Отчёт по лабораторной работе №2

дисциплина: Операционные системы

Быстров Глеб Андреевич

Содержание

# Цель работы

В данной лабораторной работе мне будет необходимо научиться оформлять отчёты с помощью легковесного языка разметки Markdown.

# Задание

Сделать отчёт по предыдущей лабораторной работе в формате Markdown. В качестве отчёта предоставить отчёты в 3 форматах: pdf, docx и md.

# Выполнение лабораторной работы

1. Создал учётную запись на платформе https://github.com. (рис. 1)

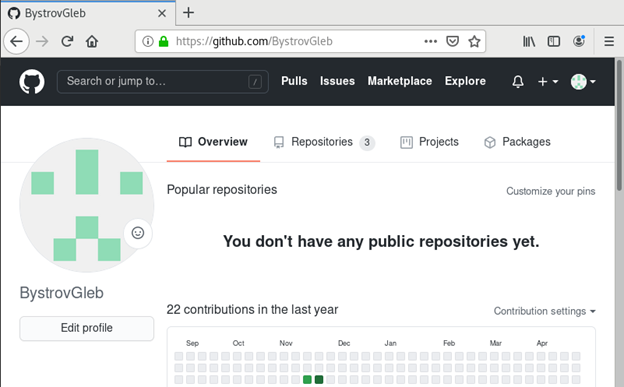


Figure 1: Учётная запись

1. Сгенерировал пару ключей (приватный и открытый). (рис. 2)

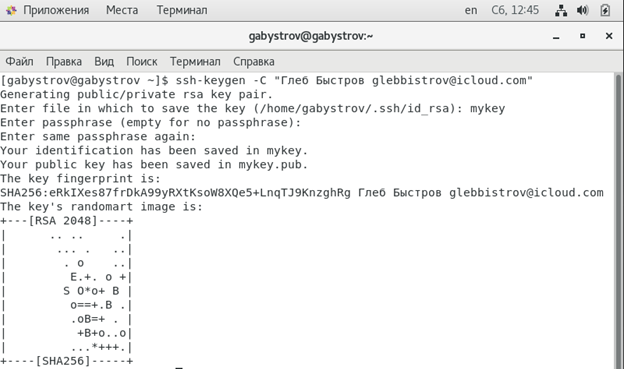


Figure 2: Генерация ключа

1. Добавил ключ на сайте. (рис. 3)

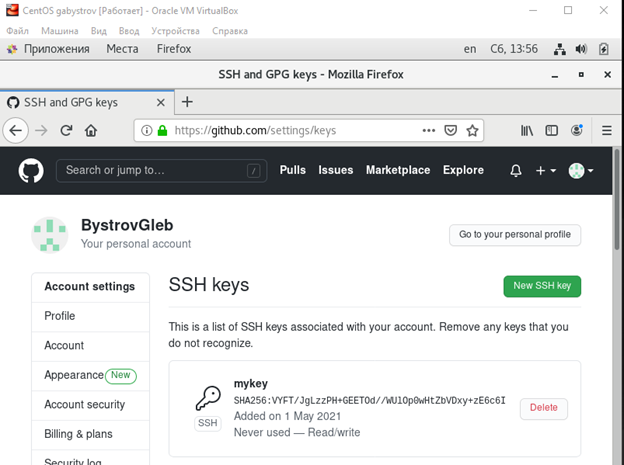


Figure 3: Добавление ключа

1. Создал структуру каталога лабораторных работ. (рис. 4)

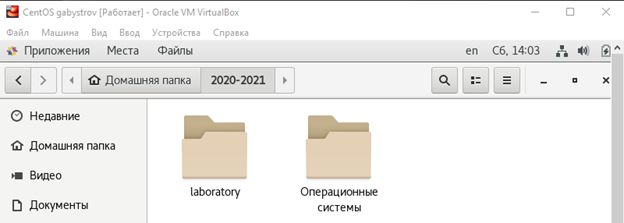


Figure 4: Структура каталога

1. Создал репозиторий на GitHub. (рис. 5)

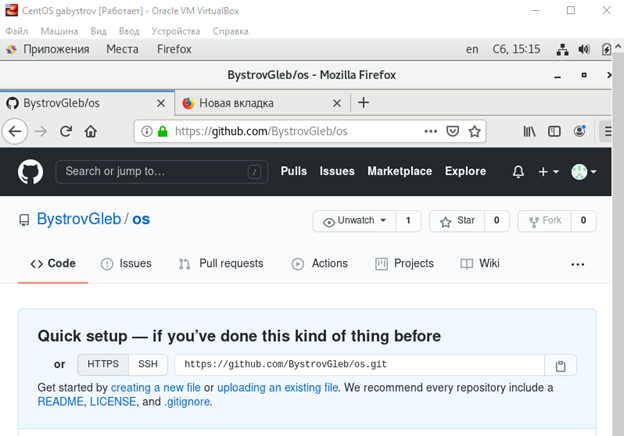


Figure 5: Репозиторий на GitHub

1. Подключил репозиторий к GitHub. (рис. 6)



Figure 6: Подключение репозитория

1. Добавил файл лицензии. (рис. 7)



Figure 7: Добавление файла лицензии

1. Добавил шаблон игнорируемых файлов. (рис. 8)

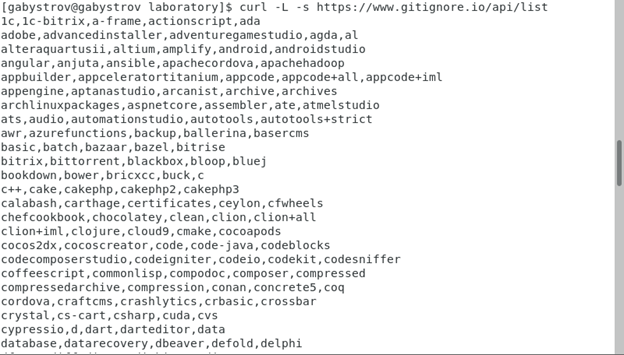


Figure 8: Шаблон игнорируемых файлов

1. Скачал шаблон для С. (рис. 9)

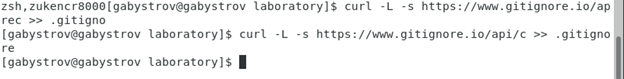


Figure 9: Шаблон для С

1. Добавил новые файлы. (рис. 10)

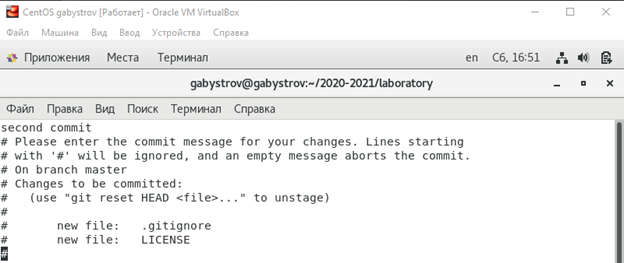


Figure 10: Новые файлы

1. Скачал git-flow. (рис. 11)

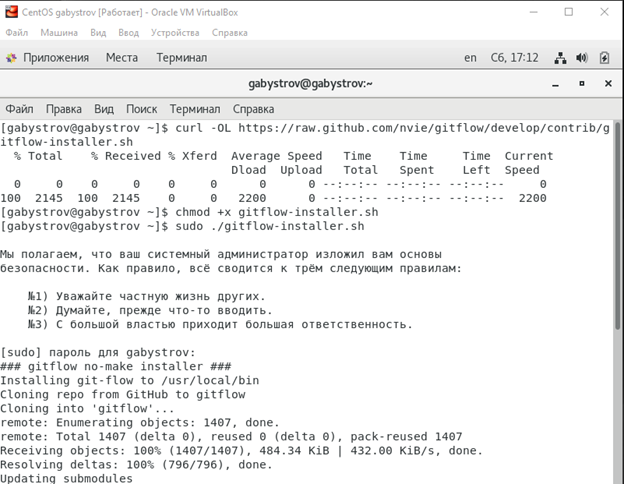


Figure 11: git-flow

1. Провёл работу с конфигурацией git-flow. (рис. 12)

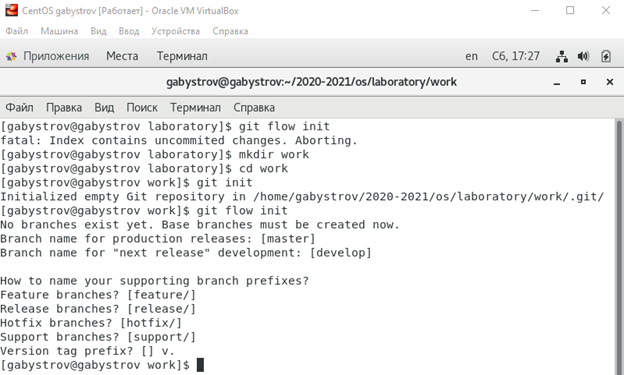


Figure 12: Конфигурация git-flow

1. Создал релиз с версией. (рис. 13)

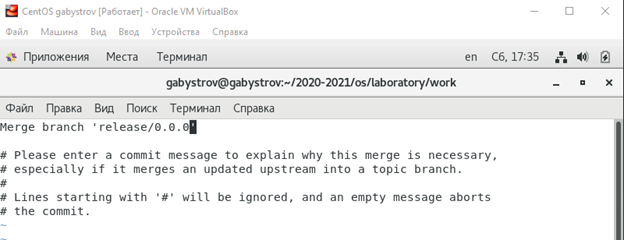


Figure 13: Релиз с версией

1. Файлы успешно созданы и загружены. (рис. 14)

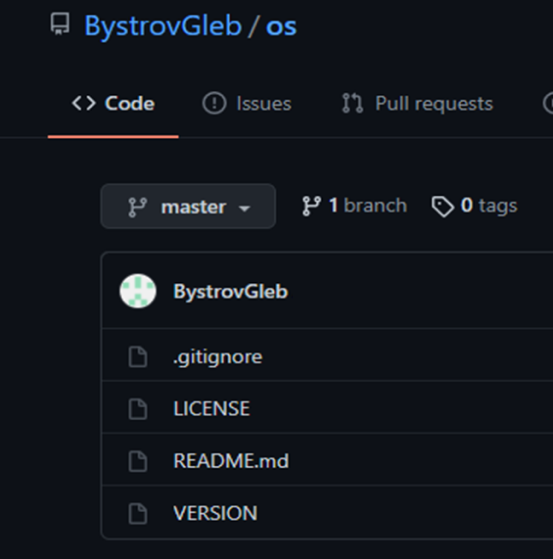


Figure 14: Репозиторий

# Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Система контроля версий —  это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии. Программистам, дизайнерам, разработчикам и другим специалистам будет удобно сохранять разные версии проектов, чтобы легко к ним возвращаться при необходимости. Благодаря системе контроля версий несколько участников могут работать с файлами и смотреть изменения каждого участника.
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

* Хранилище – традиционные системы управления версиями используют централизованную модель, когда имеется единое хранилище документов, управляемое специальным сервером, который и выполняет бо́льшую часть функций по управлению версиями.
* commit – завершив очередной этап работы над заданием, разработчик фиксирует свои изменения, передавая их на сервер.
* история – вся история изменения документов хранится на каждом компьютере, в локальном хранилище, и при необходимости отдельные фрагменты истории локального хранилища синхронизируются с аналогичным хранилищем на другом компьютере.
* рабочая копия – обычно создаётся локальная копия документа, так называемая «рабочая копия».

1. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные системы контроля версий представляют собой приложения типа клиент-сервер, когда репозиторий проекта существует в единственном экземпляре и хранится на сервере. В децентрализованной системе у каждого есть свой полноценный репозиторий. Децентрализованные системы были созданы для обмена изменениями. При использовании децентрализованной системы нет какой-то жестко заданной структуры репозиториев с центральным сервером.
2. Опишите действия с VCS при единоличной работе с хранилищем. Первым действием, которое должен выполнить разработчик, является извлечение рабочей копии проекта или той его части, с которой предстоит работать. Это действие выполняется с помощью команды извлечения версии (обычно checkout или clone). Разработчик задаёт версию, которая должна быть скопирована, по умолчанию обычно копируется последняя (или выбранная администратором в качестве основной) версия. По команде извлечения устанавливается соединение с сервером, и проект (или его часть — один из каталогов с подкаталогами) в виде дерева каталогов и файлов копируется на компьютер разработчика. Работая с проектом, разработчик изменяет только файлы основной рабочей копии. Вторая локальная копия хранится в качестве эталона, позволяя в любой момент без обращения к серверу определить, какие изменения внесены в конкретный файл или проект.
3. Опишите порядок работы с общим хранилищем VCS. Делать мелкие исправления в проекте можно путём непосредственной правки рабочей копии и последующей фиксации изменений прямо в главной ветви (в стволе) на сервере. Для изменений обычной практикой является создание ветвей (branch), то есть «отпочковывание» от ствола в какой-то версии нового варианта проекта или его части, разработка в котором ведётся параллельно с изменениями в основной версии. Ветвь создаётся специальной командой. Рабочая копия ветви может быть создана заново обычным образом (командой извлечения рабочей копии, с указанием адреса или идентификатора ветви), либо путём переключения имеющейся рабочей копии на заданную ветвь.
4. Каковы основные задачи, решаемые инструментальным средством git? Система спроектирована как набор программ, специально разработанных с учётом их использования в сценариях. Это позволяет удобно создавать специализированные системы контроля версий на базе Git или пользовательские интерфейсы. Git поддерживает быстрое разделение и слияние версий, включает инструменты для визуализации и навигации по нелинейной истории разработки. Предоставляет каждому разработчику локальную копию всей истории разработки, изменения копируются из одного репозитория в другой. Удалённый доступ к репозиториям Git обеспечивается сервером.
5. Назовите и дайте краткую характеристику командам git. Наиболее часто используемые команды 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 – добавить все изменённые и/или созданные файлы и/или каталоги: git add .
6. Приведите примеры использования при работе с локальным и удалённым репозиториями. Локальный репозиторий — она же директория “.git”. В ней хранятся коммиты и другие объекты. Удаленный репозиторий – репозиторий который считается общим, в который вы можете передать свои коммиты из локального репозитория, что бы остальные могли их увидеть.
7. Что такое и зачем могут быть нужны ветви (branches)? Ветки нужны для того, чтобы участники могли вести совместную работу над проектом и не мешать друг другу при этом. При создании проекта, Git создает базовую ветку. Она называется master веткой. Она считается центральной веткой, т.е. в ней содержится основной код приложения.
8. Как и зачем можно игнорировать некоторые файлы при commit? Часто в рабочей директории появляются файлы, которые нужно игнорировать. Зачастую, у имеется группа файлов, которые вы не только не хотите автоматически добавлять в репозиторий, но и видеть в списках не отслеживаемых. К таким файлам обычно относятся автоматически генерируемые файлы. В таком случае, вы можете создать файл. gitignore с перечислением шаблонов соответствующих таким файлам. Это защитит вас от случайного добавления в репозиторий файлов, которых вы там видеть не хотите.

# Выводы

В данной лабораторной работе мне успешно удалось приобрести практические навыки применения средств контроля версий. Также я смог научиться работать с системой контроля версий Git с помощью командной строки.