Лабораторная работа №2

Дисциплина: Операционные системы

Савостин Олег

Содержание

Список иллюстраций

Список таблиц

# 1 Цель работы

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

# 2 Задание

1. Установка программного обеспечения
2. Базовая настройка git
3. Создайте ключи ssh
4. Создайте ключи pgp
5. Настройка github
6. Добавление PGP ключа в GitHub
7. Настройка автоматических подписей коммитов git
8. Настройка gh
9. Шаблон для рабочего пространства # Теоретическое введение

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.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

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.

# 3 Выполнение лабораторной работы

## 3.1 Установка программного обеспечения

В первую очередь, устанавливаем git и gh(рис. 1).

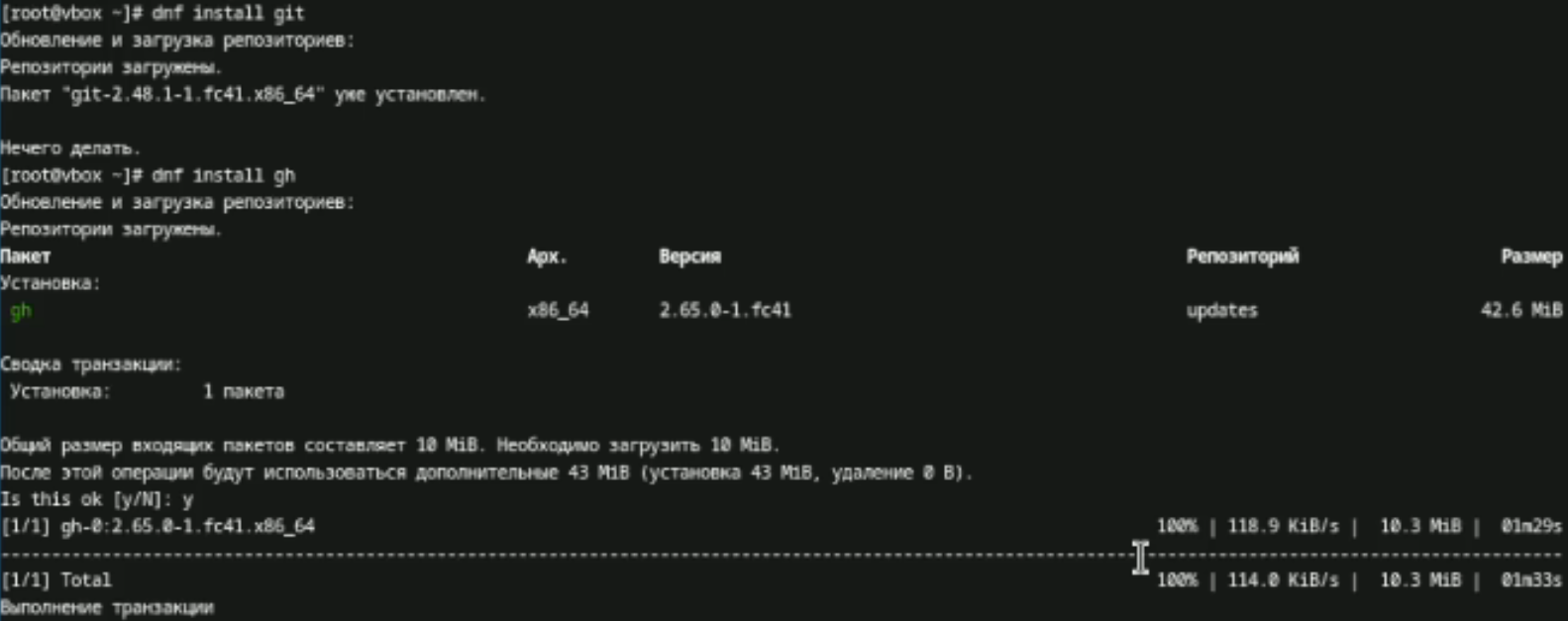


Рис. 1: Установка программного обеспечения

## 3.2 Базовая настройка git

Задаю имя и почту владельца репозитория. Так как у меня уже есть аккаунт на github’e, то я задаю данные, которые я вводил когда создавал аккаунт(рис. 2).

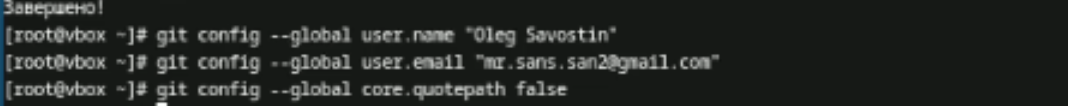


Рис. 2: Задаю имя и почту владельца

Настраиваю utf-8 в выводе сообщений гит, настраиваю верификацию и подписание коммитов git(рис. 3) (рис. 4).

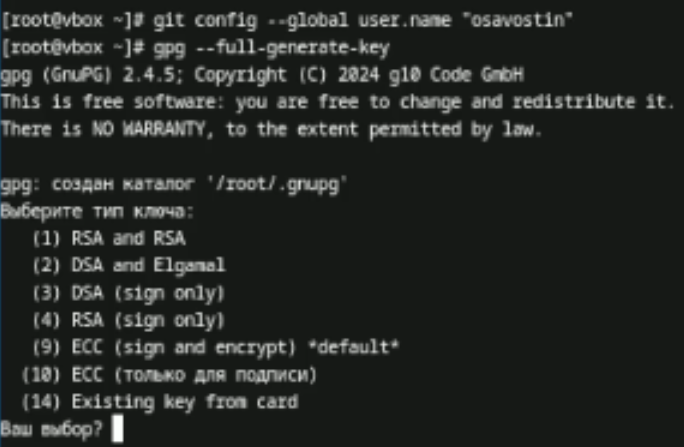


Рис. 3: utf-8

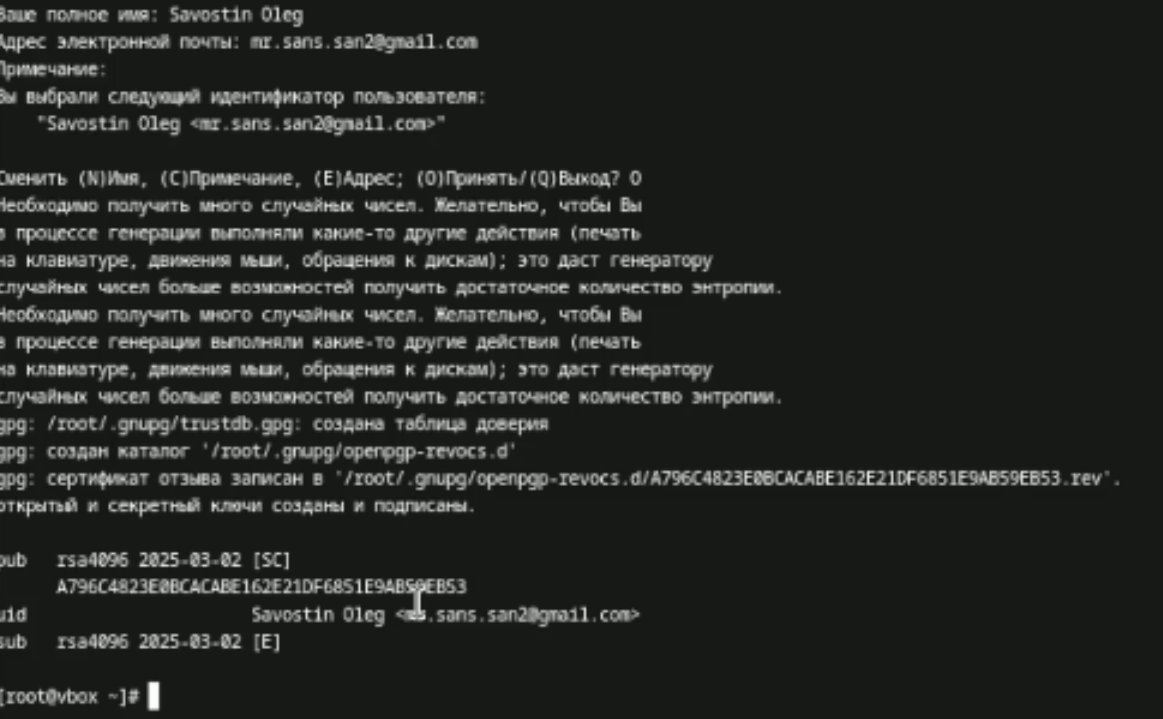


Рис. 4: Верификация и подписание коммитов

Записываю параметры autocrlf safecrl(рис. 5).

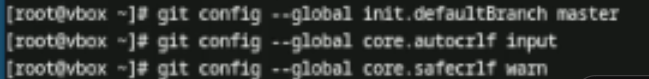


Рис. 5: Записывание параметров

## 3.3 Создайте ключи ssh

По алгоритму rsa с ключём размером 4096 создаю ключ. Также создаю ключ по алгоритму ed25519(рис. 6).

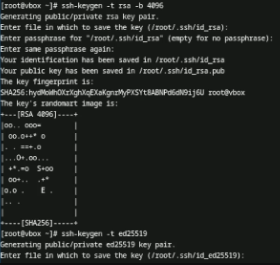


Рис. 6: Создание ключей ssh

## 3.4 Создайте ключи pgp

Следуя инструкциям, указанных в лабораторной работе, создаю ключ gpg gpg –full-generate-key (рис. 7).



Рис. 7: gpg –full-generate-key

## 3.5 Настройка github

Аккаунт заранее существовал с первого семестра. (рис. 8).

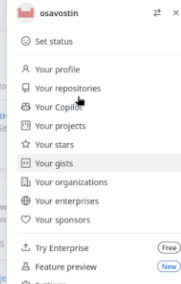


Рис. 8: Аккаунт на гитхабе

## 3.6 Добавление PGP ключа в GitHub

Вывожу список ключей и копирую отпечаток приватного ключа командой gpg –list-secret-keys –keyid-format LONG (рис. 9).

Рис. 9: Копирую ключ

Рис. 9: Копирую ключ

Копирую отпечаток ключа и вставляю его в настройки GitHub (рис. 10).

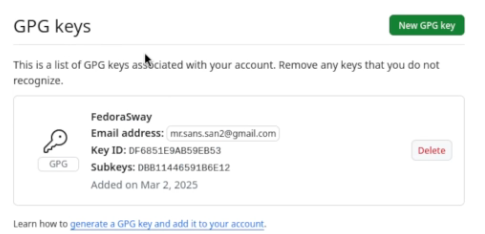


Рис. 10: Новый ключ gpg на аккаунте

## 3.7 Настройка автоматических подписей коммитов git

Используя введённый email, указываю Гит применять его при подписи коммитов (рис. 11).

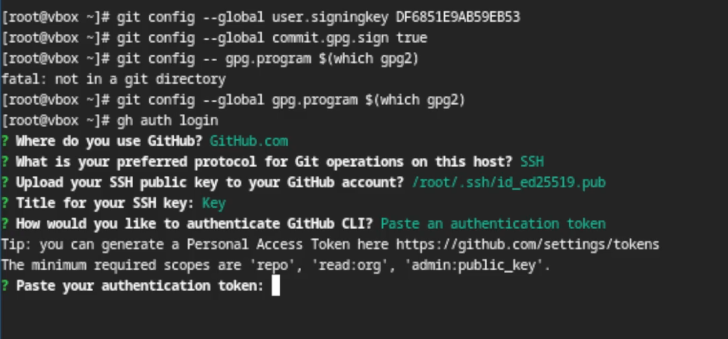


Рис. 11: Email commit

## 3.8 Настройка gh

Для пользованием gh я захожу в свой аккаунт через терминал (рис. 12).

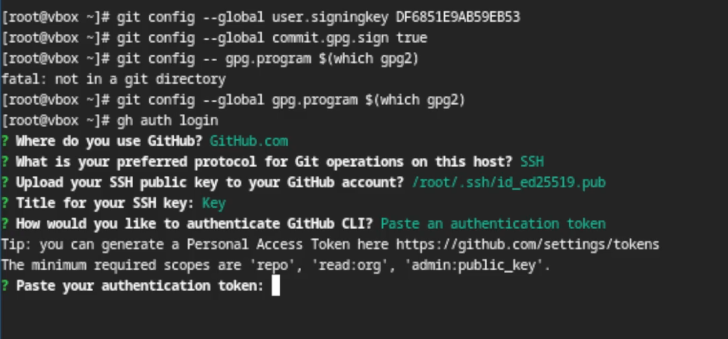


Рис. 12: gh auth login

## 3.9 Шаблон для рабочего пространства

Теперь необходимо создать шаблон рабочего пространства. Пишу в терминал следующие команды: mkdir -p ~/work/study/2022-2023/“Операционные системы” cd ~/work/study/2022-2023/“Операционные системы” gh repo create study\_2022-2023\_os-intro –template=yamadharma/course-directory-student-template –public git clone –recursive git@github.com:/study\_2022-2023\_os-intro.git os-intro (рис. 13).

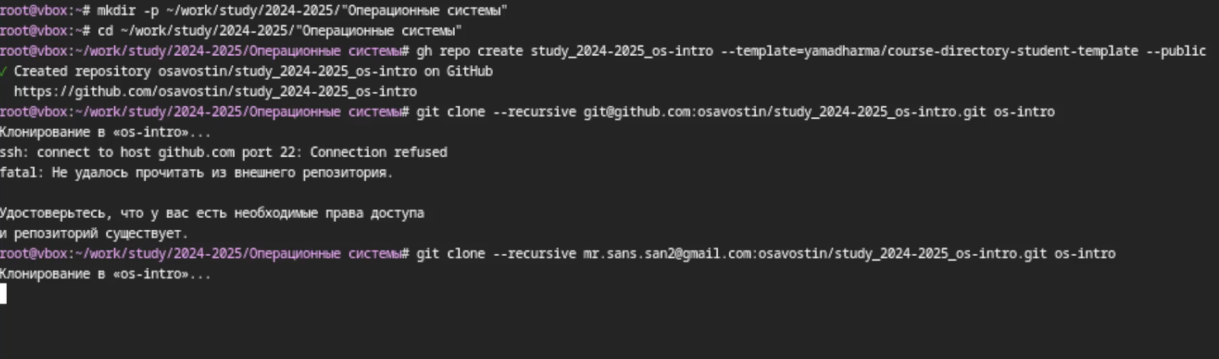


Рис. 13: Создание рабочего пространства

Перехожу в каталог курса и удаляю лишние файлы .json и затем создаю нужные каталоги (рис. 14) (рис. 15).

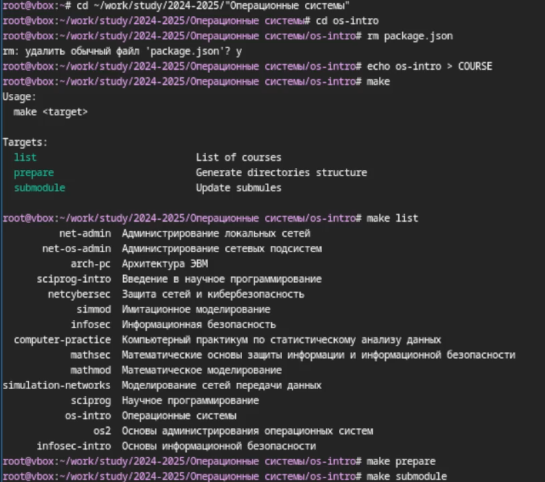


Рис. 14: Удаление лишних файлов и создание каталогов

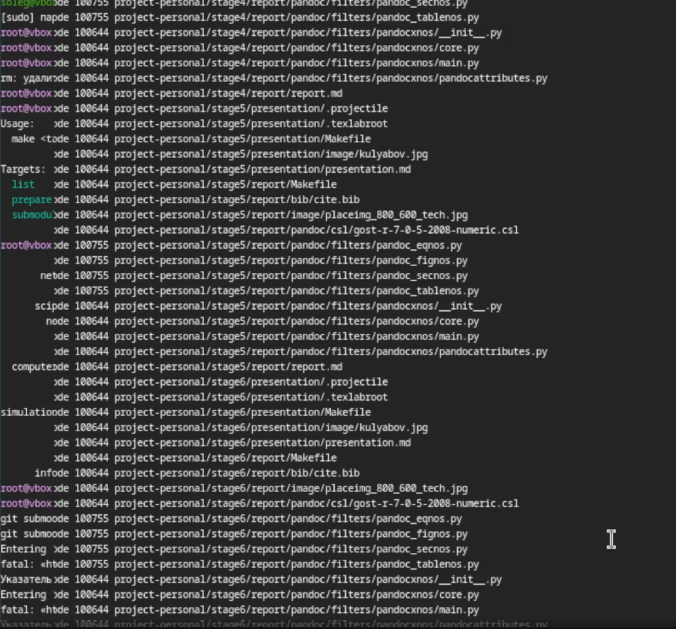


Рис. 15: Создание каталогов

Успешно отправляю файлы на сервер (рис. 16).

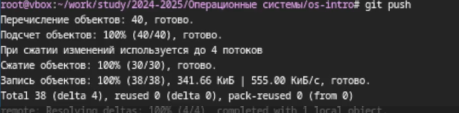


Рис. 16: Отправка файлов на сервер

# 4 Выводы

В ходе проделанной работы я изучил идеологию и применение средств контроля версий и освоил умения по работе с git.

# Список литературы

Лабораторная работа №2