

**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:** Филиппова Анна

**Группа:** НПМбд-02-20

**МОСКВА**

2021г.

## 1.Цель работы:

Целью данной работы является изучение идеологии и применение средств контроля версий.

## 2. Ход работы:

Создаем учётную запись на <https://github.com>. Учетная запись называется adfilippova. (Рис.1)

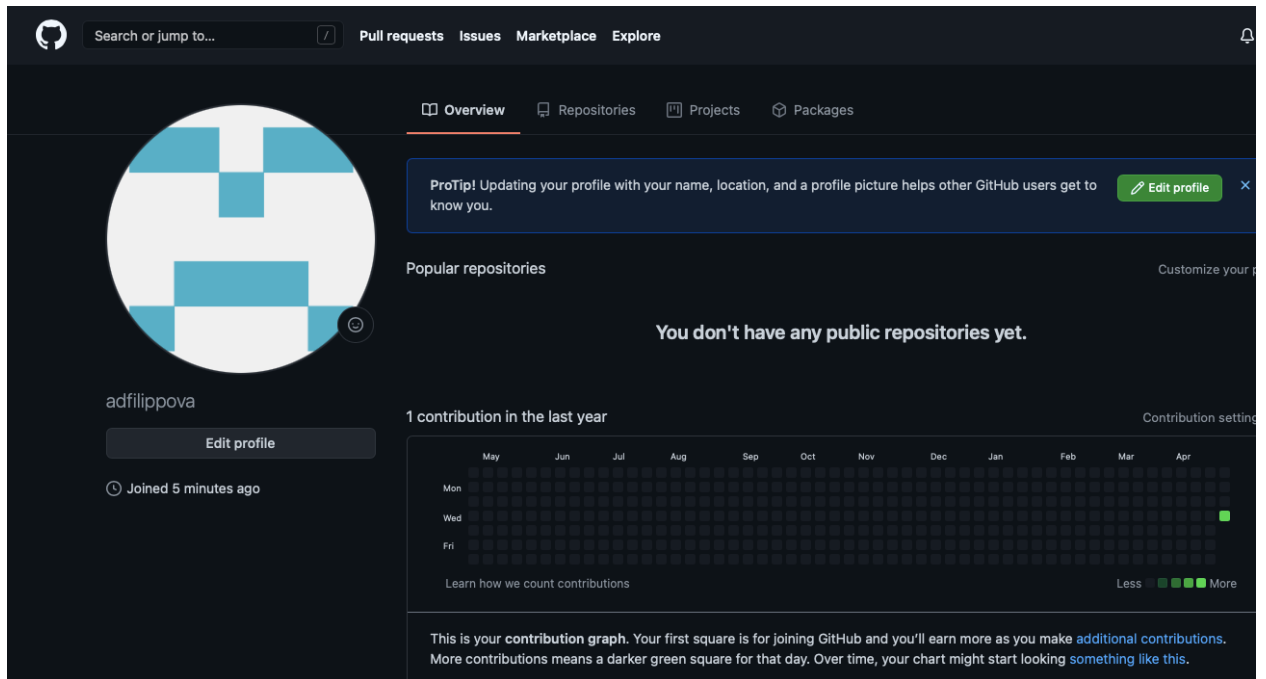


Рис.1

### 1) Настраиваем систему контроля версий git.

Синхронизируем учётную запись github с компьютером:

```
git config --global user.name "adfilippova"
```

```
git config --global user.email "annafil20162016@yandex.ru"
```

Затем создаём новый ключ на github `ssh-keygen -C "adfilippova <annafil20162016@yandex.ru>"` (Рис.2) и привязываем его к компьютеру через консоль. (Рис.3)

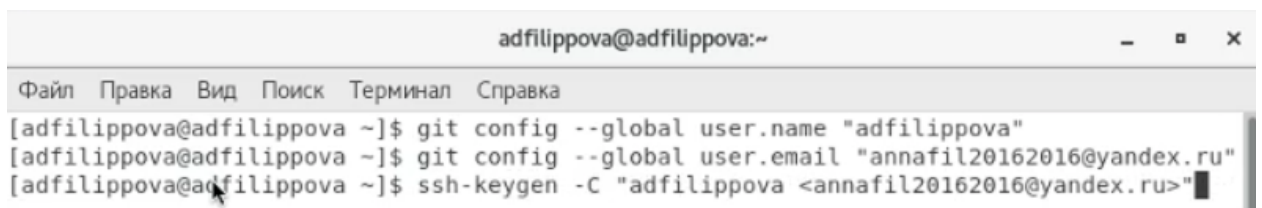



Рис.2

```
Приложения Места Терминал en Чт, 02:57
adfilippova@adfilippova:~
Файл Правка Вид Поиск Терминал Справка
[adfilippova@adfilippova ~]$ ssh-keygen -C "adfilippova <annafil20162016@yandex.ru>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/adfilippova/.ssh/id_rsa):
/home/adfilippova/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/adfilippova/.ssh/id_rsa.
Your public key has been saved in /home/adfilippova/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:psaHdbT+TM26DT0Jep69Qpz6p+9000TRNTE1t5zcGy8 adfilippova <annafil20162016@yandex.ru>
The key's randomart image is:
+---[RSA 2048]----+
|                 .|
|                o|
|               ++|
|              .o.o|
|             S.o.o.*|
|            . = o++ * +|
|           = .oo @ E..|
|          . ...oB.O .|
|         .+*=Xo.    |
+-----[SHA256]-----+
[adfilippova@adfilippova ~]$
```

```
[adfilippova@adfilippova ~]$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaClyc2EAAAADAQABAAQDh5oLo/fvLP+0jCI9RVsS/25b0nBRw8cx0g/jkAHEEdbg8JsP
zG0z+wTLYP1CfT0UQpL0LoJhVkaQHWGN1mMxshf0asAoBxB7zyGhN7v3/ruhGH9CthfYGNw+cjeXY3lnFm9XAY7
5NaVKw5hSGzF19BE/KEAL+bfVdahJabJJ5IXTpQ63JhSa7snVYRsR5GDx1jGXXCqClH6yB+SV1R+6itpjHdcm5q
FM70Y6SQwJX5tnblMCwn6Iwtf5Vd2SvywH3Ck6o7mYkps9JHW5cTPUyWDFkaliU6aUZZc0dRIJPTCs1x8e6IOH
RRMseT609D6qyNzegH0S99C/vqXeZdsz adfilippova <annafil20162016@yandex.ru>
```

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.



**adfilippova**  
SHA256:psaHdbT+TM26DT0Jep69Qpz6p+9000TRNTE1t5zcGy8  
Added on 29 Apr 2021  
Never used — Read/write

Delete

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

Рис.3

2) Созданием и подключаем репозитория к github. На сайте заходим в «repository» и создаём новый репозиторий под названием os-intro1. (Рис.4)

Owner \*  / Repository name \*

Great repository names are short, lowercase, and unique. Need inspiration? How about **improved-disco**?

os-intro1 is available.

Description (optional)

---

☒ **Public**  
 Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
 You choose who can see and commit to this repository.

---

```
[adfilippova@adfilippova ~]$ git clone https://github.com/adfilippova/os-intro1
Cloning into 'os-intro1'...
warning: You appear to have cloned an empty repository.
[adfilippova@adfilippova ~]$ ls
os-intro1  Документы  Изображения  Общедоступные  Шаблоны
Видео     Загрузки   Музыка       Рабочий стол
```

**Рис.4**

Создаем рабочий каталог. (Рис.5)

```
[adfilippova@adfilippova ~]$ cd os-intro1
[adfilippova@adfilippova os-intro1]$ mkdir 2020-2021
[adfilippova@adfilippova os-intro1]$ cd 2020-2021
[adfilippova@adfilippova 2020-2021]$ mkdir OS
[adfilippova@adfilippova 2020-2021]$ cd OS
[adfilippova@adfilippova OS]$ mkdir laboratory
[adfilippova@adfilippova OS]$ cd laboratory
[adfilippova@adfilippova laboratory]$

[adfilippova@adfilippova laboratory]$ mkdir lab02
[adfilippova@adfilippova laboratory]$ lab02
```

**Рис.5**

Добавляем первый коммит и выкладываем на github. Для того, чтобы правильно разместить первый коммит, необходимо добавить команду `git add .`, после этого с помощью команды `git commit -am "first commit"` выкладываем коммит. Сохраняем первый коммит, используя команду `git push`. (Рис.6)

```
[adfilippova@adfilippova lab02]$ touch lab2.txt
[adfilippova@adfilippova lab02]$ echo "# Лабораторные работы">> README.md
[adfilippova@adfilippova lab02]$ ls
lab2.txt  README.md
[adfilippova@adfilippova lab02]$ git add .
```

```
adfilippova@adfilippova lab02]$ git commit -am "first commit"
master la73840] first commit
1 file changed, 1 insertion(+)

[adfilippova@adfilippova lab02]$ git push -u origin master
Username for 'https://github.com': adfilippova
Password for 'https://adfilippova@github.com':
Counting objects: 14, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (14/14), 836 bytes | 0 bytes/s, done.
Total 14 (delta 0), reused 0 (delta 0)
To https://github.com/adfilippova/os-intro1
+ [new branch]      master -> master
```

**Рис.6**

### 3) Первичная конфигурация

Добавляем файл лицензии. Добавляем шаблон игнорируемых файлов.

Просматриваем список имеющихся шаблонов (Рис.7)

```
[adfilippova@adfilippova lab02]$ wget https://creativecommons.org/licenses/by/4.0/legal
code.txt -O LICENSE
--2021-04-29 04:08:50-- https://creativecommons.org/licenses/by/4.0/legalcode.txt
Распознаётся creativecommons.org (creativecommons.org)... 172.67.34.140, 104.20.151.16,
104.20.150.16, ...
Подключение к creativecommons.org (creativecommons.org)|172.67.34.140|:443... соединени
е установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: нет данных [text/plain]
Сохранение в: «LICENSE»

[ <=> ] 18 657 --.-K/s за 0,01s

2021-04-29 04:08:51 (1,53 MB/s) - «LICENSE» сохранён [18657]
```

```
[adfilippova@adfilippova lab02]$ curl -L -s https://www.gitignore.io/api
```

adfilippova@adfilippova:~/os-intro1/2020-2021/OS/laboratory/lab02

Файл Правка Вид Поиск Терминал Справка

```
salesforcedx,sam,sas,sass,sbt
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,tarinstallmate,terraform
terragrunt,test,testcomplete,testinfra,te
text,textmate,textpattern,theos-tweak,thinkphp
tla+,tortoisegit,tower,turbogears2,twincat3
tye,typings,typo3,typo3-composer,umbraco
unity,unrealengine,vaadin,vagrant,valgri
vapor,venv,vertx,video,vim
virtualenv,virtuoso,visualstudio,visualstudiocode,vivado
vlab,vs,vscod
vue,vuejs
vuvv,waf,wakanda,web,webmethods
webstorm,webstorm+all,webstorm+iml,werckercli,windows
wintersmith,wordpress,wyam,xamarinstudio,xcode
xcodeinjection,xilinx,xilinxise,xilinxvivado,xill
xoio,xtext,v86,varn,veoman
```

Рис.7

Скачиваем шаблон (например, для C) и выполняем коммит.

Отправляем на github. (команда gitpush) (Рис.8)

```
[adfilippova@adfilippova lab02]$ curl -L -s https://www.gitignore.io/api/c >> .gitignore
[adfilippova@adfilippova lab02]$ git add .
[adfilippova@adfilippova lab02]$ git commit -am "first commit"
[master e5367f9] first commit
2 files changed, 455 insertions(+)
create mode 100644 2020-2021/OS/laboratory/lab02/.gitignore
create mode 100644 2020-2021/OS/laboratory/lab02/LICENSE

[adfilippova@adfilippova lab02]$ 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': adfilippova
Password for 'https://adfilippova@github.com':
Counting objects: 13, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (8/8), 6.63 KiB | 0 bytes/s, done.
Total 8 (delta 0), reused 0 (delta 0)
To https://github.com/adfilippova/os-introl
   5367f9 master -> master
```

Рис.8

#### 4) Работаем с конфигурацией git-flow.

У нас не получилось установить git-flow, так как root этого не допустил. В связи с этим дальнейшие действия выполнить невозможно. (Рис.9)

```
[adfilippova@adfilippova lab02]$ su
Пароль:
[root@adfilippova lab02]# yum install gitflow
Загружены модули: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: mirror.reconn.ru
 * extras: mirror.reconn.ru
 * updates: centos-mirror.rbc.ru

base | 3.6 kB  00:00:00
extras | 2.9 kB  00:00:00
updates | 2.9 kB  00:00:00

Пакета с названием gitflow не найдено.
Ошибка: Выполнять нечего
```

Рис.9

### 3. Контрольные вопросы:

1) 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) 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, пользователь размещает 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) 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) 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 репозитория 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) 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) 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). Использование 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 списки 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:** я изучила идеологию и применение контроля версий.