Отчёт

Лабораторная работа № 2

Приходько Иван Иванович

Содержание

# 1 Цель работы

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

# 2 Задание

Создать базовую конфигурацию для git. Создать ключ SSH и PGP. Зарегистрироваться на Github. Создать локальный каталог для выполнения заданий поп предмету.

# 3 Выполнение лабораторной работы

Для начала установим git (рис. 1).

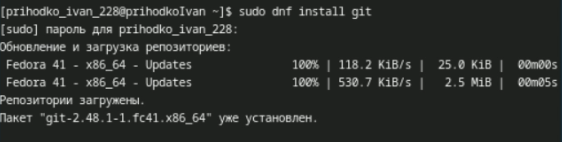


Рис. 1: Установка git

Далее установим gh (рис. 2).

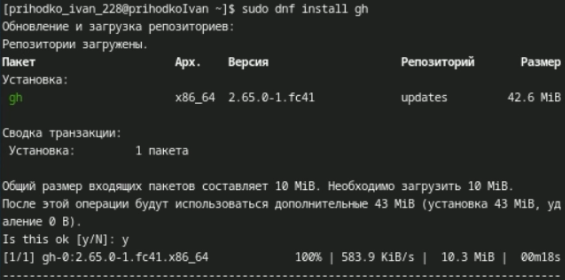


Рис. 2: Установка gh

Далее зададим имя владельца репозитория и его почту (рис. 3).

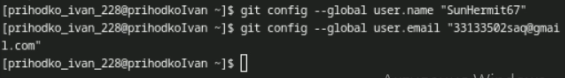


Рис. 3: Указание имени и почты

Установим имя начальной ветки, параметр autocrlf и safecrlf (рис. 4).

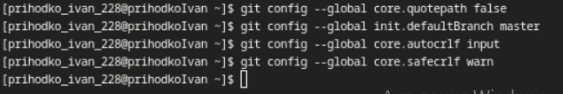


Рис. 4: Настройки git

Создадим RSA ключ размером 4096 (рис. 5).

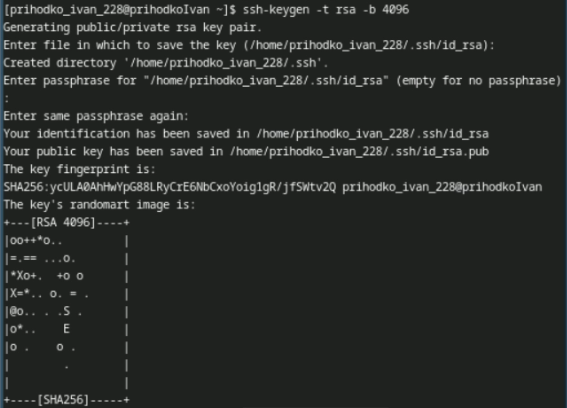


Рис. 5: Создание RSA ключа

Далее создадим ключ по алгоритму ed25519 (рис. 6).

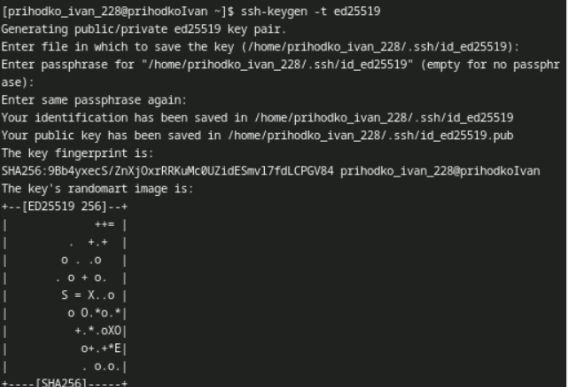


Рис. 6: Создание ключа по алгоритму ed25519

Создадим PGP ключ. Выбираем тип «RSA and RSA», на 4096 бит и срок неограничен. Далее вводим свои данные и генерируем ключ (рис. 7, 8).

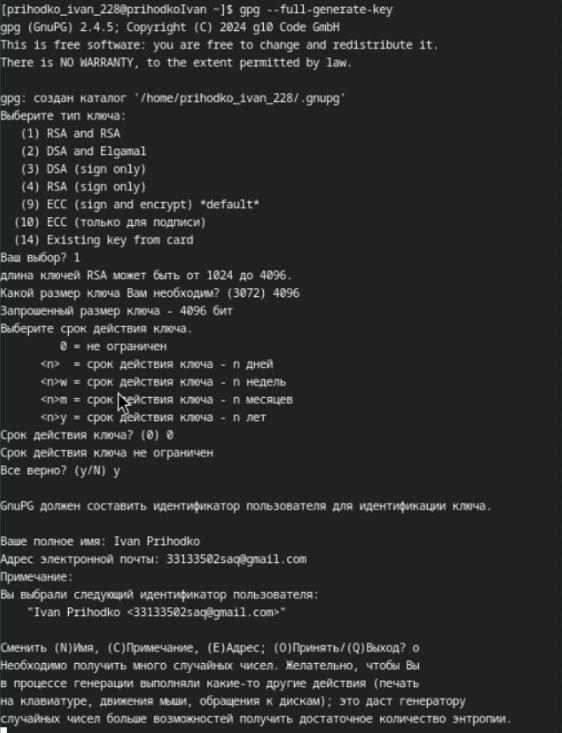


Рис. 7: Генерация PGP ключа

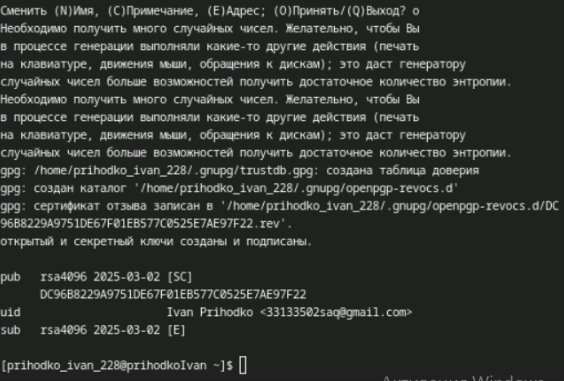


Рис. 8: Генерация PGP ключа

Выводим список pgp ключей (рис. 9).

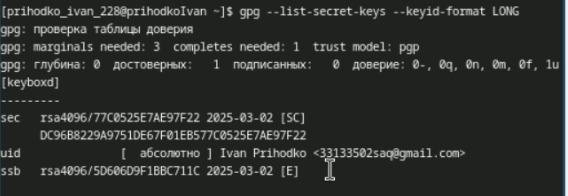


Рис. 9: Список pgp ключей

Копирем наш ключ в буфер обмена (рис. 10).

Копирование PGP ключа

Рис. 10: Копирование PGP ключа

Добавляем наш ключ на Github (рис. 11).



Рис. 11: Вставка ключа на Github

Производим найстройку автоматических подписей (рис. 12).

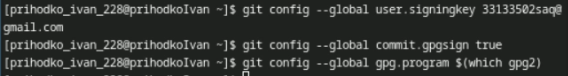


Рис. 12: Настройка автоматических подписей

После этого авторизируемся на github с помощью gh, выбираем SSH протокол, публичный ключ id\_rsa.pub и имя ключа sway (рис. 13).

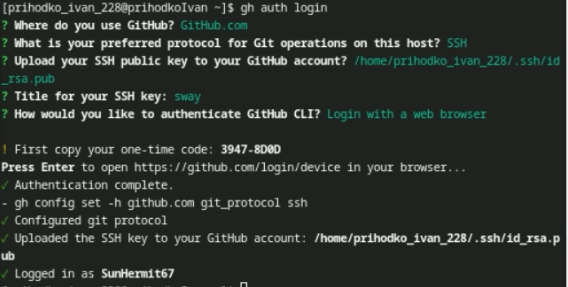


Рис. 13: Авторицазия в gh

Создаем рабочую папку (рис. 14).

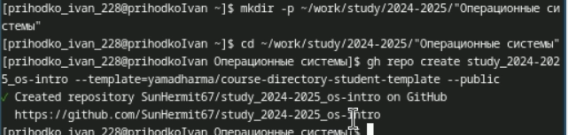


Рис. 14: Создание рабочего пространства

Копируем туда репозиторий из лабораторной работы и создаем свой (рис. 15).

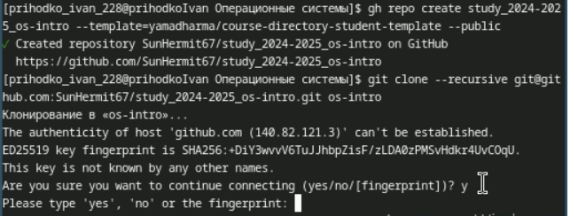


Рис. 15: Создание репозитория

Удаляем ненужный файлы и создаем необходимые каталоги. Прописываем make prepare (рис. 16).

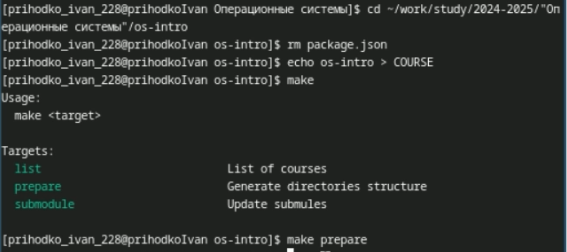


Рис. 16: Удаление ненужных файлов и работа с рабочим пространством

Добавляем нашу папку для отправки, добавляем коммит и пушим! (рис. 17-19)

Добавление папки для отправки

Рис. 17: Добавление папки для отправки

Добавление коммита

Рис. 18: Добавление коммита

Пуш

Рис. 19: Пуш

# 4 Выводы

Была произведена установка git, проведена его первоначальная настройка, были созданы ключи для авторизации и подписи, а также создан репозиторий курса из предложенного шаблона

# 5 Ответы на контрольные вопросы

1. Системы контроля версий – это системы, в которых мы можем хранить свои проекты и выкладывать их обновления, контролируя релизы и каждые внесённые изменения. Эти системы нужны для работы над проектами, чтобы иметь возможность контролировать версии проектов и в случае командной работы контролировать изменения, внесённые всеми участниками. Также, VCS позволяют откатываться на более ранние версии
2. Хранилище – репозиторий, в нём хранятся все файлы проекта и все его версии  
   commit – внесённые изменения в репозитории  
   история – это история изменений файлов проекта  
   рабочая копия – копия, сделанная из версии репозитория, с которой непосредственно работает сам разработчик
3. Централизованные системы контроля версий имеют один центральный репозиторий, с которым работают все разработчики. Примером является CVS, который является уже устаревшей системой.  
   В децентрализованных системах же используется множество репозиториев одного проекта у каждого из разработчиков, при этом репозитории можно объединять брать из каждого только то, что нужно. Примером является знакомый нам Git
4. Создаётся репозиторий, и разрабатывается проект. При внесении изменений файлы отправляются на сервер
5. Разработчик клонирует репозиторий к себе на компьютер, и после внесения изменений выгружает их на сервер в качестве отдельной версии. После этого разработчики с более высокими правами могут, например, объединить его версию с текущей
6. Хранение файлов проекта, а также обеспечение командной работы, и контроль за версиями проекта
7. git clone – клонирует проект с сервера на компьютер  
   git add – добавляет папку для выгрузки на сервер  
   git commit – фиксирует изменения репозитория  
   git push – выгружает изменения на сервер  
   git pull – получить изменения с сервера  
   git rm – удалить файл  
   git status – получить статус репозитория
8. С локальным: git commit -am “added files” – создаёт коммит С удалённым: git push – загрузить данные на удалённый сервер
9. Ветки – это несколько независимых копий проекта, в каждой из которых ведётся разработка какой-то конкретной функции, при этом ветки существуют параллельно. Они нужны, когда нужно параллельно вести разработку нескольких функций, а в конце их можно объединить в одну
10. Игнорировать файлы можно, внося их в файл .gitignore. Игнорировать файлы нужно, когда их не нужно добавлять в репозиторий. Например, это могут быть файлы виртуального окружения (venv)