# Отчёт по лабораторной №2

**Дисциплина: Операционные системы**

Воронов Александр Валерьевич

# Содержание

# Список иллюстраций

# Список таблиц

# 1 Цель работы

Изучить идеологию и применение средств контроля версий Освоить умения по работе с git # Задание 1. Создать базовую конфигурацию для работы с git. 2. Создать ключ SSH 3. Создать ключ PGP. 4. Настроить подписи git. 5. Зарегистрироваться на Github. 6. Создать локальный каталог для выполнения заданий по предмету.

# 2 Теоретическое введение

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хра В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнен

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одни

Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с не

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

# 3 Выполнение лабораторной работы

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

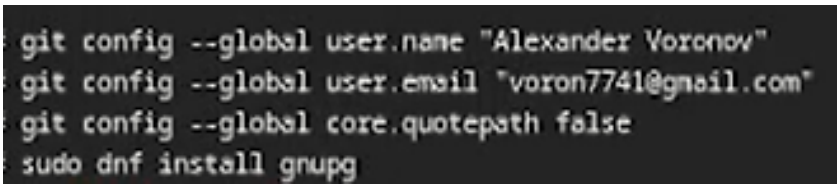


Рис. 3.1: Базовая настройка git

Создание и добавление SSH ключа на github(рис.3.2)

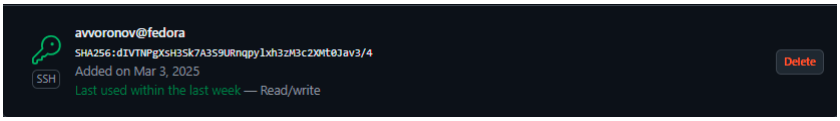


Рис. 3.2: Добавление SSH ключа на github

Генерируем GPG ключ(рис.3.3)

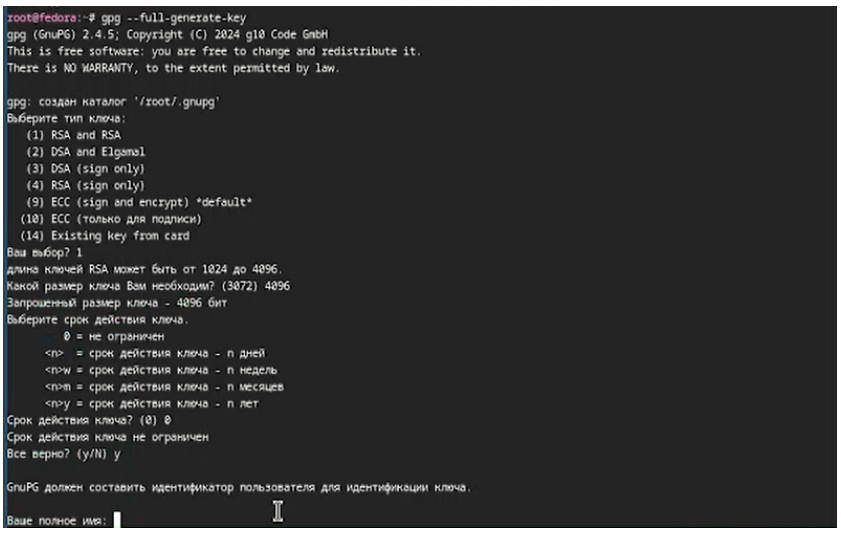


Рис. 3.3: Генерация GPG ключа

Копируем GPG в буфер обмена(рис.3.4)

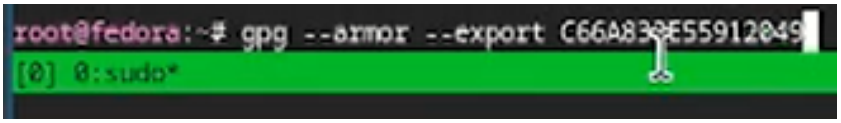


Рис. 3.4: Копирование GPG ключ

Добавляем GPG ключ на github(рис.3.5)


Рис. 3.5: Добавление GPG ключа на github

Используем введенный email, указывая git где будут применять его при подписи коммитов(рис. 3.6)

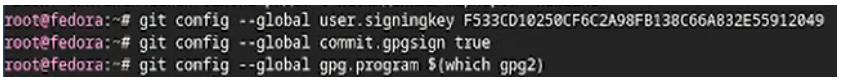


Рис. 3.6: Настройка автоматических подписей коммитов git

Авторизуемся с помощью gh auth login(рис.3.7)

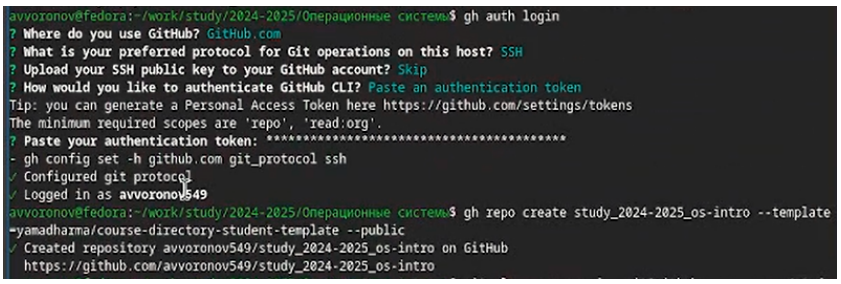


Рис. 3.7: Настройка gh

Создаем репозиторий курса на основе шаблона(рис.3.8)

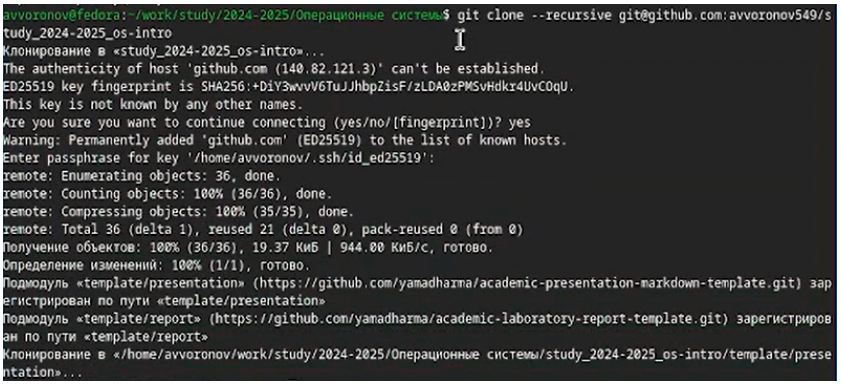


Рис. 3.8: Создание репозитория

Создаем необходимые каталоги(рис.3.9)

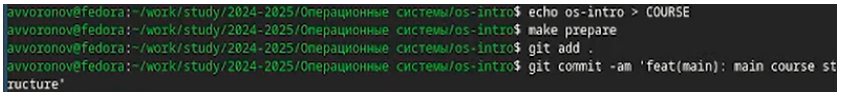


Рис. 3.9: Создание файлов для отправки

Отправляем файлы на сервер(рис.3.10)

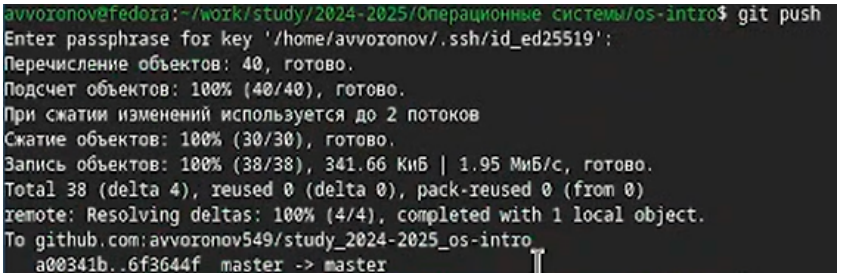


Рис. 3.10: Отправление файлов на сервер

# 4 Контрольные вопросы

Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

Система контроля версий — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления в ерсиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним вер сиям, определять, кто и когда сделал то или иное изменение, и многое другое. Системы контроля версий (Version Control Sy stem, VCS) применяются для: Хранение полной истории изменений причин всех производимых изменений Откат изменений, если что-то пошло не так Поиск причины и ответственного за появления ошибок в программе Совместная работа группы над одним проектом Возможность изменять код, не мешая работе других пользователей

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 информац ией.Commit — отслеживание изменений Рабочая копия - копия проекта, связанная с репозиторием (текущее состояние файлов проекта, основанное на версии из храни лища (обычно на последней) История хранит все изменения в проекте и позволяет при необходимости обратиться к нужным данным.

3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида

Централизованные VCS (Subversion; CVS; TFS; VAULT; AccuRev): Одно основное хранилище всего проекта Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменени я обратно Децентрализованные VCS (Git; Mercurial; Bazaar): У каждого пользователя свой вариант (возможно не один) репозитория Присутствует возможность добавлять и забирать изменения из любого репозитория . В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большин ства функций по управлению версиями осуществляется специальным сервером. В отличие от классических, в распределённых сис темах контроля версий центральный репозиторий не является обязательным.

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

Сначала создаем и подключаем удаленный репозиторий. Затем по мере изменения проекта отправлять эти изменения на сервер.

5. Опишите порядок работы с общим хранилищем VCS.

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 це нтрального хранилища и к ним можно вернуться в любой момент.

6. Каковы основные задачи, решаемые инструментальным средством git?

Первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобс тва командной работы над кодом.

7. Назовите и дайте краткую характеристику командам git.

Наиболее часто используемые команды git: создание основного дерева репозитория: git init • получение обновлений (измен ений) 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 текущи х изменения: git diff • сохранение текущих изменений: – добавить все изменённые и/или созданные файлы и/или каталоги: gi t 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 • сохр анение добавленных изменений: – сохранить все добавленные изменения и все изменённые файлы: git commit -am 'Описание ком мита' – сохранить добавленные изменения с внесением комментария через встроенный редактор git commit • создание новой ве тки, базирующейся на текущей: git checkout -b имя_ветки переключение на некоторую ветку: git checkout имя_ветки (при п ереключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой) • отправка измен ений 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 – п ринудительное удаление локальной ветки: git branch -D имя_ветки – удаление ветки с центрального репозитория: git push or igin :имя_ветки

8. Приведите примеры использования при работе с локальным и удалённым репозиториями.

git push –all (push origin master/любой branch)

9. Что такое и зачем могут быть нужны ветви (branches)?

Ветвление («ветка», branch) — один из параллельных участков истории в одном хранилище, исходящих из одной версии (точки ветвления). [3]Обычно есть

главная ветка (master), или ствол (trunk). Между ветками, то есть их концами, возможно с лияние. Используются для разработки новых функций.

10. Как и зачем можно игнорировать некоторые файлы при commit?

Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репоз иторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно пропис ать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов. # Выводы

В результате выполнения лаюораторной работы я приобрел навыки работы с гит, научился созданию репозиториев, gpg и ssh ключей, настроил каталог курса и авторизовался в gh.

# Список литературы