Балтийский государственный технический университет «ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра И5  
«Информационные системы и программная инженерия»

**Лабораторная работа № 5** по дисциплине «Компьютерный практикум»на тему **«Особенности использования систем контроля версий»**

Выполнил:

Студент Селюхов М. М.  
 Группа И582

**Преподаватель**:  
 Вальштейн К. В.

Санкт-Петербург  
2019

*Цель работы -* изучить особенности различных систем контроля версий.

Часть 1. Работа с локальным репозиторием:

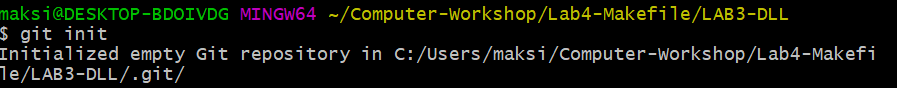
Для создания локального репозитория необходимо воспользоваться командой git init. 

Рисунок 5.1 – Результат работы команды git init

Далее в создаём файл .gitignore, располагая его в папке с проекта, и указываем файлы, которые должен игнорировать git, в нашем случае это файл сборки.

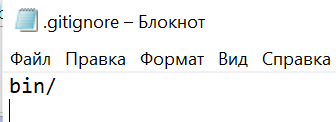


Рисунок 5.2 – Содержимое файла .gitignore

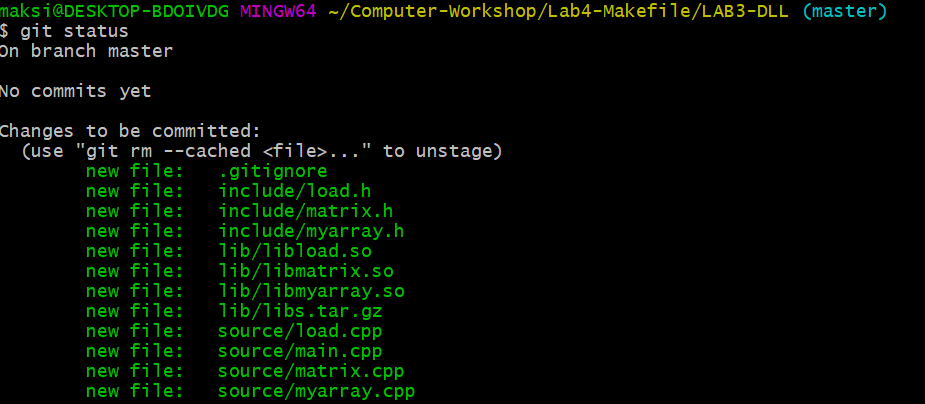
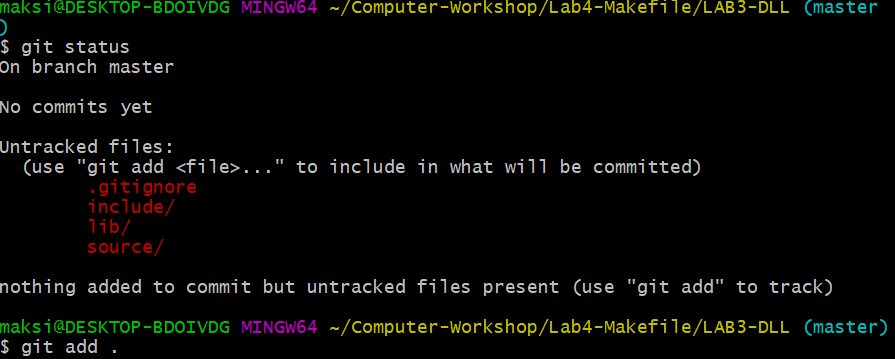
Чтобы добавить файлы в следующий коммит, нужно прописать команду git add, а для проверки состояния файлов воспользоваться командной git status. 

Рисунок 5.3 – Результат работы команд git status и git add

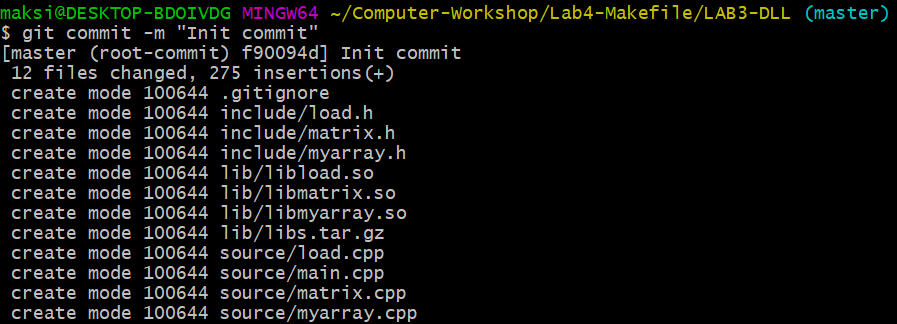
Снимок директории проекта происходит с помощью команды git commit. 

Рисунок 5.4 – Результат работы команды git commit

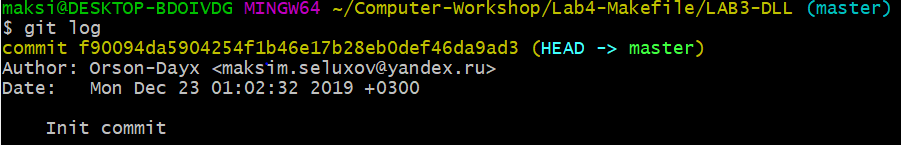
Посмотреть историю коммитов можно с помощью команды git log. 

Рисунок 5.5 – Результат работы команды git log

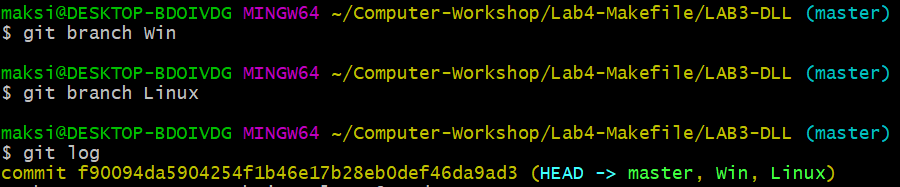
Создание ветки происходит набором команды git branch \*имя ветки\*.

Рисунок 5.6 – Результат работы команды git branch

Переход к ветке осуществляется командой git checkout. Перейдя на ветку Win удаляем поддержку ОС семейства Linux из файлов load.h и load.cpp

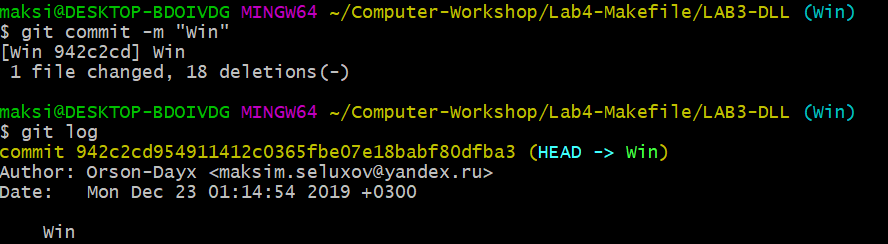
Сделаем коммит проделанной работы. 

Рисунок 5.7 – Результат работы команды git commit на ветке Win

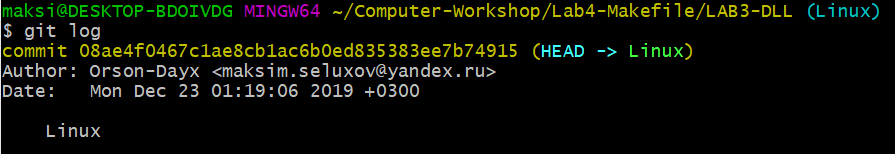
Проделаем аналогичную работу на ветке Linux. 

Рисунок 5.8 – Результат работы команды git commit на ветке Linux

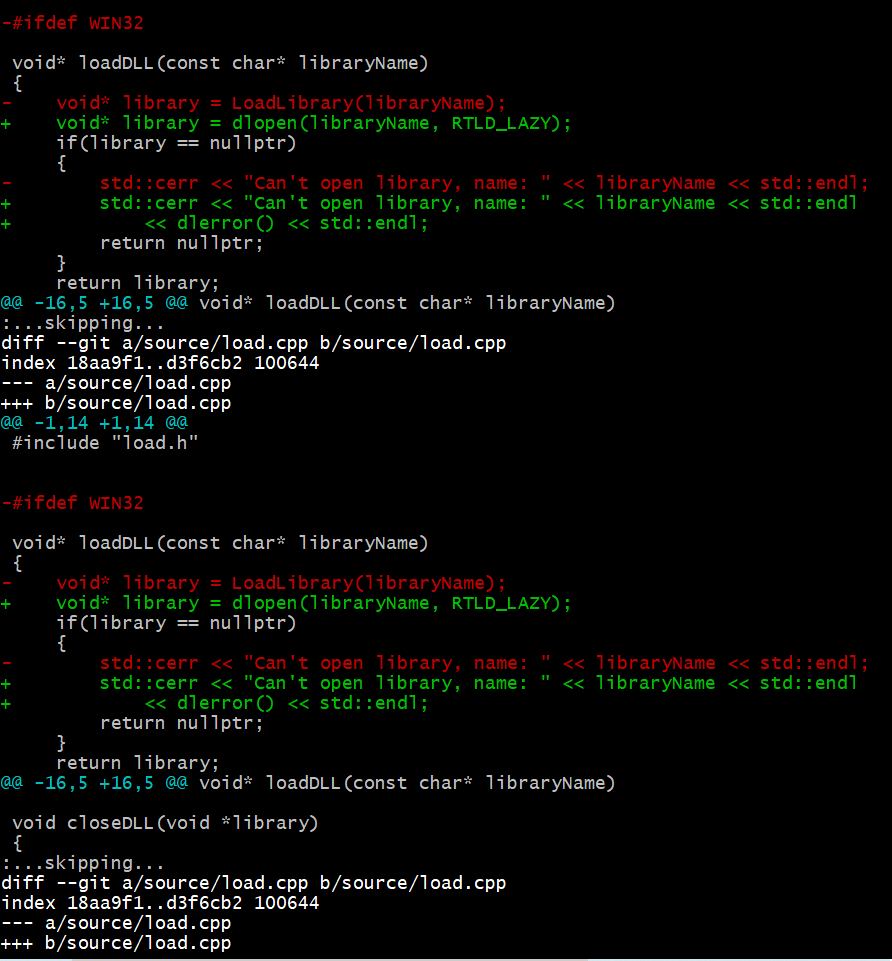
Сравнивать различия на ветках будем с помощью команды git diff. 

Рисунок 5.9 – Результат работы команды git diff Win Linux

Для восстановления изначальной версии программы достаточно перейти на ветку

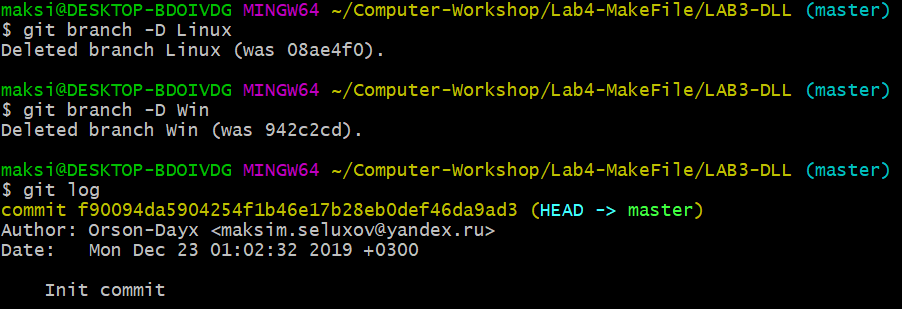
master и удалить созданные ветки. 

Рисунок 5.10 – Восстановление изначальной версии

Часть 2. Работа с глобальным репозиторием.

Основные различия CVS и SVN:

1. SVN создавалась как альтернатива CVS с целью исправить недостатки CVS;
2. Как и CVS, SVN это бесплатная система контроля версий с открытым исходным кодом. С той лишь разницей, что распространяется под лицензией Apache, а не под Открытым лицензионным соглашением GNU.
3. Для сохранения целостности базы данных SVN использует атомарные операции. При появлении новой версии к финальному продукту применяются либо все исправления, либо ни одно из них. Таким образом, код защищают от хаотичных частичных правок, которые не согласуются между собой и вызывают ошибки.
4. В то время как в CVS операции с ветками кода дорогостоящие и не предусмотрены архитектурой системы, SVN создана как раз для этого. То есть, для более крупных проектов с ветвлением кода и многими направлениями разработки.

Основные различия SVN и Git:

1. Git разрабатывалась главным образом под Linux, то именно в этой ОС она работает быстрее всего
2. B Git каждая копия рабочей папки .git содержит всю историю разработки
3. Значительное увеличение быстродействия
4. Отсутствует потребность в центральном сервере. Каждый пользователь, который имеет копию проекта является носителем всех версий проекта, за счет чего мала вероятность потери проекта
5. Гибкая работа с файлами проекта и возможность не добавлять их в версию проекта
6. Не копирует файл в новую версию, если он не был изменен, а ссылается на предыдущую версию с последним изменением файла

Описание процесса регистрации и работы с одним из рассмотренных глобальных репозиториев (GitHub):

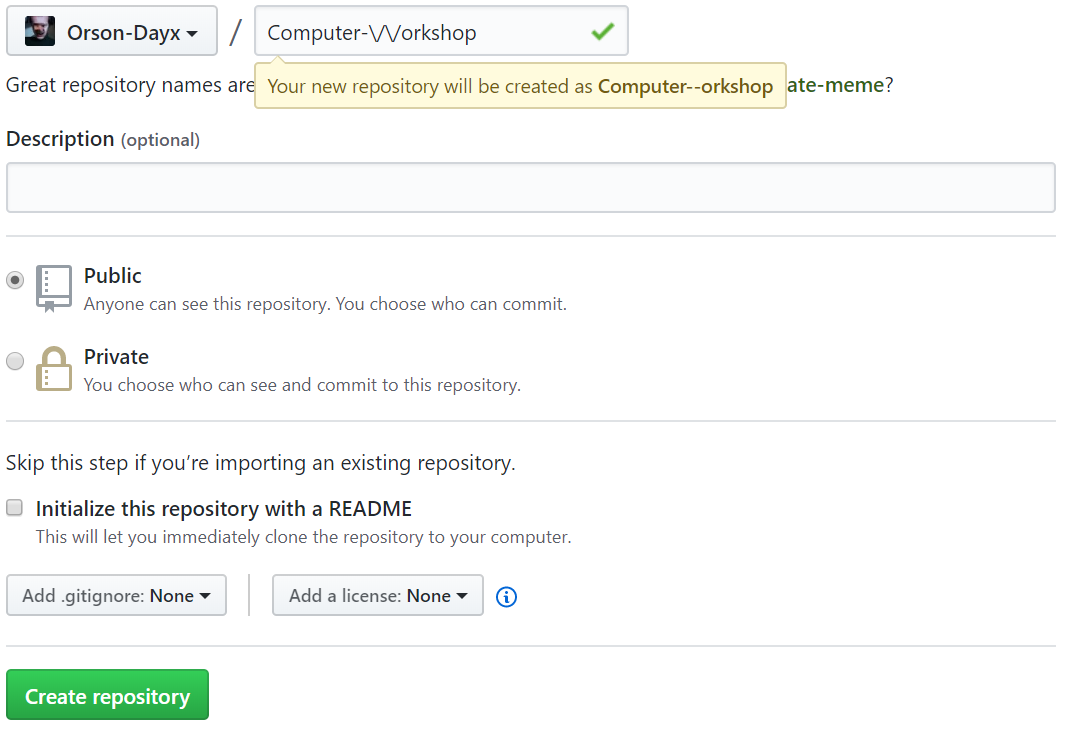
1. Если мы хотим иметь возможность сохранения проекта в интернете, создадим репозиторий на Github. Для начала нужно зарегистрироваться на сайте github.com.
2. После регистрации нажимаем кнопочку "+" и вводим название репозитория.
3. Выбираем тип Public (репозиторий всегда Public для бесплатной версии) и нажимаем Create.
4. В результате мы создали репозиторий на сайте Github. На экране мы увидим инструкцию, как соединить наш локальный репозиторий со вновь созданным.   
   

Рисунок 5.11 – Создание удаленного репозитория

1. Связать локальный репозиторий с удалённым можно с помощью команды

git remote add origin https://github.com/alohaeee/Computer-Workshop

1. Добавляем файлы в созданный репозиторий c помощью git push origin
2. Для получения обновлений с удаленного репозитория воспользуйтесь командой:

git pull

1. Если вы изменили ваши локальные файлы, то команда *git pull* выдаст ошибку. Если вы уверены, что хотите перезаписать локальные файлы, файлами из удаленного репозитория то выполните команды:

git fetch –all

git reset –hard github/master

1. В случае, если в удаленном репозитории лежат файлы с версией более новой, чем у вас в локальном, то команда git push выдаст ошибку. Если вы уверены, что хотите перезаписать файлы в удаленном репозитории несмотря на конфликт версий, то воспользуйтесь командой:

git push –f

1. Иногда возникает необходимость отложить ваши текущие изменения и поработать над файлами, которые находятся в удаленном репозитории. Для этого отложите текущие изменения командой:

git stash

1. После выполнения этой команды ваша локальная директория будет содержать файлы такие же, как и при последнем коммите. Вы можете загрузить новые файлы из удаленного репозитория командой *git pull* и после этого вернуть ваши изменения которые вы отложили командой: git stash pop
2. Клонирование существующего удалённого репозитория происходит с помощью команды git clone.