Лабораторная работа №2

Операционные системы

Кирилюк Светлана

Содержание

# 1 Цель работы

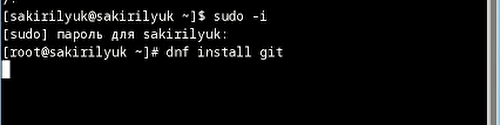
Изучить идеологию и применение средств контроля версий. Освоить умения по работе с git.

# 2 Задание

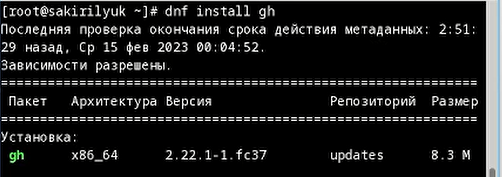
1)Создать базовую конфигурацию для работы с git. 2)Создать ключ SSH. 3)Создать ключ PGP. 4)Настроить подписи git. 5)Зарегистрироваться на Github. 6)Создать локальный каталог для выполнения заданий по предмету.

# 3 Выполнение лабораторной работы

В первую очередь я установила git (рис. ??) и gh (рис. ??).

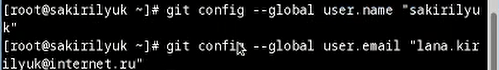


Установка git



Установка gh

Затем я задала имя и email владельца репозитория (рис. ??) и настроила utf-8 в выводе сообщений git (рис. ??).



Имя и email

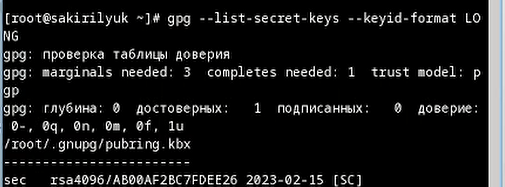


Настройка utf-8

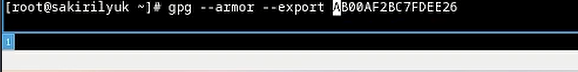
Я сгенерировала pgp ключ (рис. ??), вывела список ключей и скопировала отпечаток приватного ключа (рис. ??), чтобы скопировать сгенерированный PGP ключ в буфер обмена (рис. ??). Затем я вставила полученный ключ в строку на GitHub (рис. ??).



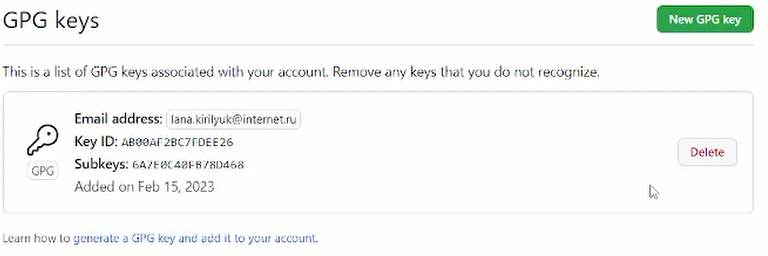
Генерация ключа pgp



Копия отпечатка ключа

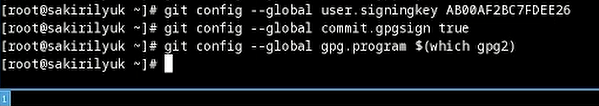


Копирование ключа в буфере обмена

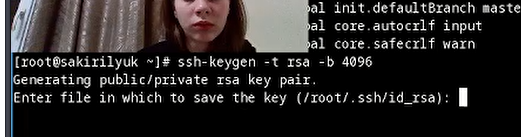


Загрузка ключа pgp на Github

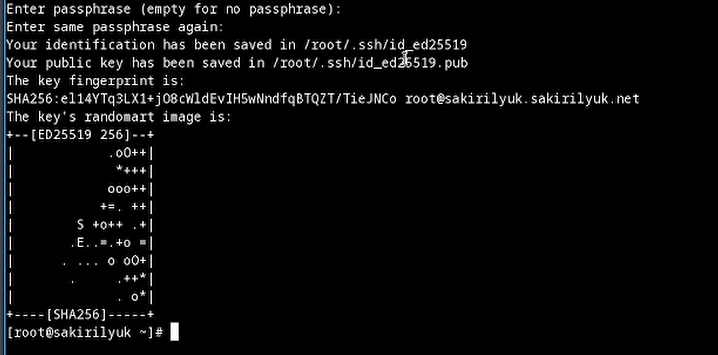
Также я настроила автоматические подписи коммитов (рис. ??), задала параметры autocrlf и safecrlf и сгенерировала ключи по двум алгоритмам (рис. ??), (рис. ??).



Автоматические подписи коммитов

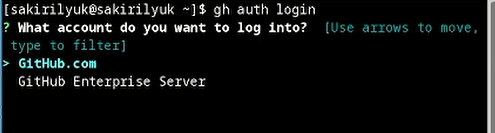


Использование алгоритма autocrlf

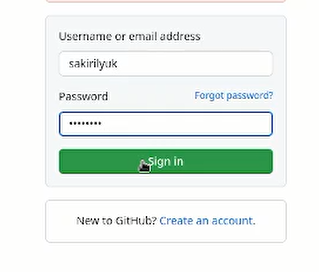


Использование алгоритма safecrlf

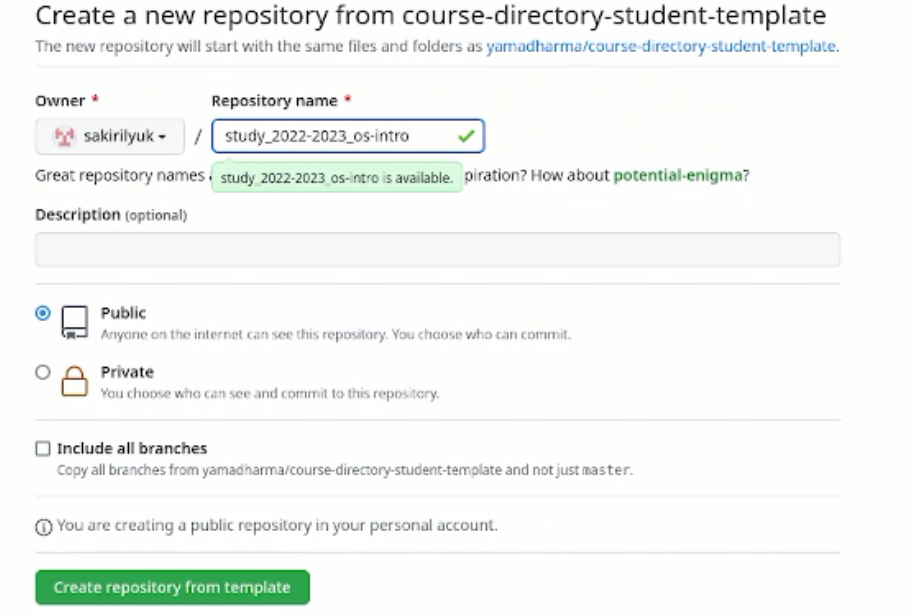
Далее я авторизовалась (рис. ??), (рис. ??). Создала шаблон репозитория (рис. ??), (рис. ??).



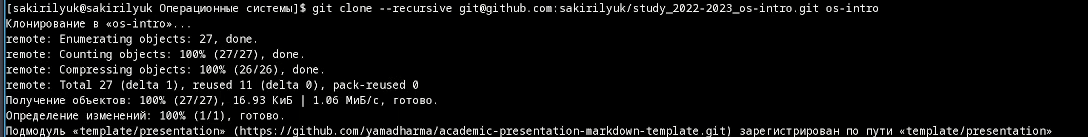
Авторизация (1)



Авторизация (2)

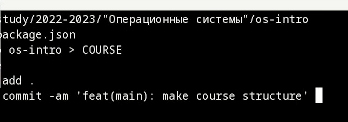


Создание шаблона репозитория (1)

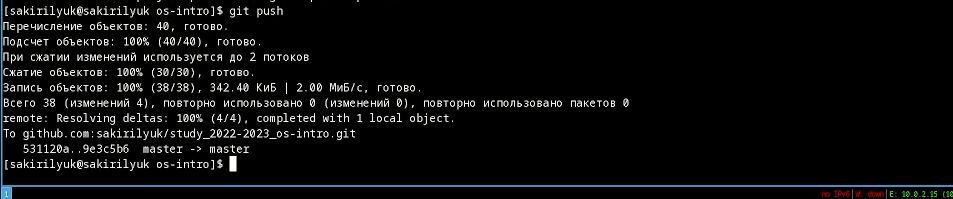


Создание шаблона репозитория (2)

Перейдя в каталог курса я удалила лишние файлы и создала необходимые каталоги (рис. ??), затем отправила файлы на сервер (рис. ??).



Удаление файлов и создание каталогов



Отправка файлов на сервер

Ответы на контрольные вопросы: 1)Система контроля версий (VCS) — это место хранения кода. Она нужна для разработки продуктов (для хранение кода, синхронизации работы нескольких человек, создания релизов).

1. 2.1 Хранилище (репозиторий) - центральное место, хранящее не только файлы, но и историю. Доступ к репозиторию осуществляется через сеть, выступая в роли сервера и инструмента контроля версий, выступающего в роли клиента. 2.2 Commit - это команда Git для записи индексированных изменений в репозиторий. 2.3 Рабочая копия – это снимок хранилища, личное рабочее место, где разработчики могут выполнять свою работу, оставаясь изолированными от остальной части команды.
2. 3.1 Централизованные VCS Одно основное хранилище всего проекта; Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно. 3.2 Децентрализованные VCS У каждого пользователя свой вариант (возможно не один) репозитория; Присутствует возможность добавлять и забирать изменения из любого репозитория.

6)Git — это система управления версиями. У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.

1. 7.1 Создание основного дерева репозитория: git init 7.2 Получение обновлений (изменений) текущего дерева из центрального репозитория: git pull 7.3 Отправка всех произведённых изменений локального дерева в центральный репозиторий: git push 7.4 Просмотр списка изменённых файлов в текущей директории: git status 7.5 Просмотр текущих изменений: git diff 7.6 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). 7.7 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) 7.8 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).
2. Ветка (англ. branch) — это последовательность коммитов, в которой ведётся параллельная разработка какого-либо функционала Основная ветка– master Ветки в GIT. Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций.

10)Игнорируемые файлы — это, как правило, артефакты сборки и файлы, генерируемые машиной из исходных файлов в репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты

# 4 Выводы

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