

# Отчёт по лабораторной работе № 3

*дисциплина: Операционные системы*

Андрианова Марина Георгиевна

## Цель работы

Целью данной работы является изучение идеологии и применение средств контроля версий, а также освоение умений по работе с git.

## Ход работы

1. Создаем учётную запись на <https://github.com>. Заполняем основные данные. Моя учетная запись называется mgandrianova. (Рис.1)

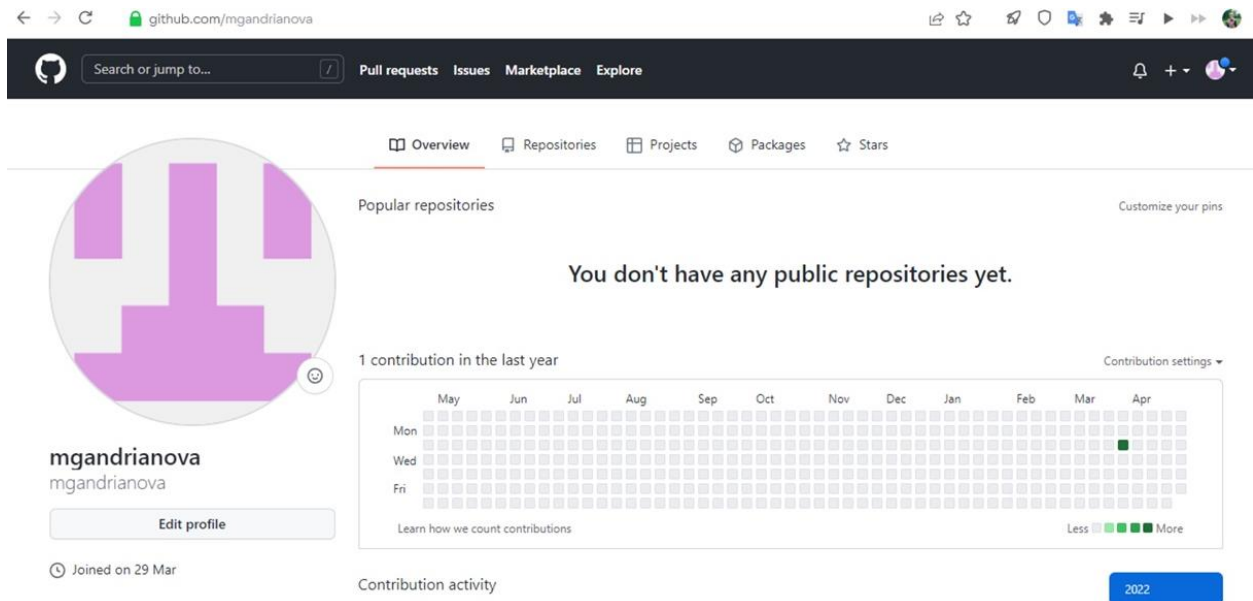


Рис.1

2. Настроим базовую конфигурацию git. Для этого зададим имя и email владельца репозитория(рис.2). Настроим utf-8 в выводе сообщений git. Настроим верификацию и подписание коммитов git: зададим имя начальной ветки(master), параметр(autocrlf), параметр safecrlf(рис.3).

```
[mgandrianova@fedora ~]$ git config --global user.name "mgandrianova"
[mgandrianova@fedora ~]$ git config --global user.email "marina.andrianova.03@mail.ru"
```

Рис.2

```
[mgandrianova@fedora ~]$ git config --global core.quotepath
false
[mgandrianova@fedora ~]$ git config --global init.defaultBranch master
[mgandrianova@fedora ~]$ git config --global core.autocrlf input
[mgandrianova@fedora ~]$ git config --global core.safecrlf warn
```

Рис.3

3. Теперь надо сгенерировать два ключа ssh и gpg и вставить их в учетную запись github для того, чтобы привязать наш компьютер к github. Создадим ключ ssh с помощью команды (рис.4)

```
ssh-keygen -t rsa -b 4096
```

и скопируем его в буфер обмена командой (рис.5).

```
cat ~/.ssh/id_rsa.pub | xclip -sel clip
```

```
[mgandrianova@fedora ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/mgandrianova/.ssh/id_rsa):
/home/mgandrianova/.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/mgandrianova/.ssh/id_rsa
Your public key has been saved in /home/mgandrianova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:FNbD859UDo+XoQMrUNyvw0QYVruMVSmkX7rkEhfx06M mgandrianova@fedora
The key's randomart image is:
+---[RSA 4096]-----+
|          =*+  ..  |
|         o.+*==.o..|
|        o..B=+.0o|
|       . .=0*+=.=|
|      So+B.E.o |
|       =+. o  |
|      . o.  |
|      .    |
|      +----[SHA256]-----+
```

Рис.4

```
[mgandrianova@fedora ~]$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
```

Рис.5

Теперь вставим ключ в аккаунт на GitHub(рис.9). Создадим ключ gpg командой

```
gpg --full-generate-key
```

и выбираем из предложенных опций варианты, которые указаны в лабораторной работе(рис.6). Выведем список ключей, чтобы скопировать отпечаток приватного ключа(рис.7).

```
[mgandrianova@fedora ~]$ gpg --full-generate-key
gpg (GnuPG) 2.3.2; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Выберите тип ключа:
  (1) RSA and RSA
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (9) ECC (sign and encrypt) *default*
 (10) ECC (только для подписи)
 (14) Existing key from card
Ваш выбор? 1
длина ключей RSA может быть от 1024 до 4096.
Какой размер ключа Вам необходим? (3072) 4096
Запрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
  0 = не ограничен
  <n> = срок действия ключа - n дней
  <n>w = срок действия ключа - n недель
  <n>m = срок действия ключа - n месяцев
  <n>y = срок действия ключа - n лет
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (y/N) y
```

Рис.6

```
[mgandrianova@fedora ~]$ gpg --armor --export C7AC6238614C4800 | xclip -sel clip
```

Рис.7

Скопируем сгенерированный gpg ключ в буфер обмена(рис.8)

```
[mgandrianova@fedora ~]$ gpg --list-secret-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/mgandrianova/.gnupg/pubring.kbx
-----
sec   rsa4096/C7AC6238614C4800 2022-04-29 [SC]
      96B0CA9AA036647383EBC357C7AC6238614C4800
uid           [ абсолютно ] Marina <marina.andrianova.03@mail.ru>
ssb   rsa4096/97BCDB0BDAC0BD41 2022-04-29 [E]
```

Рис.8

И вставим его на GitHub(рис. 9).

```
[mgandrianova@fedora ~]$ git config --global user.signingkey C7AC6238614C4800
[mgandrianova@fedora ~]$ git config --global commit.gpgsign
[mgandrianova@fedora ~]$ git config --global commit.gpgsign true
[mgandrianova@fedora ~]$ git config --global gpg.program $(which gpg2)
```

Рис.9

4.Используя введенный email, укажем Git применять его при подписи коммитов(рис.10).

```
[mgandrianova@fedora ~]$ mkdir -p ~/work/study/2021-2022/"Операционные системы"
[mgandrianova@fedora ~]$ cd ~/work/study/2021-2022/"Операционные системы"
[mgandrianova@fedora Операционные системы]$ gh repo create study_2021-2022_os-intro --template=yamadharma/course-directory-student-template --public
✓ Created repository mgandrianova/study_2021-2022_os-intro on GitHub
```

Рис.10

5.Создадим путь, где будут храниться материалы к лабораторным работам и перейдём в последнюю папку(рис.11) и скачаем шаблон репозитория(рис.11) в папку.

```
[mgandrianova@fedora Операционные системы]$ git clone --recursive git@github.com:mgandrianova/study_2021-2022_os-intro.git os-intro
```

Рис.11

Теперь создадим репозиторий на GitHub, где будут храниться только что созданные папки. 6. Перейдём в каталог курса(команда `cd os-intro`), удалим лишние файлы(команда `rm package.json`) и создадим необходимые каталоги для дальнейшей работы (`make COURSE=os-intro`) (рис.12).

```
[mgandrianova@fedora os-intro]$ git add .
[mgandrianova@fedora os-intro]$ git commit -am 'First'
[master 61f5cf0] First
149 files changed, 16590 insertions(+), 14 deletions(-)
create mode 100644 labs/lab01/presentation/Makefile
create mode 100644 labs/lab01/presentation/presentation.md
create mode 100644 labs/lab01/report/Makefile
create mode 100644 labs/lab01/report/bib/cite.bib
create mode 100644 labs/lab01/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab01/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
```

Рис.12

Теперь скопируем наш собственный репозиторий work в папку “Операционные системы”, перенесём в него все файлы из папки os-intro и отправим на сервер, чтобы они также появились в репозитории на GitHub(рис. 13-16)

```

create mode 100644 project-personal/stage6/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage6/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100644 project-personal/stage6/report/report.md
create mode 100644 structure
[mgandrianova@fedora os-intro]$ git push
Перечисление объектов: 20, готово.
Подсчет объектов: 100% (20/20), готово.
Сжатие объектов: 100% (16/16), готово.
Запись объектов: 100% (19/19), 266.50 КиБ | 2.05 МиБ/с, готово.
Всего 19 (изменений 2), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To github.com:mgandrianova/study_2021-2022_os-intro.git
 d031d33..61f5cf0  master -> master

```

Рис.13

```

[mgandrianova@fedora Операционные системы]$ cd os-intro
[mgandrianova@fedora os-intro]$ ls
config labs LICENSE Makefile project-personal README.en.md README.git-flow.md README.md structure template


```

Рис.14

## SSH keys

New SSH key

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



**d**

SHA256: FNbD859UDo+XoQMruNyvw0QYVruMVSmkX7rkEhfx06M

Added on 29 Apr 2022

Last used within the last week — Read/write


Delete

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

## GPG keys

New GPG key

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



**Email address:** marina.andrianova.03@mail.ru

**Key ID:** C7AC6238614C4800

**Subkeys:** 97BCDB0BDAC0BD41

Added on 29 Apr 2022

Delete

Рис.15

Смотрим репозиторий на сайте GitHub и убеждаемся, что мы сделали всё правильно(рис.16).


 <b>mgandrianova</b> First commit <span style="float: right;">fa3502b 6 hours ago ⌚ 2 commits</span>		
📁 config	Initial commit	6 hours ago
📁 labs	First commit	6 hours ago
📁 project-personal	First commit	6 hours ago
📁 template	Initial commit	6 hours ago
📄 .gitattributes	Initial commit	6 hours ago
📄 .gitignore	Initial commit	6 hours ago
📄 .gitmodules	Initial commit	6 hours ago
📄 LICENSE	Initial commit	6 hours ago
📄 Makefile	Initial commit	6 hours ago
📄 README.en.md	Initial commit	6 hours ago
📄 README.git-flow.md	Initial commit	6 hours ago
📄 README.md	Initial commit	6 hours ago

Рис.16

### Выводы

Я изучила идеологию и применение контроля версий, также освоила умения по работе с git.

### 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. Система контроля версий (СКВ) – программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями 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.
- 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 сохраняются 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 строки, 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 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
```