Отчет по лабораторной работе №2

Операционные системы

Курилко-Рюмин Евгений Михайлович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Цель лабораторной работы заключается в изучение применения средств контроля версий,а также освоение умения по работе с git.

2 Задание

1. Создание базовой конфигурацию для работы с git
2. Создание SSH ключ
3. Создание GPG ключ
4. Настройка подписи Git
5. Регистрирование на GitHub
6. Создание локального каталога для выполнения заданий по предмету.

3 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.

4 Выполнение лабораторной работы

4.1 Установка программного обеспечения

Устанавливаю необходимое программное обеспечение git и gh через терминал с помощью команд: dnf install git и dnf install gh (рис. fig. 4.1).

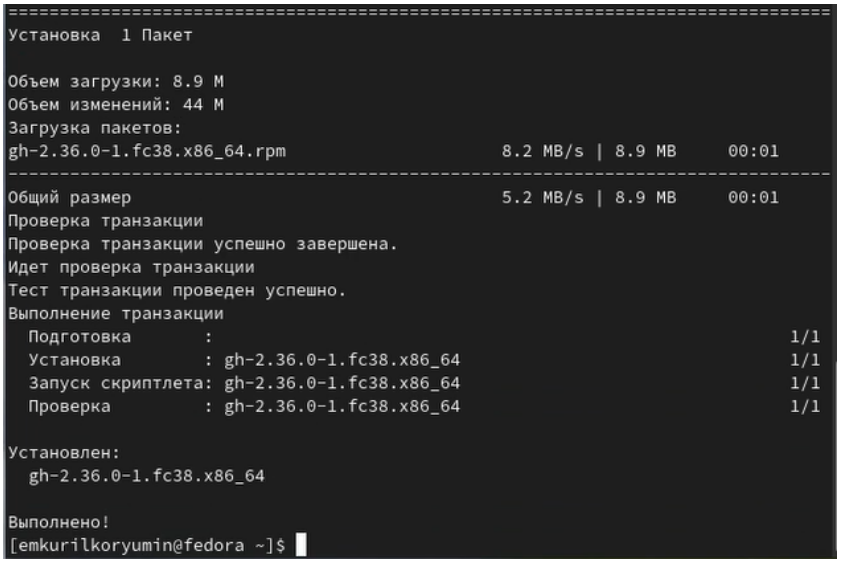

Рис. 4.1: Установка git и gh

4.2 Базовая настройка git

Задаю в качестве имени владельца репозитория и его email свое имя,фамилию и почту (рис. fig. 4.2)

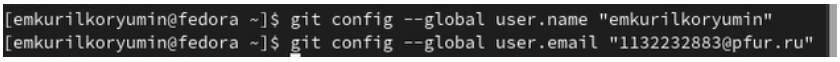

Рис. 4.2: Настройка git

Настраиваю utf-8 в выводе сообщений git для корректного отображения (рис. fig. 4.3)

Рис. 4.3: Настройка utf-8

Задаю имя для начальной ветки “master” (рис. fig. 4.4)

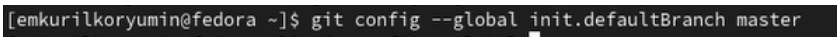

Рис. 4.4: Задача имени для начальной ветки

Далее задаю параметры для autocrlf и safecrlf для корректного отображения конца строки (рис. fig. 4.5)

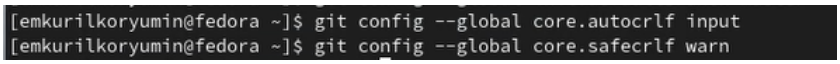

Рис. 4.5: Задача параметров autocrlf и safecrlf

4.3 Создание SSH ключа

Создаю SSH ключ размером 4096 бит по алгоритму RSA (рис. fig. 4.6)

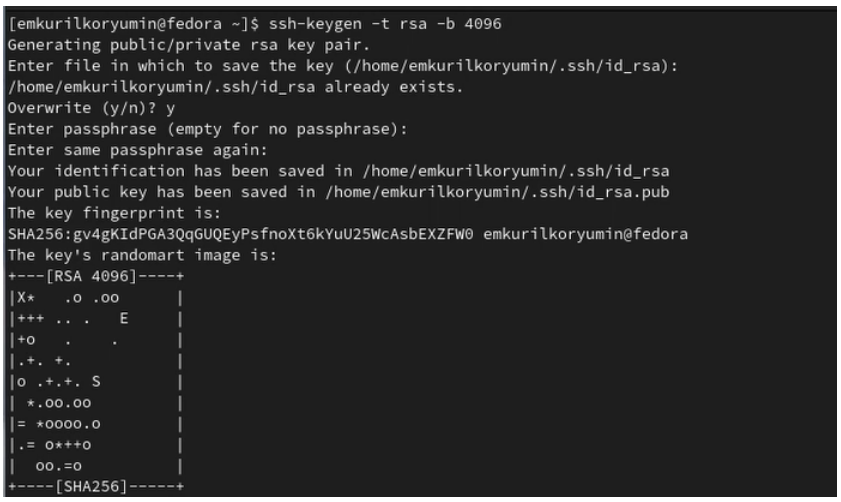

Рис. 4.6: Генерация ssh ключа по алгоритму rsa

Создаю ключ ssh по алгоритму ed25519 (рис. fig. 4.7).

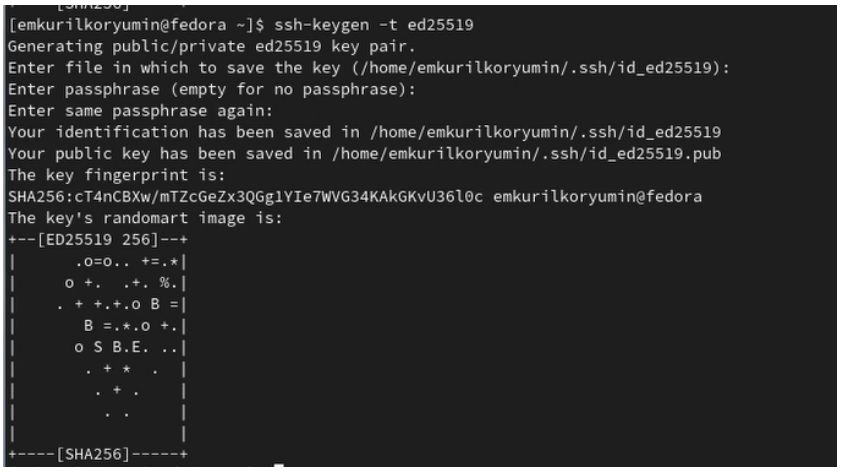

Рис. 4.7: Генерация ssh ключа по алгоритму ed25519

4.4 Создание ключа GPG

Генерирую ключ GPG, затем выбираю тип ключа RSA и RSA, задаю максимальную длину ключа: 4096, оставляю неограниченный срок действия ключа. Далее отвечаю на вопросы программы о личной информации (рис. fig. 4.8).

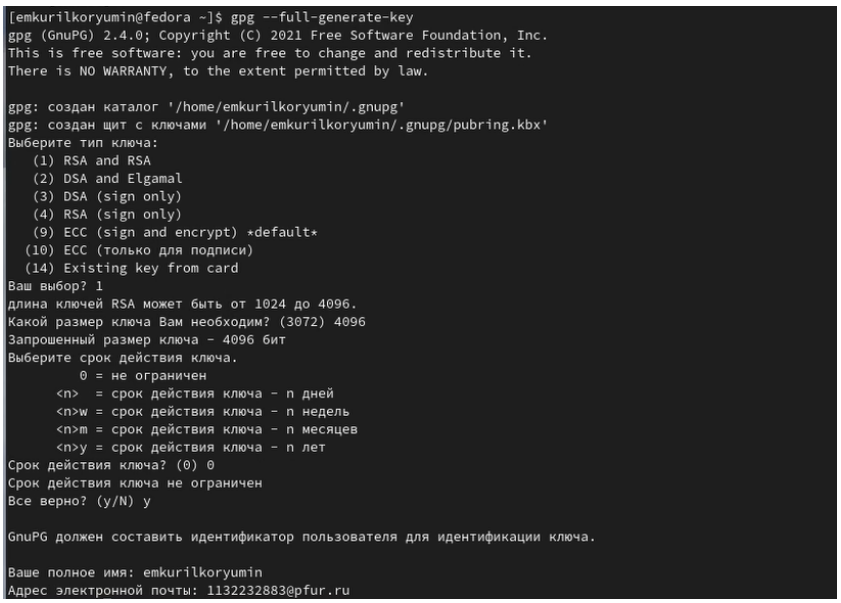

Рис. 4.8: Генерация ключа

Ввожу фразу-пароль для защиты нового ключа (рис. fig. 4.9).

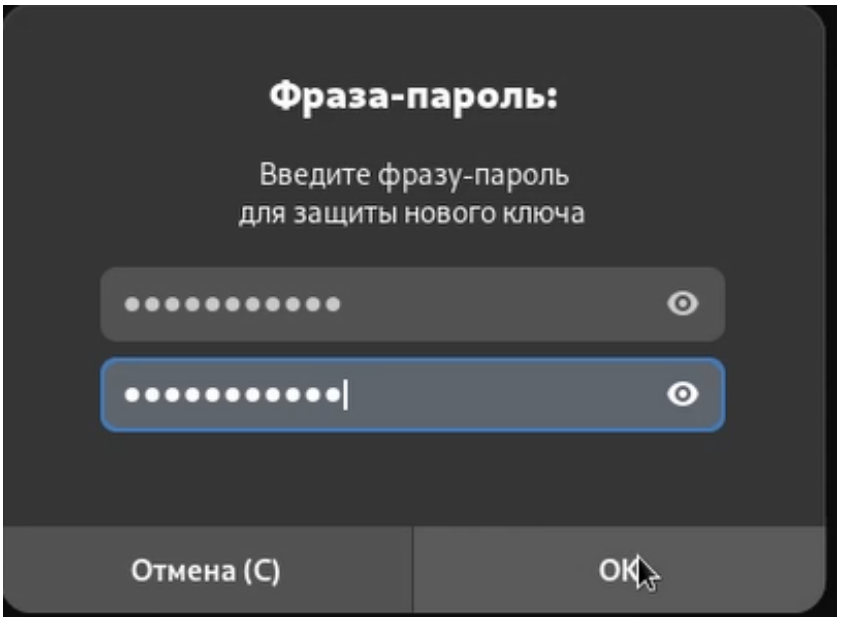

Рис. 4.9: Защита ключа GPG

##Создание учётной записи на Github

Так как у меня уже был создан аккаунт на Github,то основные данные аккаунта и его настройка уже была проведена мной,поэтому просто захожу в свой аккаунт

(рис. fig. 4.10).

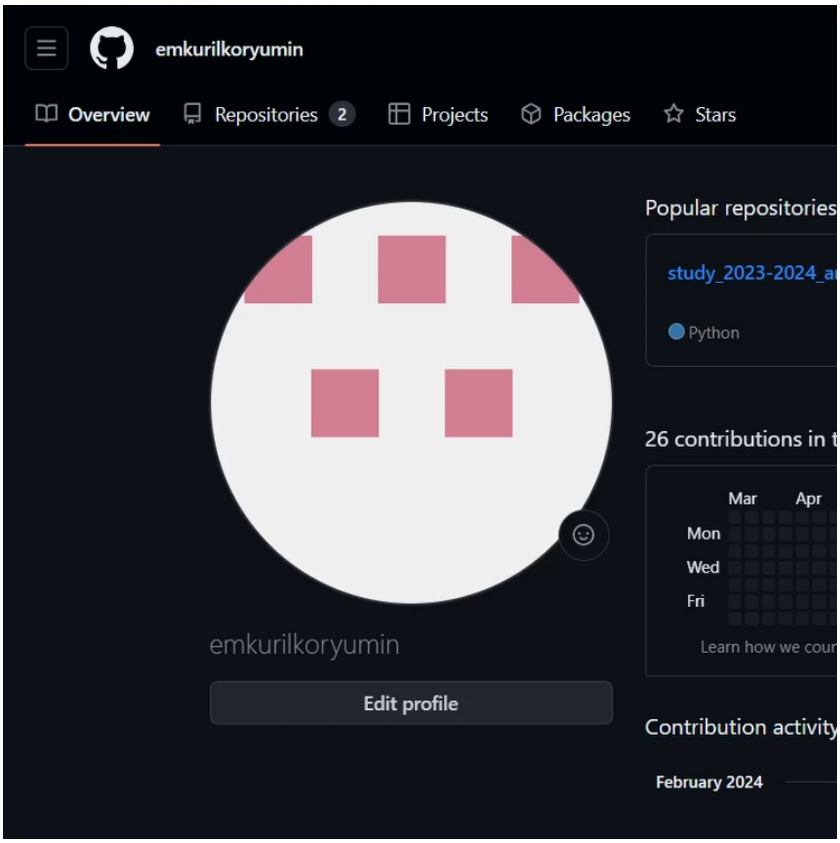

Рис. 4.10: Учетная запись Github

4.5 Добавление ключа GPG в Github

Вывожу список созданных ключей в терминал, ищу последовательность байтов для идентификации более длинного, по сравнению с самим отпечатком ключа, копирую его в буфер обмена (рис. fig. 4.11).

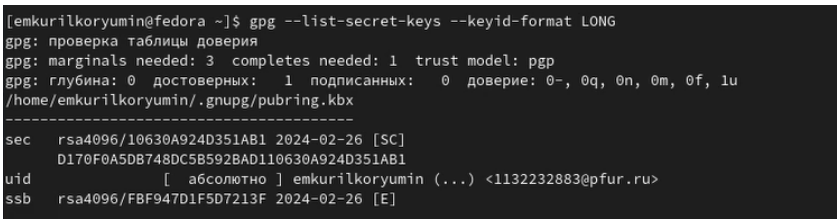

Рис. 4.11: Вывод списка ключей

Ввожу в терминале команду, с помощью которой копирую сам ключ GPG в буфер обмена, за это отвечает утилита xclip (рис. fig. 4.12).

Рис. 4.12: Копирование ключа в буфер обмена

Открываю настройки GirHub, ищу среди них добавление GPG ключа (рис. fig. 4.13).

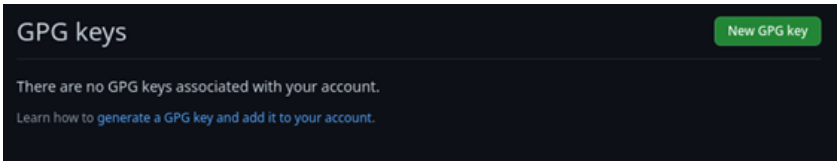

Рис. 4.13: Настройки GitHub

Нажимаю на “New GPG key” и вставляю в поле ключ из буфера обмена (рис. fig. 4.14).

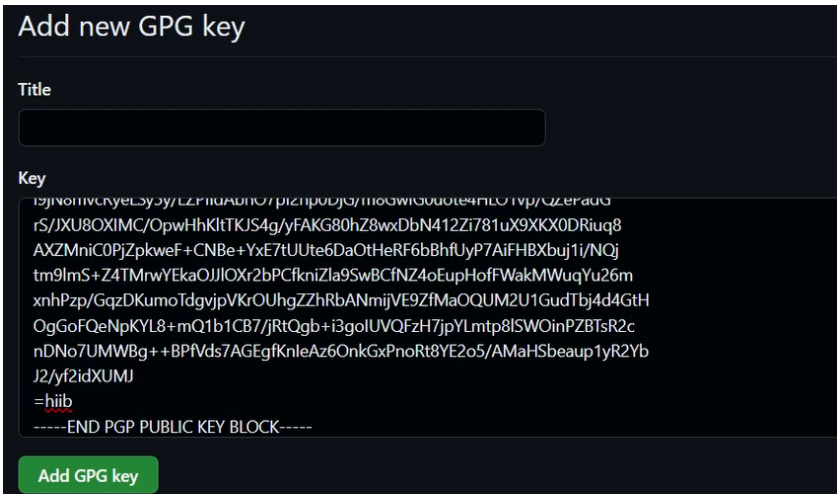

Рис. 4.14: Добавление нового PGP ключа

Ключ GPG добавлен на GitHub (рис. fig. 4.15).

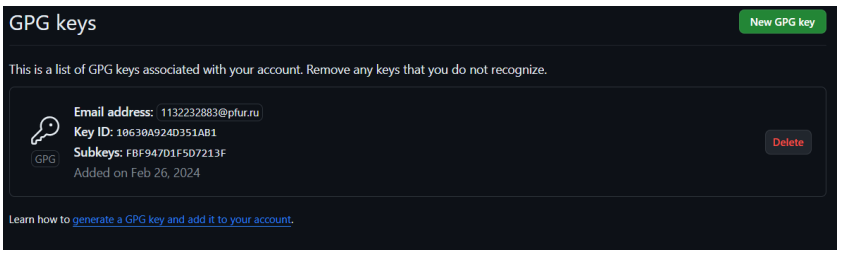

Рис. 4.15: Добавленный ключ GPG

##Настройка автоматических подписей коммитов git

Настраиваем автоматическую подпись коммитов git используя введеный ранее email,указываем git который буду использовать при создании подписей коммитов (рис. fig. 4.16).

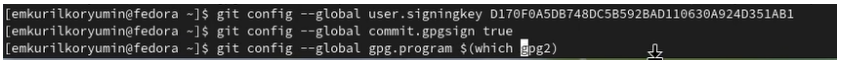

Рис. 4.16: Настройка подписей коммитов git

4.6 Настройка gh

Начинаю авторизацию в gh, отвечаю на наводящие вопросы от утилиты, в конце выбираю авторизоваться через браузер (рис. fig. 4.17).

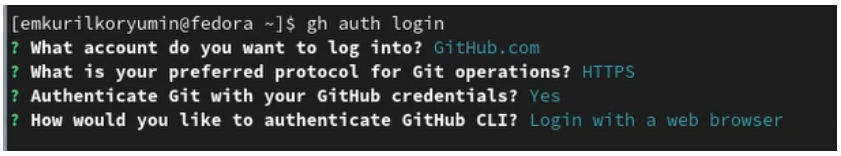

Рис. 4.17: Авторизация в gh

Завершаю авторизацию на сайте (рис. fig. 4.18).

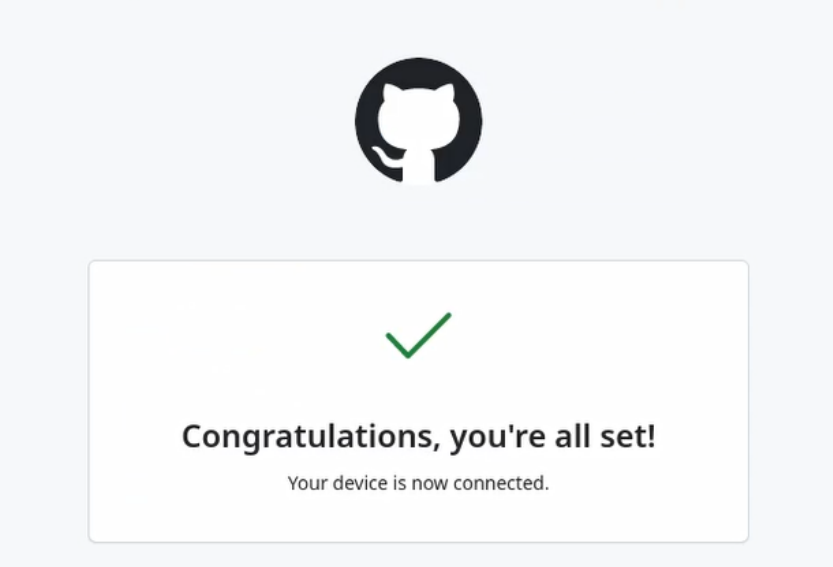

Рис. 4.18: Завершение авторизации через браузер

Вижу сообщение о завершении авторизации,в докозательство выводится мое имя emkurilkorymin (рис. fig. 4.19).

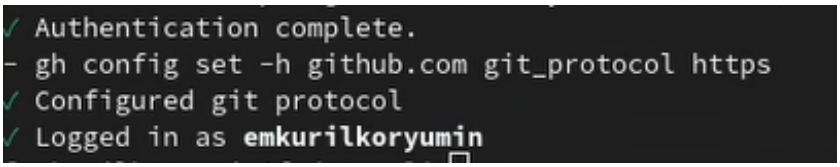

Рис. 4.19: Завершение авторизации

##Сознание репозитория курса на основе шаблона

Создаю директорию с помощью утилиты mkdir и флага -p, который позволяет установить каталоги на всем указанном пути. После этого с помощью утилиты cd перехожу в созданную директорию.Далее в терминале ввожу команду gh repo create study_2023-2024_os-intro –template yamadharma/course-directory-student-trmplate –public, чтобы создать репозиторий на основе шаблона репозитория. После этого клонирую репозиторий к себе в директорию (рис. fig. 4.20).

Рис. 4.20: Создание репозитория

Дальше перехожу в каталог курса применя утилиту cd (рис. fig. 4.21).

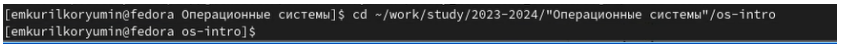

Рис. 4.21: Переход в каталог

Удаляем лишние файлы с помощью утилиты rm,затем создаю каталог используя makefile (рис. fig. 4.22).

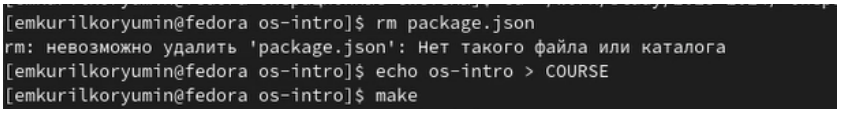

Рис. 4.22: Удаление лишних файлов

С помощью команд git add и git commit,добавляю новые файлы и комментирую их для отправки на сервер (рис. fig. 4.23).

Рис. 4.23: подготовка файлов к отправлению на сервер

Отправляю файлы на сервер (рис. fig. 4.24).

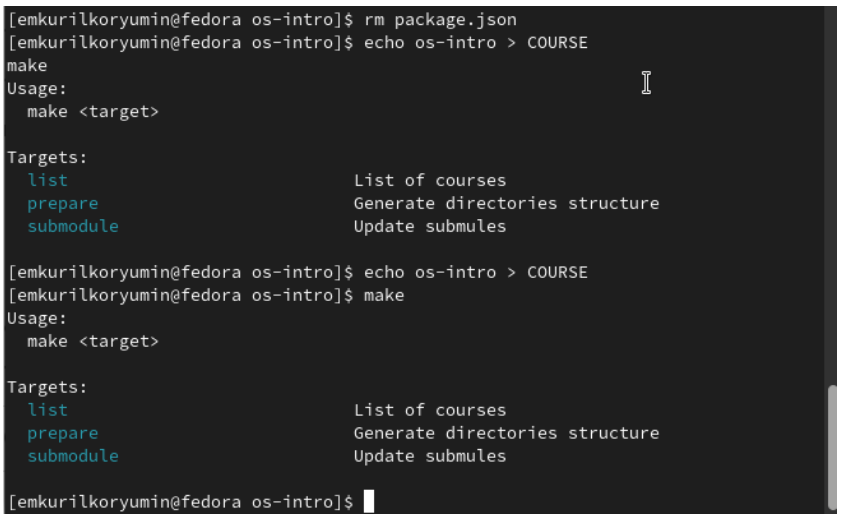

Рис. 4.24: отправление файлов на сервер

5 Выводы

В ходе выполнения лабораторной работы я научился применять средства контроля версий,а также получил знания по работе с git.

6 Ответы на контрольные вопросы.

1. Системы контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Они позволяют хранить несколько версий изменяющейся информации, одного и того же документа, может предоставить доступ к более ранним версиям документа. Используется для работы нескольких человек над проектом, позволяет посмотреть, кто и когда внес какое-либо изменение и т. д. VCS ррименяются для: Хранения понлой истории изменений, сохранения причин всех изменений, поиска причин изменений и совершивших изменение, совместной работы над проектами.

2. Хранилище – репозиторий, хранилище версий, в нем хранятся все документы, включая историю их изменения и прочей служебной информацией. commit – отслеживание изменений, сохраняет разницу в изменениях. История – хранит все изменения в проекте и позволяет при необходимости вернуться/обратиться к нужным данным. Рабочая копия – копия проекта, основанная на версии из хранилища, чаще всего последней версии.

3. Централизованные VCS (например: CVS, TFS, AccuRev) – одно основное хранилище всего проекта. Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет, затем добавляет изменения обратно в хранилище. Децентрализованные VCS (например: Git, Bazaar) – у каждого пользователя свой вариант репозитория (возможно несколько вариантов), есть возможность добавлять и забирать

изменения из любого репозитория. В отличие от классических, в распределенных (децентралиованных) системах контроля версий центральный репозиторий не является обязательным.

4. Сначала создается и подключается удаленный репозиторий, затем по мере изменения проекта эти изменения отправляются на сервер.

5. Участник проекта перед началом работы получает нужную ему версию проекта в хранилище, с помощью определенных команд, после внесения изменений пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются. К ним можно вернуться в любой момент.

6. Хранение информации о всех изменениях в вашем коде, обеспечение удобства командной работы над кодом.

7. 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 (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)

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

8. git push -all отправляем из локального репозитория все сохраненные изменения в центральный репозиторий, предварительно создав локальный репозиторий и сделав предварительную конфигурацию.

9. Ветвление - один из параллельных участков в одном хранилище, исходящих из одной версии, обычно есть главная ветка. Между ветками, т. е. их концами возможно их слияние. Используются для разработки новых функций.

10. Во время работы над проектом могут создаваться файлы, которые не следуют добавлять в репозиторий. Например, временные файлы. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов.

Список литературы

Архитектура компьютеров и ОС/Электронный ресурс