

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: Ким Реачна

Группа: НПИбд-02-20

МОСКВА

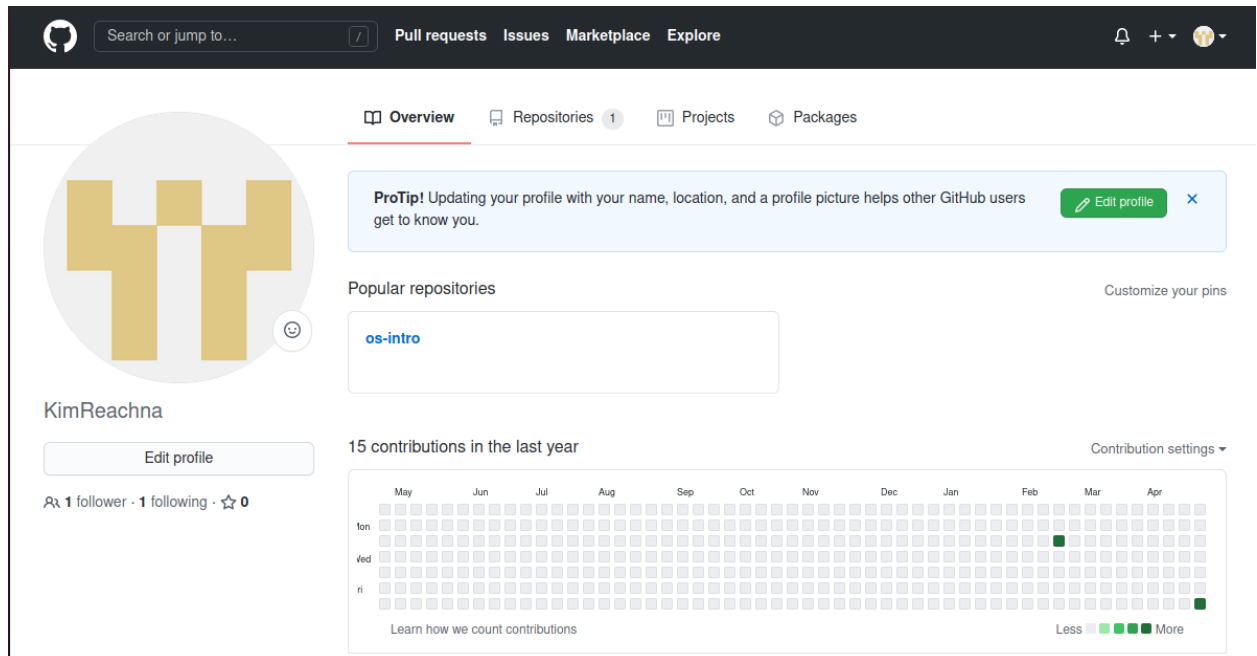
2020 г.

Цель работы

Изучить идеологию и применение средств контроля версий.

Выполнение работы:

1. Создаем учетную запись Github



2. 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"
```

```
kim@kin-VirtualBox:~$ git config --global user.name "Kim Reachna"
kim@kin-VirtualBox:~$ git config --global user.email "kimreachna@gmail.com"
```


3. Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый):

```
ssh-keygen -C "Имя Фамилия <work@mail>"
```

```
kim@kim-VirtualBox:~$ ssh-keygen -C "Kim Reachna <kimreachna@gmail.com>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kim/.ssh/id_rsa): /home/kim/.ssh/id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Passphrases do not match. Try again.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kim/.ssh/id_rsa
Your public key has been saved in /home/kim/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:cX84UxsP/2KJ+W7tvVbo3ay4svV1JISeTAiBTfjXtUw Kim Reachna <kimreachna@gmail.com>
The key's randomart image is:
+---[RSA 3072]-----+
|
|  =+.
|  o .. . .E
|  .. o.o+=.
|  .o.=.o*
|  S. 0 +.+
|      *.+o
|      +.++*
|      .. =o+B
|      .oo+*++
+---[SHA256]-----+
```

4. Откройте настройки -> SSH в github и скопируйте текст и вставьте его

```
kim@kim-VirtualBox:~$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGC1XTmkFLMcVko0X3eFbjhItvc7BFYbIg2j4My3Vv0mHhe7babc2fpb6DtA9+RZrUr4Rn+i0IMsog5S18a/d7lMvX00r+E
oWwVouP8b64MZsHI0a78ThfZIRAGn4/nrVxQbKTrMPWw6+iX8rFQCe8hvsnxI6nQftLKJmWEn9x29fiJjNuaRp/IJBcUwsPqA7gVXQ10q/uK400BK1Z5/1ALmTnjhuXT6r
qSGA+/wegwv70/+JslJyKwJ3UXK43VxOTNkP2tocGodG4NZIPLXpYEr+S2ey7Wr10tfy9ebJJc3wMtEc+9yloU3NgIXmP0D+gn3o/one+/B7K1UMasTSQZz5Po0vJ1U+7Wq0
V6ZQhBSuumbIYY4Z8Ty0YdGKRHRy7Wxkz6j+e406eFBXEcA5hpE8fMjJqEQ8deccgF704yxpfaM3/kqahJJ064UWF6MJnQTYkx9P3ISZrPCI6o8Fc6aXHQyIr6BwdDOX3bG
VuV0bE0yVAqfzL3UcnR40Cc= Kim Reachna <kimreachna@gmail.com>
```



KimReachna
Your personal account

Go to your personal profile

Account settings

Profile

Account

Appearance New

Account security

Billing & plans

Security log

Security & analysis

Emails


Notifications

SSH and GPG keys

SSH keys

New SSH key

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

**Kim**

SHA256:cX84UxsP/2KJ+W7tvVbo3ay4svV1JISeTAiBTfjXtUw

Added on May 1, 2021

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

There are no GPG keys associated with your account.

Learn how to [generate a GPG key](#) and [add it to your account](#).

5. Создаем репозиторий на GitHub, назовём его os-intro:

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * KimReachna / Repository name * os-intro ✓

Great repository names are: os-intro is available. e. Need inspiration? How about [probable-octo-spork?](#)

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.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ Add a README file

6. Перехожу в каталог laboratory используя команду `cd < полный путь до каталога >` и инициализирую системы `git`

```
kim@kim-VirtualBox:~$ cd
kim@kim-VirtualBox:~$ cd /home/kim/work/2020-2021/laboratory
kim@kim-VirtualBox:~/work/2020-2021/laboratory$ cd /home/kim/work/2020-2021/os/laboratory
bash: cd: /home/kim/work/2020-2021/os/laboratory: No such file or directory
kim@kim-VirtualBox:~/work/2020-2021/laboratory$ git init
Initialized empty Git repository in /home/kim/work/2020-2021/laboratory/.git/
```

7. Создаём заготовку для файла README.md:

```
kim@kim-VirtualBox:~/work/2020-2021/laboratory$ echo "# Лабораторные работы" >> README.md
kim@kim-VirtualBox:~/work/2020-2021/laboratory$ git add README.md
```

8. Делаем первый коммит и выкладываем на github:

```
kim@kim-VirtualBox:~/work/2020-2021/laboratory$ git commit -m "first commit"
[master (root-commit) 646db94] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md
```

```

kim@kim-VirtualBox:~/work/2020-2021/laboratory$ git remote set-url origin git@github.com:KimReachna/os-intro.git
kim@kim-VirtualBox:~/work/2020-2021/laboratory$ git remote -v
origin  git@github.com:KimReachna/os-intro.git (fetch)
origin  git@github.com:KimReachna/os-intro.git (push)
kim@kim-VirtualBox:~/work/2020-2021/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])? yes
Warning: Permanently added 'github.com,140.82.121.4' (RSA) to the list of known hosts.
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (10/10), 797 bytes | 797.00 KiB/s, done.
Total 10 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To github.com:KimReachna/os-intro.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

```

9. Добавим файл лицензии:

```

kim@kim-VirtualBox:~/work/2020-2021/laboratory$ wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O LICENSE
--2021-05-01 14:03:28--  https://creativecommons.org/licenses/by/4.0/legalcode.txt
Resolving creativecommons.org (creativecommons.org)... 172.67.34.140, 104.20.151.16, 104.20.150.16, ...
Connecting to creativecommons.org (creativecommons.org)|172.67.34.140|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: 'LICENSE'

LICENSE                                     [ <=> ] 18,22K  --.-KB/s   in 0,002s

2021-05-01 14:03:28 (7,24 MB/s) - 'LICENSE' saved [18657]

```

10.Добавим шаблон игнорируемых файлов. Просмотрим список имеющихся шаблонов:

```
kim@kim-VirtualBox:~/work/2020-2021/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,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,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
chefcookbook,chocolatey,clean,clion,clion+all
clion+iml,clojure,cloud9,cmake,cocoapods
cocos2dx,cocoscreator,code,code-java,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,fontforge
forcedotcom,forgegradle,fortran,freepascal,fsharp
```

11.Добавим новые файлы , Выполним коммит

```
zsh,zukencr8000kim@kim-VirtualBox:~/work/2020-2021/laboratory$ curl -L -s https://www.gitignore.io/api/c >> .gitignore
kim@kim-VirtualBox:~/work/2020-2021/laboratory$ git add .
kim@kim-VirtualBox:~/work/2020-2021/laboratory$ git commit -a
```

12.Отправим на github

```
kim@kim-VirtualBox:~/work/2020-2021/laboratory$ fg
git commit -a
[master 1231eac] new file
2 files changed, 753 insertions(+)
create mode 100644 .gitignore
create mode 100644 LICENSE
```

```
kim@kim-VirtualBox:~/work/2020-2021/laboratory$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 6.58 KiB | 6.58 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To github.com:KimReachna/os-intro.git
83a3df4..1231eac master -> master
```

13. Инициализируем git-flow , Префикс для ярлыков установим в v.

```
kim@kim-VirtualBox:~/work/2020-2021/laboratory$ git flow init

Which branch should be used for bringing forth production releases?
- master
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? [] v
Hooks and filters directory? [/home/kim/work/2020-2021/laboratory/.git/hooks]
```

14. Проверьте на ветке develop:

```
kim@kim-VirtualBox:~/work/2020-2021/laboratory$ git branch
* develop
master
```

15. Создадим релиз с версией 1.0.0

```
kim@kim-VirtualBox:~/work/2020-2021/laboratory$ git flow release start 1.0.0
Switched to a new branch 'release/1.0.0'

Summary of actions:
- A new branch 'release/1.0.0' was created, based on 'develop'
- You are now on branch 'release/1.0.0'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

    git flow release finish '1.0.0'
```


16. Запишем версию, Добавим в индекс

```
kim@kim-VirtualBox:~/work/2020-2021/laboratory$ echo "1.0.0" >> VERSION
kim@kim-VirtualBox:~/work/2020-2021/laboratory$ git add .
kim@kim-VirtualBox:~/work/2020-2021/laboratory$ git commit -am 'chore(main): add version'
[release/1.0.0 f13bf4a] chore(main): add version
1 file changed, 1 insertion(+)
create mode 100644 VERSION
```

17. Зальём релизную ветку в основную ветку

```
kim@kim-VirtualBox:~/work/2020-2021/laboratory$ git flow release finish 1.0.0
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
Merge made by the 'recursive' strategy.
VERSION | 1 +
1 file changed, 1 insertion(+)
create mode 100644 VERSION
Already on 'master'
Your branch is ahead of 'origin/master' by 2 commits.
(use "git push" to publish your local commits)
Switched to branch 'develop'
Merge made by the 'recursive' strategy.
VERSION | 1 +
1 file changed, 1 insertion(+)
create mode 100644 VERSION
Deleted branch release/1.0.0 (was f13bf4a).

Summary of actions:
- Release branch 'release/1.0.0' has been merged into 'master'
- The release was tagged 'v1.0.0'
- Release tag 'v1.0.0' has been back-merged into 'develop'
- Release branch 'release/1.0.0' has been locally deleted
- You are now on branch 'develop'
```

18. Отправим данные на github

```
kim@kim-VirtualBox:~/work/2020-2021/laboratory$ git push --all
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 516 bytes | 516.00 KiB/s, done.
Total 5 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To github.com:KimReachna/os-intro.git
1231eac..9cd9d87 master -> master
* [new branch] develop -> develop
kim@kim-VirtualBox:~/work/2020-2021/laboratory$ git push --tags
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 163 bytes | 163.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0)
To github.com:KimReachna/os-intro.git
* [new tag] v1.0.0 -> v1.0.0
```


Контрольные вопросы:

1. Системы контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией, позволяющее хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям.

Предназначены для работы нескольких человек над одним проектом, а также при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы.

2. Хранилище – место «памяти», в котором будет храниться новая версия файла после его изменения пользователем.

Commit. В нем содержится описание тех изменений, которые вносит пользователь в код приложения.

История – история изменений. Обычно доступна информация о том, кто из участников, когда и какие изменения вносил.

Рабочая копия – это копия, которую мы выписали в свою рабочую зону, это то, над чем мы работаем в данный момент. Привилегированный доступ только одному пользователю, работающему с файлом.

3. Централизованные VCS предполагают наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями и осуществляется специальным сервером. Пример: AccuRev

Децентрализованные VCS не имеют единого репозитория, он у каждого пользователя свой. Помимо того, они были созданы для обмена изменениями, а не для их объединения. Не имеют какой-то жестко заданной структуры репозитория с центральным сервером. Пример: Git

4. При единоличной работе с VCS каждое новое изменение в репозитории сохраняется не со всеми предыдущими версиями. Оно изменяется по системе: одно предыдущее + новая информация.

5. Для начала те действия, что совершаются один раз:

1. Создать репозиторий.

Это место, где будут лежать файлы. Теперь у нас есть общее хранилище данных, с которым и будет проходить дальнейшая работа.

2. Скачать проект из репозитория.

Далее то, что будет использоваться в работе часто:

1. Обновить проект, забрать последнюю версию из репозитория

2. Внести изменения в проект

3. Запустить код, т.е изменить код в общем хранилище

4. Создать ветку

Теперь, если нужно закоммитить изменения, они по-прежнему пойдут в основную ветку. Бранч при этом трогать НЕ будут. Так что мы можем смело коммитить новый код в *trunk*. А для показа использовать *branch*, который будет оставаться стабильным даже тогда, когда в основной ветке всё падает из-за кучи ошибок. С бранчами мы всегда будем иметь работающий код.

6. - Сохранение файлов с исходным кодом

- Защита от случайных исправлений и удалений

- Отмена изменений и удалений, если они оказались некорректными

- Одновременная поддержка рабочей версии и разработка новой

- Возврат к любой версии кода из прошлого

- Просмотр истории изменений

- Совместная работа без боязни потерять данные или затереть чужую работу

7. `git init` – создание основного дерева репозитория

`git pull` – получение обновлений (изменений) текущего дерева из центрального репозитория

`git push` – отправка всех произведённых изменений локального дерева в центральный репозиторий

`git status` – просмотр списка изменённых файлов в текущей директории

`git diff` – просмотр текущих изменений

`git add` – добавить все изменённые и/или созданные файлы и/или каталоги

`git add имена_файлов` – добавить конкретные изменённые и/или созданные файлы и/или каталоги

`git rm имена_файлов` – удалить файл и/или каталог из индекса репозитория

`git commit -am 'Описание коммита'` – сохранить все добавленные изменения и все изменённые файлы

`git commit` – сохранить добавленные изменения с внесением комментария через встроенный редактор

`git checkout -b имя_ветки` – создание новой ветки, базирующейся на текущей:

`git checkout имя_ветки` – переключение на ветку

`git push origin имя_ветки` – отправка изменений конкретной ветки в центральный репозиторий

`git merge --no-ff имя_ветки` – слияние ветки с текущим деревом

`git branch -d имя_ветки` – удаление локальной уже слитой с основным деревом ветки

`git branch -D имя_ветки` – принудительное удаление локальной ветки

`git push origin :имя_ветки` – удаление ветки с центрального репозитория

8. Локальный репозиторий – она же директория “.git”. В ней хранятся коммиты и другие объекты.

Удаленный репозиторий – тот репозиторий, который считается общим, в который мы можем передать свои коммиты из локального репозитория, чтобы остальные пользователи могли их увидеть.

Локальный репозиторий мы используем, когда работаем одни и нам нужно сохранить свои же изменения.

Удаленный репозиторий используется для групповой работы, когда в личном репозитории скопилось достаточно коммитов, мы делимся ими в удаленном для того, чтобы другие пользователи могли видеть наши изменения. Также из удаленного репозитория мы можем скачать чужие изменения.

9. Ветка – это подвижный указатель на один из коммитов. Обычно ветка указывает на последний коммит в цепочке коммитов.

В своей ветке мы можем как угодно ломать проект, основной код при этом не пострадает.

10. Игнорируемые файлы – это, как правило, специфичные для платформы файлы или автоматически созданные файлы из систем сборки. Некоторые общие примеры включают в себя:

- Файлы времени выполнения, такие как журнал, блокировка, кэш или временные файлы.
- Файлы с конфиденциальной информацией, такой как пароли или ключи API.
- Скомпилированный код, такой как .class или .o.
- Каталоги зависимостей, такие как /vendor или /node_modules.
- Создавать папки, такие как /public, /out или /dist.
- Системные файлы, такие как .DS_Store или Thumbs.db
- Конфигурационные файлы IDE или текстового редактора.

.gitignore Шаблоны

.gitignore — это простой текстовый файл, в каждой строке которого содержится шаблон, который файлы или каталоги следует игнорировать.

Он использует [шаблоны подстановки](#) для сопоставления имен файлов с подстановочными знаками. Если у вас есть файлы или каталоги, содержащие шаблон подстановки, вы можете использовать одиночную обратную косую черту () для экранирования символа.

Местный .gitignore

`.gitignore` файл `.gitignore` обычно помещается в корневой каталог репозитория. Однако вы можете создать несколько файлов `.gitignore` в разных подкаталогах вашего репозитория. Шаблоны в файлах `.gitignore` сопоставляются относительно каталога, в котором находится файл.

Шаблоны, определенные в файлах, которые находятся в каталогах (подкаталогах) более низкого уровня, имеют приоритет над шаблонами в каталогах более высокого уровня. Локальные файлы `.gitignore` используются совместно с другими разработчиками и должны содержать шаблоны, полезные для всех других пользователей репозитория.

Личные правила игнорирования

Шаблоны, специфичные для вашего локального репозитория и не подлежащие распространению в другие репозитории, должны быть установлены в файле `.git/info/exclude`.

Например, вы можете использовать этот файл, чтобы игнорировать файлы, сгенерированные из ваших личных инструментов проекта.

Глобальный `.gitignore`

Git также позволяет вам создать глобальный файл `.gitignore`, в котором вы можете определить правила игнорирования для каждого репозитория Git в вашей локальной системе.

Файл можно назвать как угодно и хранить в любом месте. Чаще всего этот файл хранится в домашнем каталоге. Вам придется вручную [создать файл](#) и настроить Git для его использования.

Вывод:

Я изучила идеологию и применение средств контроля версий.