

Лабораторная работа №2

Дисциплина: Операционные системы

Ищенко Ирина Олеговна

Содержание

| | | |
|---|--------------------------------|----|
| 1 | Цель работы | 5 |
| 2 | Выполнение лабораторной работы | 6 |
| 3 | Ответы на контрольные вопросы | 13 |
| 4 | Выводы | 16 |
| | Список литературы | 17 |

Список иллюстраций

| | | |
|------|---|----|
| 2.1 | Установка git и gh | 6 |
| 2.2 | Настройка параметров | 6 |
| 2.3 | ssh ключ | 7 |
| 2.4 | ssh ключ | 7 |
| 2.5 | pgp ключ | 8 |
| 2.6 | Копирование ключа | 8 |
| 2.7 | Добавление ключа | 9 |
| 2.8 | Настройка подписей коммитов | 9 |
| 2.9 | Авторизация | 10 |
| 2.10 | Создание репозитория | 10 |
| 2.11 | Создание репозитория | 11 |
| 2.12 | Удаление и добавление каталогов | 11 |
| 2.13 | Отправление файлов на сервер | 11 |
| 2.14 | Отправление файлов на сервер | 12 |
| 2.15 | Итоговый вид репозитория | 12 |

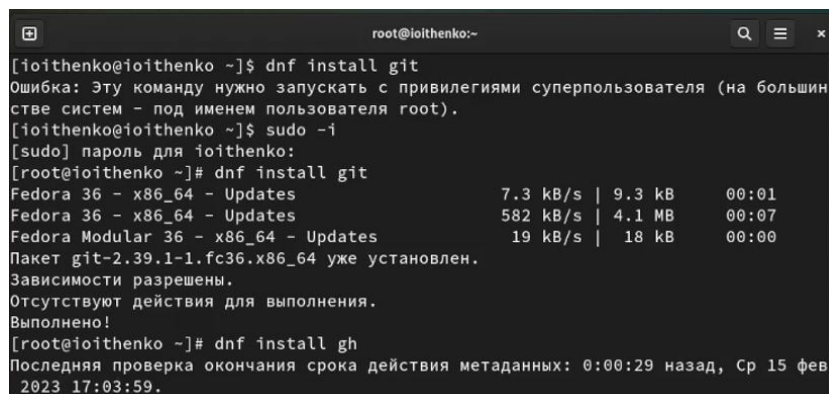
Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

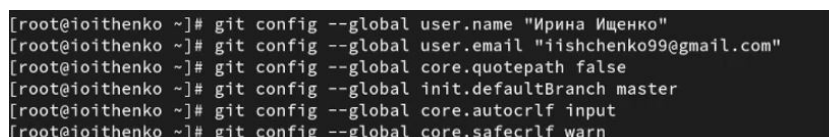
Установили git и gh (рис. 2.1).



```
root@ioithenko:~  
[ioithenko@ioithenko ~]$ dnf install git  
Ошибка: Эту команду нужно запускать с привилегиями суперпользователя (на большин  
стве систем - под именем пользователя root).  
[ioithenko@ioithenko ~]$ sudo -i  
[sudo] пароль для ioithenko:  
[root@ioithenko ~]# dnf install git  
Fedora 36 - x86_64 - Updates          7.3 kB/s | 9.3 kB      00:01  
Fedora 36 - x86_64 - Updates          582 kB/s | 4.1 MB      00:07  
Fedora Modular 36 - x86_64 - Updates  19 kB/s | 18 kB       00:00  
Пакет git-2.39.1-1.fc36.x86_64 уже установлен.  
Зависимости разрешены.  
Отсутствуют действия для выполнения.  
Выполнено!  
[root@ioithenko ~]# dnf install gh  
Последняя проверка окончания срока действия метаданных: 0:00:29 назад, Ср 15 фев  
2023 17:03:59.
```

Рис. 2.1: Установка git и gh

Зададим имя и email владельца репозитория. Настроим utf-8 в выводе сообще-
ний git. Зададим имя начальной ветки. Задаем параметры autocrlf и safecrlf (рис.
2.2).



```
[root@ioithenko ~]# git config --global user.name "Ирина Ищенко"  
[root@ioithenko ~]# git config --global user.email "iishchenko99@gmail.com"  
[root@ioithenko ~]# git config --global core.quotepath false  
[root@ioithenko ~]# git config --global init.defaultBranch master  
[root@ioithenko ~]# git config --global core.autocrlf input  
[root@ioithenko ~]# git config --global core.safecrlf warn
```

Рис. 2.2: Настройка параметров

Создаем ключи ssh: - по алгоритму rsa с ключём размером 4096 бит (рис. 2.3).

```

[root@ioithenko ~]# 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:uqu7kfUezl6x/aVAhVEi/EipI0qDBM053bZWkBXta9U root@ioithenko
The key's randomart image is:
+---[RSA 4096]-----+
|. + o ..+++.oo. |
| * . + ..++E |
| . o . o o =. . |
| . o + o + .. |
| . +..So. . |
|.o o = |
| o . o o o . |
| . = o o o |
| ++oo= o |
+---[SHA256]-----+
[root@ioithenko ~]#

```

Рис. 2.3: ssh ключ

- по алгоритму ed25519 (рис. 2.4).

```

[root@ioithenko ~]# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519
Your public key has been saved in /root/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:FyfgBkmoHryDT4HM878RjU6MxJDbpEDBi3pOsowm87M root@ioithenko
The key's randomart image is:
+--[ED25519 256]--+
|o+o oo.. |
|.oo.. .o . |
|=o*+ o o . |
|oB*.o o. + |
|.oo= = .S . |
|+ B.o . . |
|o0 ..o |
|=o+ .. |
|ooEo .. |
+---[SHA256]-----+
[root@ioithenko ~]#

```

Рис. 2.4: ssh ключ

Генерируем ключ pgr: тип RSA and RSA; размер 4096; срок действия - 0 (рис. 2.5). Указываем имя и адрес электронной почты (рис. ??).

```
[root@ioithenko ~]# 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.

gpg: создан каталог '/root/.gnupg'
gpg: создан щит с ключами '/root/.gnupg/pubring.kbx'
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? █
```

Рис. 2.5: pgr ключ

Учетная запись на github настроена с первого семестра.

Выводим список ключей и копируем отпечаток приватного ключа. Скопируем наш сгенерированный PGP ключ в буфер обмена (рис. 2.6).

```
root@ioithenko:~
pub  rsa4096 2023-02-15 [SC]
     7DF4013F58E3045AF94E51DE458CBD8967FC1F42
uid      Ирина Ищенко <iishchenko99@gmail.com>
sub  rsa4096 2023-02-15 [E]

[root@ioithenko ~]# 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: 2  подписанных: 0  доверие: 0-, 0q, 0n, 0m, 0f, 2
u
-----
sec  rsa4096/B12277B5CB96AEF2 2023-02-15 [SC]
     91208A142A3406D568038B2DB12277B5CB96AEF2
uid      [ абсолютно ] Ирина Ищенко <iishchenko99@gmail.com>
ssb  rsa4096/4C9F10B3E282CAFE 2023-02-15 [E]

sec  rsa4096/458CBD8967FC1F42 2023-02-15 [SC]
     7DF4013F58E3045AF94E51DE458CBD8967FC1F42
uid      [ абсолютно ] Ирина Ищенко <iishchenko99@gmail.com>
ssb  rsa4096/9F50F4BC796A294B 2023-02-15 [E]

[root@ioithenko ~]# gpg --armor --export B12277B5CB96AEF2 | xclip -sel clip
[root@ioithenko ~]#
```

Рис. 2.6: Копирование ключа

В настройках github добавляем полученный ключ (рис. 2.7).

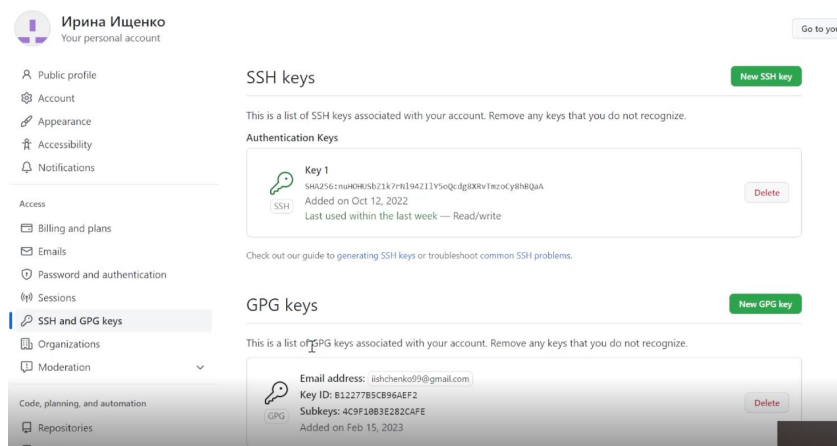


Рис. 2.7: Добавление ключа

Используя введённый email, укажем Git применять его при подписи коммитов (рис. 2.8).

```

root@ioithenko:~# gpg --armor --export B12277B5CB96AEF2 | xclip -sel clip
-----BEGIN PGP PUBLIC KEY BLOCK-----
uid          [ абсолютно ] Ирина Ищенко <iishchenko99@gmail.com>
ssb          rsa4096/4C9F10B3E282CAFE 2023-02-15 [E]

sec          rsa4096/458CBD8967FC1F42 2023-02-15 [SC]
uid          [ абсолютно ] Ирина Ищенко <iishchenko99@gmail.com>
ssb          rsa4096/9F50F4BC796A294B 2023-02-15 [E]

[root@ioithenko ~]# gpg --armor --export B12277B5CB96AEF2 | xclip -sel clip
[root@ioithenko ~]# git config --global user.signingkey B12277B5CB96AEF2
[root@ioithenko ~]# git config --global commit.gpgsign true
[root@ioithenko ~]# git config --global gpg.program $(which gpg2)
[root@ioithenko ~]# gh auth login
? What account do you want to log into? GitHub.com
? You're already logged into github.com. Do you want to re-authenticate? Yes
? What is your preferred protocol for Git operations? HTTPS
? How would you like to authenticate GitHub CLI? Login with a web browser

```

Рис. 2.8: Настройка подписей коммитов

Авторизовываемся (рис. 2.9).

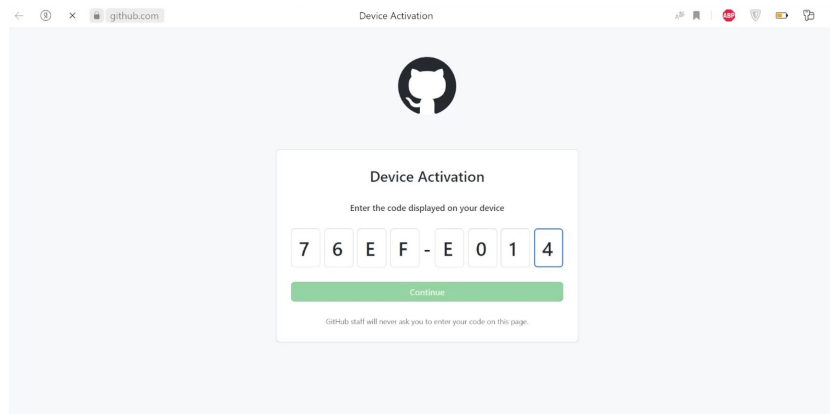


Рис. 2.9: Авторизация

С помощью шаблона создаем новый репозиторий (рис. 2.10).

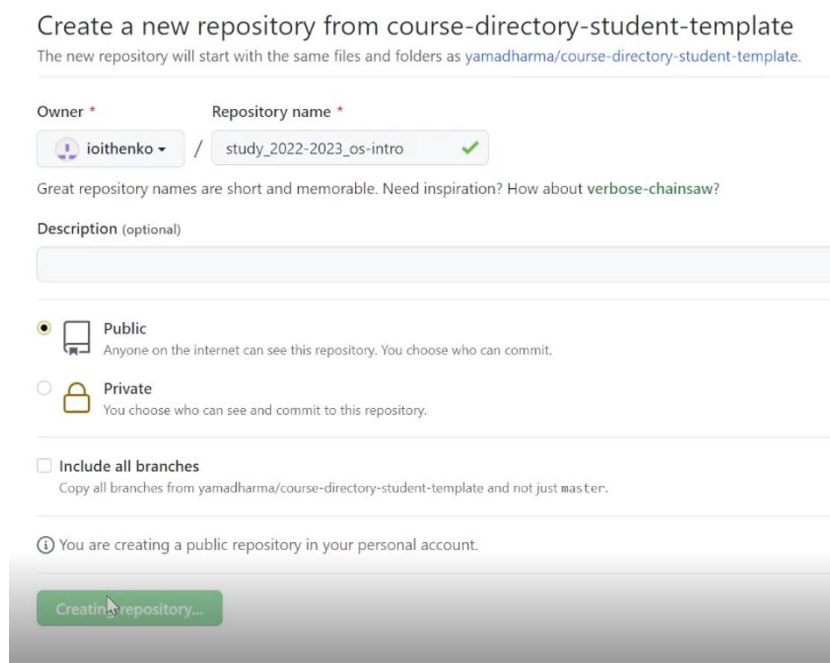


Рис. 2.10: Создание репозитория

Создаем каталог “Операционные системы”, переходим в него, создаем структуру с помощью шаблона, клонируем репозиторий (рис. 2.11).

```
ioithenko@ioithenko:~/work/study/2022-2023/Операционные системы — git clone --recursive git@github.com:ioithenko/study_2022-2023_os-intro
root@ioithenko:~
[ioithenko@ioithenko ~]$ mkdir -p ~/work/study/2022-2023/"Операционные системы"
[ioithenko@ioithenko ~]$ cd ~/work/study/2022-2023/"Операционные системы"
[ioithenko@ioithenko Операционные системы]$ gh repo create study_2022-2023_os-intro --template=yamadharma/course-directory-student-template --public
To get started with GitHub CLI, please run:  gh auth login
Alternatively, populate the GH_TOKEN environment variable with a GitHub API authentication token.
[ioithenko@ioithenko Операционные системы]$ git clone --recursive git@github.com:ioithenko/study_2022-2023_os-intro.git os-intro
bash: owner: Нет такого файла или каталога
[ioithenko@ioithenko Операционные системы]$ git clone --recursive git@github.com:ioithenko/study_2022-2023_os-intro.git os-intro
Клонирование в «os-intro»...
```

Рис. 2.11: Создание репозитория

Удаляем лишний каталог. Создаем необходимые каталоги (рис. 2.12).

```
[ioithenko@ioithenko Операционные системы]$ cd ~/work/study/2022-2023/"Операционные системы"/os-intro
[ioithenko@ioithenko os-intro]$ rm package.json
[ioithenko@ioithenko os-intro]$ echo os-intro > COURSE
[ioithenko@ioithenko os-intro]$ make
[ioithenko@ioithenko os-intro]$
```

Рис. 2.12: Удаление и добавление каталогов

Отправляем файлы на сервер (рис. 2.13) и (рис. 2.14).

```
[ioithenko@ioithenko os-intro]$ git add .
[ioithenko@ioithenko os-intro]$ git commit -am 'make a course structure'
```

Рис. 2.13: Отправление файлов на сервер

```
ioithenko@ioithenko:~/work/study/2022-2023/Операционные системы/os-intro
root@ioithenko:~#
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_fignos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_secnos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/core.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/main.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 project-personal/stage6/report/report.md
[ioithenko@ioithenko os-intro]$ git push
Перечисление объектов: 40, готово.
Подсчет объектов: 100% (40/40), готово.
Сжатие объектов: 100% (30/30), готово.
Запись объектов: 100% (38/38), 342.41 КиБ | 1.07 МБ/с, готово.
Всего 38 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано 0 пакетов
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:ioithenko/study_2022-2023_os-intro.git
d5bc0a2..0214a3f master -> master
[ioithenko@ioithenko os-intro]$
```

Рис. 2.14: Отправление файлов на сервер

Репозиторий настроен (рис. 2.15).

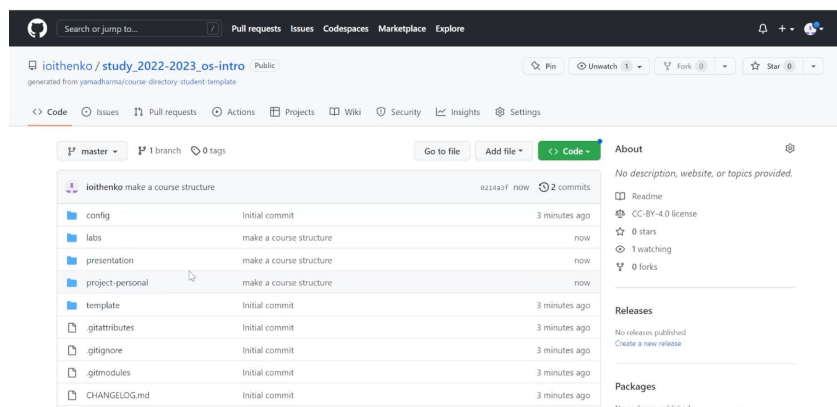


Рис. 2.15: Итоговый вид репозитория

3 Ответы на контрольные вопросы

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

Система контроля версий — это программные инструменты, помогающие командам разработчиков управлять изменениями в исходном коде с течением времени. VCS применяется при работе нескольких разработчиков над одним общим проектом. Система контроля версий хранит все изменения, внесенные в проект, а также разработчики могут вернуться к любой более ранней версии проекта.

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

Хранилище - место расположения файлов и каталогов проекта, изменения которых отслеживаются. Коммит - операция отправки в репозиторий изменений, которые внес разработчик в свою копию проекта. История - информация о всех более ранних версиях проекта. Рабочая копия - текущее состояние файлов проекта, полученных из хранилища и используемых разработчиков в настоящее время.

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

Централизованная система контроля версий предназначена для решения основной проблемы локальной системы контроля версий. Для организации такой

системы контроля версий используется единственный сервер, который содержит все версии файлов. Пример централизованной VCS - CVS, Subversion, Perforce. Децентрализованные системы контроля версий - это такие системы, как Git, Mercurial, Bazaar или, например, Darcs. Каждый пользователь имеет свою версию репозитория, может вносить в нее изменения.

Различие между централизованными и децентрализованными VCS заключается в количестве репозиториях.

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

Разработчик работает с главной веткой master, при необходимости может создавать дополнительные ветки для отдельных задач проекта. При внесении изменений он сохраняет файлы в общем хранилище.

5. Опишите порядок работы с общим хранилищем VCS.

Каждый разработчик работает с отдельной веткой проекта и вносит изменения в локальный репозиторий. При завершении работы ему требуется запустить файлы на сервер, внести изменения в главную ветку.

6. Каковы основные задачи, решаемые инструментальным средством git?

Основные задачи: возможность коллективной работы над проектом и сохранение всей информации об изменениях, внесенных в проект.

7. Назовите и дайте краткую характеристику командам 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` удаление ветки с центрального репозитория: `git push origin :имя_ветки`

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

Локальный репозиторий используется одним человеком и расположен на его компьютере. Удаленный репозиторий может использоваться группой разработчиков, в центральный репозиторий они пушат из локальных репозиториях изменения, внесенные в проект.

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

Ветка - это указатель на один из коммитов. Ветки используются для работы над одной из частей проекта в отдельности от других его частей. Каждая ветка представляет собой отдельную копию проекта, что позволяет работать над разными версиями одного проекта.

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

Игнорировать файлы при commit можно с помощью файла `.gitignore`. Там помещаются файлы, не требующиеся в итоговой версии проекта.

4 Выводы

В ходе выполнения лабораторной работы я изучила идеологию и применение средств контроля версий и освоила умения по работе с git.

Список литературы