Лабораторная работа №2

‘Операционные системы’

Дроздова Дарья Игоревна

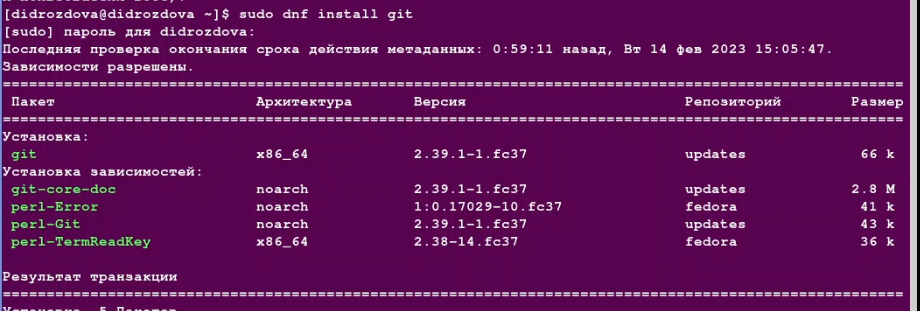
Содержание

# 1 Цель работы

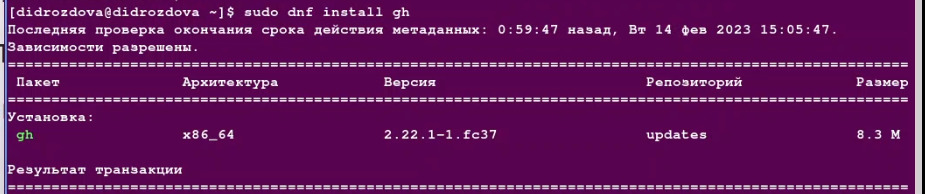
Изучить идеологию и применение средств контроля версий, освоить умения по работе с git.

# 2 Выполнение лабораторной работы

1. **Установка git**
   * Установим git(в моем случае гит был установлен заранее)

* 
* Figure 1: Установим git

1. **Установка gh**

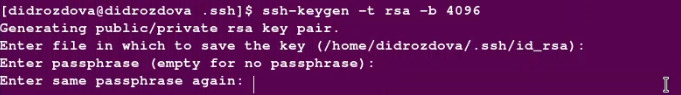
* 
* Figure 2: Установим gh

1. **Базовая настройка git**
   * Зададим имя и email владельца репозитория и настроим utf-8 в выводе сообщений git:

* + Figure 3: Информация о владельце репозитория
  + Figure 3: Информация о владельце репозитория
  + Настроим верификацию и подписание коммитов git (в моем случае подписание коммитов настроено, проверим командой):

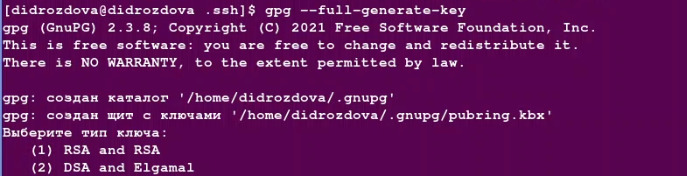
* 
* Figure 4: Настройка коммитов

1. **Создайте ключи ssh**
   * по алгоритму rsa с ключём размером 4096 бит:

* 
* Figure 5: Настройка ssh rsa ключа
  + по алгоритму ed25519:

* Figure 6: Настройка ssh ed25519 ключа
* Figure 6: Настройка ssh ed25519 ключа

1. **Создайте ключи pgp**
   * Генерируем ключ

* 
* Figure 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

1. **Настройка github**

* (в моем случае профиль на гитхабе уже имеется)

1. **Добавление PGP ключа в GitHub**
   * Выводим список ключей и копируем отпечаток приватного ключа:

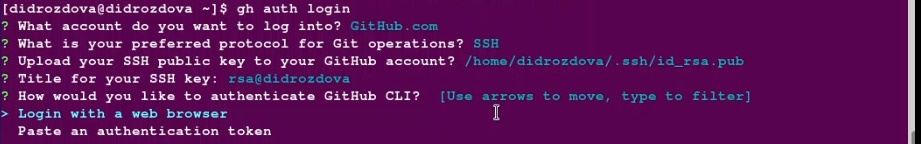
* Figure 8: Копирование pgp ключа
* Figure 8: Копирование pgp ключа
* Отпечаток ключа — это последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа.
* Формат строки:
* sec Алгоритм/Отпечаток*ключа Дата*создания [Флаги] [Годен\_до] ID\_ключа
  + Cкопируем сгенерированный PGP ключ в буфер обмена:

* Figure 9: Копирование pgp ключа
* Figure 9: Копирование pgp ключа
  + Перейдем в настройки GitHub (https://github.com/settings/keys), нажмем на кнопку New GPG key и вставим полученный ключ в поле ввода.

1. **Настройка автоматических подписей коммитов git**
   * Используя введёный email, указываем Git применять его при подписи коммитов:

* Figure 10: Настройка подписи pgp ключа
* Figure 10: Настройка подписи pgp ключа

1. **Настройка gh**
   * Для начала необходимо авторизоваться

* 
* Figure 11: Авторизация gh
* Утилита задаст несколько наводящих вопросов. Авторизоваться можно через браузер.

1. **Создание репозитория курса на основе шаблона**

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

1. **Настройка каталога курса**

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

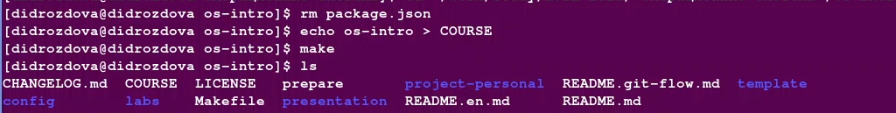


Figure 12: Настройка каталога курса

# 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.

1. 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

1. 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

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

* Создание репозитория на распределенном VCS и последовательное ведение проекта(pet-проета, 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 понимании истории вашего проекта и возможности скопировать репозиторий.

1. Опишите порядок работы с общим хранилищем VCS.

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

1. 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.

1. Назовите и дайте краткую характеристику командам git.

* Зададие 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
* Зададание 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

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

* см. предыдущие впр под №7, №5, №4

1. 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.

1. 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 p.