

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: Антипина Кристина
Алексеевна

Группа: НБИбд-01-21

Ст. билет №: 1032211397

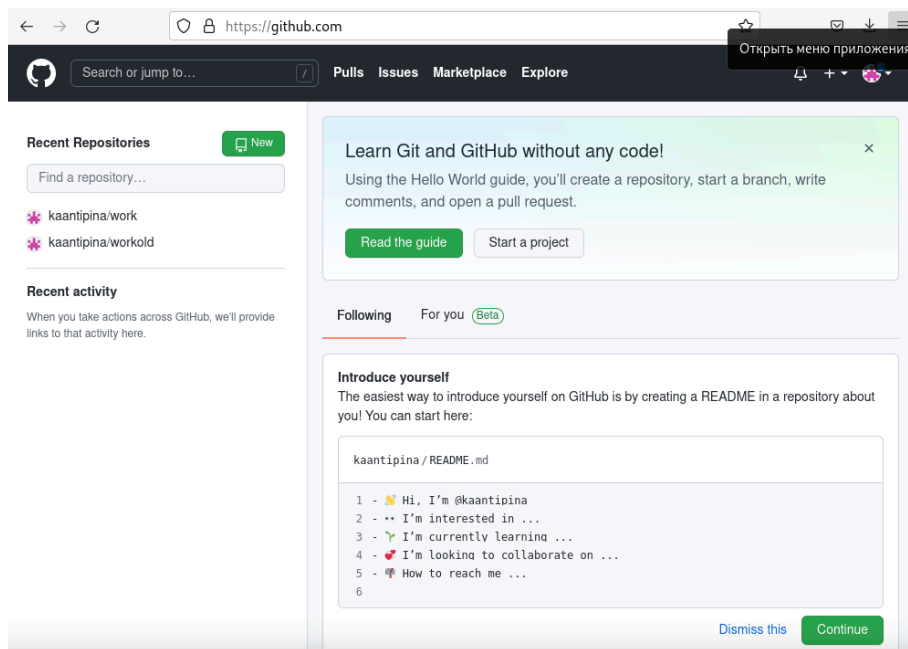
Москва
2022 г.

Цель работы:

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

Ход работы:

Создаем учётную запись на <https://github.com> (рис. 1).



(рис. 1)

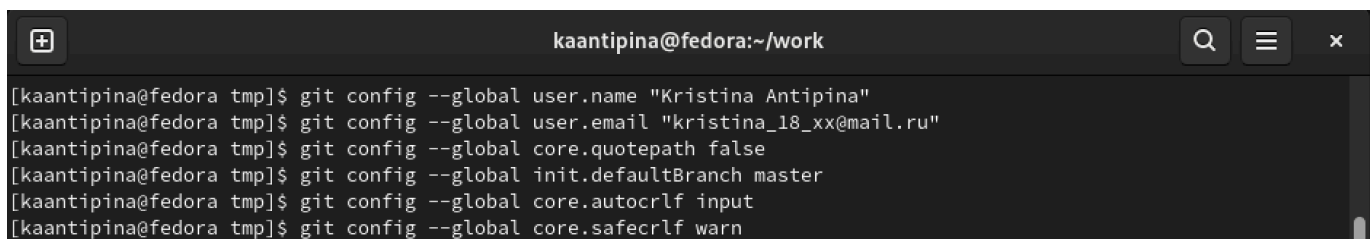
1) Настраиваем систему контроля версий git.

Синхронизируем учётную запись github с компьютером (рис. 2):

```
git config --global user.name «Kristina Antipina»
```

```
git config --global user.email "kristina_18_xx@mail.ru»
```

Затем создаём новый ключ на github `ssh-keygen -C "Kristina Antipina <kristina_18_xx@mail.ru>"` (рис. 3, рис. 4. рис.5) и привязываем его к компьютеру через консоль (рис. 6).



(рис.2)

```
[kaantipina@fedora tmp]$ cd
[kaantipina@fedora ~]$ mkdir tutorial
[kaantipina@fedora ~]$ cd tutorial
[kaantipina@fedora tutorial]$ git init
Инициализирован пустой репозиторий Git в /home/kaantipina/tutorial/.git/
[kaantipina@fedora tutorial]$ echo 'hello world' > hello.txt
[kaantipina@fedora tutorial]$ git add hello.txt
[kaantipina@fedora tutorial]$ git commit -am 'Новый файл'
[master (корневой коммит) d4edcf1] Новый файл
1 file changed, 1 insertion(+)
create mode 100644 hello.txt
[kaantipina@fedora tutorial]$ git status
На ветке master
ничего коммитить, нет изменений в рабочем каталоге
[kaantipina@fedora tutorial]$ curl -L -s https://www.gitignore.io/api/list
1c,1c-bitrix,a-frame,actionsript,ada
adobe,advancedinstaller,adventuregamestudio,agda,al
alteraquartusii,altium,amplify,android,androidstudio
angular,anjuta,ansible,ansibletower,apachecordova
apachehadoop,appbuilder,appcelerator titanium,appcode,appcode+all
appcode+iml,appengine,aptanastudio,arcanist,archive
archives,archlinuxpackages,aspnetcore,assembler,ate
atmelstudio,ats,audio,automationstudio,autotools
autotools+strict,awr,azurefunctions,azurite,backup
ballerina,basercms,basic,batch,bazaar
bazel,bitrise,bitrix,bittorrent,blackbox
bloop,bluej,bookdown,bower,bricxcc
buck,c,c++,cake,cakephp
cakephp2,cakephp3,calabash,carthage,certificates
ceylon,cfwheels,cheftcookbook,chocolatey,circuitpython
clean,clion,clion+all,clion+iml,clojure
cloud9,cmake,cocoapods,cocos2dx,cocoscreator
codeblocks,codecomposerstudio,codeigniter,codeio,codekit
```

(рис.3)

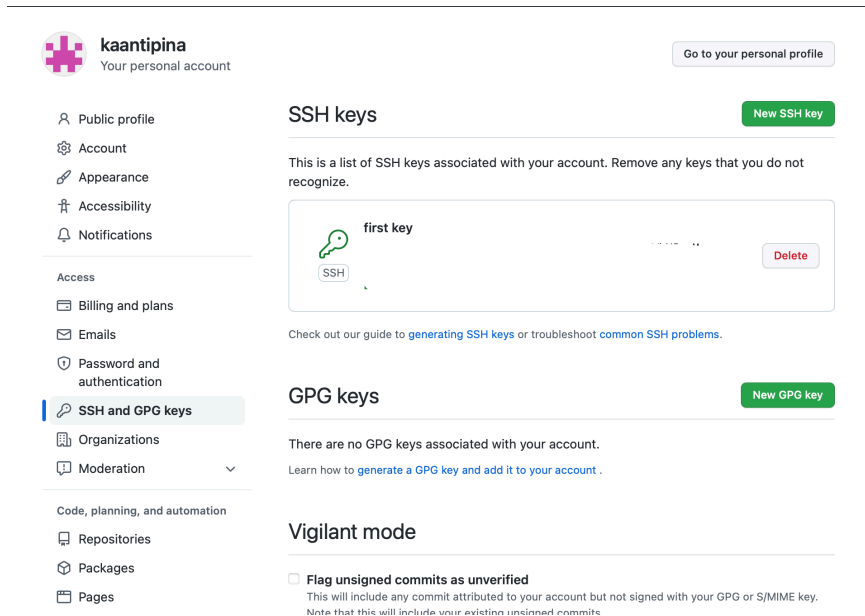
```
zephir,zig,zsh,zukencr8000[kaacurl -L -s https://www.gitignore.io/api/c >> .gitignore
[kaantipina@fedora tutorial]$ curl -L -s https://www.gitignore.io/api/c >> .gitignore
[kaantipina@fedora tutorial]$ curl -L -s https://www.gitignore.io/api/c++ >> .gitignore
[kaantipina@fedora tutorial]$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
```

(рис.4)

```
[kaantipina@fedora tmp]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/kaantipina/.ssh/id_ed25519):
Created directory '/home/kaantipina/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kaantipina/.ssh/id_ed25519
Your public key has been saved in /home/kaantipina/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:mDKLABYWyHr/gSPw+s/GGHtLGcGahVod83A08+CQo3k kaantipina@fedora
The key's randomart image is:
+---[ED25519 256]---+
|o+. .B.. |
|o.. =o=@ |
|o. = ..+ |
|= = E .o |
|. + oo.ooS |
|. +.++= |
| ..*. = . |
|. o.+ . |
| ..+o |
+-----[SHA256]-----+
[kaantipina@fedora tmp]$ 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.

gpg: создан каталог '/home/kaantipina/.gnupg'
gpg: создан щит с ключами '/home/kaantipina/.gnupg/pubring.kbx'
Выберите тип ключа:
(1) RSA and RSA
(2) DSA and Elgamal
(3) DSA (sign only)
(4) RSA (sign only)
```

(рис.5)



(рис.6)

```
kaantipina@fedora:~/work
[kaantipina@fedora tutorial]$ sudo git push -u origin master
[sudo] пароль для kaantipina:
The authenticity of host 'github.com (140.82.121.4)' can't be established.
ED25519 key fingerprint is SHA256:+D1Y3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvC0qU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
root@github.com: Permission denied (publickey).
fatal: Не удалось прочитать из внешнего репозитория.

Удостоверьтесь, что у вас есть необходимые права доступа
и репозиторий существует.
[kaantipina@fedora tutorial]$ git remote add origin ssh://github.com/kaantipina/work.git
error: внешний репозиторий origin уже существует
[kaantipina@fedora tutorial]$ git push -u origin main
error: src refspec main does not match any
error: не удалось отправить некоторые ссылки в «ssh://github.com/kaantipina/work.git»
[kaantipina@fedora tutorial]$ git branch -M main
[kaantipina@fedora tutorial]$ git push -u origin main
kaantipina@github.com: Permission denied (publickey).
fatal: Не удалось прочитать из внешнего репозитория.

Удостоверьтесь, что у вас есть необходимые права доступа
и репозиторий существует.
[kaantipina@fedora tutorial]$ git remote set-url origin ssh://github.com/kaantipina/work.git
[kaantipina@fedora tutorial]$ git push -u origin main
kaantipina@github.com: Permission denied (publickey).
fatal: Не удалось прочитать из внешнего репозитория.

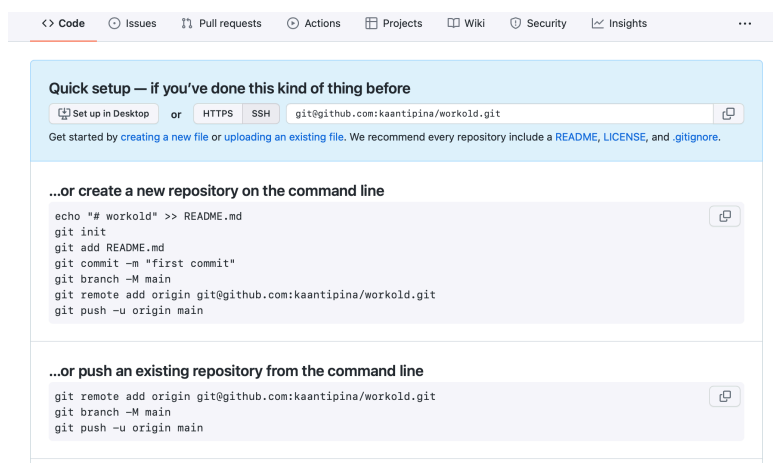
Удостоверьтесь, что у вас есть необходимые права доступа
и репозиторий существует.
[kaantipina@fedora tutorial]$ git branch -M main
[kaantipina@fedora tutorial]$ git push -u origin main
kaantipina@github.com: Permission denied (publickey).
fatal: Не удалось прочитать из внешнего репозитория.

Удостоверьтесь, что у вас есть необходимые права доступа
```

(рис.7)

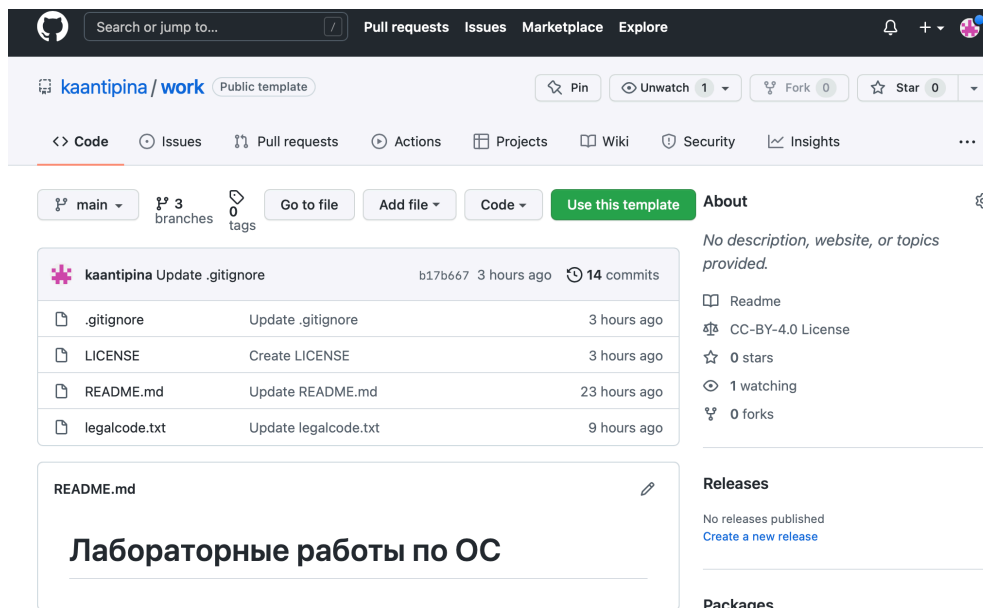
2) Создаем и подключаем репозиторий к github.

На сайте заходим в «repository» и создаём новый репозиторий под названием work. Переносим его на наш компьютер (рис. 8).



(рис.8)

Создаем рабочий каталог



(рис. 9)

Добавляем первый commit и выкладываем на github. Для того, чтобы правильно разместить первый коммит, необходимо добавить команду `git add .`, после этого с помощью команды `git commit -m "first commit"` выкладываем коммит. Сохраняем первый коммит, используя команду `git push` (рис. 10).

```
[kaantipina@fedora 2021-2022]$ ls
'Операционные системы'
[kaantipina@fedora 2021-2022]$ cd
[kaantipina@fedora ~]$ cd ~/tutorial/work/2021-2022
bash: cd: /home/kaantipina/tutorial/work/2021-2022: Нет такого файла или каталога
[kaantipina@fedora ~]$ ls
tutorial  work  Видео  Документы  Загрузки  Изображения  Музыка  Общедоступные  'Рабочий стол'  Шаблоны
[kaantipina@fedora ~]$ cd ..
[kaantipina@fedora home]$ cd ..
[kaantipina@fedora /]$ cd ~/work/study/2021-2022
[kaantipina@fedora 2021-2022]$ ls
'Операционные системы'
[kaantipina@fedora 2021-2022]$ cd "Операционные системы"
[kaantipina@fedora Операционные системы]$ ls
[kaantipina@fedora Операционные системы]$ cd laboratory
```

(рис. 10)

3)Первичная конфигурация.

Добавляем файл лицензии. Добавляем шаблон игнорируемых файлов. Просматриваем список имеющихся шаблонов (рис.11). Скачиваем шаблон (например, для C) и выполняем коммит. Отправляем на github (команда `git push`) (рис. 12).

```
kaantipina@fedora:~/work
[kaantipina@fedora tutorial]$ git status
На ветке master
ничего коммитить, нет изменений в рабочем каталоге
[kaantipina@fedora tutorial]$ curl -L -s https://www.gitignore.io/api/list
1c,1c-bitrix,a-frame,actionsript,ada
adobe,advancedinstaller,adventuregamestudio,agda,al
alteraquartusii,altium,amplify,android,androidstudio
angular,anjuta,ansible,ansibletower,apachecordova
apachehadoop,appbuilder,appcelerator titanium,appcode,appcode+all
appcode+iml,appengine,aptanastudio,arcanist,archive
archives,archlinuxpackages,aspnetcore,assembler,ate
atmelstudio,ats,audio,automationstudio,autotools
autotools+strict,awr,azurefunctions,azurite,backup
ballerina,basercms,basic,batch,bazaar
bazel,bitrise,bitrix,bittorrent,blackbox
bloop,bluej,bookdown,bower,brickcc
buck,c,c++,cake,cakephp
cakephp2,cakephp3,calabash,carthage,certificates
ceylon,cfwheels,cheffcoookbook,chocolatey,circuitpython
clean,clion,clion+all,clion+iml,clojure
cloud9,cmake,cocoapods,cocos2dx,cocoscreator
codeblocks,codecomposerstudio,codeigniter,codeio,codekit
codesniffer,coffeescript,commonlisp,compodoc,composer
compressed,compressedarchive,compression,conan,concrete5
coq,cordova,craftcms,crashlytics,crbasic
crossbar,crystal,cs-cart,csharp,cuda
cvs,cypressio,d,dart,darteditor
data,database,datarecovery,dbeaver,defold
delphi,dframe,diff,direnv,diskimage
django,dm,docfx,docpress,docz
dotenv,dotfilessh,dotnetcore,dotsettings,dreamweaver
dropbox,drupal,drupal7,drupal8,e2studio
eagle,easybook,eclipse,eiffelstudio,elasticbeanstalk
elisp,elixir,elm,emacs,ember
ensime,episerver,erlang,espresso,executable
exercism,expressionengine,extjs,fancy,fastlane
finale,firebase,flashbuilder,flask,flatpak
flex,flexbuilder,floobits,flutter,font
```

(рис.11)

```
kaantipina@fedora:~/work/study/2021-2022/Операционные системы/laboratory
[kaantipina@fedora laboratory]$ ls
hello.txt lab01 lab02 lab03 README.md
[kaantipina@fedora laboratory]$
```

(рис.12)

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 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 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 изменений: `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
```

Вывод:

Я изучила идеологию и применение контроля версий.