Отчёт по лабораторной работе №2

Операционные системы

Шатохина Виктория Сергеевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задания

Создать базовую конфигурацию для работы с git. Создать ключSSH.–Создать ключPGP. Настроить подписи git. Зарегистрироваться наGithub. Создатьлокальный каталогдля выполнения заданий по предмету.

3 Выполнение лабораторной работы

Задаём имя и email владельца репозитория, настраиваем utf-8 в выводе сообщений git (рис.3.1).

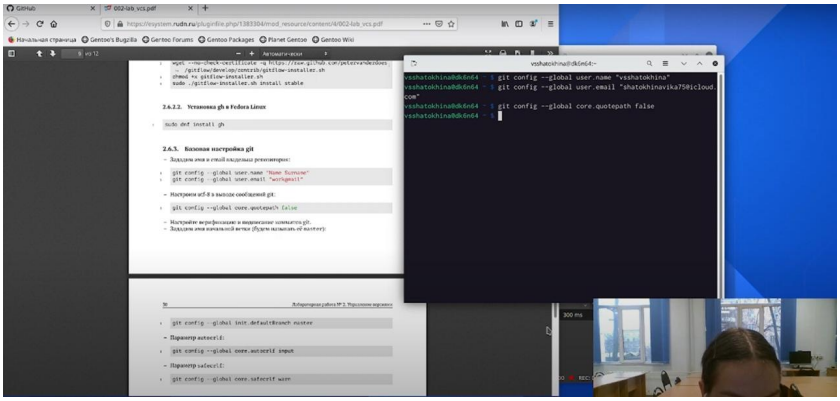

Рис. 3.1: «Создаём базовую конфигурацию, задаём имя, настраиваем utf-8»

Настраиваем верификацию и подписание коммитов git. Задаём имя начальной ветки (master), параметр autocrlf и параметр safecrlf. (рис.3.2)

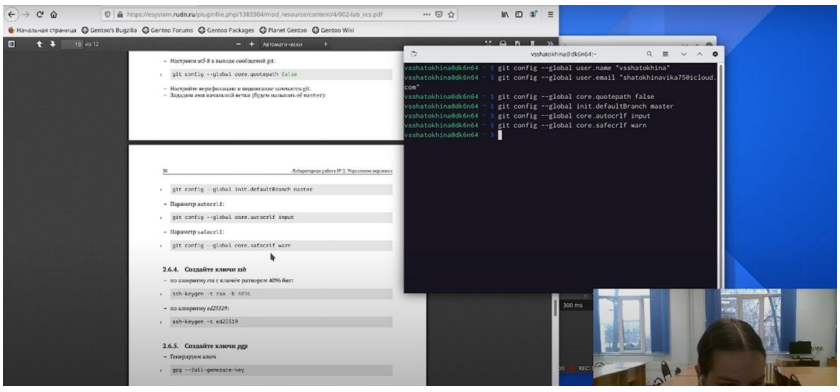

Рис. 3.2: «Настройка верификации, подписание коммитов, задача имени и параметров»

По алгоритму rsa с ключом размером 4096 бит настраиваем ключ.(рис.3.3) (рис.3.4)

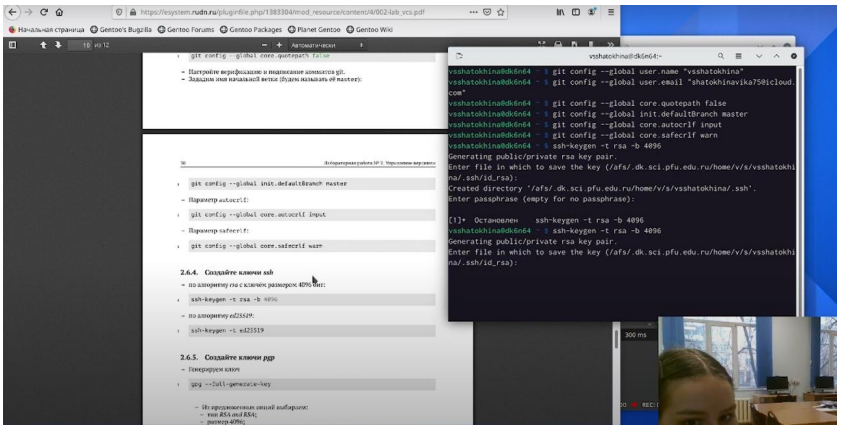

Рис. 3.3: «Выполнение по алгоритму rsa»

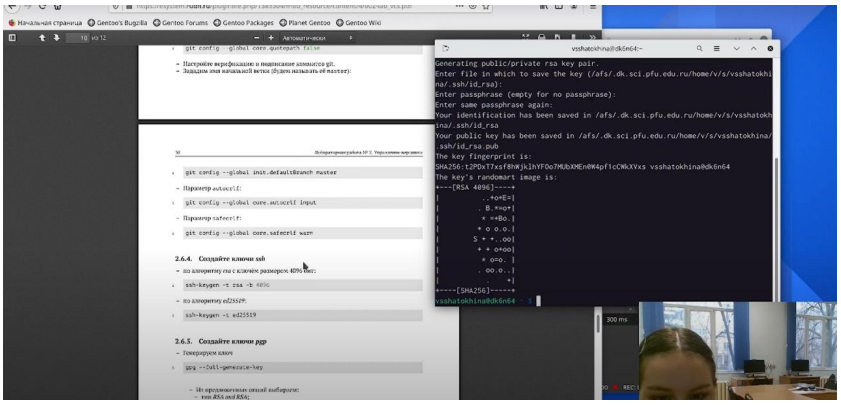

Рис. 3.4: «Создание ключа ssh»

Далее работаем с сервером репозиториев. Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый): затем, скопировав из локальной консоли ключ в буфер обмена, вставляем ключ в появившееся на сайте поле.(рис.3.5) (рис.3.6)

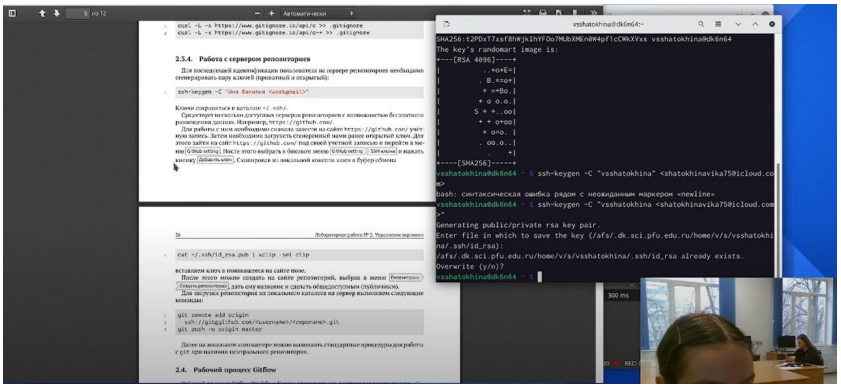

Рис. 3.5: «Генерируем ключи»

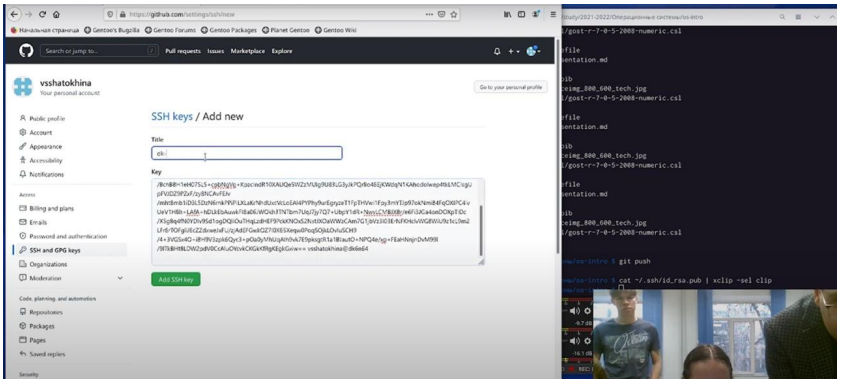

Рис. 3.6: «Вставляем ключ на сайт»

Для загрузки репозитория из локального каталога на сервер выполняем следующие команды: git remote add origin ssh://git@github.com//.git git push -u origin master

Создаём репозиторий.(рис.3.7) (рис.3.8)

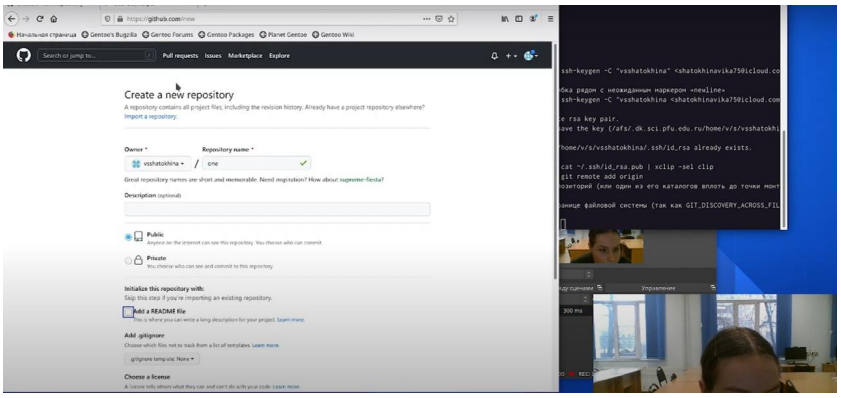

Рис. 3.7: «Создание репозитоия»

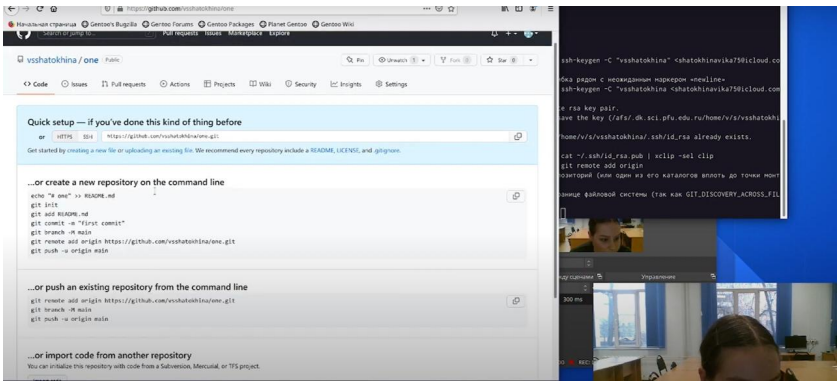

Рис. 3.8: «Репозиторий создан»

Создать ключ PGP Генерируем ключ и выбираем нужные нам опции(рис.3.9) (рис.3.10)

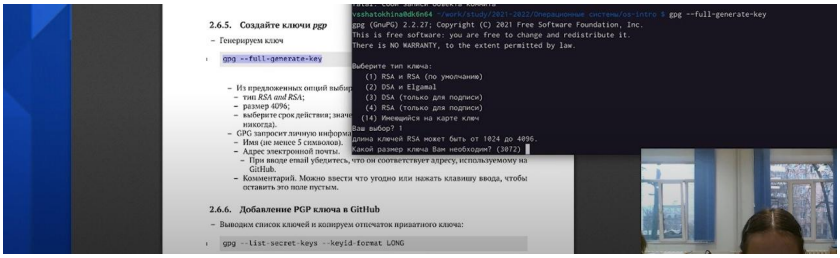

Рис. 3.9: «Генерируем ключ»

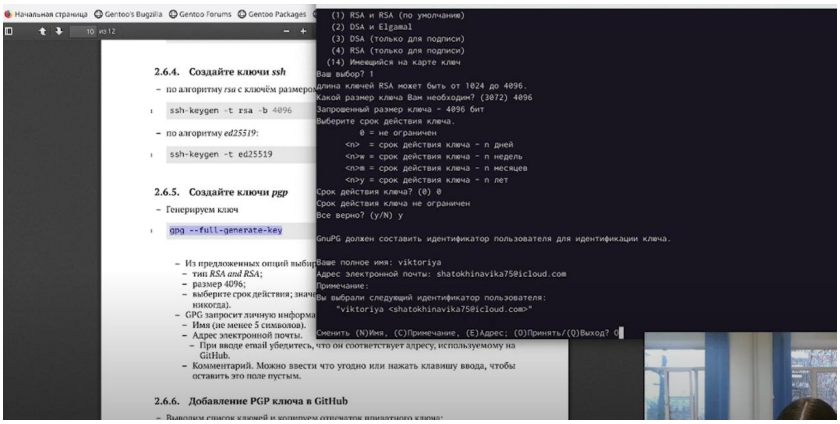

Рис. 3.10: «Выбор нужных условий»

Далее следует ввести пароль. (рис.3.11)

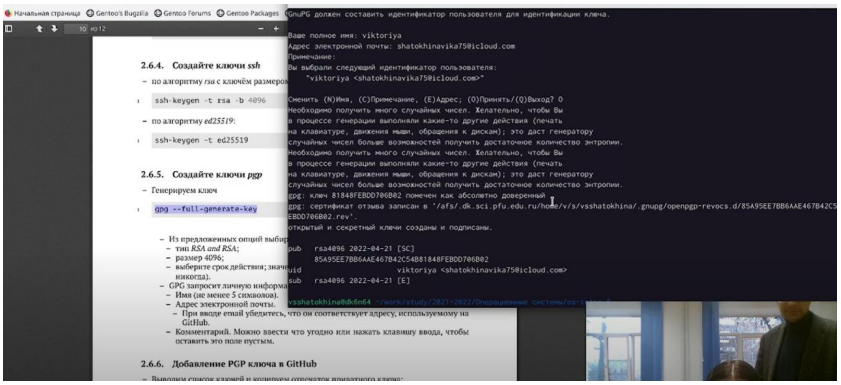

Рис. 3.11: «Вводим пароль»

Выполняем это задание следующим образом: используя введённый email, указываем Git (применять его при подписи коммитов) (рис.3.12)

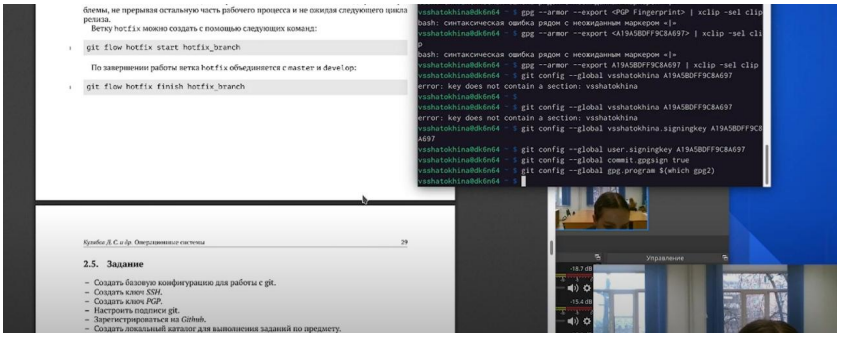

Рис. 3.12: «Выполнение задание №4»

Во время выполнения лабораторной работы я выполнила регистрацию.(рис.3.13)

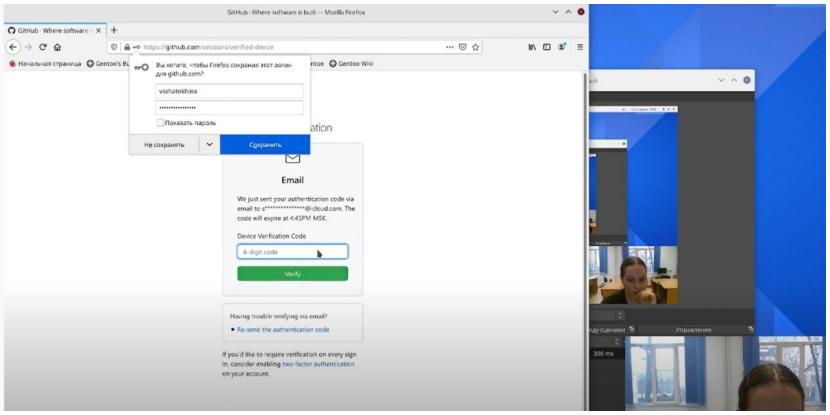

Рис. 3.13: «Регистрация на Github»

Задание было выполнено мною. Каталог создан.(рис.3.14)

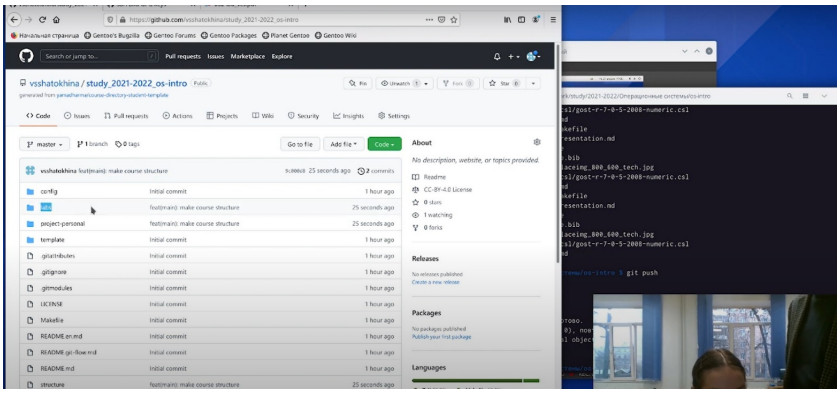

Рис. 3.14: «Созданный каталог»

4 Выводы

Мною были изучены идеология и применение средств контроля знаний, освоены умения по работе с git.

Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Система контроля версий (VCS) — это место хранения кода. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое. Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы.

2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Репозиторий - хранилище версий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией Commit («[трудовой] вклад», не переводится) — процесс создания новой версии Рабочая копия (working copy) — текущее состояние файлов проекта, основанное на версии, загруженной из хранилища (обычно на последней).

Версия (revision), или ревизия, — состояние всех файлов на определенный момент времени, сохраненное в репозитарии, с дополнительной информацией

3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные системы — это системы, которые используют архитектуру клиент / сервер,

где один или несколько клиентских узлов напрямую подключены к центральному серверу. (Пример — Wikipedia.) В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. (Пример — Bitcoin)

4. Опишите действия с VCS при единоличной работе с хранилищем.

5. Опишите порядок работы с общим хранилищем VCS.

6. Каковы основные задачи, решаемые инструментальным средством git? У Git есть две основные задачи: хранить информацию обо всех изменениях в коде, начиная с самой первой строчки, и обеспечить удобства командной работы над кодом.

7. Назовите и дайте краткую характеристику командам git. – 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 изменения: 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 – сл ияние 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 9. Что такое и зачем могут быть нужны ветви (branches)? ‘Git branch’ – это команда для управления ветками в репозитории Git. Ветка – это просто «скользящий» указатель на один из коммитов. Когда мы создаём новые коммиты, указатель ветки автоматически сдвигается вперёд, к вновь созданному коммиту. Ветки используются для разработки одной части функционала изолированно от

других. Каждая ветка представляет собой отдельную копию кода проекта. Ветки позволяют одновременно работать над разными версиями проекта. Ветвление («ветка», branch) — один из параллельных участков истории в одном хранилище, исходящих из одной версии (точки ветвления). Ветки нужны для того, чтобы программисты могли вести совместную работу над проектом и не мешать друг другу при этом.

10. Как и зачем можно игнорировать некоторые файлы при commit? Игнорируемые файлы обычно представляют собой файлы, специфичные для платформы, или автоматически созданные из сборочных систем. Временно игнорировать изменения в файле можно командой: git update-index —assume-unchanged