

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

Дисциплина: Операционные системы

Преподаватель: Кулябов Дмитрий Сергеевич

Студент: Тулеева Валерия

Группа: НБИбд-01-20

МОСКВА 2021 г.

Управление версиями

Оглавление:

1. Введение: а) Основные команды git б) Цель работы
2. Описание результатов выполнения задания;
3. Контрольные вопросы;
4. Вывод;
5. Библиография.

Введение:

Системы контроля версиями записывают и сохраняют несколько изменений в файлах. Благодаря этому можно вернуться к определенной точке истории изменения файла или проекта. Некоторые системы, такие как Subversion, отслеживают историю отдельных файлов. Другие, такие как Git и Mercurial, отслеживают историю целых репозиториев. Управление версиями подобно системе безопасности. Если вы внесли изменения, которые позже вызвали проблемы, можно будет вернуть файл или весь проект к определенной точке вместо того, чтобы начинать все с нуля. Распределенное управление версиями является популярным благодаря таким системам, как Git и Mercurial. Они широко применяются для организации совместной работы в проектах с открытым исходным кодом. Из-за особенностей настройки клонирование всей базы кода проекта для каждой равноправной системы позволяет получить больше свободы, когда дело касается рабочих процессов и совместной работы. Система контроля Git представляет собой набор программ командной строки.

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`
- 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`
- 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 с текущим деревом: `git merge --no-ff имя_ветки`
- удаление локальной уже слитой с основным деревом ветки: `git branch -d имя_ветки`
- принудительное удаление локальной ветки: `git branch -D имя_ветки`

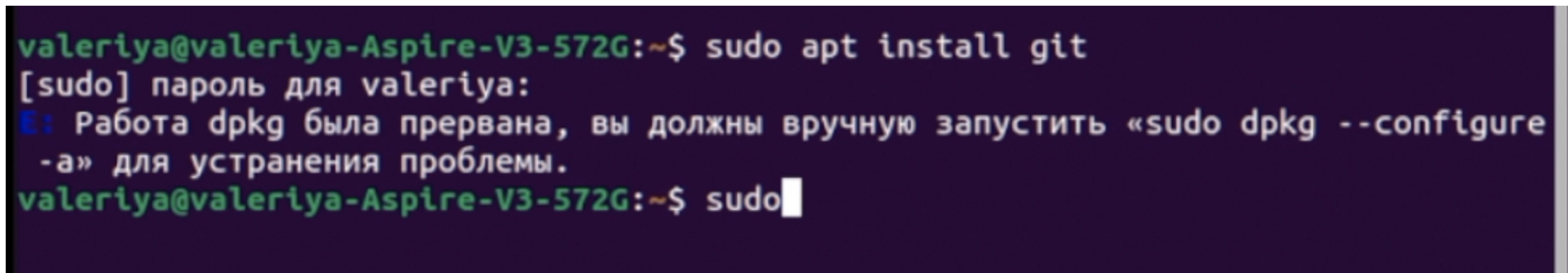
Цель работы:

Изучить идеологию и применение средств контроля версий.

Описание результатов выполнения задания:

- Установила команду git:

```
sudo apt install git (рис 1.1)
```



```
valeriya@valeriya-Aspire-V3-572G:~$ sudo apt install git
[sudo] пароль для valeriya:
E: Работа dpkg была прервана, вы должны вручную запустить «sudo dpkg --configure -a» для устранения проблемы.
valeriya@valeriya-Aspire-V3-572G:~$ sudo
```

Рис 1.1. Установка

- Вручную устранила проблемы:

```
sudo dpkg --configure -a *(рис 2.1)*
```

Рис 2.1. Устранение проблем

для устранения проблемы.

```
valeriya@valeriya-Aspire-V3-572G:~$ sudo dpkg --configure -a
Настраивается пакет virtualbox-dkms (6.1.16-dfsg-6~ubuntu1.20.04.1) ...
Removing old virtualbox-6.1.16 DKMS files...

----- Uninstall Beginning -----
Module: virtualbox
Version: 6.1.16
Kernel: 5.8.0-50-generic (x86_64)
-----
Status: This module version was INACTIVE for this kernel.
depmod...■
```

Рис 2.1. Устранение проблемы

3. Создала учетную запись на [github](#): (*рис 3.1 и рис 3.2*)

Join GitHub

Create your account

Username *

Valeriya851



Email address *

leratuleeva@gmail.com|



Password *

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

Email preferences

Send me occasional product updates, announcements, and offers.

Verify your account

Рис 3.1. Создание учетной записи

Создание учетной записи

Welcome to GitHub

Woohoo! You've joined millions of developers who are doing their best work on GitHub. Tell us what you're interested in. We'll help you get there.

What kind of work do you do, mainly?

Software Engineer

I write code

Student

I go to school

Product Manager

I write specs

UX & Design

I draw interfaces

Data & Analytics

I write queries

Marketing & Sales

I look at charts

4. Создала локальный репозиторий: (рис 4.1)

```
git config --global user.name "Tuleeva Valeriya"  
git config --global user.email leratuleeva@gmail.com
```

```
valeriya@valeriya-Aspire-V3-572G:~$ git config --global user.name "Tuleeva Valeriya"  
valeriya@valeriya-Aspire-V3-572G:~$ git config --global user.email leratuleeva@gmail.com  
valeriya@valeriya-Aspire-V3-572G:~$ █
```

Рис 4.1. Создание локального репозитория

5. Создала репозиторий на Github и назвала его os-intro: (рис 5.1)

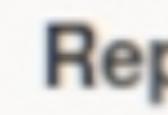
different workflows, or because your integrations still require “master” as the default branch. You can change the default branch name on individual repositories. [Learn more about default branches](#)

New

os-intro

Update

Repositories



Repositories



Deleted repositories

Valeriya851



Valeriya851/os-intro

0 Bytes

0 collaborators

Рис 5.1. Создание репозитория на сайте

6. Создала каталог work и в нем каталог laboratory: (рис 6.1)

```
mkdir work  
cd work  
mkdir laboratory  
cd laboratory
```

```
valeriya@valeriya-Aspire-V3-572G:~$ cd laboratory  
bash: cd: laboratory: Нет такого файла или каталога  
valeriya@valeriya-Aspire-V3-572G:~$ cd work  
valeriya@valeriya-Aspire-V3-572G:~/work$ cd laboratory
```

Рис 6.1. Каталоги

7. Инициализировала систему git: (рис 7.1)

```
git init
```

```
valeriya@valeriya-Aspire-V3-572G:~/work/laboratory$ git init  
Переинициализирован существующий репозиторий Git в /home/valeriya/work/laboratory/.git/  
valeriya@valeriya-Aspire-V3-572G:~/work/laboratory$ echo "#
```

Рис 7.1. Инициализация

8. Создала заготовку для файла README.md: (рис 8.1)

```
echo "# Лабораторные работы" >> README.md
```

```
Переинициализирован существующий репозиторий Git в /home/valeriya/work/laborator  
y/.git/  
valeriya@valeriya-Aspire-V3-572G:~/work/laboratory$ echo "# Лабораторные работы"  
>> README.md  
valeriya@valeriya-Aspire-V3-572G:~/work/laboratory$ git add README.md
```

Рис 8.1. Создание заготовки для файла

9. Сделала первый коммит и выложила на github: (рис 9.1 и рис 9.2)

```
git commit -m "first commit"  
git remote add origin git@github.com:Valeriya851/sciproc-intro.git  
git push -u origin master
```

The screenshot shows a terminal window with the following session:

```
valeriya@valeriya-Aspire-V3-572G: ~/work/laboratory
valeriya@valeriya-Aspire-V3-572G:~$ cd laboratory
bash: cd: laboratory: Нет такого файла или каталога
valeriya@valeriya-Aspire-V3-572G:~$ cd work
valeriya@valeriya-Aspire-V3-572G:~/work$ cd laboratory
valeriya@valeriya-Aspire-V3-572G:~/work/laboratory$ git init
Переинициализирован существующий репозиторий Git в /home/valeriya/work/laboratory/.git/
valeriya@valeriya-Aspire-V3-572G:~/work/laboratory$ echo "# Лабораторные работы"
>> README.md
valeriya@valeriya-Aspire-V3-572G:~/work/laboratory$ git add README.md
valeriya@valeriya-Aspire-V3-572G:~/work/laboratory$ git commit -m "first commit"
[master (корневой коммит) 005d83f] first commit
 1 file changed, 2 insertions(+)
 create mode 100644 README.md
```

Рис 9.1. Создание первого коммита

Обзор Терминал ●

апр 27 13:43 ●

valeriya@valeriya-Aspire-V3-572G: ~/laboratory

```
valeriya@valeriya-Aspire-V3-572G:~$ cd/home/os-intro
bash: cd/home/os-intro: Нет такого файла или каталога
valeriya@valeriya-Aspire-V3-572G:~$ mkdir laboratory
valeriya@valeriya-Aspire-V3-572G:~$ cd laboratory
valeriya@valeriya-Aspire-V3-572G:~/laboratory$ git init
Инициализирован пустой репозиторий Git в /home/valeriya/laboratory/.git/
valeriya@valeriya-Aspire-V3-572G:~/laboratory$ ls =al
ls: невозможно получить доступ к '=al': Нет такого файла или каталога
valeriya@valeriya-Aspire-V3-572G:~/laboratory$ ls -al
итого 12
drwxrwxr-x 3 valeriya valeriya 4096 апр 27 13:33 .
drwxr-xr-x 23 valeriya valeriya 4096 апр 27 13:33 ..
drwxrwxr-x 7 valeriya valeriya 4096 апр 27 13:33 .git
valeriya@valeriya-Aspire-V3-572G:~/laboratory$ echo "# os-intro" >> README.md
valeriya@valeriya-Aspire-V3-572G:~/laboratory$ git init
Переинициализирован существующий репозиторий Git в /home/valeriya/laboratory/.git/
valeriya@valeriya-Aspire-V3-572G:~/laboratory$ git add README.md
valeriya@valeriya-Aspire-V3-572G:~/laboratory$ git commit -m "first commit"
[master (корневой коммит) 345188c] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
valeriya@valeriya-Aspire-V3-572G:~/laboratory$ git branch -M os-intro
valeriya@valeriya-Aspire-V3-572G:~/laboratory$ git remote add origin git@github.com:Valeriya851/os-intro.git
valeriya@valeriya-Aspire-V3-572G:~/laboratory$ git push -u origin os-intro
Перечисление объектов: 3, готово.
Подсчет объектов: 100% (3/3), готово.
Запись объектов: 100% (3/3), 227 bytes | 15.00 KiB/s, готово.
Всего 3 (изменения 0), повторно использовано 0 (изменения 0)
To github.com:Valeriya851/os-intro.git
 * [new branch]      os-intro -> os-intro
Ветка «os-intro» отслеживает внешнюю ветку «os-intro» из «origin».
valeriya@valeriya-Aspire-V3-572G:~/laboratory$
```

Рис 9.2

10. Добавила файл лицензии: (рис 10.1)

```
wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O LICENSE
```

```
wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O LICENSE
```

```
и репозитории существует.  
valeriya@valeriya-Aspire-V3-572G:~/work/laboratory$ wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O LICENSE  
--2021-04-26 16:04:04-- https://creativecommons.org/licenses/by/4.0/legalcode.txt  
Распознаётся creativecommons.org (creativecommons.org)... 104.20.150.16, 172.67.34.140, 104.20.151.16, ...  
Подключение к creativecommons.org (creativecommons.org)|104.20.150.16|:443... соединение установлено.  
HTTP-запрос отправлен. Ожидание ответа... 200 OK  
Длина: нет данных [text/plain]  
Сохранение в: «LICENSE»  
  
LICENSE [ <=> ] 18,22K ---KB/s за 0s  
  
2021-04-26 16:04:05 (53,4 MB/s) - «LICENSE» сохранён [18657]
```

Рис 10.1. Добавление файла лицензии

11. Добавила шаблон игнорируемых файлов: (рис 11.1)

```
curl -L -s https://www.gitignore.io/api/list
```

```
valeriya@valeriya-Aspire-V3-572G:~/work/laboratory$ curl -L -s https://www.gitignore.io/api/list
```

Рис 11.1. Шаблон игнорируемых файлов

12. Скачала шаблон C: (рис 12.1 и рис 12.2)

```
curl -L -s https://www.gitignore.io/api/c >> .gitignore
```

```
yii,yii2,zendframework,zephir,zig  
zsh,zukencr8000valeriya@valeriya-Aspire-V3-572G:~/work/laboratory$ curl -L -s ht  
tp://www.gitignore.io/api/c >> .gitignore  
valeriya@valeriya-Aspire-V3-572G:~/work/laboratory$ █
```

Рис 12.1. Скачивание шаблона C

12.1

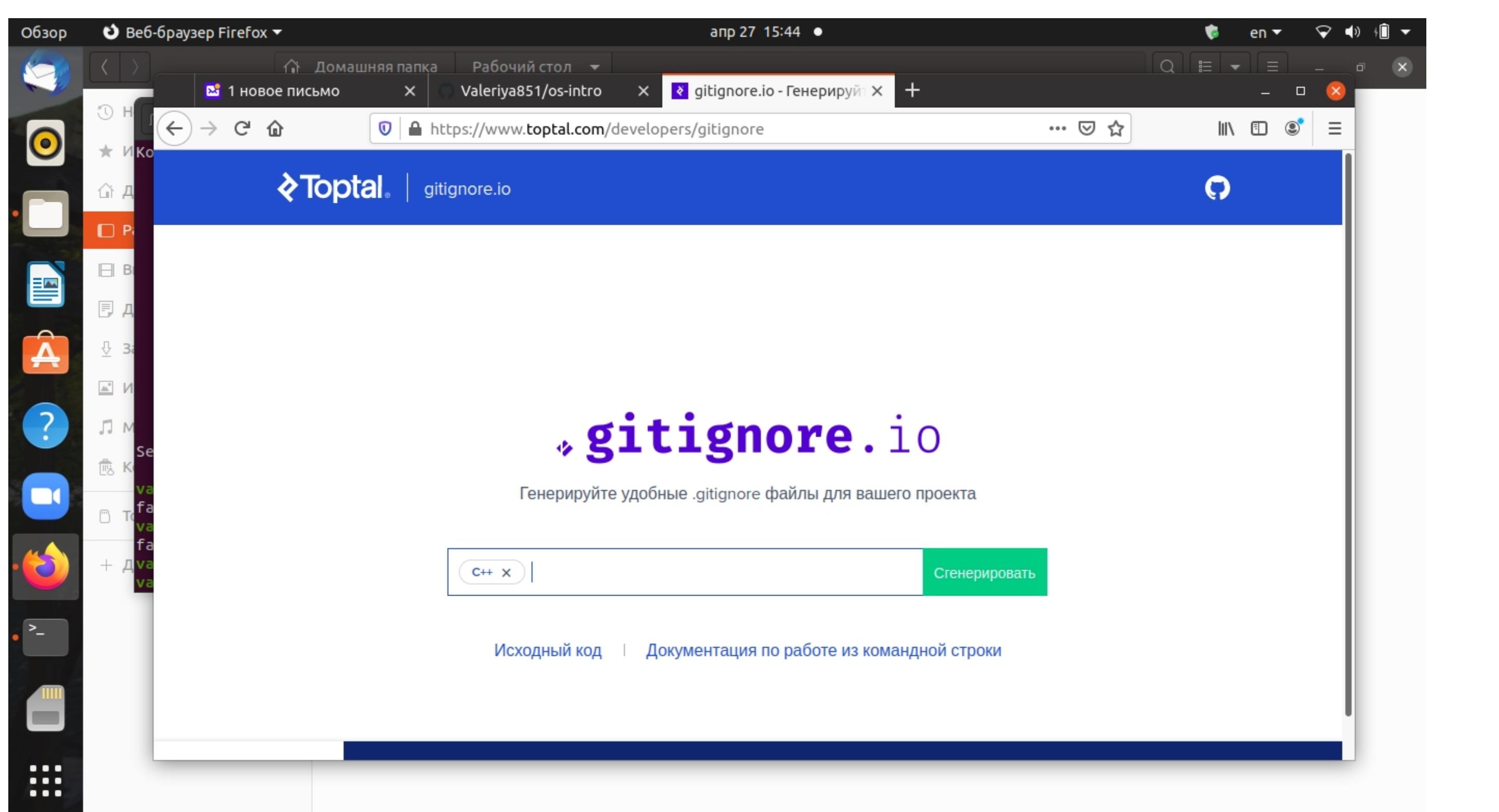


Рис 12.2. Генерация файла

13. Добавила новые файлы: (рис 13.1)

```
git add .gitignore
```

Выполнила коммит:

```
git commit -m
```

Отправила на github:

```
git push
```

valeriya@valeriya-Aspire-V3-572G:~/laboratory\$ curl -L -s https://www.gitignore.io/api/c >> .gitignore
valeriya@valeriya-Aspire-V3-572G:~/laboratory\$ git add .gitignore
valeriya@valeriya-Aspire-V3-572G:~/laboratory\$ git commit -m "gitignore"
[os-intro 2ded068] gitignore
1 file changed, 59 insertions(+)
valeriya@valeriya-Aspire-V3-572G:~/laboratory\$ git push
To github.com:Valeriya851/os-intro.git

Рис. 13.1

14. Сгенерировала ключ: (рис 14.1)

ssh-keygen -t ed25519 -C leratuleeva@gmail.com

valeriya@valeriya-Aspire-V3-572G:~/.ssh\$ ssh-keygen -t ed25519 -C "leratuleeva@gmail.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/valeriya/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:

```
Enter same passphrase again:  
Your identification has been saved in /home/valeriya/.ssh/id_ed25519  
Your public key has been saved in /home/valeriya/.ssh/id_ed25519.pub  
The key fingerprint is:  
SHA256:gseQ7lMOAMCjYP1z16jSpXM+CbV2tfPSJ1QFjBTCSEM leratuleeva@gmail.com  
The key's randomart image is:  
+--[ED25519 256]---+  
|= . oEo..o+.. |  
|.= . . .... . . |  
|+ o + o . . |  
|. o * . = . . . |  
| + 0 S . . . . |  
| . * B + . o. |  
| o o * o .+ |  
| . + ..o. |  
| . . . . . |  
+---[SHA256]---+  
valeriya@valeriya-Aspire-V3-572G:~/.ssh$
```

Рис 14.1. Генерация ключа

15. Установила xclip: (рис 15.1)

sudo apt install xclip

```
valeriya@valeriya-Aspire-V3-572G:~/.ssh$ sudo apt install xclip  
[sudo] пароль для valeriya: █
```

Рис 15.1. Установка

16. Скопировала ключ: (рис 16.1)

```
cat ~/.ssh/id_ed25519.pub | xclip -selclip
```

```
valeriya@valeriya-Aspire-V3-572G:~/.ssh$ cat ~/.ssh/id_ed25519.pub | xclip -selclip
```

Рис 16.1. Копирование ключа

17. Вставила ключ в появившееся на сайте поле. (рис 17.1)

SSH keys / Add new

Title

Key

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGiWMLWALao+RUgLkJwmOYmYDeZ9oks7vPoTzeM10S6Q  
leratuleeva@gmail.com
```



Рис 17.1

18. Загрузила git-flow: (рис 18.1)

```
sudo apt-get install git-flow
```

A terminal window with a dark background and light-colored text. The text shows a command being entered: 'valeriya@valeriya-Aspire-V3-572G:~\$ sudo apt-get install git-flow'. Below this, a password prompt is shown: '[sudo] пароль для valeriya: [REDACTED]' where the password field is redacted.

Рис 18.1. Установка

19. Инициализировала git-flow: (рис 19.1)

```
git flow init
```

```
valeriya@valeriya-Aspire-V3-572G:~$ git flow init
Инициализирован пустой репозиторий Git в /home/valeriya/.git/
No branches exist yet. Base branches must be created now.
Branch name for production releases: [master]
Branch name for "next release" development: [develop]
```

Рис 19.1. Инициализация

20. Проверила, что я на ветке develop: (рис 20.1)

```
git branch
```

```
valeriya@valeriya-Aspire-V3-572G:~$ git branch
* develop
  master
```

Рис 20.1

21. Создала релиз с версией 1.0.0: (рис 21.1)

```
git flow release start 1.0.0
```

```
master
valeriya@valeriya-Aspire-V3-572G:~$ git flow release start 1.0.0
Переключено на новую ветку «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'
```

Рис 21.1. Создание релиза

22. Записала версию: (рис 22.1)

```
echo "1.0.0" >> VERSION
```

```
valeriya@valeriya-Aspire-V3-572G:~$ echo "1.0.0" >> VERSION
```

Рис 22.1. Запись

23. Добавила в индекс: (рис 23.1)

```
git add .
git commit -am 'chore(main): add version'
```

```
valeriya@valeriya-Aspire-V3-572G:~/work/laboratory$ git add .
valeriya@valeriya-Aspire-V3-572G:~/work/laboratory$ git commit -am 'chore(main):
add vesion'
[develop c6ea8e9] chore(main): add vesion
 1 file changed, 1 insertion(+)
create mode 100644 VERSION
```

Рис 23.1. Добавление

24. Залила релизную ветку в основную ветку: (рис 24.1)

```
git flow release finish 1.0.0
```

```
valeriya@valeriya-Aspire-V3-572G:~/work/laboratory$ git flow release finish 1.0.
0
```

Рис 24.1

25. Отправила данные на github:

```
git push -all (рис 25.1)
```

```
git push -tags (рис 25.2)
```

```
valeriya@valeriya-Aspire-V3-572G:~/work/laboratory$ git push --all
Error: Repository not found
```

Рис 25.1

```
valeriya@valeriya-Aspire-V3-572G:~/work/laboratory$ git push --tags
```

Рис 25.2

26. Репозиторий (рис 26.1)

[Code](#)[Issues](#)[Pull requests](#)[Actions](#)[Projects](#)[Wiki](#)[Security](#)[Insights](#)[Settings](#)

os-intro ▾

1 branch

0 tags

[Go to file](#)[Add file ▾](#)[Code ▾](#)

About



No description, website, or topics provided.

[Readme](#)[CC-BY-4.0 License](#)

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

 Valeriya851 Create lab02	22077c0 now	9 commits
 Lab01	Add files via upload	1 minute ago
 Lab02	Create lab02	now
 .gitignore	Create .gitignore	4 hours ago
 LICENSE	Add files via upload	4 hours ago
 README.md	Update README.md	4 hours ago
 VERSION	Add files via upload	4 hours ago
 Лаб 1 OC.md	Add files via upload	2 hours ago

[README.md](#)

Лабораторные работы

Рис 26.1

Контрольные вопросы (лабораторная работа №2)

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

Version Control System, VCS или Revision Control System) — программное обеспечение для облегчения работы с изменяющейся информацией. Система контроля версий — это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии. Для контроля версий файлов в этой книге в качестве примера будет использоваться исходный код программного обеспечения, хотя на самом деле вы можете использовать контроль версий практически для любых типов файлов. Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы.

2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

Хранилище:

Место хранения, из которого пакеты программного обеспечения могут быть получены и установлены на компьютере.

Коммит

Можно сказать, что коммит это основной объект в любой системе управления версиями. В нем содержится описание тех изменений, которые вносит пользователь в код приложения. В Git коммит состоит из нескольких так называемых объектов. Для простоты понимания можно считать, что коммиты это односвязный список, состоящий из объектов в которых содержаться измененные файлы, и ссылка на предыдущий коммит.

История:

Системы контроля версий (VCS) первого поколения отслеживали изменения в отдельных файлах, а редактирование поддерживалось только локально и одним пользователем за раз. Системы строились на предположении, что все пользователи будут заходить по своим учётными записям на один и тот же общий узел Unix.

Рабочая копия:

Рабочая копия - копия проекта, связанная с репозиторием.

3. Что представляет собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида?

Системы контроля версий можно разделить на две группы: распределенные и централизованные.

Централизованные системы контроля версий

Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. Доступ к нему осуществлялся через специальное клиентское приложение. В качестве примеров таких программных продуктов можно привести CVS, Subversion.

CVS (Concurrent Versions System, Система одновременных версий) одна из первых систем получивших широкое распространение среди разработчиков, она возникла в конце 80-х годов прошлого века. В настоящее время этот продукт не развивается, это в первую очередь связано с рядом ключевых недостатков, таких как невозможность переименования файлов, неэффективное их хранение, практически полное отсутствие контроля целостности. Subversion (SVN) – система контроля версий, созданная на замену CVS. SVN была разработана в 2004 году и до сих пор используется. Несмотря на многие преимущества по сравнению с CVS у SVN все-таки есть недостатки, такие как проблемы с переименованием, невозможность удаления данных из хранилища, проблемы в операции слияния ветвей и т.д. В целом SVN был (и остается) значительным шагом вперед по сравнению с CVS.

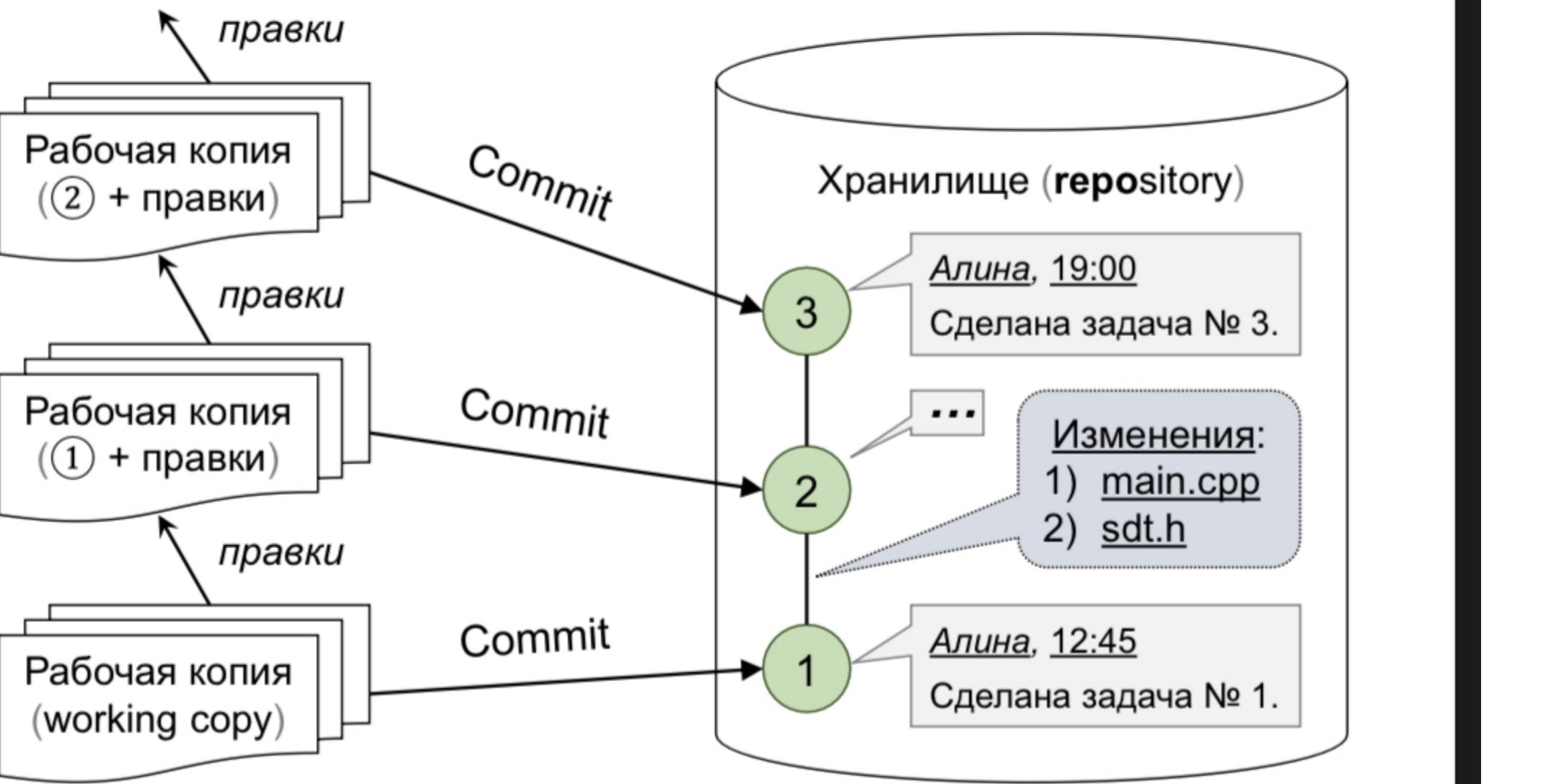
Децентрализованная система:

- В DVCS у каждого есть свой полноценный репозиторий
- DVCS были созданы для обмена изменениями
- При использовании DVCS нет какой-то жестко заданной структуры репозиториев с центральным сервером

К децентрализованным видам деятельности можно отнести: предметы снабжения, используемые для обслуживания, ремонта и эксплуатации, мелкие закупки, обеспечение покупными комплектующими изделиями (состоящие из широкого ассортимента, характеризующиеся постоянной корректировкой ведомостей, зачастую одновременно со строительством).

4. Опишите действия с VCS при единоличной работе с хранилищем.

Единоличная работа с VCS



5. Опишите порядок работы с общим хранилищем VCS.

Каждая система управления версиями имеет свои специфические особенности в наборе команд, порядке работы пользователей и администрировании. Тем не менее, общий порядок работы для большинства VCS совершенно стереотипен. Здесь предполагается, что проект, каким бы он ни был, уже существует и на сервере размещён его репозиторий, к которому разработчик получает доступ.

По мере внесения изменений, хранилища, принадлежащие разным разработчикам, начинают различаться, и возникает необходимость в их синхронизации.

6. Каковы основные задачи, решаемые инструментальным средством git?

Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией. Git — система управления версиями с распределенной архитектурой. В отличие от некогда популярных систем вроде CVS и Subversion (SVN), где полная история версий проекта доступна лишь в одном месте, в Git каждая рабочая копия кода сама по себе является репозиторием. Это позволяет всем разработчикам хранить историю изменений в полном объеме.

*7. Назовите и дайте краткую характеристику командам git. *

• принудительное удаление локальной ветки: `git branch -D имя_ветки`

• создание основного дерева репозитория: `git init`

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

• принудительное обновление локальной ветки: `git pull --force`

• создание нового ветви: `git branch новая_ветка`

• удаление локальной ветки: `git branch -d новая_ветка`

- 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`
- 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`
- 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 с текущим деревом: `git merge --no-ff имя_ветки`
- удаление локальной уже слитой с основным деревом ветки: `git branch -d имя_ветки`

8. Приведите примеры использования при работе с локальным и удаленным репозиториями.

- Локальный репозиторий — она же директория `.git`. В ней хранятся коммиты и другие объекты.
- Удаленный репозиторий — тот самый репозиторий который считается общим, в который вы можете передать свои коммиты из локального репозитория, что бы остальные программисты могли их увидеть. Удаленных репозиториев может быть несколько, но обычно он бывает один.

Работа с удалёнными репозиториями

Для того, чтобы внести вклад в какой-либо Git-проект, вам необходимо уметь работать с удалёнными репозиториями. Удалённые репозитории представляют собой версии вашего проекта, сохранённые в интернете или ещё где-то в сети. У вас может быть несколько удалённых репозиториев, каждый из которых может быть доступен для чтения или для чтения-записи. Взаимодействие с другими пользователями предполагает управление удалёнными репозиториями, а также отправку и получение данных из них. Управление репозиториями включает в себя как умение добавлять новые, так и умение удалять устаревшие репозитории, а также умение управлять различными удалёнными ветками, объявлять их отслеживаемыми или нет и так далее. В данном разделе мы рассмотрим некоторые из этих навыков.

9. Что такое и зачем могут быть нужны ветви branches?

Ветвь в системах управления версиями — направление разработки, независимое от других. Ветвь представляет собой копию части хранилища, в которую можно вносить изменения, не влияющие на другие ветви. Документы в разных ветвях имеют одинаковую историю до точки ветвления и разные — после неё. Ветки нужны для того, чтобы программисты могли вести совместную работу над проектом и не мешать друг другу при этом. При создании проекта, Git создает базовую ветку. Она называется `master` веткой.

10. Как и зачем можно игнорировать некоторые файлы при commit?

Игнорируемые файлы обычно представляют собой файлы, специфичные для платформы, или автоматически созданные из сборочных систем. Ниже приводятся некоторые общие примеры:

- Runtime файлы, такие как `log`, `lock`, `cache` или временные файлы (`tmp`).
- Файлы с конфиденциальной информацией, такие как пароли или ключи API.
- Каталоги зависимостей, такие как `/vendor` или `/node_modules`.
- Build каталоги, такие как `/public` или `/dist`.
- Системные файлы, такие как `.DS_Store` или `Thumbs.db`.
- Конфигурационные файлы IDE или текстового редактора.

Шаблоны `.gitignore`

.gitignore - это простой текстовый файл, в каждой строке которого содержится шаблон файла или каталога, который необходимо проигнорировать.

.gitignore использует glob шаблоны для сопоставления имен файлов с символами подстановки (Это что-то вроде регулярных выражений). Если у вас есть файлы или каталоги, содержащие шаблон подстановки (например *), вы можете использовать один обратный слеш (*), чтобы экранировать такой символ.

Вывод:

В данной лабораторной работе, я изучила идеологию и применение средств контроля версий.

Библиография:

<https://www.internet-technologies.ru/articles/newbie/chto-takoe-upravlenie-versiyami.html>

<https://esystem.rudn.ru/user/policy.php>