Лабораторная работа № 2

Первоначальная настройка git.

Захаренко Анастасия Викторовна

Содержание

Список иллюстраций

Список таблиц

Цель работы

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

Задание

Создать базовую конфигурацию для работы с git. Создать ключ SSH. Создать ключ PGP. Настроить подписи git. Зарегистрироваться на Github. Создать локальный каталог для выполнения заданий по предмету.

Теоретическое введение

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. Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

Выполнение лабораторной работы

Установка программного обеспечения

Установим git и gh:

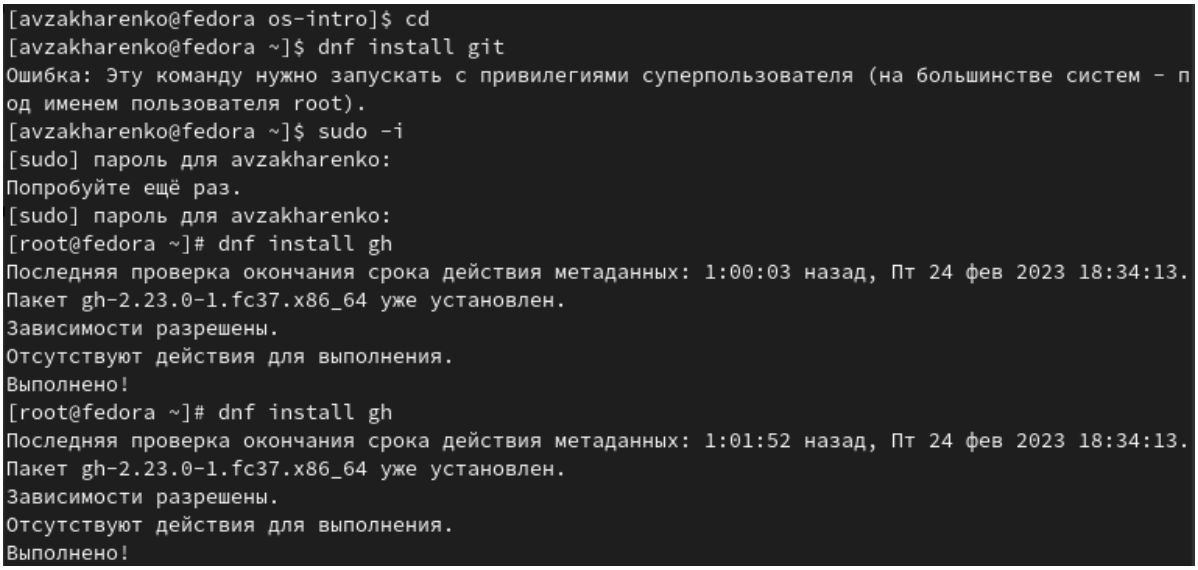

Рис. 0.1: установили git и Fedora

Базовая настройка git

Зададим имя и email владельца репозитория:

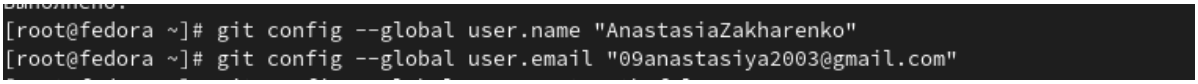

Рис. 0.2: имя и email владельца

Настроим utf-8 в выводе сообщений git:

```
[root@fedora ~]# git config --global core.quotepath false
```

Рис. 0.3: настройка utf-8 в выводе сообщений git

Зададим имя начальной ветки (будем называть её master):

```
[root@fedora ~]# git config --global init.defaultBranch master
```

Рис. 0.4: имя начальной ветки

Параметр autocrlf:

```
[root@fedora ~]# git config --global core.autocrlf input
```

Рис. 0.5: autocrlf

Параметр safecrlf:

```
[root@fedora ~]# git config --global core.safecrlf warn
```

Рис. 0.6: safecrlf

Создаем ключи ssh

по алгоритму rsa с ключём размером 4096 бит:

```
[root@fedora ~]# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:8Wm0haQTiS4jNDNB1GoFuTg4V0cybEGsQvp6PfnaRNI root@fedora
The key's randomart image is:
+---[RSA 4096]----+
| o=O=o.....      |
| .* Boo .+ .     |
|+o @ o  + o .    |
|B B o..  = +     |
| B ..oE S =      |
|  .  o   .       |
| . . ..          |
|. . +o           |
| .  .+o          |
+----[SHA256]-----+
```

Рис. 0.7: rsa ключ

по алгоритму ed25519:

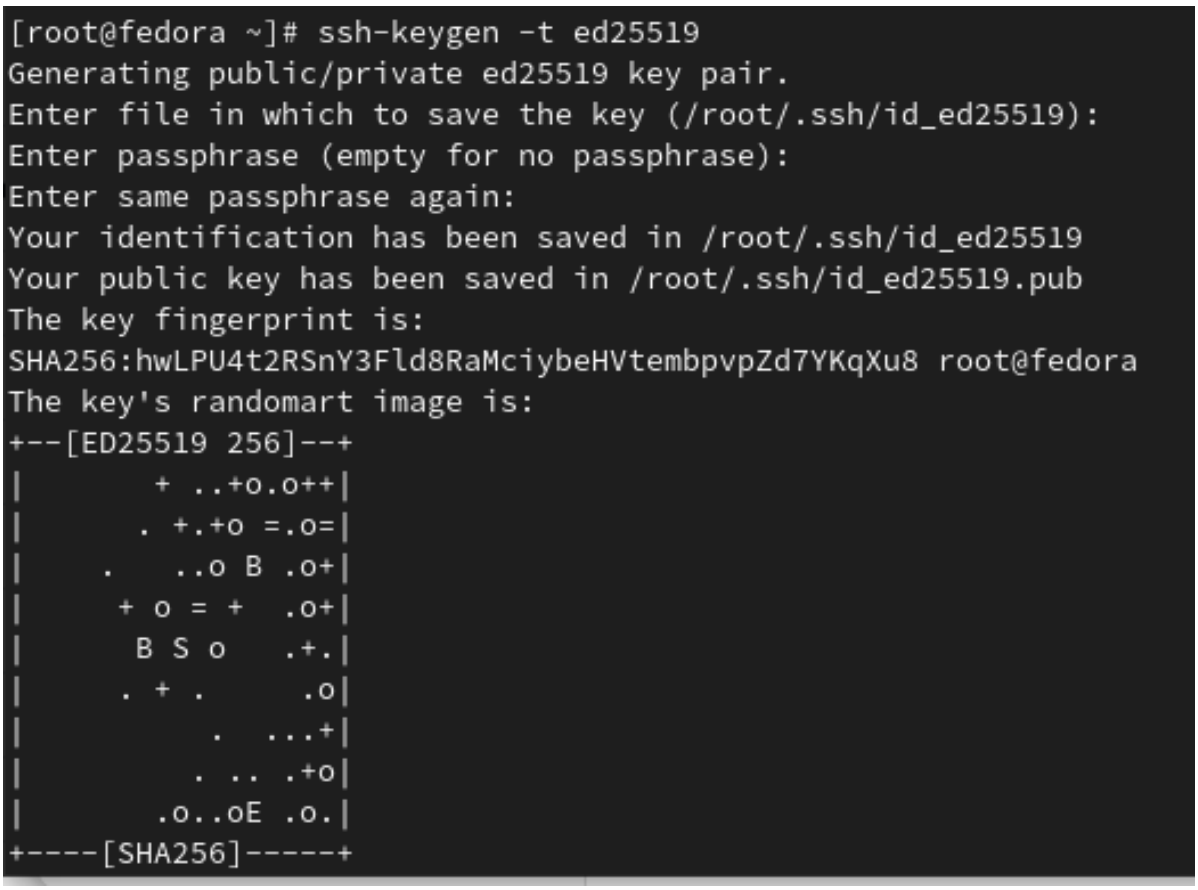

Рис. 0.8: ed25519 ключ

Создаем ключи pgp

Генерируем ключ

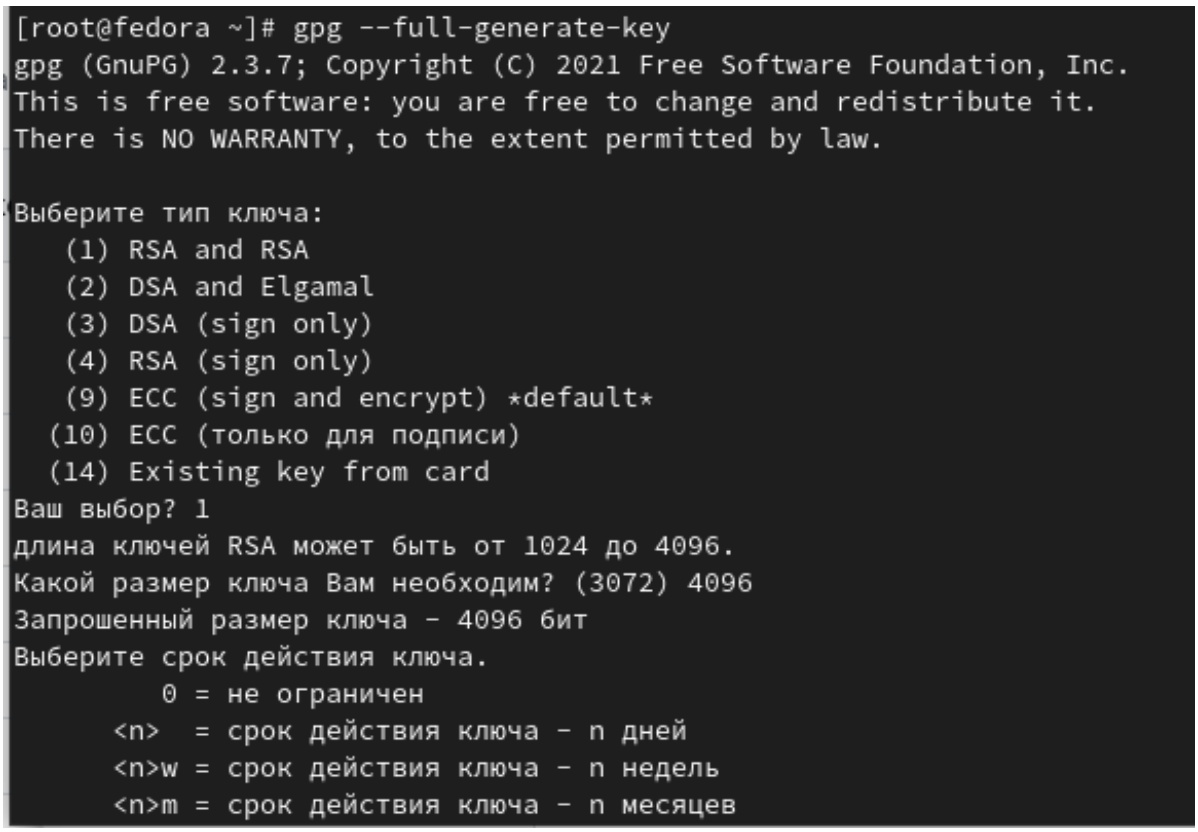
[root@fedora ~]# gpg --full-generate-key
gpg (GnuPG) 2.3.7; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Выберите тип ключа:
(1) RSA and RSA
(2) DSA and Elgamal
(3) DSA (sign 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

Рис. 0.9: gpg

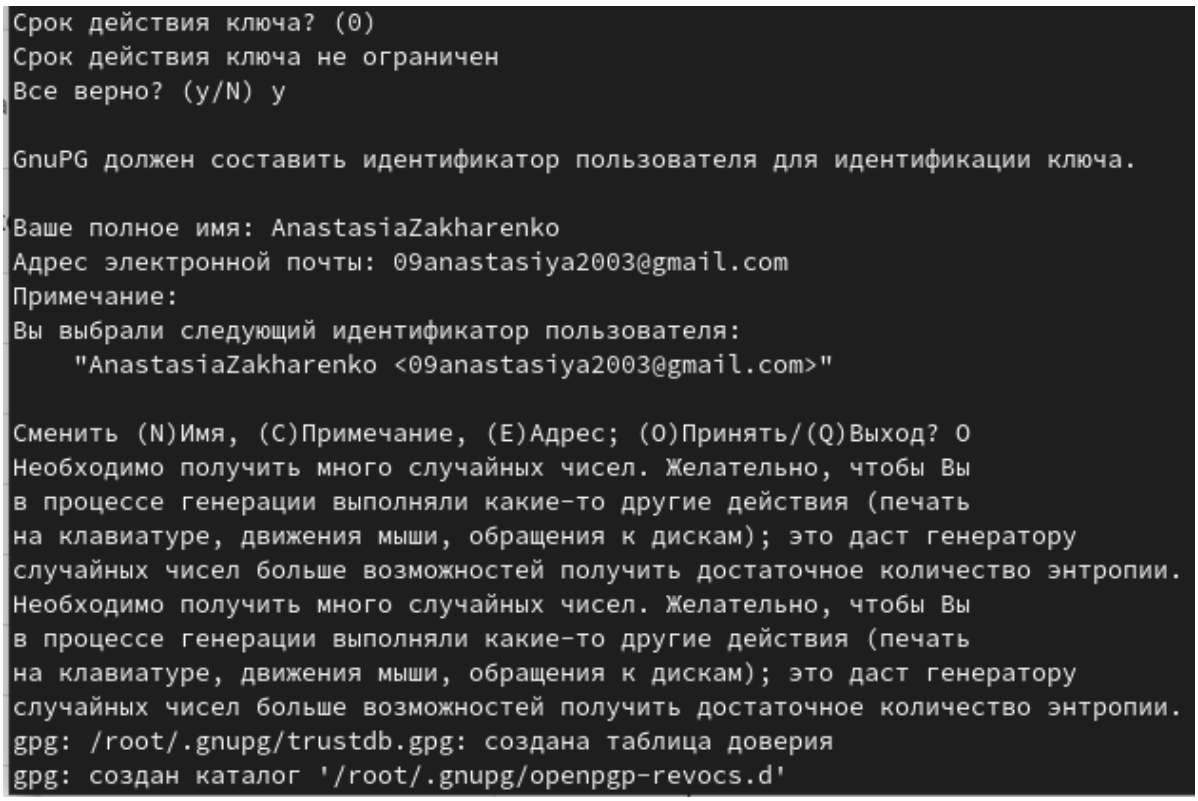

Рис. 0.10: gpg

Из предложенных опций выбрали: 1. тип RSA and RSA; 2. размер 4096; 3. выберите срок действия; значение по умолчанию — 0 (срок действия не истекает никогда). Ввели личную информацию, которая сохранится в ключе

Добавление PGP ключа в GitHub

Выводим список ключей и копируем отпечаток приватного ключа:

```
[root@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
/root/.gnupg/pubring.kbx
------------------------
sec   rsa4096/C358E97582A98677 2023-02-24 [SC]
      1D6194CA2A11E1B86A68865EC358E97582A98677
uid                 [  абсолютно ] AnastasiaZakharenko <09anastasiya2003@gmail.com>
ssb   rsa4096/58B9588A674E1296 2023-02-24 [E]
```

Рис. 0.11: добавляем ключи в GitHub

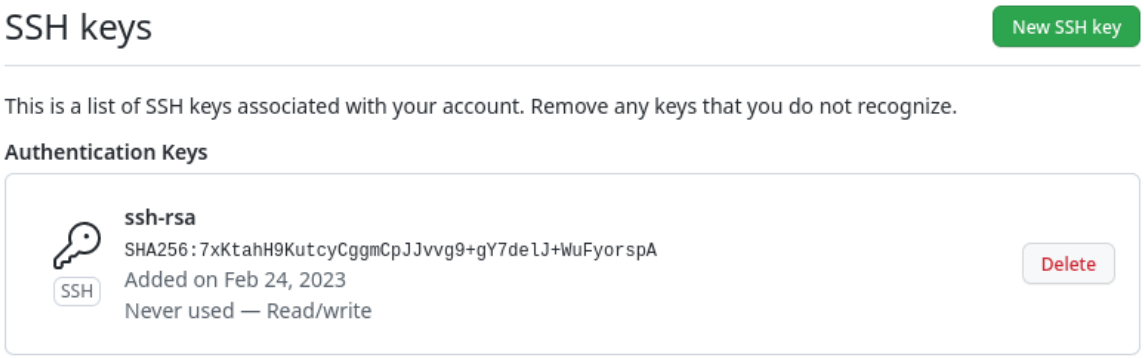

Рис. 0.12: ssh

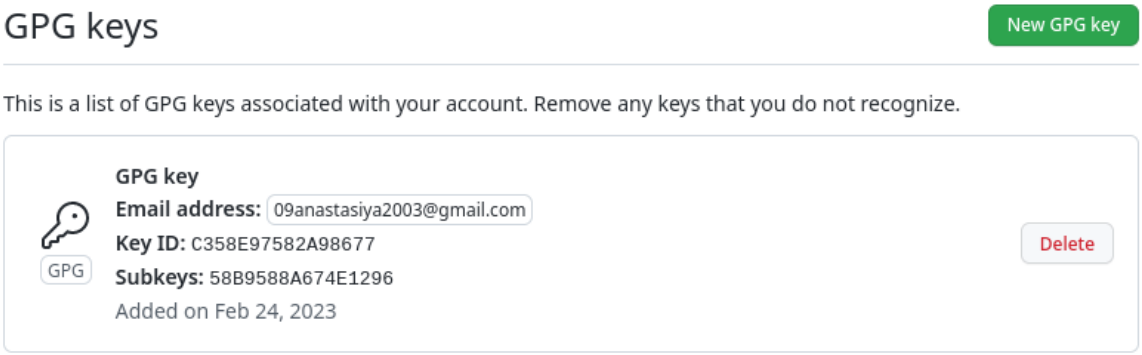

Рис. 0.13: gpg

Настройка автоматических подписей коммитов git

Используя введёный email, указываем Git применять его при подписи коммитов:

```
[root@fedora ~]# gpg --armor --export C358E97582A98677 | xclip -sel clip
[root@fedora ~]# cat ~/.ssh/id_rsa.pub | xclip -sel clip
[root@fedora ~]# git config --global user.signingkey C358E97582A98677
[root@fedora ~]# git config --global commit.gpgsign true
[root@fedora ~]# git config --global gpg.program $(which gpg2)
```

Рис. 0.14: настройка

Настройка gh

Для начала необходимо авторизоваться

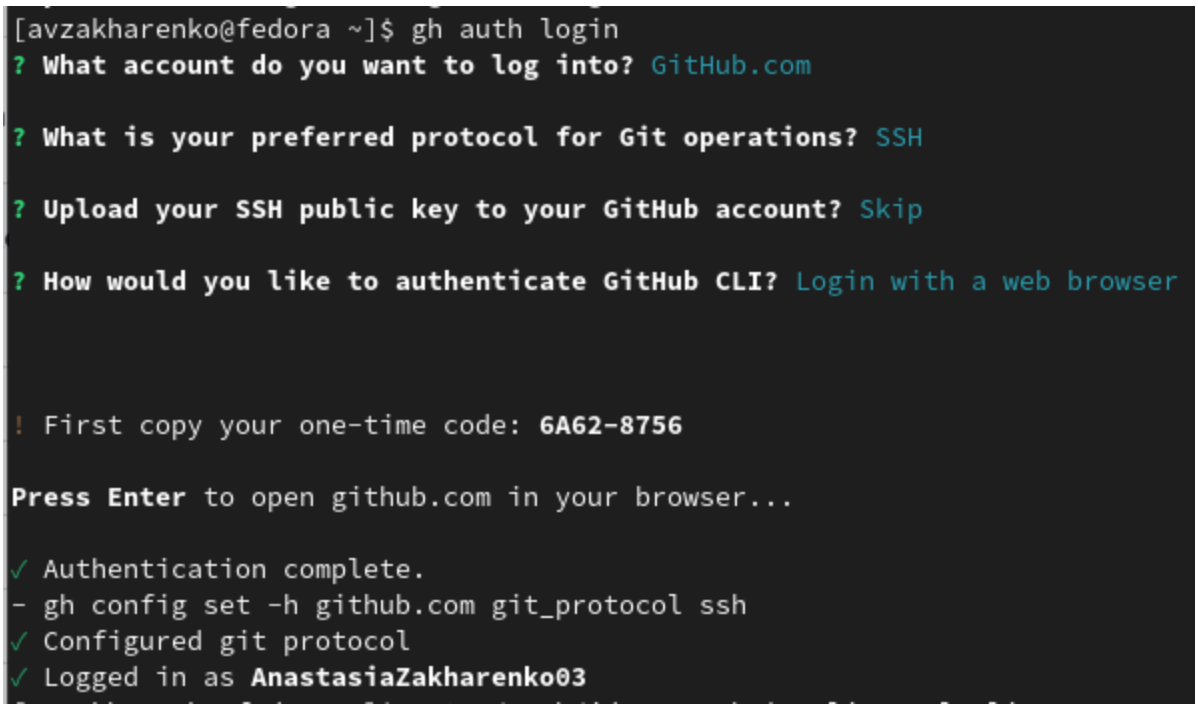

Рис. 0.15: авторизация

Сознание репозитория курса на основе шаблона

создать шаблон рабочего пространства

```
[avzakharenko@fedora ~]$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
[avzakharenko@fedora ~]$ cd ~/work/study/2022-2023/"Операционные системы"
[avzakharenko@fedora Операционные системы]$ gh repo create study_2022-2023_os-intro --template=
yamadharma/course-directory-student-template --public
✓ Created repository AnastasiaZakharenko03/study_2022-2023_os-intro on GitHub
[avzakharenko@fedora Операционные системы]$ git clone --recursive https://github.com/AnastasiaZ
akharenko03/study_2022-2023_os-intro.git os-intro
Клонирование в «os-intro»...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 27 (delta 1), reused 11 (delta 0), pack-reused 0
Получение объектов: 100% (27/27), 16.94 КиБ | 385.00 КиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown
-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.
git) зарегистрирован по пути «template/report»
Клонирование в «/home/avzakharenko/work/study/2022-2023/Операционные системы/os-intro/template/
presentation»...
```

Рис. 0.16: создание репозитория

Настройка каталога курса

Перейдите в каталог курса:

```
[avzakharenko@fedora Операционные системы]$ cd ~/work/study/2022-2023/"Операционные системы"/os
-intro
```

Рис. 0.17: переход

Удалите лишние файлы:

```
[avzakharenko@fedora os-intro]$ rm package.json
```

Рис. 0.18: удаление

Создайте необходимые каталоги:

```
[avzakharenko@fedora os-intro]$ echo os-intro > COURSE
[avzakharenko@fedora os-intro]$ make
```

Рис. 0.19: создание

Отправьте файлы на сервер:

```
[avzakharenko@fedora os-intro]$ git add .
[avzakharenko@fedora os-intro]$ git commit -am 'feat(main): make course structure'
[master fff6d70] feat(main): make course structure
 361 files changed, 100327 insertions(+), 14 deletions(-)
 create mode 100644 labs/README.md
 create mode 100644 labs/README.ru.md
 create mode 100644 labs/lab01/presentation/Makefile
 create mode 100644 labs/lab01/presentation/image/kulyabov.jpg
 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. 0.20: отправка

```
[avzakharenko@fedora os-intro]$ git push
Username for 'https://github.com': AnastasiaZakharenko03
Password for 'https://AnastasiaZakharenko03@github.com':
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 343.05 КиБ | 2.77 МиБ/с, готово.
Всего 38 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
```

Рис. 0.21: отправка

Рис. 0.22: отправка

#Контрольные вопросы 1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? 2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. 3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. 4. Опишите действия с VCS при единоличной работе с хранилищем. 5. Опишите порядок работы с общим хранилищем VCS. 6. Каковы основные задачи, решаемые инструментальным средством git? 7. Назовите и дайте краткую характеристику командам git. 8. Приведите примеры использования при работе с локальным и удалённым репозиториями. 9. Что такое и зачем могут быть нужны ветви (branches)? 10. Как и зачем можно игнорировать некоторые файлы при commit?

#Ответы на контрольные вопросы

1. Контроль версий, также известный как управление исходным кодом, — это практика отслеживания изменений программного кода и управления ими. Системы контроля версий — это программные инструменты, помогающие командам разработчиков управлять изменениями в исходном коде с течением времени.
2. Репозиторий - хранилище версий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией. Рабочая копия -

копия проекта, связанная с репозиторием Коммит, фиксация commit, checkin-сохранение изменений в репозитории

3. Централизованные VCS. Клиент-серверная модель: один центральный репозиторий, с которым разработчики взаимодействуют по сети.Примеры:CVS- одна из первых систем второго поколения (1986г.). Обладает множеством недостатков и считается устаревшей.Subversion (SVN) – система второго поколения, созданная для замены CVS. Одна из самых распространенных систем контроля версий. Распределенные системы контроля версий (Distributed Version Control System, DVCS) позволяют хранить репозиторий (его копию) у каждого разработчика, работающего с данной системой. При этом можно выделить центральный репозиторий (условно), в который будут отправляться изменения из локальных и, с ним же эти локальные репозитории будут синхронизироваться. Mercurial,Git
4. 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 не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.
5. 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 зависи-

мости 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.

6. Примеры использования gitСистема 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.
7. Основные команды git: Перечислим 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 с текущим деревом: git merge –no-ff имя_ветки Удаление ветки: удаление локальной уже слитой с основным деревом ветки: git branch -d имя_ветки принудительное уда-

ление локальной ветки: git branch -D имя_ветки удаление ветки с центрального репозитория: git push origin :имя_ветки

8. 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” Настроим 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 После это в каталоге tutorial появится каталог .git, в котором будет храниться история изменений. Создадим тестовый текстовый файл hello.txt и добавим его в локальный репозиторий: echo ‘hello world’ > hello.txt git add hello.txt git commit -am ‘Новый файл’ Воспользуемся командой status для просмотра изменений в рабочем каталоге, сделанных с момента последней ревизии: git status 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 список имеющихся 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
9. 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. Можно принудительно сделать коммит игнорируемого файла в репозиторий с помощью команды git add с параметром -f (или –force) Этот способ хорош, если у вас задан общий шаблон (например, *.log), но вы хотите сделать коммит определенного файла. Однако еще лучше в этом случае задать исключение из общего правила:

Выводы

Мы изучили идеологию и применение средств контроля версий и освоили умения по работе с git.

Список литературы