Титульный лист

Отчёт по лабораторной работе №2 Ибрагимов Улугбек Ботырхонович

Цель работы

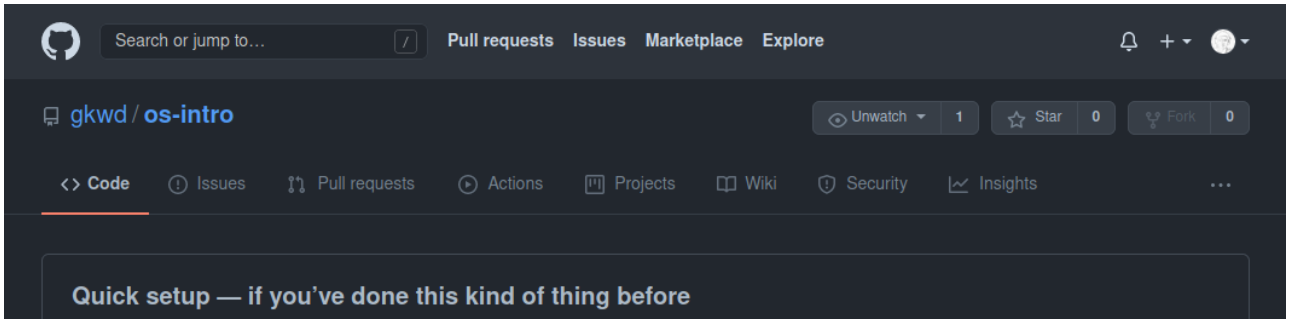
Изучить идеологию и применение средств контроля версий.

Задание

1. Создайте учётную запись на https://github.com.
2. Настройте систему контроля версий git, как это описано выше с использованием сервера репозиториев https://github.com/.
3. Создайте структуру каталога лабораторных работ согласно пункту М.2.
4. Подключите репозиторий: 4.1. Создайте репозиторий на GitHub. Для примера назовём его os-intro. 4.2. Рабочий каталог будем обозначать как laboratory. Вначале нужно перейти в этот каталог: `cd laboratory` 4.3. Инициализируем системы git: `git init` 4.4. Создаём заготовку для файла `README.md`: `echo "# Лабораторные работы" >> README.md git add README.md` 4.5. Делаем первый коммит и выкладываем на github: `git commit -m "first commit" git remote add origin ↪ git@github.com:<username>/sciproc-intro.git git push -u origin master`
5. Первичная конфигурация: 5.1. Добавим файл лицензии: `wget https://creativecommons.org/licenses/by/4.0/legalcode.txt ↪ -O LICENSE` 5.2. Добавим шаблон игнорируемых файлов. Просмотрим список имеющихся шаблонов: `curl -L -s https://www.gitignore.io/api/list` Затем скачаем шаблон, например, для C: `curl -L -s https://www.gitignore.io/api/c >> .gitignore` 5.3. Можно это же сделать через web-интерфейс на сайте `https://www.gitignore.io/.` 5.4. Добавим новые файлы: `git add .` 5.5. Выполним коммит: `git commit -a` 5.6. Отправим на github: `git push`
6. Конфигурация git-flow: 6.1. Инициализируем git-flow `git flow init` Префикс для ярлыков установим в v. 6.2. Проверьте, что Вы на ветке `develop`: `git branch` 6.3. Создадим релиз с версией 1.0.0 `git flow release start 1.0.0` 6.4. Запишем версию: `echo "1.0.0" >> VERSION` 6.5. Добавим в индекс: `git add . git commit -am 'chore(main): add version'` 6.6 Зальём релизную ветку в основную ветку `git flow release finish 1.0.0` 6.7 Отправим данные на github `git push --all git push --tags` 6.8 Создадим релиз на github

Ход выполнения работы

1. Создали репозиторий `os-intro`



2. Создаем рабочий каталог `laboratory`, переходим в него

```
ubibragimov@ubibragimov-VirtualBox:~/os-intro$ mkdir laboratory
ubibragimov@ubibragimov-VirtualBox:~/os-intro$ cd laboratory
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ git init
Инициализирован пустой репозиторий Git в /home/ubibragimov/os-intro/laboratory/.git/
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ echo "#Лабораторные работы" >> README.md
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ git add README.md
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ git commit -m "first commint"
```

)

3. Инициализируем системы Git, делаем первый коммит в виде файла к README.md

```
ubibragimov@ubibragimov-VirtualBox:~/os-intro$ mkdir laboratory
ubibragimov@ubibragimov-VirtualBox:~/os-intro$ cd laboratory
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ git init
Инициализирован пустой репозиторий Git в /home/ubibragimov/os-intro/laboratory/.git/
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ echo "#Лабораторные работы" >> README.md
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ git add README.md
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ git commit -m "first commint"
```

) и

```
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ git commit -m "first commint"
[master (корневой коммит) 319cbce] first commint
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

)

4. Выкладываем на git

```
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ git remote add origin git@github.com:<gkwd>/os-intro.git
bash: gkwd: Нет такого файла или каталога
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ git remote add origin git@github.com:gkwd/os-intro.git
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ git push -u origin master
The authenticity of host 'github.com (140.82.121.4)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'github.com,140.82.121.4' (RSA) to the list of known hosts.
Перечисление объектов: 3, готово.
Подсчет объектов: 100% (3/3), готово.
Запись объектов: 100% (3/3), 245 bytes | 245.00 KiB/s, готово.
Всего 3 (изменения 0), повторно использовано 0 (изменения 0)
To github.com:gkwd/os-intro.git
 * [new branch]      master -> master
Ветка «master» отслеживает внешнюю ветку «master» из «origin».
```

)

5. Добавляем лицензию

```
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ wget https://creativecommons.org/licenses/by/4.0/legalcode.t
xt -O LICENSE
--2021-05-01 23:50:11--  https://creativecommons.org/licenses/by/4.0/legalcode.txt
Распознаётся creativecommons.org (creativecommons.org)… 104.20.150.16, 172.67.34.140, 104.20.151.16, ...
Подключение к creativecommons.org (creativecommons.org)|104.20.150.16|:443... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа… 200 OK
Длина: нет данных [text/plain]
Сохранение в: «LICENSE»

LICENSE                         [ <=>                                       ]  18.22K  --.-KB/s    за 0.001s

2021-05-01 23:50:12 (34.8 MB/s) - «LICENSE» сохранён [18657]
```

)

6. Скачиваем и добавляем gitignore

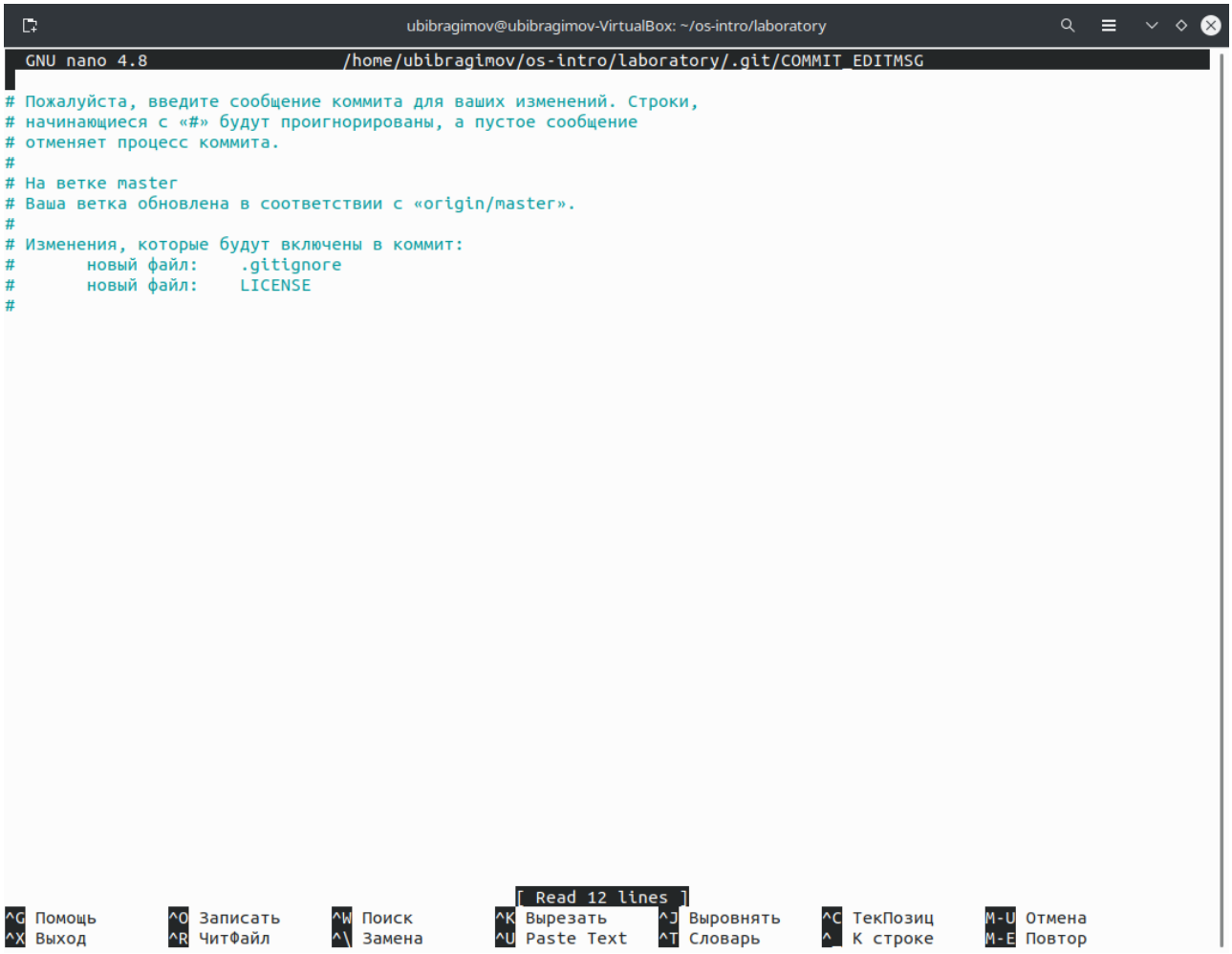
```
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ curl -L -s https://www.gitignore.io/api/list
1c,1c-bitrix,a-frame,actionscript,ada
adobe,advancedinstaller,adventuregamestudio,agda,al
alteraquartusii,altium,amplify,android,androidstudio
angular,anjuta,ansible,apachecordova,apachehadoop
appbuilder,appceleratortitanium,appcode,appcode+all,appcode+iml
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
```

)

```
vapor,venv,vertx,video,vim
virtualenv,virtuoso,visualstudio,visualstudiocode,vivado
vlab,vs,vscode,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,yarn,yeoman
yii,yii2,zendframework,zephir,zig
zsh,zukencr8000ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ curl -L -s https://www.gitignore.io/api/c >>
.gitignore
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$
```

)

7. Публикуем коммиты



)

```
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ git commit -a
[master 9139858] Hello world
 2 files changed, 455 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 LICENSE
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ git push
Warning: Permanently added the RSA host key for IP address '140.82.121.3' to the list of known hosts.
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (4/4), готово.
Запись объектов: 100% (4/4), 6.43 KiB | 3.21 MiB/s, готово.
Всего 4 (изменения 0), повторно использовано 0 (изменения 0)
To github.com:gkwd/os-intro.git
   319cbce..9139858  master -> master
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$
```

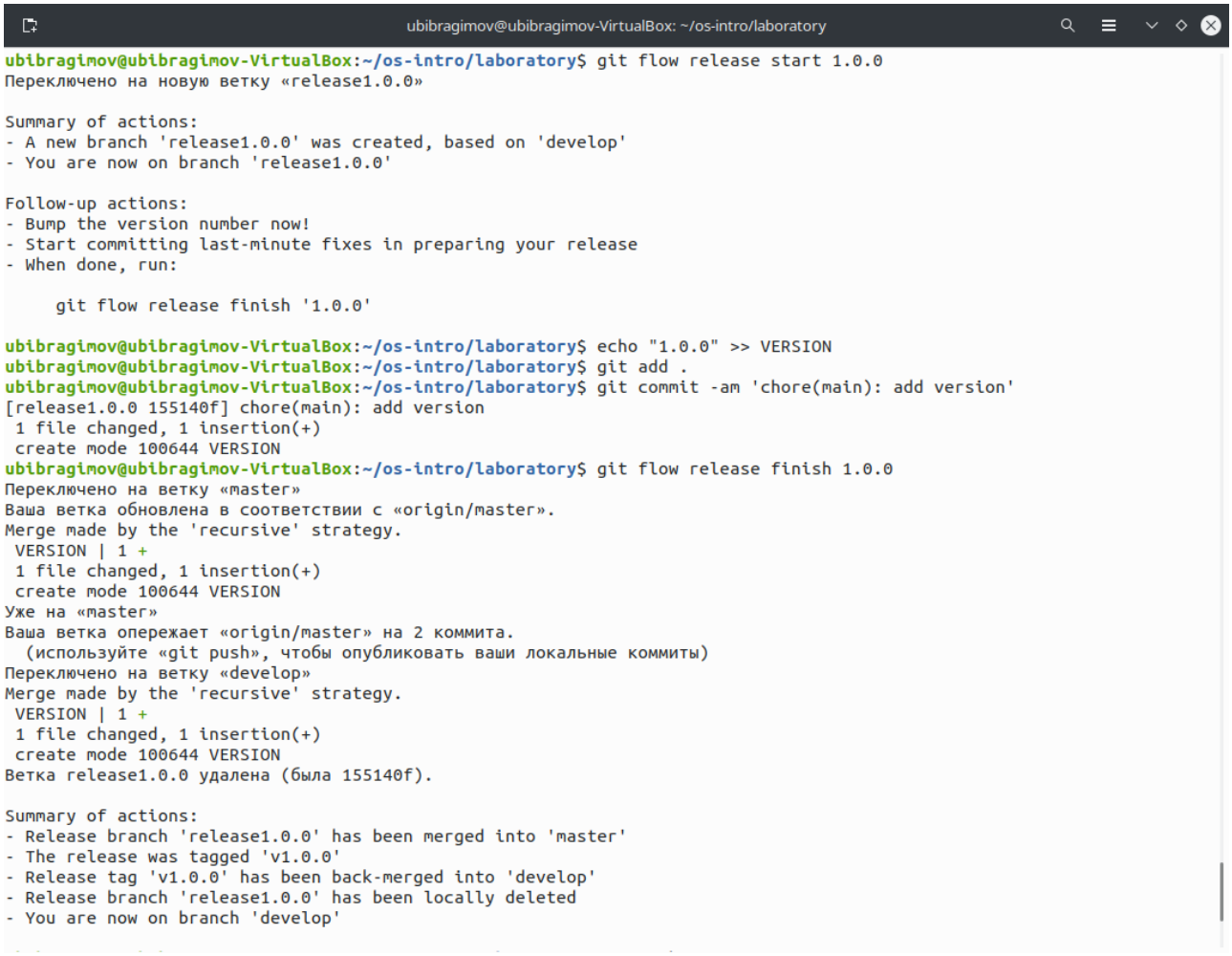
)

8. Инициализурем git-flow и потоки параллельной разработки

```
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ git branch
* master
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ git flow init

Which branch should be used for bringing forth production releases?
   - master
Branch name for production releases: [master] develop
Fatal: Local branch 'develop' does not exist.
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ 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/] feature
Bugfix branches? [bugfix/] hostfix
Release branches? [release/] release
Hotfix branches? [hotfix/] hotfix
Support branches? [support/] support
Version tag prefix? [] v
Hooks and filters directory? [/home/ubibragimov/os-intro/laboratory/.git/hooks]
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$
```

)

9. Создаем релиз 1.0.0



)

10. Отправим на git

```
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ git push --all
Перечисление объектов: 6, готово.
Подсчет объектов: 100% (6/6), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (4/4), готово.
Запись объектов: 100% (5/5), 488 bytes | 488.00 KiB/s, готово.
Всего 5 (изменения 3), повторно использовано 0 (изменения 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To github.com:gkwd/os-intro.git
   9139858..6d8dd52  master -> master
 * [new branch]      develop -> develop
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$ git push --tags
Перечисление объектов: 1, готово.
Подсчет объектов: 100% (1/1), готово.
Запись объектов: 100% (1/1), 154 bytes | 154.00 KiB/s, готово.
Всего 1 (изменения 0), повторно использовано 0 (изменения 0)
To github.com:gkwd/os-intro.git
 * [new tag]         v1.0.0 -> v1.0.0
ubibragimov@ubibragimov-VirtualBox:~/os-intro/laboratory$
```

)

Выводы

В ходе выполнения Лабораторной работы №2, были приобретены навыки по администрированию и взаимодействию с децентрализованной системой контроля версий и программой git для параллельной разработки и поддержки программного кода.

Ответы на контрольные вопросы

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

> Ответ: Системы контроля версий (VCS) применяются при работе 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.

> Ответ:
>
> - Хранилище - общее пространство для хранения файлов
> - Commit - команда для записииндексированных изменений в репозиторий
> - История - в истории сохраняются все коммиты, по которым можно отследить автора, сообщение, дату и хэш коммита
> - Рабочая копия - все файлы кроме `.git/` называются рабочей копией, и принадлежать пользователю (-лям)

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

> Ответ: Централизованные системы контроля версий - сохраняют проект и его файлы на один общий сервер, децентрализованные системы контроля версий - при каждом копировании данных удаленного репозитория, происходит полное копирование данных в локальный репозиторий. Пример ЦСКВ - SVN, MS TFS, ClearCase; ДСКВ - Git, Mercurial, Bazaar.

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

> Ответ:
>
> 1. Создаем репозиторий, 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.

> Ответ:
>
> 1. Создаем репозиторий, именуем его или присоединяемся к нему в качестве `contibutor`
> 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?

> Ответ: Систематизация, параллельность разработки программного обеспечения, единое место для хранение файлов проекта

7. Назовите и дайте краткую характеристику командам git.

> Ответ: Создание репозитория (`git init`), Клонирование репозитория (`git clone`), Добавление изменений в индекс (`git add`), Удаление изменений из индекса (`git reset`), Коммиты (`git commit`), Удаление файла (`git rm`).

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

> Ответ: Для написания черновых работ по лабораторным работам я использую локальные репозитории, для их распространения или для оценивания преподователем я использую удаленный репозиторий `git`

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

> Ответ: Ветви служат для паралельной разработки программного обеспечения, тестирования, отладки и улучшения

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

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