Отчет по лабораторной работе №2

Архитектура компьютеров и операционные системы

Дмитрий Константинович Кобзев

Содержание

# 1 Цель работы

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

# 2 Задание

Создать базовую конфигурацию для работы с git.  
Создать ключ SSH.  
Создать ключ PGP.  
Настроить подписи git.  
Зарегистрироваться на Github.  
Создать локальный каталог для выполнения заданий по предмету.

# 3 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы.

Например, в табл. 1 приведено краткое описание стандартных каталогов Unix.

Таблица 1: Описание некоторых каталогов файловой системы GNU Linux

| Имя каталога | Описание каталога |
| --- | --- |
| / | Корневая директория, содержащая всю файловую |
| /bin | Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям |
| /etc | Общесистемные конфигурационные файлы и файлы конфигурации установленных программ |
| /home | Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя |
| /media | Точки монтирования для сменных носителей |
| /root | Домашняя директория пользователя root |
| /tmp | Временные файлы |
| /usr | Вторичная иерархия для данных пользователя |

Более подробно про Unix см. в [1–4].

# 4 Выполнение лабораторной работы

Устанавливаем git и gh (рис. 1).

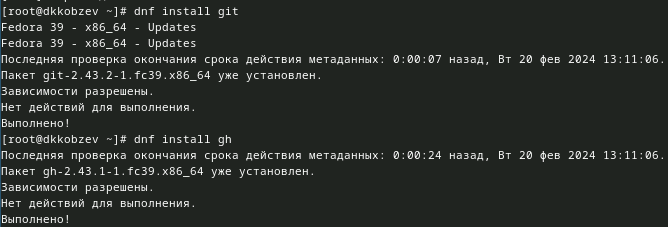


Рис. 1: Установка git и gh

Задаем имя и email владельца репозитория и настраиваем utf-8 в выводе сообщений git (рис. 2)

Владелец репозитория

Рис. 2: Владелец репозитория

Задаем имя начальной ветки, задаем параметры autocrlf и safecrlf (рис. 3).

Имя начальной ветки и параметры autocrlf и safecrlf

Рис. 3: Имя начальной ветки и параметры autocrlf и safecrlf

Создаем shh ключ по алгоритму rsa с ключем размером 4096 бит (рис. 4)

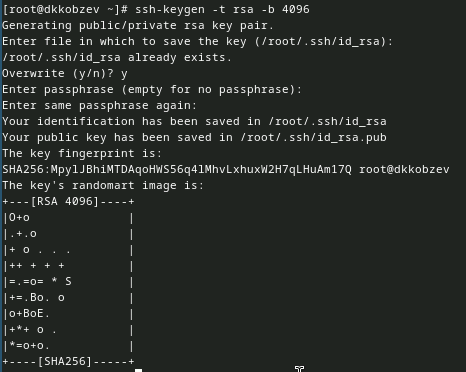


Рис. 4: Ключ ssh по алгоритму rsa

Создаем shh ключ по алгоритму ed25519 (рис. 5)

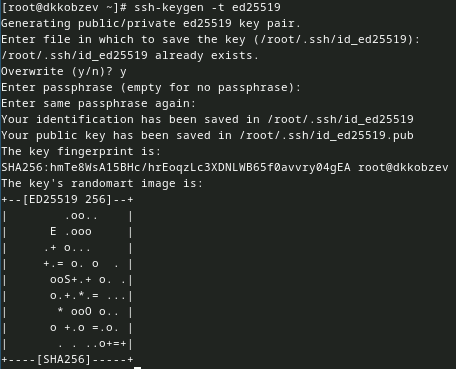
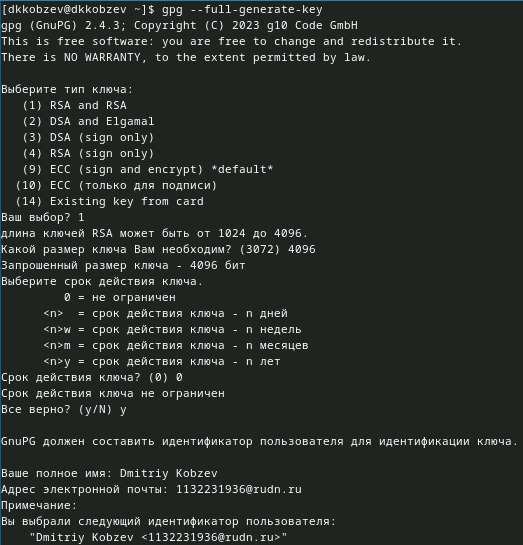
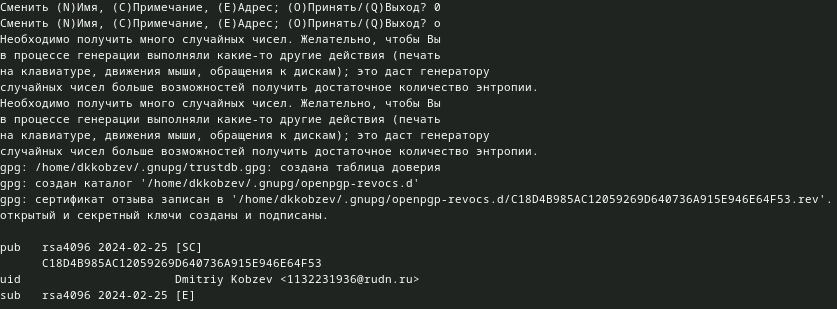


Рис. 5: Ключ ssh по алгоритму ed25519

Генерируем ключ pgp (рис. **¿fig:006?**).

Выводим список ключей и копируем отпечаток приватного ключа. Копируем наш сгенерированный PGP ключ в буфер обмена (рис. 6).

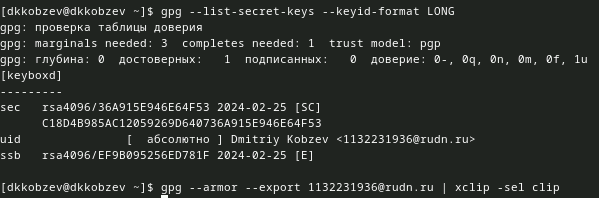


Рис. 6: Добавление PGP ключа в GitHub

Переходим в настройки GitHub, нажимаем на кнопку New PGP key и вставляем полученный ключ в поле ввода (рис. 7).

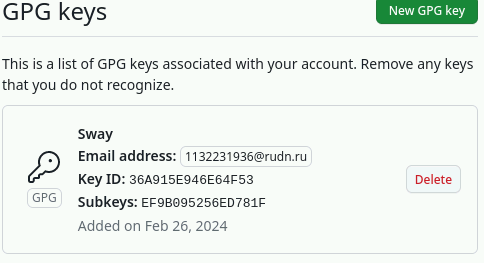


Рис. 7: Новый PGP ключ

Используя введеный email, указываем Git применять его при подписи коммитов (рис. 8).

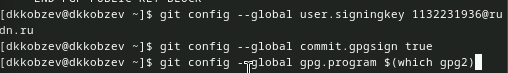


Рис. 8: Настройка автоматических подписей коммитов

Авторизовываемся через gh (рис. 9).

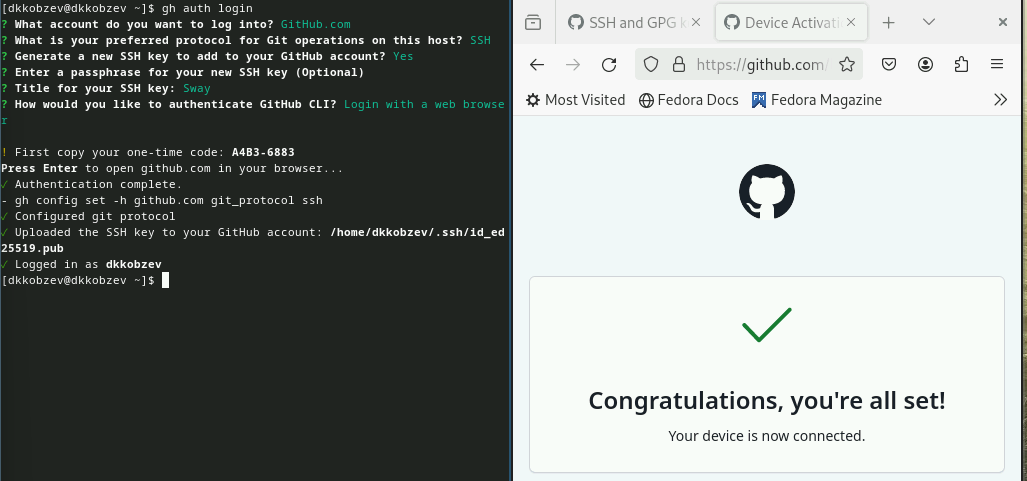
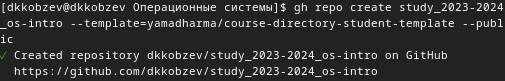
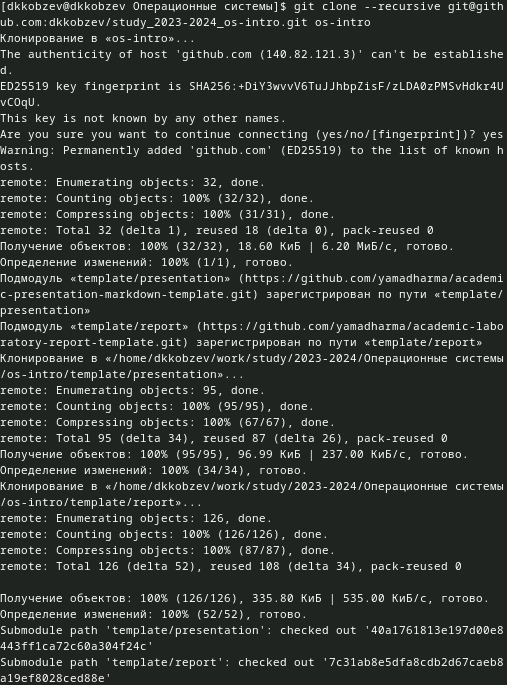


Рис. 9: Настройка gh

Создаем репозиторий курса на основе шаблона (рис. **¿fig:012?**), (рис. **¿fig:013?**), (рис. **¿fig:014?**).

Создание каталога  

Переходим в каталог курса,удаляем лишние файлы и создаем необходимые каталоги (рис. 10).

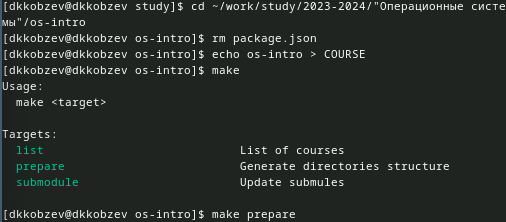
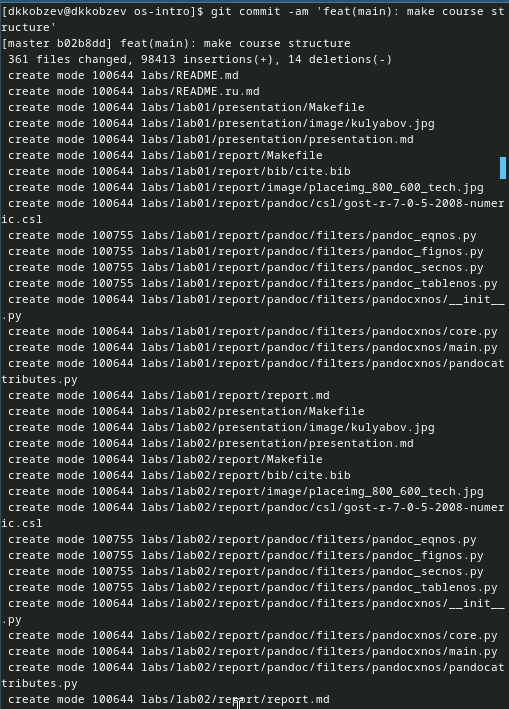
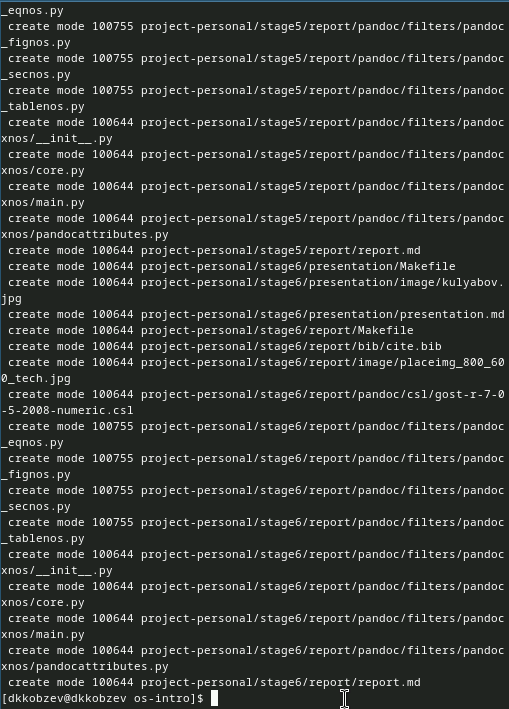
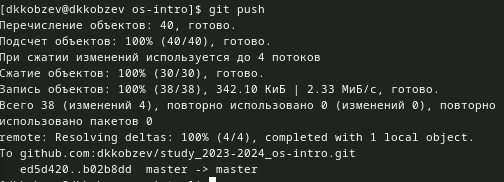


Рис. 10: Настройка каталога курса

Отправляем файлы на сервер (рис. **¿fig:016?**), (рис. **¿fig:017?**), (рис. **¿fig:018?**), (рис. **¿fig:019?**).

Команда git add .   

# 5 Контрольные вопросы

1. Системы контроля версий (VCS) предназначены для управления изменениями в исходном коде и других файлов проекта, позволяя отслеживать изменения, управлять версиями, совместно работать, создавать резервные копии, отслеживать авторство, управлять конфликтами, а также экспериментировать и ветвить проекты.
2. Хранилище - место, где хранятся файлы проекта и история их изменений. Commit - сохранение текущего состояния файлов проекта с описанием изменений. История - записи о всех сделанных изменениях в проекте. Рабочая копия - локальная копия файлов проекта, позволяющая вносить изменения и синхронизироват их с хранилищем.
3. Централизованные VCS: В таких системах основной репозиторий находится на центральном сервере. Разработчики обычно работают с локальными копиями файлов, синхронизируя их с центральным сервером при необходимости. Все действия происходят через этот центральный репозиторий. Примеры централизованных VCS: Subversion (SVN) CVS (Concurrent Versions System) Децентрализованные VCS: В децентрализованных системах каждый разработчик имеет локальную копию полного репозитория, включая всю его историю и метаданные. Это означает, что разработчики могут работать независимо друг от друга и без постоянного подключения к центральному серверу. Синхронизация изменений происходит напрямую между локальными репозиториями. Примеры децентрализованных VCS: Git Mercurial
4. Создание репозитория, добавление файлов для работы, фиксация изменений, отправка файлов на сервер.
5. Клонирование репозитория, работа с файлами, добавление файлов, фиксация изменений, получение изменений, отправка изменений на сервер.
6. Управление версиями файлов проекта. Совместная разработка между несколькими разработчиками. Отслеживание изменений и истории проекта. Ветвление и слияние изменений для разработки новых функций. Резервное копирование и восстановление данных. Работа с удаленными репозиториями для совместной работы и обмена изменениями.
7. Команда git add добавляет содержимое рабочего каталога в индекс для последующего коммита. Команда git commit берёт все данные, добавленные в индекс с помощью git add, и сохраняет их. Команда git clone клонирует существующий репозиторий Git, создавая локальную копию. Команда git init cоздает новый репозиторий Git в текущем каталоге. Команда git status показывает состояния файлов в рабочем каталоге и индексе. Команда git push отправляет локальные коммиты в удаленный репозиторий. Команда git pull получает изменения из удаленного репозитория и объединяет их с локальными изменениями. Команда git diff используется для вычисления разницы между любыми двумя Git деревьями. Команда git branch gоказывает список веток, а также создает, удаляет или переключается между ними.
8. Локальный: git commit - фиксирует изменения Удаленный: git push - отправляет изменения на сервер
9. Ветви (branches) в системах контроля версий, представляют собой механизм, позволяющий разработчикам работать параллельно над разными версиями проекта, изолируя изменения и эксперименты от основной линии разработки.
10. Игнорировать файлы при commit можно с помощью файла ‘.gitignore’. Этот файл содержит шаблоны путей к файлам и каталогам, которые не должны быть добавлены в репозиторий Git. Как использовать: Создаем файл ‘.gitignore’ в корневом каталоге проекта и добавляем в него шаблоны путей к файлам и каталогам, которые хотим проигнорировать. Добавляем файл в репозиторий Git и выполняем коммит. Зачем: Игнорировать временные файлы, не являющиеся частью исходного кода. Предотвращение попадания конфеденциальных данных в репозиторий. Улучшение производительности Git, исключив большие или ненужные файлы из репозитория.

# 6 Выводы

Мною были изучены идеология и применение средств контроля версий иосвоены умения по работе с git.

# Список литературы

1. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.

2. Robbins A. Bash Pocket Reference. O’Reilly Media, 2016. 156 с.

3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.

4. Newham C. [Learning the bash Shell: Unix Shell Programming](http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658). O’Reilly Media, 2005. 354 с.