

Отчёт по лабораторной работе №2

Операционные системы

Рогожина Надежда Александровна, НКАбд-02-22

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
3.1	Примеры использования git:	7
3.2	Основные команды git:	7
4	Выполнение лабораторной работы	10
5	Выводы	17
6	Контрольные вопросы	18
	Список литературы	23

Список иллюстраций

4.1	Установка системы контроля версий GIT	10
4.2	Первичная настройка git	10
4.3	Базовая настройка git	10
4.4	Создание ssh-ключа	11
4.5	Создание gpg-ключа	11
4.6	Создание gpg-ключа (прод.)	12
4.7	Ввод ключевой фразы	12
4.8	Создание gpg-ключа	12
4.9	Аккаунт github	13
4.10	Вывод списка ключей	13
4.11	Копирование отпечатка ключа	13
4.12	Добавление ключа на github.com	14
4.13	Настройка коммитов	14
4.14	Авторизация в gh	14
4.15	Подтверждение авторизации через браузер	15
4.16	Подтверждение авторизации	15
4.17	Создание репозитория и копирование шаблона	15
4.18	Подтверждение	16
4.19	Настройка каталогов	16
4.20	Отправка файлов на сервер	16

Список таблиц

3.1	Описание некоторых команд системы контроля версий Git	7
-----	---	---

1 Цель работы

- Изучить идеологию и применение средств контроля версий.
- Освоить умения по работе с git.

2 Задание

1. Установка программного обеспечения
 - Установка git
 - Установка gh
2. Базовая настройка git
3. Создайте ключи ssh
4. Создайте ключи pgp
5. Настройка github
6. Добавление PGP ключа в GitHub
7. Настройка автоматических подписей коммитов git
8. Настройка gh
9. Шаблон для рабочего пространства
 - Создание репозитория курса на основе шаблона
 - Настройка каталога курса

3 Теоретическое введение

3.1 Примеры использования git:

- Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями.
- Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией.

3.2 Основные команды git:

Например, в табл. 3.1 приведено краткое описание основных команд Git.

Таблица 3.1: Описание некоторых команд системы контроля версий Git

Команда	Описание команды
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

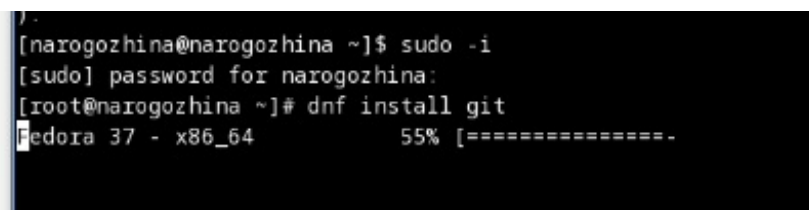
Команда	Описание команды
git status	Просмотр списка изменённых файлов в текущей директории
git diff	Просмотр текущих изменений
git add .	Добавление все изменённые и/или созданные файлы и/или каталоги
git rm име- на_фай- лов	Удаление файлов и/или каталогов из индекса репозитория
git commit -am 'Описание коммита'	Сохранение всех добавленных изменений и всех изменённых файлов
git commit	Сохранение добавленных изменений с внесением комментария через встроенный редактор
git checkout -b имя_ветки	Создание новой ветки, базирующейся на текущей
git branch -d имя_ветки	Удаление локальной уже слитой с основным деревом ветки

Команда	Описание команды
<code>git branch -D имя_ветки</code>	Принудительное удаление локальной ветки

Полный список команд можно посмотреть на официальном сайте: [Github.com](https://github.com)

4 Выполнение лабораторной работы

Первоначально установим гит(рис. 4.1):

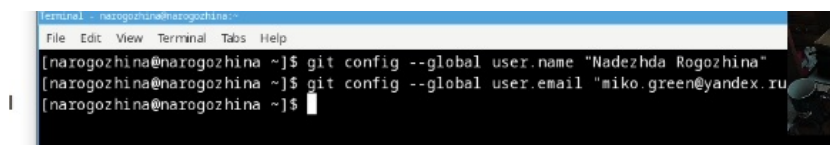


```
[narogozhina@narogozhina ~]$ sudo -i
[sudo] password for narogozhina:
[root@narogozhina ~]# dnf install git
Fedora 37 - x86_64                    55% [=====
```

Рис. 4.1: Установка системы контроля версий GIT

Производим первичную настройку параметров git, а именно:

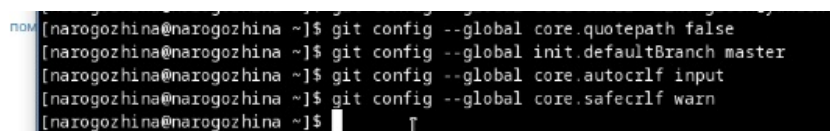
1. Зададим имя и email владельца репозитория(рис. 4.2):



```
[narogozhina@narogozhina ~]$ git config --global user.name "Nadezhda Rogozhina"
[narogozhina@narogozhina ~]$ git config --global user.email "miko.green@yandex.ru"
[narogozhina@narogozhina ~]$
```

Рис. 4.2: Первичная настройки git

2. Настроим utf-8 в выводе сообщений, а также зададим имя начальной ветки, установим параметры **autocrlf** и **safecrlf**(рис. 4.3):



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

Рис. 4.3: Базовая настройка git

3. Создадим ключ *ssh* по алгоритму *rsa* с ключём размером 4096 бит(рис. 4.4):

```
[narogozhina@narogozhina ~]$ git config --global core.safecrlf warn
[narogozhina@narogozhina ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/narogozhina/.ssh/id_rsa):
Created directory '/home/narogozhina/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/narogozhina/.ssh/id_rsa
Your public key has been saved in /home/narogozhina/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:Dm5ncZ1jOSblXyA2Yt5pQHlR0K5iHwU6DrKovw3U20 narogozhina@narogozhina.narogozhina.net
The key's randomart image is:
+----[RSA 4096]-----+
|
|..o+.o
|...+.
|o+o..=
|*o++o+
|..o.S.+&
|..o.+E+=
|o.+=
|..=o
|...o
+-----[SHA256]-----+
[narogozhina@narogozhina ~]$
```

Рис. 4.4: Создание ssh-ключа

4. Создадим *gpg*-ключ(рис. 4.5)(рис. 4.6)(рис. 4.7)(рис. 4.8):

```
+-----[SHA256]-----+
[narogozhina@narogozhina ~]$ gpg --full-generate-key
gpg (GnuPG) 2.3.8; 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: directory '/home/narogozhina/.gnupg' created
gpg: keybox '/home/narogozhina/.gnupg/pubring.kbx' created
Please select what kind of key you want:
(1) RSA and RSA
(2) DSA and Elgamal
(3) DSA (sign only)
(4) RSA (sign only)
(9) ECC (sign and encrypt) *default*
(10) ECC (sign only)
(14) Existing key from card
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 4096
Requested keysize is 4096 bits
Please specify how long the key should be valid.
0 = key does not expire
<n> = key expires in n days
<n>w = key expires in n weeks
<n>m = key expires in n months
<n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y
```

Рис. 4.5: Создание gpg-ключа

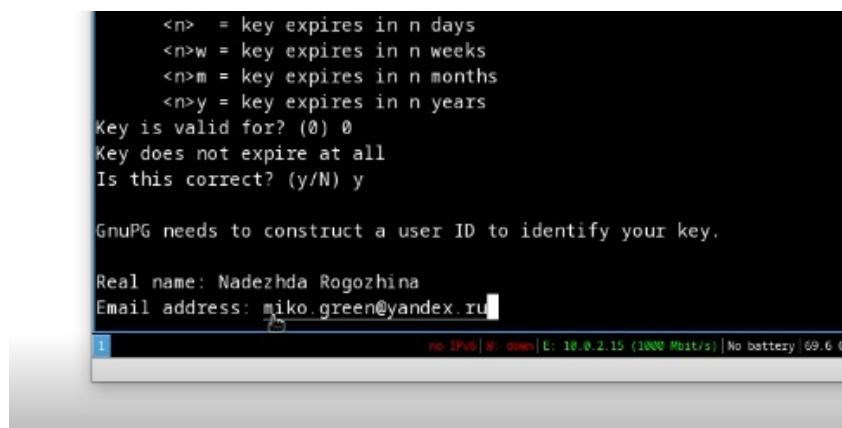


Рис. 4.6: Создание gpg-ключа (прод.)



Рис. 4.7: Ввод ключевой фразы

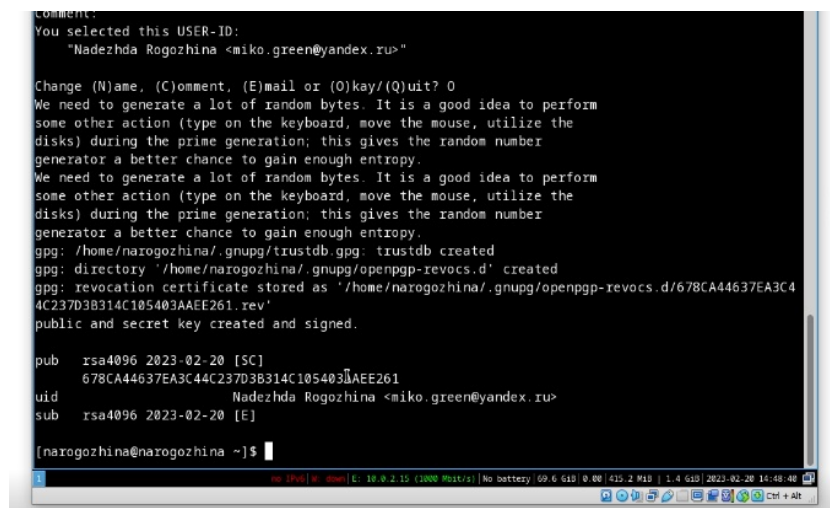


Рис. 4.8: Создание gpg-ключа

5. Настройка github(рис. 4.9):

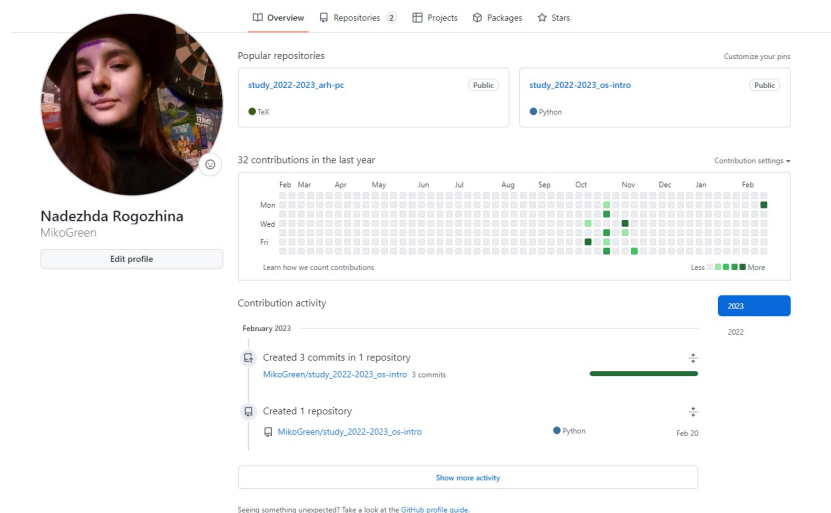


Рис. 4.9: Аккаунт github

6. Добавим PGP ключ в GitHub(рис. 4.10)(рис. 4.11)(рис. 4.12):

```
[narogozhina@narogozhina ~]$ gpg --list-secret-keys --keyid-format LONG
gpg: checking the trustdb
gpg: marginal: needed: 3 completes needed: 1 trust model: pgp
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: /home/narogozhina/.gnupg/pubring.kbx
-----
sec  rsa4096/14C105403AAEE261 2023-02-20 [SC]
      678CA44637EA3C44C237D3B314C105403AAEE261
uid          [ultimate] Nadezhda Rogozhina <miko.green@yandex.ru>
ssb  rsa4096/868D914635838A35 2023-02-20 [E]

[narogozhina@narogozhina ~]$
```

Рис. 4.10: Вывод списка ключей

```
[narogozhina@narogozhina ~]$ gpg --armor --export 14C105403AAEE261 | xclip -sel clip
[narogozhina@narogozhina ~]$
```

Рис. 4.11: Копирование отпечатка ключа

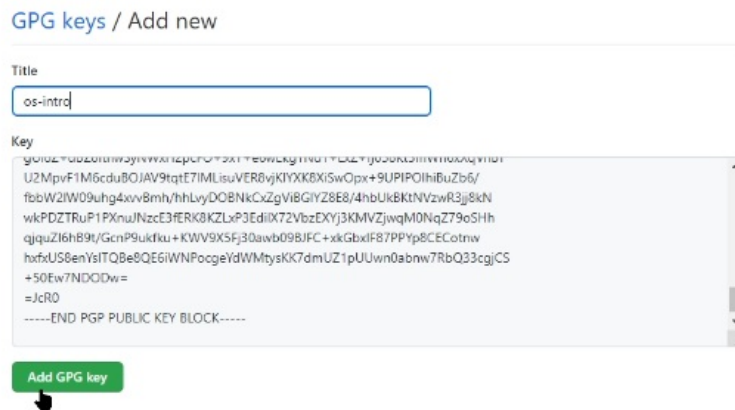


Рис. 4.12: Добавление ключа на github.com

7. Настройка автоматических подписей коммитов git(рис. 4.13):

```
[narogozhina@narogozhina ~]$ gpg --armor --export 14C105403AAEE261 | xclip -sel clip
[narogozhina@narogozhina ~]$ git config --global user.signinkey 14C105403AAEE261
[narogozhina@narogozhina ~]$ git config --global commit.gpgsign true
[narogozhina@narogozhina ~]$ git config --global gpg.program $(which gpg2)
[narogozhina@narogozhina ~]$
```

Рис. 4.13: Настройка коммитов

8. Настройка и авторизация в gh(рис. 4.14)(рис. 4.15)(рис. 4.16):

```
[narogozhina@narogozhina ~]$ git config --global gpg.program $(which gpg2)
[narogozhina@narogozhina ~]$ gh auth login
? What account do you want to log into? GitHub.com
? Upload your SSH public key to your GitHub account? /home/narogozhina/.ssh/id_rsa.pub
? Title for your SSH key: os-intro
? How would you like to authenticate GitHub CLI? Login with a web browser

First copy your one-time code: CEDA-DF08
Press Enter to open github.com in your browser...
```

Рис. 4.14: Авторизация в gh

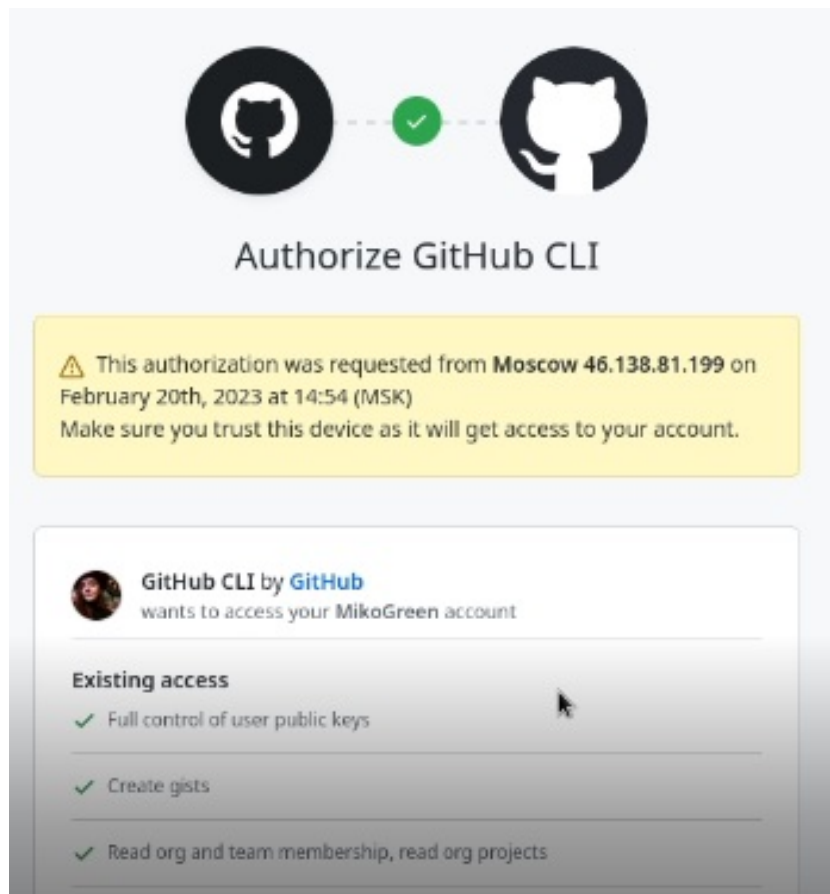


Рис. 4.15: Подтверждение авторизации через браузер

```

✓ Authentication complete.
- gh config set -h github.com git_protocol ssh
✓ Configured git protocol

✓ Uploaded the SSH key to your GitHub account: /home/narogozhina/.ssh/id_rsa.pub
✓ Logged in as MikoGreen
[narogozhina@narogozhina ~]$
[narogozhina@narogozhina ~]$

```

Рис. 4.16: Подтверждение авторизации

9. Создание репозитория курса на основе шаблона(рис. 4.17)(рис. 4.18):

```

[narogozhina@narogozhina ~]$
[narogozhina@narogozhina ~]$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
[narogozhina@narogozhina ~]$ cd

[narogozhina@narogozhina Операционные системы]$ gh repo create study_2022-2023_os-intro --templa
te=yamadharma/course-directory-student-template --public
✓ Created repository MikoGreen/study_2022-2023_os-intro on GitHub
[narogozhina@narogozhina Операционные системы]$ git clone --recursive git@github.com:MikoGreen/s

```

Рис. 4.17: Создание репозитория и копирование шаблона

```
Created repository MikoGreen/study_2022-2023_os-intro on GitHub
[narogozhina@narogozhina Операционные системы]$ git clone --recursive git@github.com:MikoGreen/s
tudy_2022-2023_os-intro.git os-intro
Cloning into 'os-intro'...
The authenticity of host 'github.com (140.82.121.4)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TujJhbpZisF/zLDA0zPMSvHdkr4UvC0qU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Enter passphrase for key '/home/narogozhina/.ssh/id_rsa':
```

Рис. 4.18: Подтверждение

10. Настройка репозитория и каталогов(рис. 4.19):

Удаление лишний файлов, а также создание необходимых каталогов:

```
[narogozhina@narogozhina os-intro]$ rm package.json
[narogozhina@narogozhina os-intro]$ echo os-intro > COURSE
[narogozhina@narogozhina os-intro]$ make
[narogozhina@narogozhina os-intro]$ git add .
```

Рис. 4.19: Настройка каталогов

11. Отправка файлов на сервер(рис. 4.20):

```
[narogozhina@narogozhina os-intro]$ git add .
[narogozhina@narogozhina os-intro]$ git commit -am 'feat(main): make course structure'
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
[narogozhina@narogozhina os-intro]$ git push
Enter passphrase for key '/home/narogozhina/.ssh/id_rsa':
Enumerating objects: 40, done.
Counting objects: 100% (40/40), done.
Compressing objects: 100% (30/30), done.
Writing objects: 100% (38/38), 343.06 KiB | 2.70 MiB/s, done.
Total 38 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:MikoGreen/study_2022-2023_os-intro.git
a229ba2..510f0a4 master -> master
[narogozhina@narogozhina os-intro]$
```

Рис. 4.20: Отправка файлов на сервер

5 Выводы

В ходе данной лабораторной работы мы приобрели навыки работы с системой контроля версий git.

6 Контрольные вопросы

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

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.

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

Хранилище - место хранения всех версий и служебной информации.

Commit - (==версия) создание новой версии.

История - журнал изменений.

Рабочая копия - текущее состояние файлов проекта, основанное на версии из хранилища.

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

Централизованные VCS имеют одно основное хранилище всего проекта, достаточно просты в использовании. 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 не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Необходимо подключение к сети. Например:

- Subversion
- CVS
- TFS, VAULT
- AccuRev

Децентрализованные VCS характеризуются тем, что у каждого пользователя свой вариант репозитория. Присутствует возможность добавлять и забирать изменения из любого репозитория. Подключение к сети не нужно. Например:

- Git
- Mercurial
- Bazaar

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

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить

рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

5. Опишите порядок работы с общим хранилищем VCS.

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом. 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.

6. Каковы основные задачи, решаемые инструментальным средством git?

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.

Репозиторий Git — это место, где хранится ваш код и вся информация о его изменениях. Репозитории могут находиться у вас на компьютере, на компьютерах ваших коллег и на удалённом сервере.

7. Назовите и дайте краткую характеристику командам git.

Основные команды git:

Команда	Описание команды
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 .	Добавление все изменённые и/или созданные файлы и/или каталоги
git rm имена_файлов	Удаление файлов и/или каталогов из индекса репозитория
git commit -am 'Описание коммита'	Сохранение всех добавленных изменений и всех изменённых файлов
git commit	Сохранение добавленный изменений с внесением комментария через встроенный редактор
git checkout -b имя_ветки	Создание новой ветки, базирующейся на текущей
git branch -d имя_ветки	Удаление локальной уже слитой с основным деревом ветки
git branch -D имя_ветки	Принудительное удаление локальной ветки

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

Удаленный репозиторий – это версии вашего проекта, сохраненные на удаленном сервере. Доступ к репозиторию на таком сервере может осуществляться по интернету или по локальной сети. Если мы подключим удаленный репозиторий к своему локальному, то у нас появятся копии всех ссылочных объектов удаленного репозитория. То есть, например, у удаленного репозитория есть ветка `main`, а у нас будет копия этой ветки – `origin/main`. Все такие ссылочные объекты (указатели, ветки и теги) удаленного репозитория хранятся почти там же, где и у локального – в директории `.git/refs/remotes/`.

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

Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка специально для экспериментов.

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

Git рассматривает каждый файл в вашей рабочей копии как файл одного из трех нижеуказанных типов.

Отслеживаемый файл — файл, который был предварительно проиндексирован или зафиксирован в коммите. Неотслеживаемый файл — файл, который не был проиндексирован или зафиксирован в коммите. Игнорируемый файл — файл, явным образом помеченный для Git как файл, который необходимо игнорировать. Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты.

Вы можете создать файл `.gitignore` в корневом каталоге репозитория, чтобы сообщить Git, какие файлы и каталоги следует игнорировать при фиксации. Чтобы поделиться правилами игнорирования с другими пользователями, которые клонируют репозиторий, зафиксируйте файл `.gitignore` в репозитории.

Список литературы

1. Руководство по лабораторной работе №2
2. Основная информация LINUX
3. Система контроля версий. Горвиц Евгений
4. Руководство пользования GIT