

# **Лабораторная работа №2**

**‘Операционные системы’**

Дроздова Дарья Игоревна

# Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Выводы	10
4	Ответы на вопросы	11
5	Список литературы	14

## Список иллюстраций

2.1	Установим git . . . . .	5
2.2	Установим gh . . . . .	5
2.3	Информация о владельце репозитория . . . . .	6
2.4	Настройка коммитов . . . . .	6
2.5	Настройка ssh rsa ключа . . . . .	6
2.6	Настройка ssh ed25519 ключа . . . . .	6
2.7	Настройка pgr ключа . . . . .	7
2.8	Копирование pgr ключа . . . . .	7
2.9	Копирование pgr ключа . . . . .	8
2.10	Настройка подписи pgr ключа . . . . .	8
2.11	Авторизация gh . . . . .	8
2.12	Настройка каталога курса . . . . .	9

# 1 Цель работы

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

## 2 Выполнение лабораторной работы

### 1. Установка git

- Установим git(в моем случае гит был установлен заранее)

```
[didrozdova@didrozdova ~]$ sudo dnf install git
[sudo] пароль для didrozdova:
Последняя проверка окончания срока действия метаданных: 0:59:11 назад, Вт 14 фев 2023 15:05:47.
Зависимости разрешены.
=====
Пакет                Архитектура      Версия            Репозиторий       Размер
=====
Установка:
git                  x86_64           2.39.1-1.fc37     updates           66 k
Установка зависимостей:
git-core-doc         noarch           2.39.1-1.fc37     updates           2.8 M
perl-Error           noarch           1:0.17029-10.fc37 fedora            41 k
perl-Git             noarch           2.39.1-1.fc37     updates           43 k
perl-TermReadKey     x86_64           2.38-14.fc37     fedora            36 k
Результат транзакции
=====
Установлено: 5 Пакетов
```

Рис. 2.1: Установим git

### 2. Установка gh

```
[didrozdova@didrozdova ~]$ sudo dnf install gh
Последняя проверка окончания срока действия метаданных: 0:59:47 назад, Вт 14 фев 2023 15:05:47.
Зависимости разрешены.
=====
Пакет                Архитектура      Версия            Репозиторий       Размер
=====
Установка:
gh                  x86_64           2.22.1-1.fc37     updates           8.3 M
Результат транзакции
=====
```

Рис. 2.2: Установим gh

### 3. Базовая настройка git

- Зададим имя и email владельца репозитория и настроим utf-8 в выводе сообщений git:

```
[didrozdova@didrozdova ~]$ git config --global user.name "Daria Drozdova"
[didrozdova@didrozdova ~]$ git config --global user.email "ddrozdova2004@gmail.com"
[didrozdova@didrozdova ~]$ git config --global quotepath false
```

Рис. 2.3: Информация о владельце репозитория

- Настроим верификацию и подписание коммитов git (в моем случае подписание коммитов настроено, проверим командой):

```
[didrozdova@didrozdova ~]$ git config --list --show-origin
(bash: git: команда не найдена)
[didrozdova@didrozdova ~]$ git config --list --show-origin
file:/home/didrozdova/.gitconfig      user.name=Daria Drozdova
file:/home/didrozdova/.gitconfig      user.email=ddrozdova2004@gmail.com
file:/home/didrozdova/.gitconfig      user.signingkey=Daria Drozdova <ddrozdova2004@gmail.com>
file:/home/didrozdova/.gitconfig      core.quotepath=false
file:/home/didrozdova/.gitconfig      core.autocrlf=input
file:/home/didrozdova/.gitconfig      core.safecrlf=warn
file:/home/didrozdova/.gitconfig      commit.gpgsign=true
file:/home/didrozdova/.gitconfig      gpg.program=/usr/bin/gpg2
```

Рис. 2.4: Настройка коммитов

#### 4. Создайте ключи ssh

- по алгоритму rsa с ключём размером 4096 бит:

```
[didrozdova@didrozdova .ssh]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/didrozdova/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again: |
```

Рис. 2.5: Настройка ssh rsa ключа

- по алгоритму ed25519:

```
[didrozdova@didrozdova .ssh]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/didrozdova/.ssh/id_ed25519):
```

Рис. 2.6: Настройка ssh ed25519 ключа

## 5. Создайте ключи pgp

- Генерируем ключ

```
[didrozdova@didrozdova .ssh]$ 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: создан каталог '/home/didrozdova/.gnupg'
gpg: создан щит с ключами '/home/didrozdova/.gnupg/pubring.kbx'
Выберите тип ключа:
  (1) RSA and RSA
  (2) DSA and Elgamal
```

Рис. 2.7: Настройка pgp ключа

- Из предложенных опций выбираем: тип RSA and RSA; размер 4096; выберите срок действия; значение по умолчанию — 0 (срок действия не истекает никогда). GPG запросит личную информацию, которая сохранится в ключе: Имя (не менее 5 символов). Адрес электронной почты. При вводе email убедитесь, что он соответствует адресу, используемому на GitHub. Комментарий. Можно ввести что угодно или нажать клавишу ввода, чтобы оставить это поле пустым

## 6. Настройка github

(в моем случае профиль на гитхабе уже имеется)

## 7. Добавление PGP ключа в GitHub

- Выводим список ключей и копируем отпечаток приватного ключа:

```
[didrozdova@didrozdova private-keys-v1.d]$ gpg --list-secret-keys --keyid-format LONG
gpg: /home/didrozdova/.gnupg/trustdb.gpg: создана таблица доверия
```

Рис. 2.8: Копирование pgp ключа

Отпечаток ключа — это последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа.

Формат строки:

sec Алгоритм/Отпечатокключа Датасоздания [Флаги] [Годен\_до] ID\_ключа

- Скопируем сгенерированный PGP ключ в буфер обмена:

```
[didrozdova@didrozdova ~]$ gpg --armor --export 5EEE34555C360B67 | xclip -sel clip
[didrozdova@didrozdova ~]$
```

Рис. 2.9: Копирование pgr ключа

- Перейдем в настройки GitHub (<https://github.com/settings/keys>), нажмем на кнопку New GPG key и вставим полученный ключ в поле ввода.

## 8. Настройка автоматических подписей коммитов git

- Используя введенный email, указываем Git применять его при подписи КОММИТОВ:

```
[didrozdova@didrozdova ~]$ git config --global commit.gpgsign true
[didrozdova@didrozdova ~]$ git config --global gpg.program $(which gpg2)
```

Рис. 2.10: Настройка подписи pgr ключа

## 9. Настройка gh

- Для начала необходимо авторизоваться

```
[didrozdova@didrozdova ~]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? SSH
? Upload your SSH public key to your GitHub account? /home/didrozdova/.ssh/id_rsa.pub
? Title for your SSH key: rsa@didrozdova
? How would you like to authenticate GitHub CLI? [Use arrows to move, type to filter]
> Login with a web browser
Paste an authentication token
```

Рис. 2.11: Авторизация gh

Утилита задаст несколько наводящих вопросов. Авторизоваться можно через браузер.



## 10. Создание репозитория курса на основе шаблона

- Необходимо создать шаблон рабочего пространства (в моем случае рабочее пространство уже было создано).

## 12. Настройка каталога курса

- Перейдем в каталог курса, удалим лишние файлы, создадим необходимые каталоги и отправим файлы на сервер:

```
[didrozdova@didrozdova os-intro]$ rm package.json
[didrozdova@didrozdova os-intro]$ echo os-intro > COURSE
[didrozdova@didrozdova os-intro]$ make
[didrozdova@didrozdova os-intro]$ ls
CHANGELOG.md  COURSE  LICENSE  prepare  project-personal  README.git-flow.md  template
config        labs   Makefile  presentation  README.en.md      README.md
```

Рис. 2.12: Настройка каталога курса

## 3 Выводы

В ходе выполнения данной лабораторной работы я приобрела теоретические навыки работы контролем версий, а также освоила базовые умения взаимодействия с распределённой системой управления версиями git.

## 4 Ответы на вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?
  - 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. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.
  - Хранилище версий - репозиторий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией.
  - Commit - команда совершающая выгрузку проиндексированных файлов в репозиторий
  - История - история ваших последовательных коммитов
  - Рабочая копия - копия основного рабочего репозитория на ваше локальное хранилище
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.
  - Централизованные VCS: одно основное хранилище всего проекта, каждый пользователь копирует себе необходимые ему файлы из этого репозитория,

изменяет и, затем, добавляет свои изменения обратно, примеры: Subversion, CVS

- Децентрализованные VCS: У каждого пользователя свой вариант (возможно не один) репозитория, Присутствует возможность добавлять и забирать изменения из любого репозитория, примеры: Git, Mercurial

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

- Создание репозитория на распределенном VCS и последовательное ведение проекта(пер-проекта, коммерческого) во-первых, для создания активного профиля в качестве портфолио, во-вторых, возможность откатиться в случае появления критической ошибки в проекте или тупиковой ветви развития, в-третьих, возможность привлечения иных разработчиков, за счет понимания истории вашего проекта и возможности скопировать репозиторий.

5. Опишите порядок работы с общим хранилищем VCS.

- инициализировать, создать первый коммит, по мере работы(важные части проекта) выгружать обновленные файлы проекта

6. Каковы основные задачи, решаемые инструментальным средством git?

- Поддерживается автономная работа; локальные фиксации изменений могут быть отправлены позже. Каждое рабочее дерево в Git содержит хранилище с полной историей проекта. Ни одно хранилище Git не является по своей природе более важным, чем любое другое. Скорость работы, ветвление делается быстро и легко.

7. Назовите и дайте краткую характеристику командам git.

- Задание имени и email владельца репозитория: `git config --global user.name "Name Surname"`; `git config --global user.email "work@mail"`
- Настройка utf-8 в выводе сообщений git: `git config --global core.quotePath false`

- Задавание имени начальной ветки (будем называть её master): `git config --global init.defaultBranch master`
  - Параметр `autocrlf`: `git config --global core.autocrlf input`
  - Параметр `safecrlf`: `git config --global core.safecrlf warn`
  - Настройка автоматических подписей коммитов git: `git config --global user.signingkey PGP Fingerprint`; `git config --global commit.gpgsign true`; `git config --global gpg.program $(which gpg2)`
  - Отправка файлов на сервер: `git add .`; `git commit -am 'feat(main): make course structure'`; `git push`
8. Приведите примеры использования при работе с локальным и удалённым репозиториями.
- см. предыдущие впр под №7, №5, №4
9. Что такое и зачем могут быть нужны ветви (branches)?
- Ветви нужны для командного взаимодействия в основном с удаленным репозиториями при распределении задач между разработчиками занимающимися разными задачами, но решающими их в рамках данного репозитория. Разработчики создают ветку в репозитория на удаленном хранилище(Github например) беря копию репозитория, обычно при этом ветка называется по тому с какой задачей работает разработчик. В дальнейшем выгрузка обновлений происходит на эту ветку с локального репозитория, а уже с нее обновления отправляют в main.
10. Как и зачем можно игнорировать некоторые файлы при commit?
- Ваши выбранные файлы, которые не нужно отслеживать, помещаются в специальную папку “git-ignore” и при обновлении этих файлов они не будут отправляться в репозиторий.

## 5 Список литературы

1. Немет Э. et al. Unix и Linux: руководство системного администратора. 4-е изд. Вильямс, 2014. 1312 р.