РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

ОТЧЕТ

по лабораторной работе № 2

дисциплина: Операционные системы

Студент: Николаев Дмитрий Иванович

Группа: НКНбд-01-20

**МОСКВА**

2021 г.

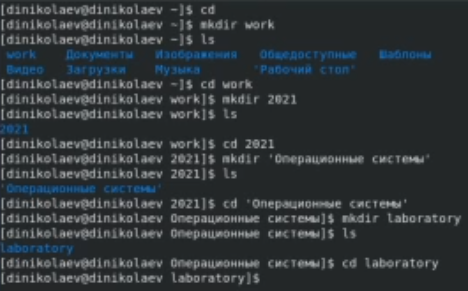
# Цель работы:

Изучить идеологию и применение средств контроля версий. Научиться использовать систему контроля версии в своих проектах.

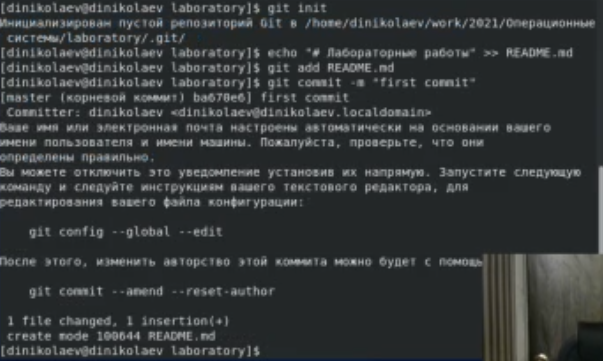
**Ход работы:**

Настроил систему git на своём компьютере.

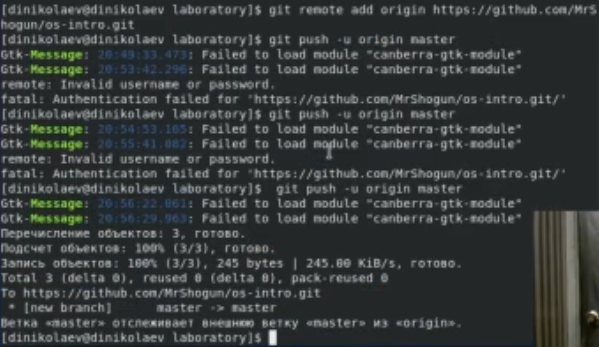
Создание структуры каталога (М2)



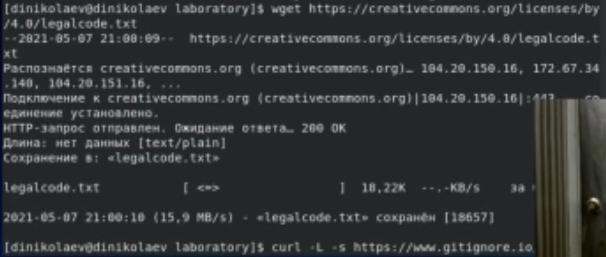
Создал пустой репозиторий на github и заготовку README, после сделал первый коммит



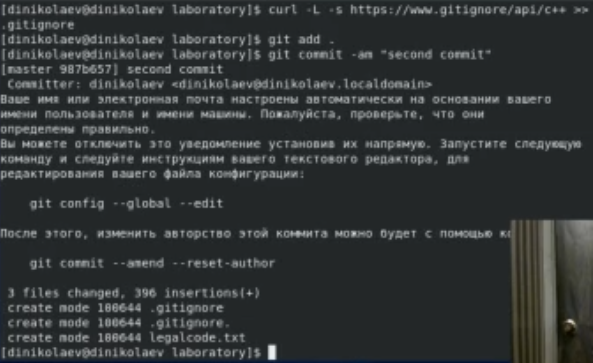
Выложил первый коммит на github



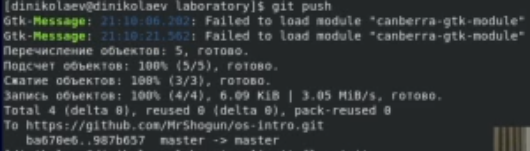
Добавил файл лицензии и шаблон игнорируемых файлов



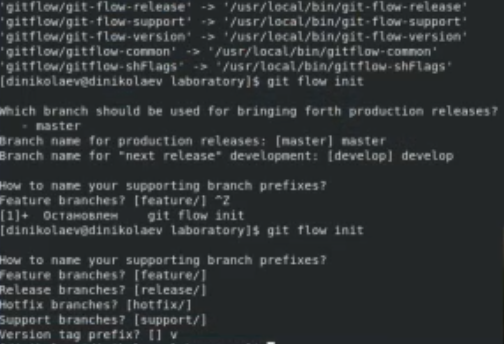
Добавил шаблон C++, добавил новые файлы, сделал второй коммит



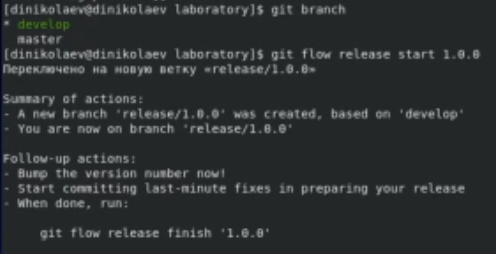
Отправил на github



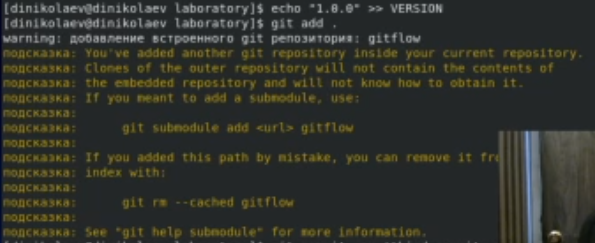
Инициализировал git-flow (настроил конфигурацию)



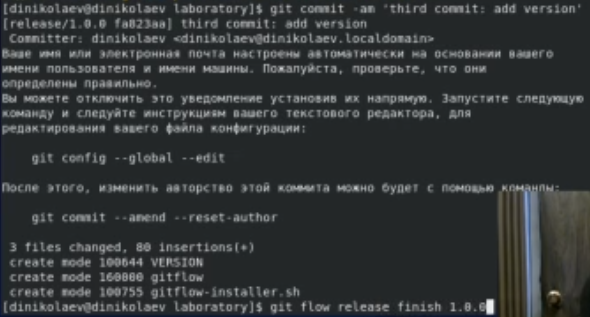
Проверил, что я на ветке develop и создал релиз версии 1.0.0



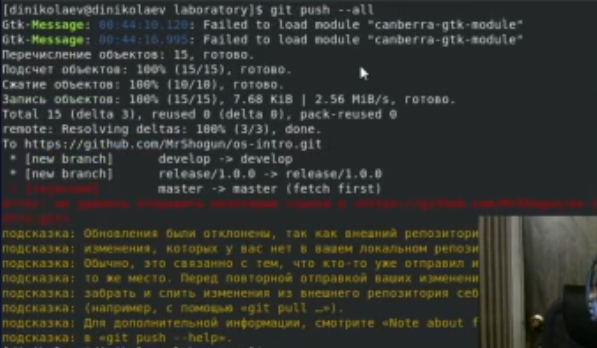
Записал версию и добавил файлы (на эту ветку)

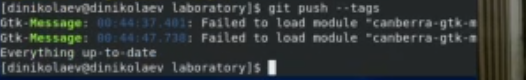


Добавил в индекс и залил релизную ветку в основную



Запушил все ветки на github





**Контрольные вопросы:**

* 1. Системы контроля версий (Version Control System, VCS) - это программное обеспечение, которое используется для облегчения работы с изменяющейся информацией, обычно - с проектами. Чаще всего применяются при работе нескольких человек над одним проектом.
  2. Хранилище в системе контроля версий - это удаленный репозиторий, где хранятся все файлы проекта.

Commit - это фиксация изменений перед загрузкой файлов в систему контроля версий.

История хранит все изменения проекта, и в случае необходимости позволяет откатиться к нужному месту.

Рабочая копия - это копия проекта на компьютере разработчика. Если другой член команды изменил проект, необходимо загрузить новую версию проекта себе на компьютер.

* 1. Централизованные системы контроля версий хранят данные о проекте на едином сервере, и в случае его отключения, доступ к данным будет утерян (Perforce). В децентрализованных системах у каждого из участников проекта на компьютере хранится полная копия проекта. Среди централизованных VCS наиболее известны CVS, Subversion, а среди децентрализованных — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.
  2. Сначала надо создать и подключить удаленный репозиторий. Затем, так как никто кроме пользователя не изменяет проект, то по мере изменения проекта пушить изменения на сервер, из-за чего нет необходимости загружать изменения.
  3. Каждый раз перед работой с проектом нужно загрузить актуальную версию проекта на свой компьютер, и только потом работать над ним. После работы необходимо закоммитить изменения и запушить на сервер.
  4. Основные задачи: упрощение обмена информацией, ускорение разработки, устранение ошибок и недоработок во время разработки (защита от случайных исправлений и удалений).
  5. 1. git init - инициализирует локальный репозиторий
     + 1. git add - добавляет файлы в репозиторий
       2. git commit - коммит версии (сохранение изменений)
       3. git pull - загружает актуальную версию проекта
       4. git push - отправляет измененный проект на сервер
       5. git checkout - позволяет переключаться между ветками
       6. git status - текущий статус проекта
       7. git branch - просмотр доступных веток
       8. git remote add - добавление удаленного репозитория
       9. git diff - просмотр текущих изменений
       10. git rm - удаление файла и/или каталог из индекса репозитория.
  6. git branch,(🡪) git init,(🡪) git remote add,(🡪) git add,(🡪) git commit,(🡪) git push (git push --all).
  7. Ветки нужны для разделения разработки. Например, можно разработать часть проекта в отдельной ветке, а не в основном проекте. В случае успешной разработки, эту ветку сливают с основной. Так убираются риски багов, ошибок, а также утечки данных.
  8. Есть временные и системные файлы, которые засоряют проект и не нужны. Игнорируемые файлы – это, как правило, специфичные для платформы файлы или автоматически созданные файлы из систем сборки. Путь к ним можно добавить в файл .gitignore, тогда они не будут добавляться в проект.

**Вывод:**

Выполняя данную лабораторную работу, я изучил идеологию и применение средств контроля версий, создал аккаунт и репозиторий на github, выполнил важнейшие команды: инициализирование локального репозитория, подключение удаленных репозиториев, добавление и удаление нужных файлов, синхронизирование данных. Также я научился использовать Git Flow, который очень сильно упрощает разработку проектов и навигацию между ветками.