Отчёт по лабораторной работе №2

Операционные системы

Мурашов Иван Вячеславович

Содержание

# 1 Цель работы

Целью данной лабораторной работы является изучение идеологии, применение средств контроля версий и освоение умений по работе с git.

# 2 Задание

1. Установить программное обеспечение
2. Создать базовую конфигурацию для работы с git
3. Создать ключ SSH
4. Создать ключ PGP
5. Создать профиль на GitHub и авторизоваться на устройстве
6. Создать репозиторий курса на основе шаблона
7. Настроить каталог курса

# 3 Теоретическое введение

Целью данной лабораторной работы является изучение идеологии, применение средств контроля версий и освоение умений по работе с git. 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.

# 4 Выполнение лабораторной работы

## 4.1 Установка программного обеспечения

Для начала я устанавливаю git (рис. 1).

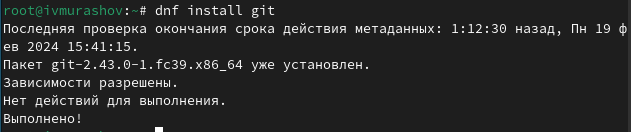


Рис. 1: Установка git

Затем устанавливаю gh (рис. 2).

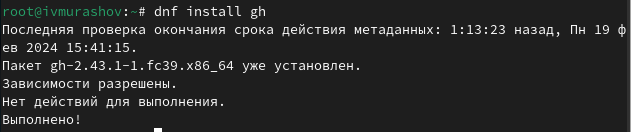


Рис. 2: Установка gh

## 4.2 Базовая настройка git

Задаю имя и email владельца репозитория (рис. 3).

Задание имени и email

Рис. 3: Задание имени и email

Настравиваю utf-8 в выводе сообщений git и задаю имя начальной ветки (рис. 4).

Настройка utf-8

Рис. 4: Настройка utf-8

Настраиваю параметры autocrlf и safecrlf (рис. 5).

Настройка autocrlf и safecrlf

Рис. 5: Настройка autocrlf и safecrlf

## 4.3 Создание ключей

Создаю ключ SSH по алгоритму rsa с ключом размером 4096 бит (рис. 6).

Создание ssh ключа

Рис. 6: Создание ssh ключа

А затем по алгоритму ed25519 (рис. 7).

Создание ssh ключа

Рис. 7: Создание ssh ключа

Генерирую ключ PGP, выбирая из предложенных опций тип ‘RSA and RSA’, размер 4096, без срока действия. Указываю имя и адрес email (рис. 8).

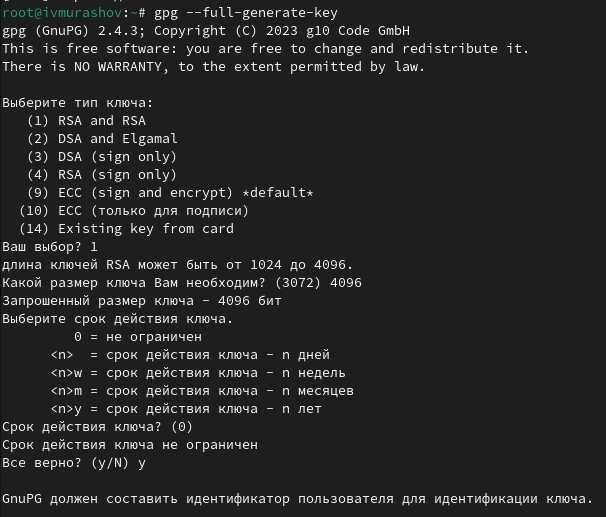


Рис. 8: Создание pgp ключа

## 4.4 Настройка github и добавление ключей

У меня уже был создан профиль на github (рис. 9).

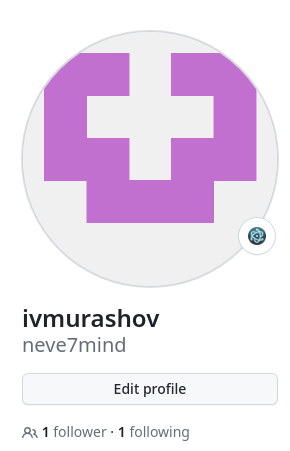


Рис. 9: Профиль github

Вывожу список ключей (рис. 10).

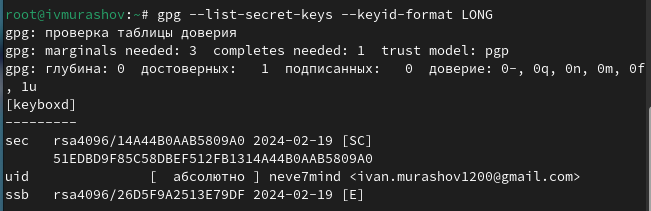


Рис. 10: Список ключей

Копирую сгенерированный pgp ключ в буфер обмена (рис. 11).

Копирование pgp ключа

Рис. 11: Копирование pgp ключа

Перехожу в настройки GitHub, нажимаю на кнопку ‘New GPG key’ и вставляю скопированный ключ в поле ввода (рис. 12).

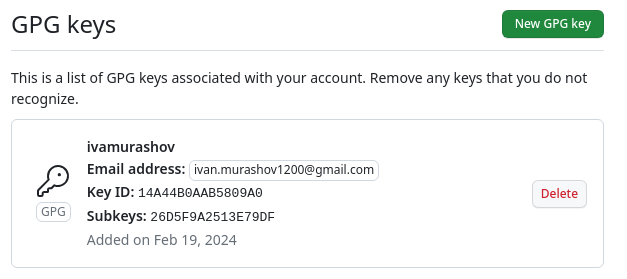


Рис. 12: Список ключей

## 4.5 Настройка автоматических подписей коммитов git и авторизация

Используя введёный email, указываю Git применять его при подписи коммитов (рис. 13).

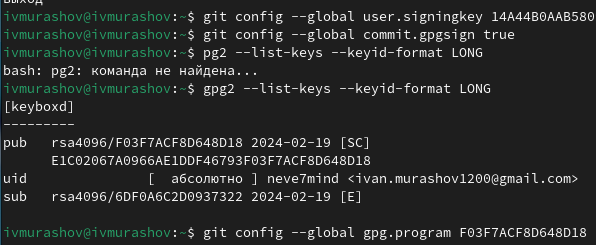


Рис. 13: Настройка автоматических подписей

Авторизуюсь с помощью браузера (gh login) (рис. 14).

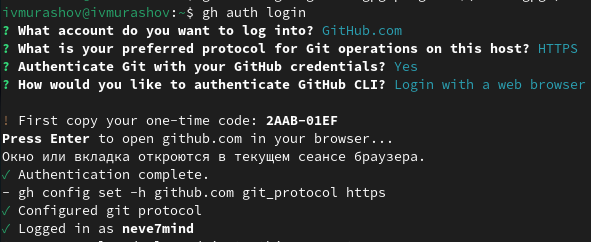


Рис. 14: Авторизация

## 4.6 Создание репозитория курса на основе шаблона и настройка каталога

Создаю каталог курса, перемещаюсь в него и создаю репозиторий на основе шаблона (рис. 15).

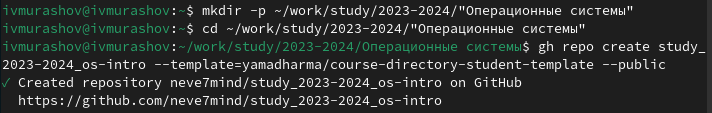


Рис. 15: Создание репозитория

Клонирую репозиторий (рис. 16).

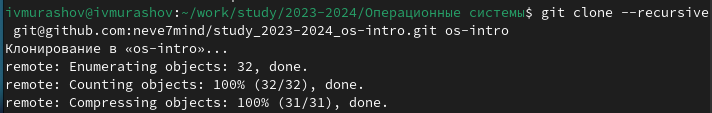


Рис. 16: Клонирование репозитория

Удаляю лишние файлы, создаю необходимые каталоги (рис. 17).

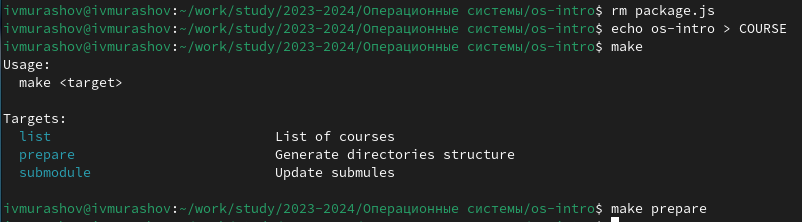


Рис. 17: Настройка каталога курса

Отправляю файлы на сервер (рис. 18).

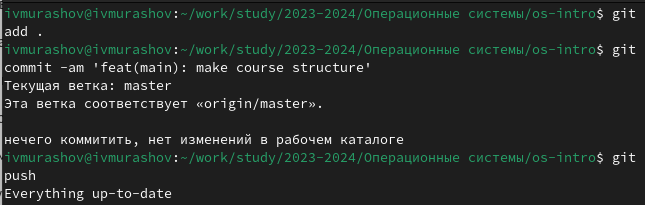


Рис. 18: Отправка файлов на сервер

# 5 Ответы на 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.

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

Хранилище - это репозиторий, в котором хрянятся все документы, включая историю изменений.

Commit - отслеживание и сохранение изменений.

История - сохраняет в себе изменения проекта на всех его этапах.

Рабочая копия - копия проекта на основе версии из хранилища.

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

Централизованные VCS (CVS, SVN, TFS) - одно главное хранилище всего проекта, каждый участник проекта добавляет изменения в данное хранилище.

Децентрализованные VCS (Git, Bazaar) - у каждого участника свой вариант репозитория, что позволяет соединять впоследствии нужные ветки проекта.

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

Создаётся и подключается удалённый репозиторий. Участник проекта при взаимодействии с данным репозиторием отправляет все изменения на сервер.

1. Опишите порядок работы с общим хранилищем VCS.

Участник получает определённую версию проекта в хранилище. Участник отправляет все изменения на сервер. При этом старые версии проекта сохраняются.

1. Каковы основные задачи, решаемые инструментальным средством git?

Хранение информации об изменениях проекта, возможность совместной работы над одним проектом.

1. 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

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  
  
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 удаление локальной ветки: git branch -D имя\_ветки  
  
удаление ветки с центрального репозитория: git push origin :имя\_ветки

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

git commit -am ‘feat(main): make course structure’ - сохранение всех добавленных изменений и всех изменённых файлов

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

Ветви - это варианты развития проекта в хранилище. Они нужны для удобства коллективной работы с хранилищем, а так же для создания “пробных” версий проекта

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

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

# 6 Выводы

В ходе выполнения данной лабораторной работы я изучил идеологии и применение средств контроля версий и освоил умения по работе с git.