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: Бабина Юлия Олеговна

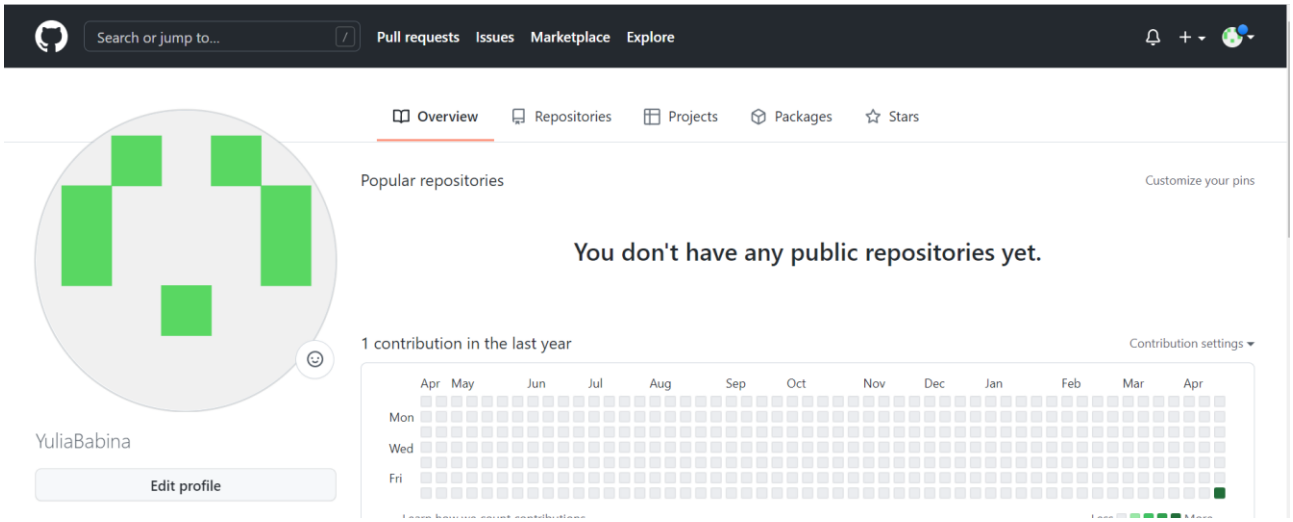
Группа: НПМбд-01-21

МОСКВА

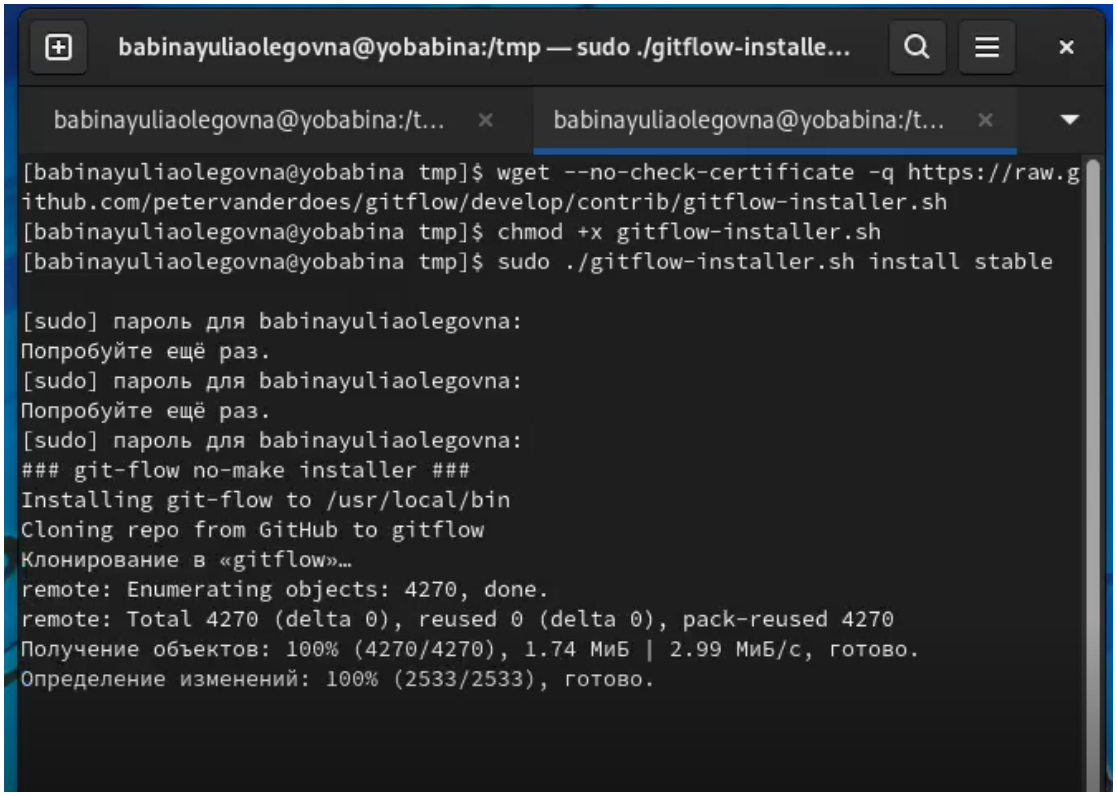
2022г.

1. Цель работы: Изучить идеологию и применение средств контроля версий. Освоить умения по работе с git.

2. Ход работы:
Создаем учетную запись на https://github.com.



Установим git-flow в Fedora Linux при помощи терминала данных команд:
cd /tmp
wget --no-check-certificate -q https://raw.github.com/petervanderdoes ↪
↪/gitflow/develop/contrib/gitflow-installer.sh
chmod +x gitflow-installer.sh
sudo ./gitflow-installer.sh install stable



Синхронизируем учётную запись github с компьютером:

git config --global user.name "YuliaBabina"

git config --global user.email "iuliiare03@gmail.com"

```
[babinayuliaolegovna@yobabina tmp]$ git config --global user.name "YuliaBabina"
[babinayuliaolegovna@yobabina tmp]$ git config --global user.email "iuliiare03@g
```

Произведем другие базовые настройки:

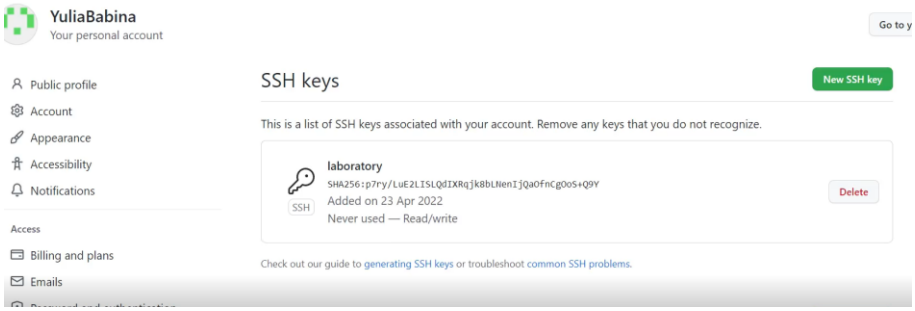
- Настроим utf-8 в выводе сообщений git
- Настроим верификацию и подписание коммитов git
- Зададим имя начальной ветки (будем называть её master)
- Параметр autocrlf
- Параметр safecrlf

```
[babinayuliaolegovna@yobabina tmp]$ git config --global core.quotepath false
[babinayuliaolegovna@yobabina tmp]$ git config --global init.defaultBranch maste
r
[babinayuliaolegovna@yobabina tmp]$ git config --global core.autocrlf input
[babinayuliaolegovna@yobabina tmp]$ git config --global core.safecrlf warn
```

Создадим ключ ssh.

```
[babinayuliaolegovna@yobabina tmp]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/babinayuliaolegovna/.ssh/id_rsa):
Created directory '/home/babinayuliaolegovna/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/babinayuliaolegovna/.ssh/id_rsa
Your public key has been saved in /home/babinayuliaolegovna/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:p7ry/LuE2LISLQdIXRqjk8bLNenIjQaOfnCgOoS+Q9Y babinayuliaolegovna@yobabina
The key's randomart image is:
+---[RSA 4096]----+
|  .o..           |
|..o.=            |
|oO =             |
|@ @ .            |
|+@.*    S .      |
|*o=Eoo . o       |
|=o =o o o        |
| o+ .+ o         |
| ....o=o+o       |
+----[SHA256]-----+
[babinayuliaolegovna@yobabina tmp]$
```

После этого скопируем ключ в буфер обмена, с помощью команды cat ~/.ssh/id_rsa.pub | xclip -sel clip. Добавляем его в поле “SSH Keys” в разделе “Settings” → SSH and GPG Keys.



Создадим ключ gpg.

```
Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? 0
Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? o
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
gpg: /home/babinayuliaolegovna/.gnupg/trustdb.gpg: создана таблица доверия
gpg: ключ C19B488B7ABCC637 помечен как абсолютно доверенный
gpg: создан каталог '/home/babinayuliaolegovna/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/babinayuliaolegovna/.gnupg/openpgp-revoc
s.d/AE035569CFA8B6D2041ED167C19B488B7ABCC637.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2022-04-23 [SC]
      AE035569CFA8B6D2041ED167C19B488B7ABCC637
uid                      yobabina <iuliiare03@gmail.com>
sub   rsa4096 2022-04-23 [E]
```

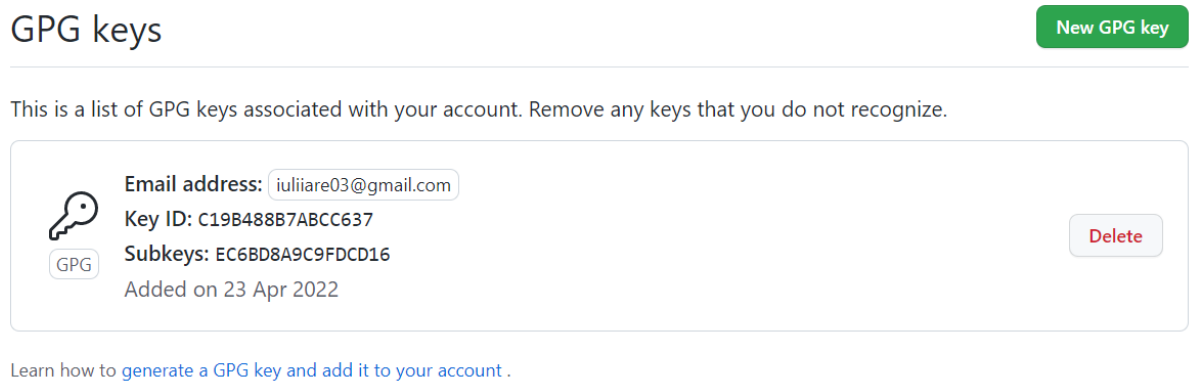
Далее воспользуемся командой *gpg2 --list-keys --keyid-format LONG*, чтобы перечислить длинную форму ключей gpg, для которых есть открытый и закрытый доступ.

```
[babinayuliaolegovna@yobabina tmp]$ gpg2 --list-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: глубина: 0  достоверных:   1  подписанных:   0  доверие: 0-, 0q, 0n, 0m, 0f
, 1u
/home/babinayuliaolegovna/.gnupg/pubring.kbx
------------------------------------------------
pub   rsa4096/C19B488B7ABCC637 2022-04-23 [SC]
      AE035569CFA8B6D2041ED167C19B488B7ABCC637
uid                 [  абсолютно ] yobabina <iuliiare03@gmail.com>
sub   rsa4096/EC6BD8A9C9FDCD16 2022-04-23 [E]
```

Воспользуемся командой gpg –amor –export C19B488B7ABCC637 | xclip –sel clip, чтобы скопировать ключ в буфер обмена.

```
[babinayuliaolegovna@yobabina tmp]$ gpg --armor --export C19B488B7ABCC637 | xcli
p -sel clip
```

Добавим ключ в поле “GPG Keys” в раздеде “Settings” → “SSH and GPG Keys”.



Настроим автоматические подписи коммитов git.

```
[babinayuliaolegovna@yobabina tmp]$ git config --global commit.gpgsign true
[babinayuliaolegovna@yobabina tmp]$ git config --global gpg.program $(which gpg2
)
```

Произведем авторизацию и настройку gh.

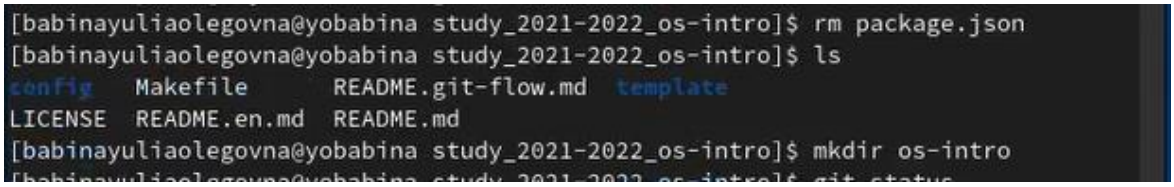
```
[babinayuliaolegovna@yobabina tmp]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Paste an authentication token
Tip: you can generate a Personal Access Token here https://github.com/settings/t
okens
The minimum required scopes are 'repo', 'read:org', 'workflow'.
? Paste your authentication token: ****************************************
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as YuliaBabina
```

Создадим папку для локального репозитория, после чего скопируем шаблон в глобальный репозиторий, а из глобального скопируем в локальный.

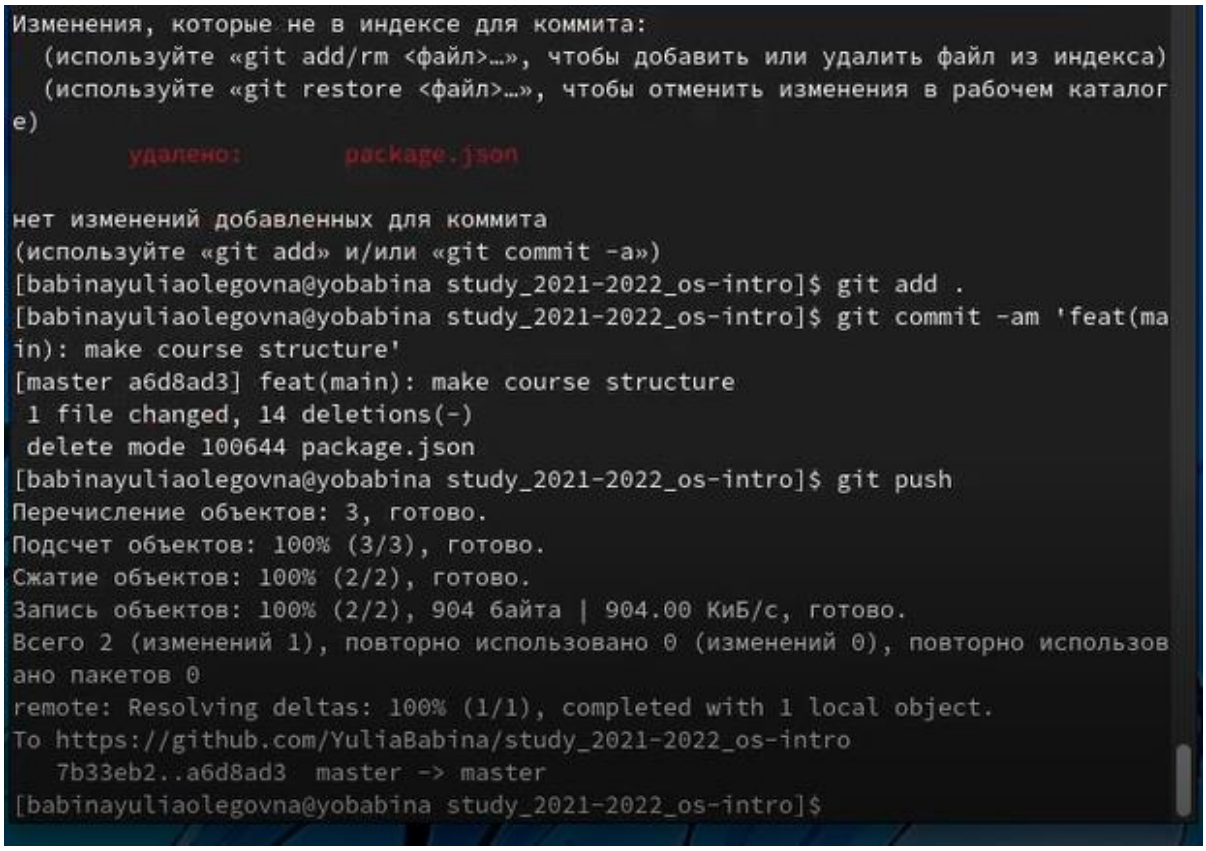
```
[babinayuliaolegovna@yobabina tmp]$ cd ~/work/study/2021-2022/"Операционные сист
емы"
[babinayuliaolegovna@yobabina Операционные системы]$ gh repo create study_2021-2
022_os-intro --template=yamadharma/course-directory-student-template --public
```

```
[babinayuliaolegovna@yobabina Операционные системы]$ git clone https://github.co
m/YuliaBabina/study_2021-2022_os-intro
Клонирование в «study_2021-2022_os-intro»…
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 20 (delta 2), reused 15 (delta 2), pack-reused 0
Получение объектов: 100% (20/20), 12.49 КиБ | 12.49 МиБ/с, готово.
Определение изменений: 100% (2/2), готово.
```

В локальном репозитории создадим удалим файл с расширением json и создадим новый каталог.



Далее добавим файлы в фазу сохранения(git add .), произведем сохранение (git commit – am “message”) и отправим версию в глобальный репозиторий(git push).



3. Контрольные вопросы:

1)Система контроля версий Git реализована в виде набора программ командной строки. Команды имеют следующий синтаксис: git <опция>. Системы контроля версий (VCS) применяются во время работы нескольких человек над одним проектом.

2) Хранилище – сервер, куда пользователь размещает новую версию проекта. При этом предыдущие версии не удаляются из центрального хранилища; к ним можно вернуться в любой момент. Сервер сохраняет только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Коммит – команда сохранения определенной версии в локальном репозитории в рамках системы управления версиями Git.

История версии содержит информацию об изменениях и служебную информацию.

Рабочая копия - созданная клиентской программой локальная копия части данных из хранилища.

3) В основе централизованных систем лежит архитектура клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу.(Wikipedia)

В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. (Bitcoin)

4) 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) 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.

Кроме этого, для отправки изменений на сервер используется команда git push.

6) В рамках Git решаются две основные задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.

7) Основные команды git:

Наиболее часто используемые 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–сохранение текущих изменений:–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 – сохранение добавленных изменений: – 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 стекущим деревом:git merge --no-ff имя_ветки–удаление 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) Использования git при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий):

git add hello.txt

git commit -am 'Новый файл’

9) 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) 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 списоки меняющихся 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

4. Вывод: В ходе данной лабораторной работы я изучила идеологию и применение средств контроля версий. Освоила умения по работе с git.