создал релиз с версией 1.0.0

```
aaurchenko@aau:~/2020-2021/laboratory$ git flow init

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

How to name your supporting branch prefixes?
Feature branches? [feature/] v.
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? [] v.
Hooks and filters directory? [/home/aaurchenko/2020-2021/laboratory/.git/hooks]
aaurchenko@aau:~/2020-2021/laboratory$ git branch
* develop
  master
```

```
aaurchenko@aau:~/2020-2021/laboratory$ 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'

aaurchenko@aau:~/2020-2021/laboratory$ echo "1.0.0" >> VERSION
aaurchenko@aau:~/2020-2021/laboratory$ git add .
aaurchenko@aau:~/2020-2021/laboratory$ git commit -am 'chore(main): add version'
[release/1.0.0 d37f3f1] chore(main): add version
 1 file changed, 1 insertion(+)
 create mode 100644 VERSION
```

```
aaurchenko@aau:~/2020-2021/laboratory$ git flow release finish 1.0.0
Переключено на ветку «master»
Ваша ветка обновлена в соответствии с «origin/master».
Merge made by the 'recursive' strategy.
 VERSION | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 VERSION
Уже на «master»
Ваша ветка опережает «origin/master» на 2 коммита.
  (используйте «git push», чтобы опубликовать ваши локальные коммиты)
fatal: нет описания метки?
Fatal: Tagging failed. Please run finish again to retry.
aaurchenko@aau:~/2020-2021/laboratory$ git push --all
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
Сжатие объектов: 100% (3/3), готово.
Запись объектов: 100% (4/4), 378 байтов | 378.00 КиБ/с, готово.
Всего 4 (изменения 2), повторно использовано 0 (изменения 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To github.com:aayurchenko/os-intro.git
   44ea864..93086af  master -> master
 * [new branch]      develop -> develop
 * [new branch]      release/1.0.0 -> release/1.0.0
aaurchenko@aau:~/2020-2021/laboratory$
```



Вывод: Я изучил идеологию и применение средств контроля версий

Ответы на контрольные вопросы

1. Система контроля версий (Version Control System, VCS) — программное обеспечение для облегчения работы с изменяющейся информацией. VCS позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

2. Хранилище – место хранения файлов и их версий, служебной информации.

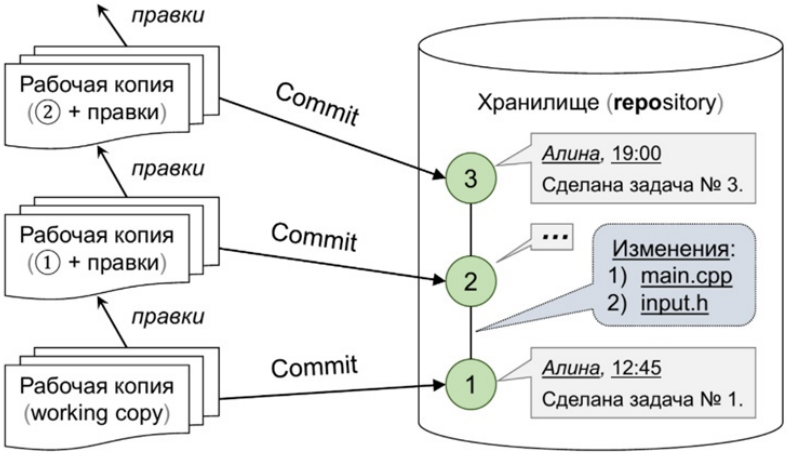
Commit – процесс создания новой версии; иногда синоним версии. История – одна из самых важных частей git, где сохраняются все коммиты, по которым можно посмотреть автора коммита, commite message, дфту коммита и его хэш. Рабочая копия – текущее состояние файлов проекта (любой версии), полученных из хранилища и, возможно, измененных.

3. Централизованные системы контроля версий предполагают сохранение версий проектов на общий сервер, с которого потом получают нужные версии клиенты.

В децентрализованных системах контроля версий при каждом копировании удалённого репозитория (расположенного на сервере) происходит полное копирование данных в локальный репозиторий (установленный на рабочем компьютере). Каждая копия содержит все данные, хранящиеся в удалённом репозитории. В случае возникновения технической неисправности на стороне сервера, удаленный репозиторий можно перезаписать с любой сохраненной копии. Примеры распределенных VCS: Git.

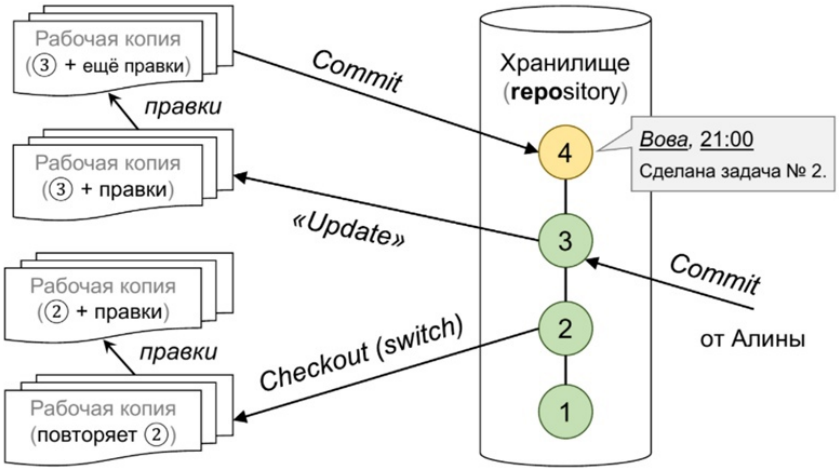
4:

Единоличная работа с VCS



5:

Работа с общим хранилищем



6. У Git есть две основные задачи: хранить информацию обо всех изменениях в коде, начиная с самой первой строчки, и обеспечить удобства командной работы над кодом.

7. Команда git add добавляет содержимое рабочей директории в индекс (staging area) для последующего коммита. По умолчанию git commit использует лишь этот индекс, так что вы можете использовать git add для сборки слепка вашего следующего коммита.

Команда git status показывает состояния файлов в рабочей директории и индексе: какие файлы изменены, но не добавлены в индекс; какие ожидают коммита в индексе. Вдобавок к этому выводятся подсказки о том, как изменить состояние файлов.

Команда git diff используется для вычисления разницы между любыми двумя Git деревьями. Это может быть разница между вашей рабочей директорией и индексом (собственно git diff), разница между индексом и последним коммитом (git diff –staged), или между любыми двумя коммитами (git diff master branchB)

Команда git reset, как можно догадаться из названия, используется в основном для отмены изменений. Она изменяет указатель HEAD и, опционально, состояние индекса. Также эта команда может изменить файлы в рабочей директории при использовании параметра –hard, что может привести к потере наработок при неправильном использовании, так что убедитесь в серьёзности своих намерений прежде чем использовать его.

Команда git rm используется в Git для удаления файлов из индекса и рабочей директории.

Команда git mv — это всего лишь удобный способ переместить файл, а затем выполнить git addдля нового файла и git rm для старого

Команда git tag используется для задания постоянной метки на какой-либо момент в истории проекта. Обычно она используется для релизов.

9. Ветка – это просто «скользящий» указатель на один из коммитов. Когда мы создаём новые коммиты, указатель ветки автоматически сдвигается вперёд, к вновь созданному коммиту. Ветки используются для разработки одной части функционала изолированно от других. Каждая ветка представляет собой отдельную копию кода проекта. Ветки позволяют одновременно работать над разными версиями проекта.

10. .gitignore – это простой текстовый файл, в каждой строке которого содержится шаблон файла или каталога, который необходимо проигнорировать. Строки, начинающиеся со знака #, являются комментариями и игнорируются. Пустые строки могут быть использованы для улучшения читабельности файла и группировки связанных строк шаблонов.