МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Казанский (Приволжский) федеральный университет»
Елабужский институт (филиал) КФУ

УТВЕРЖДАЮ
Директор
Елабужского института КФУ
_____ Е.Е. Мерзон
«22» 05 2024 г.
МП

Программа дисциплины (модуля)
Системы контроля версий программного обеспечения

Направление подготовки/специальность: 23.03.01 Технология транспортных процессов
Направленность (профиль) подготовки: Проектирование и управление интеллектуальными транспортными системами
Квалификация выпускника: бакалавр
Форма обучения: заочная
Язык обучения: русский
Год начала обучения по образовательной программе: 2024

=

Содержание

Программу дисциплины разработал(а)(и) старший преподаватель, б/с Галимуллина Э.З., Бочкарева А.В. (Кафедра математики и прикладной информатики).

1. Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с планируемыми результатами освоения ОПОП ВО

Обучающийся, освоивший дисциплину (модуль), должен обладать следующими компетенциями:

Шифр компетенции	Расшифровка приобретаемой компетенции
ПК-4	Способен разрабатывать алгоритмы и программы, пригодные для практического применения
ПК-4.1.	Знать технологии разработки алгоритмов и программ, пригодных для практического применения
ПК-4.2.	Уметь разрабатывать алгоритмы и программы, пригодные для практического применения
ПК-4.3.	Владеть способностью разрабатывать алгоритмы и программы, пригодные для практического применения

Обучающийся, освоивший дисциплину (модуль):

Должен знать:

- знать технологии разработки алгоритмов и программ, пригодных для практического применения в будущей профессиональной деятельности

Должен уметь:

- уметь разрабатывать под руководством наставника алгоритмы и программы, пригодные для практического применения в будущей профессиональной деятельности.

Должен владеть:

- владеть способностью разрабатывать под руководством наставника алгоритмы и программы, пригодные для практического применения в будущей профессиональной деятельности

2. Место дисциплины (модуля) в структуре ОПОП ВО

Данная дисциплина (модуль) включена в раздел «Б1.В.04.03 Дисциплины (модули)» основной профессиональной образовательной программы 23.03.01 Технология транспортных процессов и относится к части, формируемой участниками образовательных отношений.
Осваивается на 4 курсе в 8 семестре.

3. Объем дисциплины (модуля) в зачетных единицах с указанием количества часов, выделенных на контактную работу обучающихся с преподавателем (по видам учебных занятий) и на самостоятельную работу обучающихся

Общая трудоемкость дисциплины составляет 2 зачетных(ые) единиц(ы) на 72 часа(ов).

Контактная работа - 10 часа(ов), в том числе лекции - 4 часа(ов), практические занятия - 0 часа(ов), лабораторные работы – 6 часа(ов), контроль самостоятельной работы - 0 часа(ов).

Самостоятельная работа - 58 часа(ов).

Контроль (зачёт / экзамен) – 4 часа(ов).

Форма промежуточного контроля дисциплины: зачет в 8 семестре.

4. Содержание дисциплины (модуля), структурированное по темам (разделам) с указанием отведенного на

них количества академических часов и видов учебных занятий

4.1 Структура и тематический план контактной и самостоятельной работы по дисциплине (модулю)

N	Разделы дисциплины / модуля	Семестр	Виды и часы контактной работы, их трудоемкость (в часах)			Самостоятельная работа
			Лекции	Практические занятия	Лабораторные работы	
1.	Тема 1. Введение в теоретические основы системы контроля версий программного обеспечения.	8	1	0	0	12
2.	Тема 2. Создание и использование репозитория системы контроля версий программного обеспечения.	8	1	0	2	12
3.	Тема 3. Обзор текущего состояния локального репозитория, коммит изменений.	8	0	0	1	12
4.	Тема 4. Создание и использование ветвления системы контроля версий программного обеспечения.	8	1	0	1	12
5.	Тема 5. Распределённая работа и использование системы контроля версий в окружениях.	8	1	0	2	10
	Итого		4	0	6	58

4.2 Содержание дисциплины (модуля)

Тема 1. Введение в теоретические основы системы контроля версий программного обеспечения.

Базовые понятия и основные направления системы контроля версий программного обеспечения. Установка и конфигурация систем контроля версий программного обеспечения в различных окружениях.

Тема 2. Создание и использование репозитория системы контроля версий программного обеспечения.

Базовая работа с системой контроля версий программного обеспечения. Понятийный аппарат команд создания, клонирования, сохранения директория в репозиторий. Дополнительные возможности команд системы контроля версий. Работа с удаленным репозиторием.

Тема 3. Обзор текущего состояния локального репозитория, коммит изменений.

Разработка пользовательских панелей и организация системы контроля версий программного обеспечения. Возможности создания многоуровневых репозиториев с использование локального директория. Команды отмены версий, клонирования, сохранения локального директория в репозиторий.

Тема 4. Создание и использование ветвления системы контроля версий программного обеспечения.

Понятия и основные определения ветвления. Команды извлечения данных сервера, переноса изменений с удаленного хранилища в локальный, соединения с удаленным репозиторием. Конфликты слияние и их решения. Удаленные ветки.

Тема 5. Распределённая работа и использование системы контроля версий в окружениях.

Понятия и основные определения инструментов жизненного цикла DevOps. Управление публичными и приватными репозиториями. Управление пользователями и группами, правами доступа к репозиторию. Отслеживание ошибок, деплой, анализ кода. Интеграция с разными CI-системами CI (Jenkins и т. п.), организация самостоятельного процесса CI посредством встроенных средств.

5. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине (модулю)

Самостоятельная работа обучающихся выполняется по заданию и при методическом руководстве преподавателя, но без его непосредственного участия. Самостоятельная работа подразделяется на самостоятельную работу на аудиторных занятиях и на внеаудиторную самостоятельную работу. Самостоятельная работа обучающихся включает как полностью самостоятельное освоение отдельных тем (разделов) дисциплины, так и проработку тем

(разделов), осваиваемых во время аудиторной работы. Во время самостоятельной работы обучающиеся читают и конспектируют учебную, научную и справочную литературу, выполняют задания, направленные на закрепление знаний и отработку умений и навыков, готовятся к текущему и промежуточному контролю по дисциплине.

Организация самостоятельной работы обучающихся регламентируется нормативными документами, учебно-методической литературой и электронными образовательными ресурсами, включая:

Порядок организации и осуществления образовательной деятельности по образовательным программам высшего образования - программам бакалавриата, программам специалитета, программам магистратуры (утвержден приказом Министерства образования и науки Российской Федерации от 5 апреля 2017 года №301)

Письмо Министерства образования Российской Федерации №14-55-996ин/15 от 27 ноября 2002 г. "Об активизации самостоятельной работы студентов высших учебных заведений"

Устав федерального государственного автономного образовательного учреждения "Казанский (Приволжский) федеральный университет"

Правила внутреннего распорядка федерального государственного автономного образовательного учреждения высшего профессионального образования "Казанский (Приволжский) федеральный университет"

Локальные нормативные акты Казанского (Приволжского) федерального университета

6. Фонд оценочных средств по дисциплине (модулю)

Фонд оценочных средств по дисциплине (модулю) включает оценочные материалы, направленные на проверку освоения компетенций, в том числе знаний, умений и навыков. Фонд оценочных средств включает оценочные средства текущего контроля и оценочные средства промежуточной аттестации.

В фонде оценочных средств содержится следующая информация:

- соответствие компетенций планируемым результатам обучения по дисциплине (модулю);
- критерии оценивания сформированности компетенций;
- механизм формирования оценки по дисциплине (модулю);
- описание порядка применения и процедуры оценивания для каждого оценочного средства;
- критерии оценивания для каждого оценочного средства;
- содержание оценочных средств, включая требования, предъявляемые к действиям обучающихся, демонстрируемым результатам, задания различных типов.

Фонд оценочных средств по дисциплине находится в Приложении 1 к программе дисциплины (модулю).

7. Перечень литературы, необходимой для освоения дисциплины (модуля)

Освоение дисциплины (модуля) предполагает изучение учебной литературы. Литература может быть доступна обучающимся в одном из двух вариантов (либо в обоих из них):
- в электронном виде - через электронные библиотечные системы на основании заключенных КФУ договоров с правообладателями;
- в печатном виде - в Научной библиотеке Елабужского института КФУ. Обучающиеся получают учебную литературу на абонементе по читательским билетам в соответствии с правилами пользования Научной библиотекой.
Электронные издания доступны дистанционно из любой точки при введении обучающимся своего логина и пароля от личного кабинета в системе "Электронный университет". При использовании печатных изданий библиотечный фонд должен быть укомплектован ими из расчета не менее 0,25 экземпляра на каждого обучающегося из числа лиц, одновременно осваивающих данную дисциплину
Перечень литературы, необходимой для освоения дисциплины (модуля), находится в Приложении 2 к рабочей программе дисциплины. Он подлежит обновлению при изменении условий договоров КФУ с правообладателями электронных изданий и при изменении комплектования фондов Научной библиотеки Елабужского института КФУ.

8. Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины (модуля)

Системы контроля версий программного обеспечения - https://devpractice.ru/git-for-beginners-part-1-what-is-vcs/

Системы контроля версий программного обеспечения - https://habr.com/ru/post/552872/

Руководство по использованию системы контроля версий программного обеспечения – https://git-scm.com/book/ru/v2

9. Методические указания для обучающихся по освоению дисциплины (модуля)

Вид работ	Методические рекомендации
лекции	Лекционные занятия проводятся с использованием интерактивных технологий и предполагают активное участие студентов. Для подготовки к занятиям рекомендуется выделять в материале проблемные вопросы, затрагиваемые преподавателем в лекции, и группировать информацию вокруг них. Желательно выделять в используемой литературе постановки вопросов, на которые разными авторам могут быть даны различные ответы. На основании постановки таких вопросов следует собирать аргументы в пользу различных вариантов решения поставленных проблем.
лабораторные работы	Лабораторные занятия - это одна из разновидностей практического занятия, являющаяся эффективной формой учебных занятий в организации высшего образования. Лабораторные занятия имеют выраженную специфику в зависимости от учебной дисциплины, углубляют и закрепляют теоретические знания. На этих занятиях студенты осваивают конкретные методы изучения дисциплины, обучаются экспериментальным способам анализа, умению работать с приборами и современным оборудованием. Лабораторные занятия дают наглядное представление об изучаемых явлениях и процессах, студенты осваивают постановку и ведение эксперимента, учатся умению наблюдать, оценивать полученные результаты, делать выводы и обобщения. Отчёт по итогам выполненных лабораторных работ выполняется на листах белой бумаги формата А4 в печатном или рукописном виде. При оформлении отчёта используется сквозная нумерация страниц, считая титульный лист первой страницей. Номер страницы на титульном листе не ставится. Номера страницы ставятся по центру вверху. При оформлении отчёта в печатном виде желательно соблюдать следующие требования. Для заголовков: полужирный шрифт, 14 пт, центрированный. Для основного текста: нежирный шрифт, 14 пт, выравнивание по ширине. Во всех случаях тип шрифта - Times New Roman, отступ абзаца 1.25 см, полуторный междустрочный интервал. Поля: левое - 3 см, правое - 1 см, верхнее и нижнее - 2 см. Отчет должен содержать следующие элементы: 1) Титульный лист с обязательным указанием варианта; 2) Цель работы; 3) Задание; 4) Основная часть; 5) Вывод.
самостоя-тельная работа	Самостоятельная работа студентов по дидактической сути представляет собой комплекс условий обучения, организуемых преподавателем и направленных на самоподготовку учащихся. Учебная деятельность протекает без непосредственного участия преподавателя и заключается в проработке лекционного материала, подготовке к лабораторным занятиям; изучении учебной литературы из основного и дополнительного списка.
зачет	Зачет является формой оценки качества освоения студентом образовательной программы по дисциплине. По результатам зачета студенту выставляется оценка "зачтено" или "не зачтено". Зачет может проводиться в форме устного опроса по билетам (вопросам) или без билетов, с предварительной подготовкой или без подготовки, по усмотрению кафедры. Преподаватель может проставить зачет без опроса или собеседования тем студентам, которые активно участвовали на лабораторных занятиях.

10. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень программного обеспечения и информационных справочных систем (при необходимости)

Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень программного обеспечения и информационных справочных систем, представлен в Приложении 3 к рабочей программе дисциплины (модуля).

11. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю)

Учебные аудитории № 61 (423600, Республика Татарстан, г. Елабуга, ул. Казанская, д. 89) для проведения занятий лекционного типа, занятий семинарского типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, помещение для самостоятельной работы. Комплект мебели (посадочных мест) 29 шт. Комплект мебели (посадочных мест) для преподавателя 1 шт. Компьютерный класс: Компьютеры intel core i5 15 шт. Мониторы ViewSonic 22d 15 шт. Проектор EPSON EB-535W 1 шт. Интерактивная доска IQBoard DVT TN082 1 шт. Трибуна 1 шт. Кондиционер 1 шт. Настенные полки 6 шт. Шкаф двухстворчатый с полками 1 шт. Веб-камера 1 шт. Выход в Интернет, внутривузовская компьютерная сеть, доступ в электронную информационно-образовательную

среду. Набор учебно-наглядных пособий: комплект презентаций в электронном формате по преподаваемой дисциплине 3-5 шт.

12. Средства адаптации преподавания дисциплины к потребностям обучающихся инвалидов и лиц с ограниченными возможностями здоровья

При необходимости в образовательном процессе применяются следующие методы и технологии, облегчающие восприятие информации обучающимися инвалидами и лицами с ограниченными возможностями здоровья:

- создание текстовой версии любого нетекстового контента для его возможного преобразования в альтернативные формы, удобные для различных пользователей;

- создание контента, который можно представить в различных видах без потери данных или структуры, предусмотреть возможность масштабирования текста и изображений без потери качества, предусмотреть доступность управления контентом с клавиатуры;

- создание возможностей для обучающихся воспринимать одну и ту же информацию из разных источников - например, так, чтобы лица с нарушениями слуха получали информацию визуально, с нарушениями зрения - аудиально;

- применение программных средств, обеспечивающих возможность освоения навыков и умений, формируемых дисциплиной, за счёт альтернативных способов, в том числе виртуальных лабораторий и симуляционных технологий;

- применение дистанционных образовательных технологий для передачи информации, организации различных форм интерактивной контактной работы обучающегося с преподавателем, в том числе вебинаров, которые могут быть использованы для проведения виртуальных лекций с возможностью взаимодействия всех участников дистанционного обучения, проведения семинаров, выступления с докладами и защиты выполненных работ, проведения тренингов, организации коллективной работы;

- применение дистанционных образовательных технологий для организации форм текущего и промежуточного контроля;

- увеличение продолжительности сдачи обучающимся инвалидом или лицом с ограниченными возможностями здоровья форм промежуточной аттестации по отношению к установленной продолжительности их сдачи:

- продолжительности сдачи зачёта или экзамена, проводимого в письменной форме, - не более чем на 90 минут;

- продолжительности подготовки обучающегося к ответу на зачёте или экзамене, проводимом в устной форме, - не более чем на 20 минут;

- продолжительности выступления обучающегося при защите курсовой работы - не более чем на 15 минут.

Программа составлена в соответствии с требованиями ФГОС ВО и учебным планом по направлению 23.03.01 "Технология транспортных процессов" и профилю подготовки " Проектирование и управление интеллектуальными транспортными системами".

Приложение №1
к рабочей программе дисциплины (модуля)
Б1.В.04.03 Системы контроля версий программного обеспечения

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
"Казанский (Приволжский) федеральный университет"
Елабужский институт (филиал)

Фонд оценочных средств по дисциплине (модулю)
Б1.В.04.03 Системы контроля версий программного обеспечения

Направление подготовки: 23.03.01 Технология транспортных процессов
Профиль подготовки: Проектирование и управление интеллектуальными транспортными системами
Квалификация выпускника: бакалавр
Форма обучения: заочное
Язык обучения: русский
Год начала обучения по образовательной программе: 2024

СОДЕРЖАНИЕ

1. Соответствие компетенций планируемым результатам обучения по дисциплине (модулю)

Код и наименование компетенции	Индикаторы достижения компетенций для данной дисциплины	Оценочные средства текущего контроля и промежуточной аттестации
ПК-4 - Способен проектировать и управлять IT-проектами, осуществлять тестирование компонентов информационных систем, в том числе интеллектуальных	Знать технологии разработки алгоритмов и программ, пригодных для практического применения в будущей профессиональной деятельности Уметь разрабатывать под руководством наставника алгоритмы и программы, пригодные для практического применения в будущей профессиональной деятельности Владеть способностью разрабатывать под руководством наставника алгоритмы и программы, пригодные для практического применения в будущей профессиональной деятельности	**Текущий контроль:** Лабораторные работы по темам Тема 1. Введение в теоретические основы системы контроля версий программного обеспечения. Тема 2. Создание и использование репозитория системы контроля версий программного обеспечения. Тема 3. Обзор текущего состояния локального репозитория, коммит изменений. Тема 4. Создание и использование ветвления системы контроля версий программного обеспечения. Тема 6. Распределенная работа и использование системы контроля версий в окружениях. Практические здания по темам: Лабораторные работы по темам Тема 1. Введение в теоретические основы системы контроля версий программного обеспечения. Тема 2. Создание и использование репозитория системы контроля версий программного обеспечения. Тема 3. Обзор текущего состояния локального репозитория, коммит изменений. Тема 4. Создание и использование ветвления системы контроля версий программного обеспечения. Тема 6. Распределенная работа и использование системы контроля версий в окружениях. **Промежуточная аттестация:** *Зачет*

2. Критерии оценивания сформированности компетенций

Компетенция	Зачтено			Не зачтено
	Высокий уровень (отлично)	**Средний уровень (хорошо)**	**Низкий уровень (удовлетворительно)**	**Ниже порогового уровня (неудовлетворительно)**
ПК-4	Знает рациональные технологии разработки алгоритмов и программ,	Знает технологии разработки алгоритмов и программ, пригодных	Знает базовые технологии разработки типовых алгоритмов и	Не знает базовые технологии разработки типовых алгоритмов и

	пригодных для практического применения в будущей профессиональной деятельности, а также перечень основных команд для указания пользовательских настроек, добавления файлов в индекс, создания и работы с репозиторием, коммита изменений в файлах проекта, создания и использования ветвления, для работы с удаленным репозиторием.	для практического применения в будущей профессиональной деятельности, а также перечень основных команд для указания пользовательских настроек, добавления файлов в индекс, создания и работы с репозиторием, коммита изменений в файлах проекта, создания и использования ветвления.	программ, пригодных для решения конкретных практических задач а также перечень основных команд для указания пользовательских настроек, добавления файлов в индекс, создания и работы с репозиторием, коммита изменений в файлах проекта.	программ, пригодных для решения конкретных практических задач а также перечень основных команд для указания пользовательских настроек, добавления файлов в индекс, создания и работы с репозиторием, коммита изменений в файлах проекта.
	Умеет самостоятельно разрабатывать алгоритмы и программы, пригодные для практического применения в будущей профессиональной деятельности, а также практически применять команды для помещения изменений программного файла на удаленный репозиторий, Умеет создавать, изменять и удалять ветки репозитория и подключаться к удаленному репозиторию.	Умеет разрабатывать под руководством наставника алгоритмы и программы, пригодные для практического применения в будущей профессиональной деятельности а также практически применять команды для помещения изменений программного файла на удаленный репозиторий, Умеет создавать, изменять и удалять ветки репозитория.	Умеет разрабатывать под руководством наставника типовые алгоритмы и программы, пригодные для решения конкретных практических задач, а также практически применять команды для помещения изменений программного файла на удаленный репозиторий,	Не умеет разрабатывать под руководством наставника типовые алгоритмы и программы, пригодные для решения конкретных практических задач, а также практически применять команды для помещения изменений программного файла на удаленный репозиторий,
	Владеет способностью самостоятельно разрабатывать алгоритмы и программы, пригодные для практического применения в будущей профессиональной деятельности, а также владеет способностью самостоятельно разрабатывать программные файлы, помещать в репозиторий сервера, изменять, клонировать и распределять работу.	Владеет способностью разрабатывать под руководством наставника алгоритмы и программы, пригодные для практического применения в будущей профессиональной деятельности, а также владеет способностью самостоятельно разрабатывать программные файлы, помещать в репозиторий сервера, изменять, клонировать.	Владеет способностью разрабатывать под руководством наставника типовые алгоритмы и программы, пригодные для решения конкретных практических задач, а также владеет способностью самостоятельно, помещать программные файлы в репозиторий сервера, изменять, клонировать.	Не владеет способностью разрабатывать под руководством наставника типовые алгоритмы и программы, пригодные для решения конкретных практических задач, а также владеет способностью самостоятельно, помещать программные файлы в репозиторий сервера, изменять, клонировать

3. Распределение оценок за формы текущего контроля и промежуточную аттестацию

8 семестр:

Текущий контроль:
Лабораторные работы по темам
Тема 1. Введение в теоретические основы системы контроля версий программного обеспечения.

Тема 2. Создание и использование репозитория системы контроля версий программного обеспечения.
Тема 3. Обзор текущего состояния локального репозитория, коммит изменений.
Тема 4. Создание и использование ветвления системы контроля версий программного обеспечения.
Тема 5. Распределённая работа и использование системы контроля версий в окружениях.

Практическая работа по темам:
Тема 1. Введение в теоретические основы системы контроля версий программного обеспечения.
Тема 2. Создание и использование репозитория системы контроля версий программного обеспечения.
Тема 3. Обзор текущего состояния локального репозитория, коммит изменений.
Тема 4. Создание и использование ветвления системы контроля версий программного обеспечения.
Тема 5. Распределённая работа и использование системы контроля версий в окружениях.

Промежуточная аттестация - зачет.
Промежуточная аттестация проводится после завершения изучения дисциплины или ее части в форме, определяемой учебным планом образовательной программы с целью оценить работу обучающегося, степень усвоения теоретических знаний, уровень сформированности компетенций.
Преподаватель, принимающий зачет обеспечивает случайное распределение вариантов зачетных заданий между обучающимися с помощью билетов и/или с применением компьютерных технологий; вправе задавать обучающемуся дополнительные вопросы и давать дополнительные задания помимо тех, которые указаны в билете.
Зачет проводится по билетам. В каждом билете два оценочных средства: устный или письменный ответ на вопрос и решение задачи.

Выполнение каждого задания за промежуточную аттестацию оценивается по шкале: отлично, хорошо, удовлетворительно, неудовлетворительно.
Общая оценка за промежуточную аттестацию представляет собой среднее значение между полученными оценками за все оценочные средства промежуточной аттестации.
В случае невозможности установления среднего значения оценки за промежуточную аттестацию (например, «хорошо» или «отлично»), итоговая оценка выставляется преподавателем, исходя из принципа справедливости и беспристрастности на основании общего впечатления о качестве и добросовестности освоения обучающимся дисциплины (модуля).

Для зачета:
зачтено
не зачтено

4. Оценочные средства, порядок их применения и критерии оценивания
4.1. Оценочные средства текущего контроля
4.1.1. Лабораторные работы
Тема 1. Введение в теоретические основы системы контроля версий программного обеспечения.
Тема 2. Создание и использование репозитория системы контроля версий программного обеспечения.
Тема 3. Обзор текущего состояния локального репозитория, коммит изменений.
Тема 4. Создание и использование ветвления системы контроля версий программного обеспечения.
Тема 5. Распределённая работа и использование системы контроля версий в окружениях.

4.1.1.1. Порядок проведения.
В аудитории, оснащённой соответствующим оборудованием, обучающиеся проводят учебные эксперименты и тренируются в применении практико-ориентированных технологий. Оцениваются знание материала и умение применять его на практике, умения и навыки по работе с оборудованием в соответствующей предметной области.
Лабораторные работы по дисциплине «Системы контроля версий программного обеспечения» проводятся преподавателем согласно разработанному и утвержденному на кафедре рабочей программе. Каждая лабораторно-практическая работа выполняется по определенной теме программы в соответствии с заданием.
Перед выполнением каждой работы студенты-бакалавры должны проработать соответствующий материал, используя конспекты теоретических занятий, периодические издания, учебно-методические пособия и учебники
На каждом занятии студенты выполняют работу в соответствии с ее содержанием и методическими указаниями.
По окончании занятий студенты оформляют отчет по каждой работе, соблюдая следующую форму:
- Наименование темы;
- Цель работы;
- Задание и содержание выполненной работы,
- Письменные ответы на контрольные вопросы.
- Выводы по проделанной работе.
- Список использованных источников.

4.1.1.2. Критерии оценивания

6 семестр

Оценка «отлично» ставится, если обучающийся:

Правильно выполнил все задания. Продемонстрировал высокий уровень владения материалом. Проявлены превосходные способности применять знания и умения к выполнению конкретных заданий.

Оценка «хорошо» ставится, если обучающийся:

Правильно выполнил большую часть заданий. Присутствуют незначительные ошибки. Продемонстрирован хороший уровень владения материалом. Проявлены средние способности применять знания и умения к выполнению конкретных заданий.

Оценка «удовлетворительно» ставится, если обучающийся:

Задания выполнил более чем наполовину. Присутствуют серьёзные ошибки. Продемонстрирован удовлетворительный уровень владения материалом. Проявлены низкие способности применять знания и умения к выполнению конкретных заданий.

Оценка «неудовлетворительно» ставится, если обучающийся:

Задания выполнил менее чем наполовину. Продемонстрирован неудовлетворительный уровень владения материалом. Проявлены недостаточные способности применять знания и умения к выполнению конкретных заданий.

4.1.1.3. Содержание оценочного средства

Лабораторная работа №1. Последовательность работ с локальным репозиторием

Git обладает необычайной легкостью в использовании не только как распределенная система контроля версий, но и в работе с локальными проектами. Давайте разберем обычный цикл — начиная с создания репозитария — работы разработчика git над собственным персональным проектом:

1. Создаем рабочую директорию проекта;
2. Создаем репозиторий в директории;
3. Индексируем все существующие файлы проекта (добавляем в репозиторий) и создаем инициализирующий коммит;
4. Создаем новую ветку;
5. Переключение в новую ветку (можно сделать в один шаг);
6. Далее, после непосредственной работы с кодом;
7. Индексируем внесенные изменения и совершаем коммит;
8. Переключаемся в основную ветку;
9. Смотрим отличия между последним коммитом активной ветки и последним коммитом экспериментальной;
10. Проводим слияние;
11. Если был конфликт, то разрешаем его и повторяем слияние;
12. Ну и на всякий случай оценим проведенную за последний день работу.

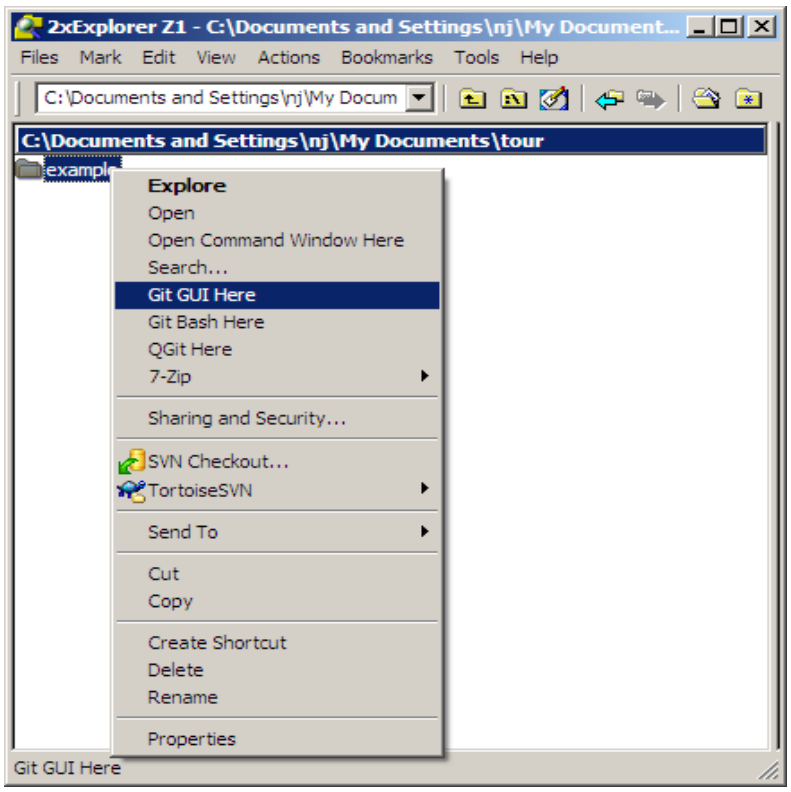
Почему именно так? Зачем отказываться от линейной модели? Хотя бы даже потому, что у программиста появляется дополнительная гибкость: он может переключаться между задачами (ветками); под рукой всегда остается «чистовик» — ветка master; коммиты становятся мельче и точнее.

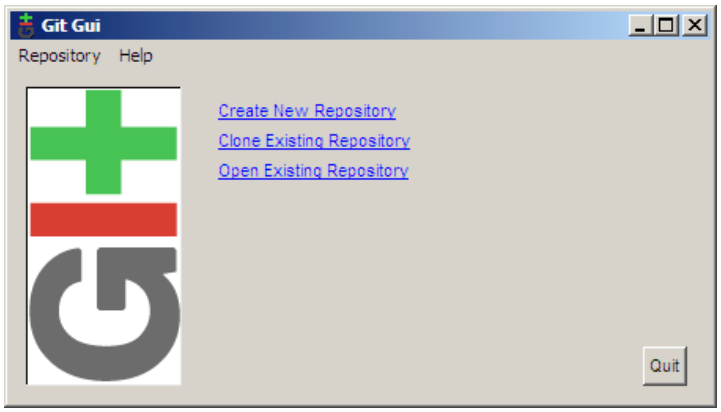
Лабораторная работа №2. Создание хранилища (GIT GUI)

Работать можно в командной строке (инструкции см. в соответствующем файле), или в интерфейсе Git Gui, или в интерфейсе TortoiseGit (на усмотрение студента).

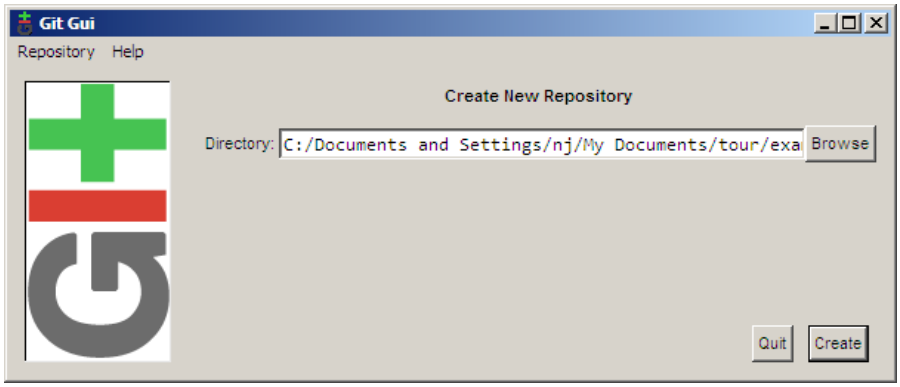
Далее подробно рассматривается работа с Git Gui.

Что бы создать хранилище, сначала создайте папку в которой ваш проект будет жить. Далее, правый щелчок мышки на этой папке и выберете *Git GUI*. Так-как в папке пока что не содержится git хранилища вам будет показан диалог создания.

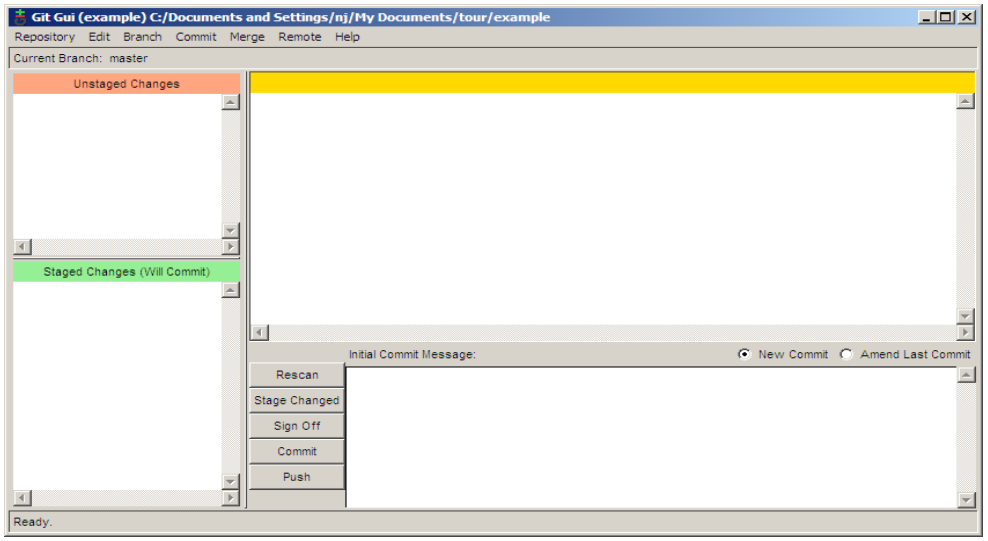




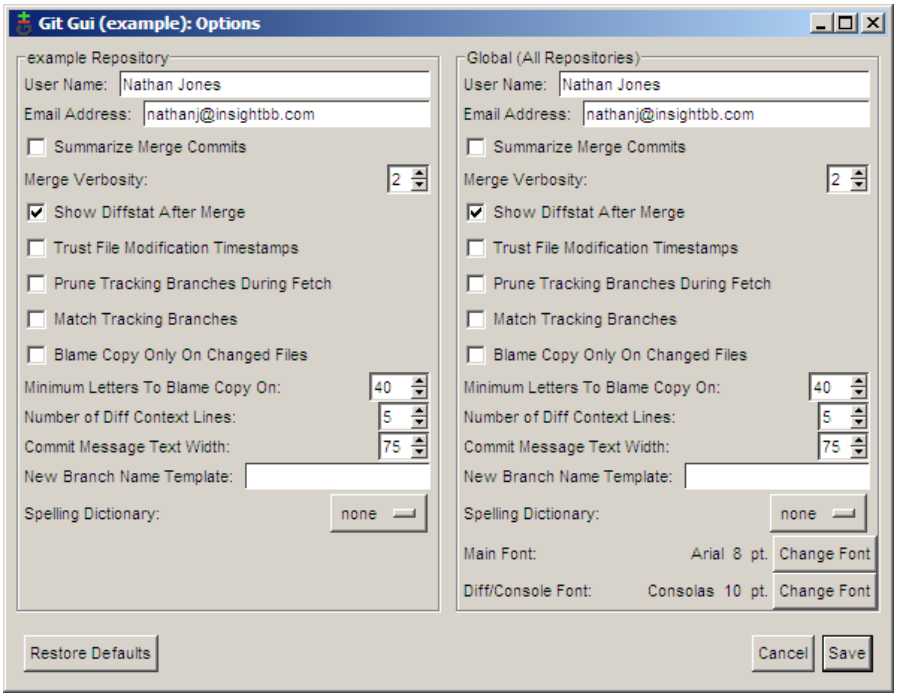
Выбор *Create New Repository* приводит нас к следующему диалогу.



Заполните путь к вашей новой директории и щёлкните *Create*. Далее вы увидите главный интерфейс git gui, который в дальнейшем будет показываться когда вы будете делать правый щелчок на вашей папке и выбирать *Git GUI*.



Теперь когда хранилище создано, вам надо сообщить git-у кто вы такой, что бы сообщении фиксации (commit message) был отмечен правильный автор. Что бы сделать это, выберете *`Edit` → `Options` (`Редактировать` → `Настойки`)*.



В диалоге опций расположены 2 варианта на выбор. С левой стороны диалога опции которые влияют только на это хранилище, в то время как правая сторона содержит глобальные опции применяемые ко всем хранилищам. Значения по умолчанию приемлемы, так что заполните только имя пользователя и email, пока что. Если у вас есть любимый шрифт, вы можете выставить его сейчас, так же.

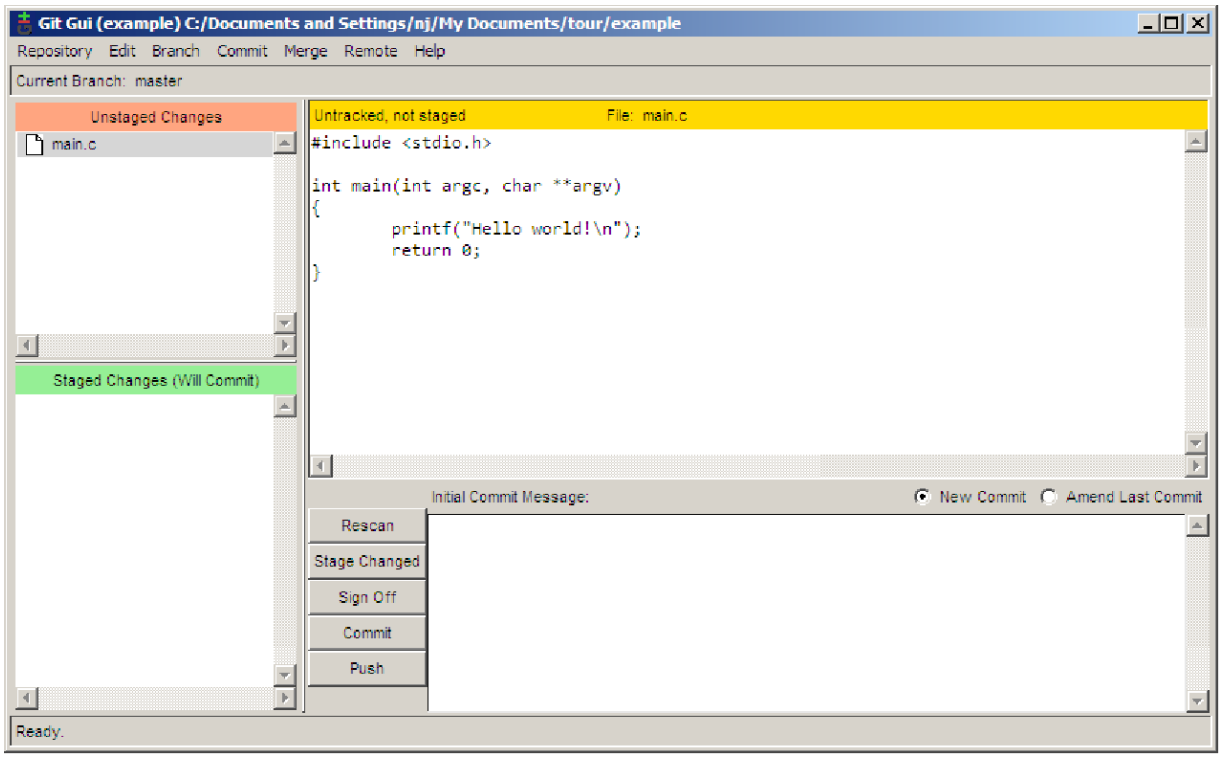
Лабораторная работа №3. Фиксация изменений (commiting)

Теперь, когда хранилище было создано, пора создать что-нибудь для фиксации. Для этого примера я создал файл main.c со следующим содержимым (редактирование выполняется в блокноте или другом редакторе):

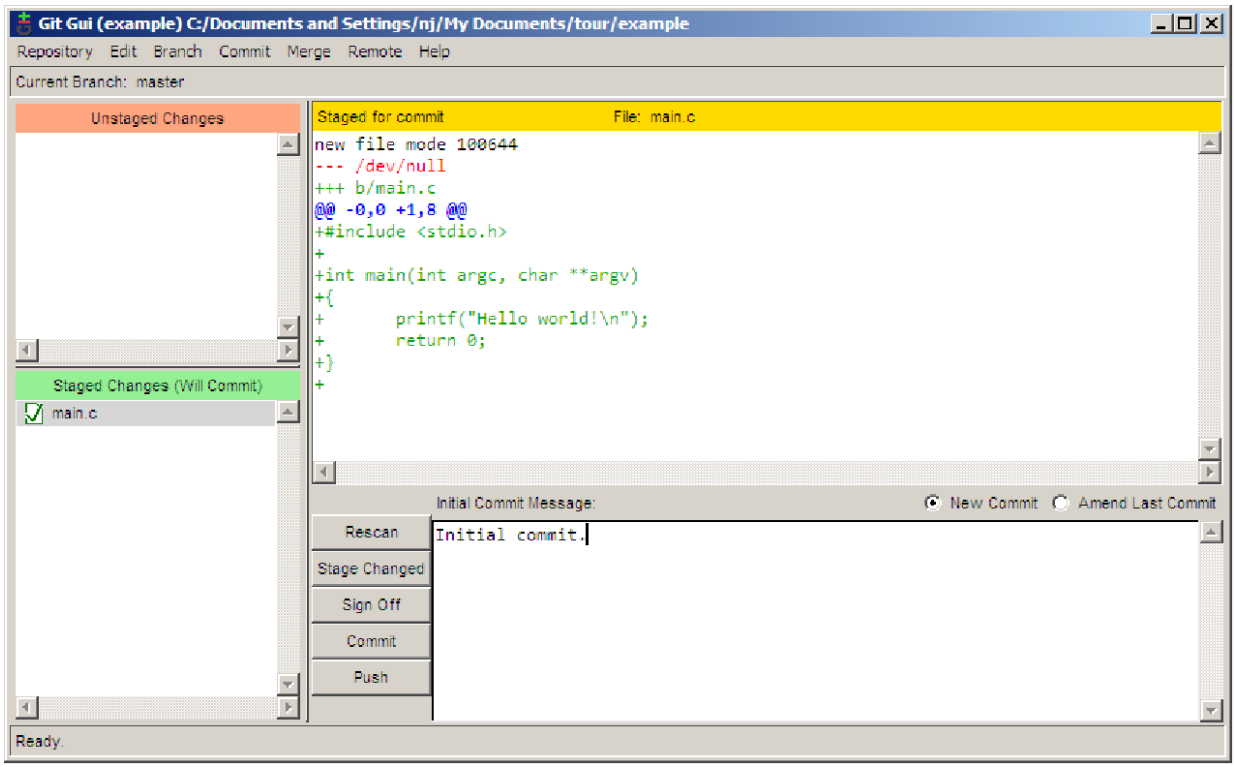
```
#include <stdio.h>

int main(int argc, char **argv)
{
    printf("Hello world!\n");
```

```
    return 0;
}
```

Щелчок на кнопку *Rescan (Перечитать)* в git gui заставит его искать новые, измененные и удалённые файлы в директории. На следующем скриншоте git gui нашёл новый файл.



Что бы добавить этот файл в фиксацию, щёлкните на иконке слева от имени файла. Файл будет перемещён с *Unstaged Changes* (Измененено) панели на *Staged Changes* (Подготовлено) панель. Теперь мы можем добавить сообщение фиксации (commit message) и зафиксировать изменения *Commit* (Сохранить) кнопкой.



Говорить 'hello world' это конечно хорошо, но я хочу, чтобы моя программа была более персонализирована. Давайте скажем 'hello' пользователю. Вот как будет выглядеть измененный код (редактирование выполняется в блокноте или другом редакторе):

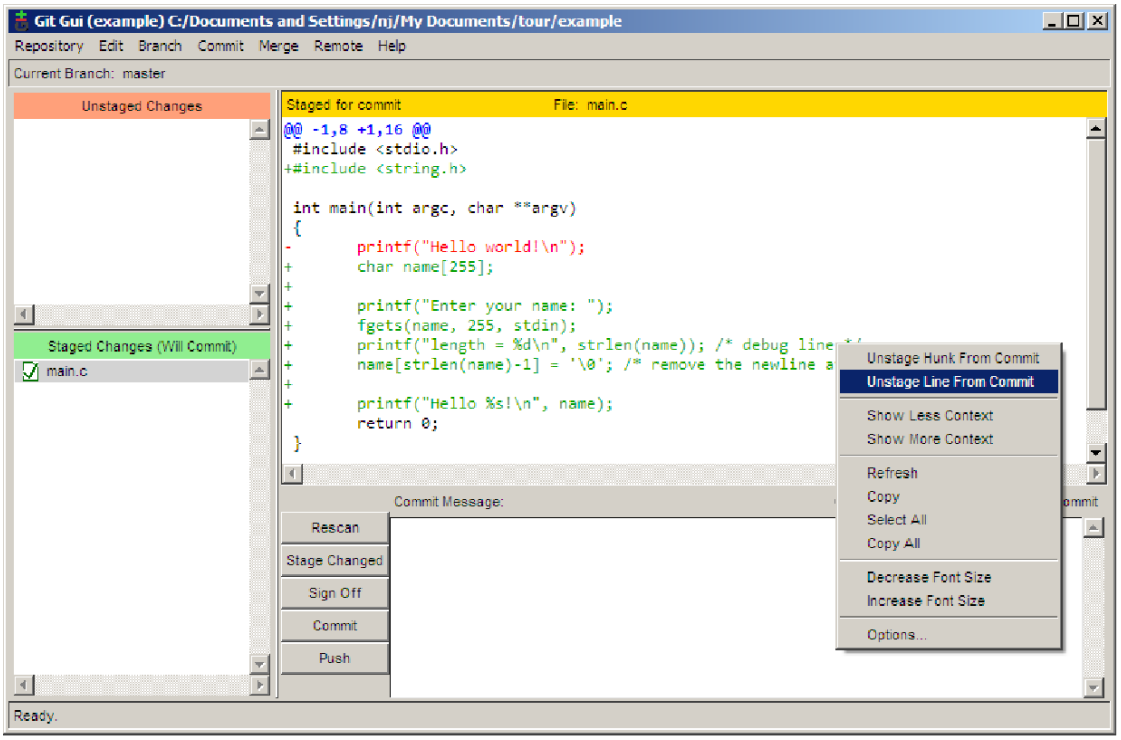
```
#include <stdio.h>
#include <string.h>

int main(int argc, char **argv)
{
    char name[255];

    printf("Enter your name: ");
    fgets(name, 255, stdin);
    printf("length = %d\n", strlen(name)); /* debug line */
    name[strlen(name)-1] = '\0'; /* remove the newline at the end */

    printf("Hello %s!\n", name);
    return 0;
}
```

У меня были проблемы с тем что новая линия печаталась после имени пользователя, поэтому я добавил отладочную линию кода, чтобы помочь себе найти причину. Я бы хотел зафиксировать изменение без этой отладочной линии, но я хочу сохранить эту линию в моей рабочей копии что бы продолжить отладку. С git gui это не проблема. Сначала щёлкните *Rescan* (перечитать) для поиска изменений.

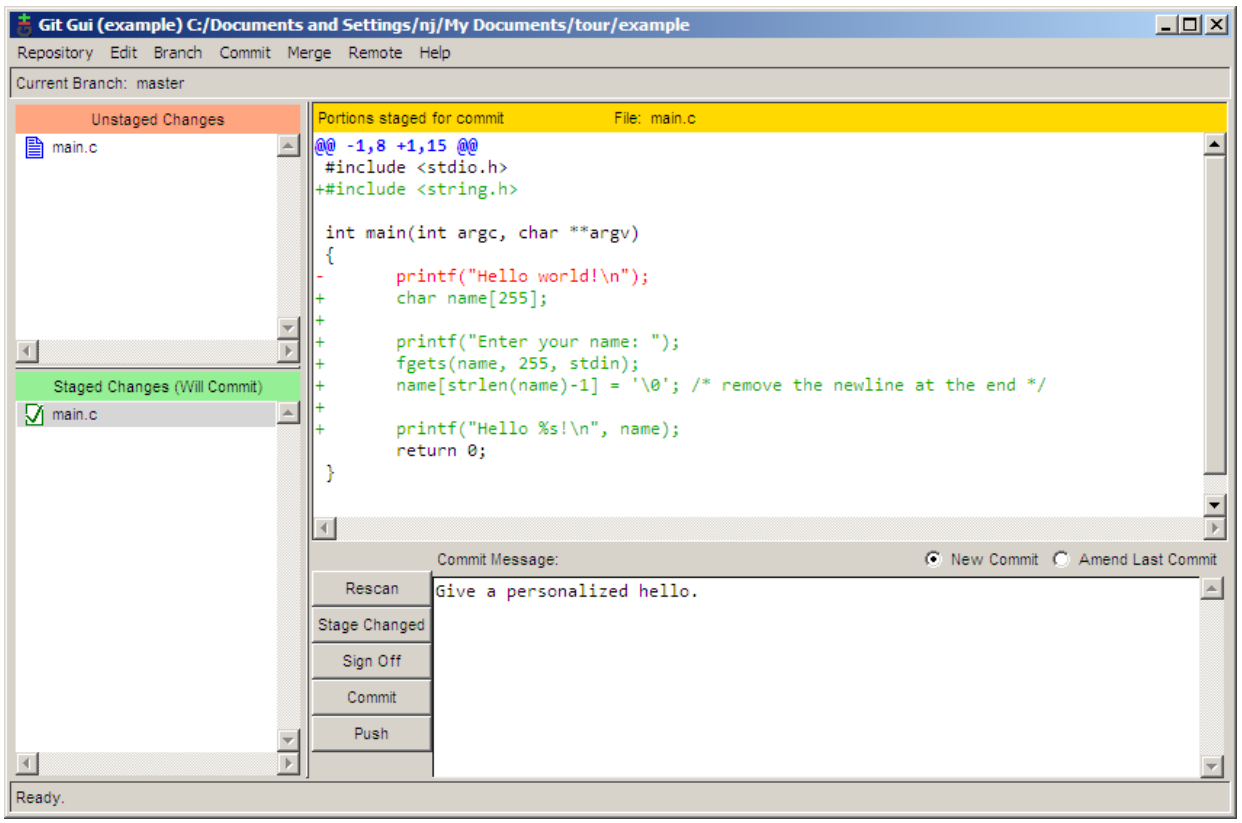
Изменения выделены красным (удаленные линии) или зеленым (добавленные линии).

Далее щёлкните на иконке слева от файла что бы подготовить (stage) изменения к фиксации. Файл будет перенесен в нижнее окно.

Затем правый щелчок на отладочной линии и выберите *Unstage Line From Commit (Убрать строку из подготовленного)*.

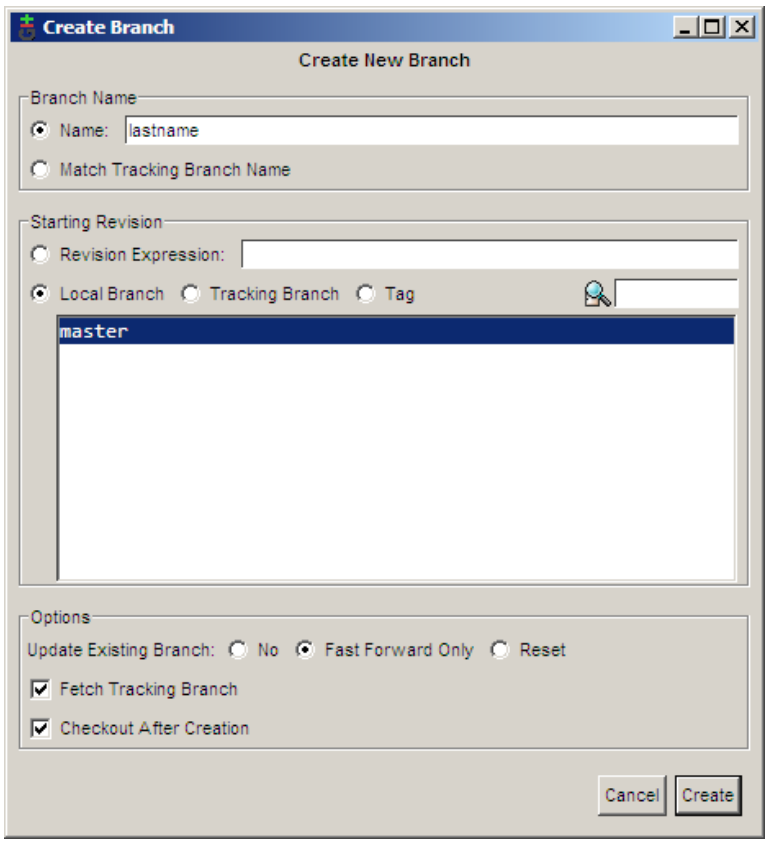


Теперь отладочная линия не была подготовлена (unstaged) к фиксации, в то время как остальные изменения были. Осталось только заполнить сообщение фиксации и зафиксировать изменения щёлкнув по *Commit (Сохранить)*.



Лабораторная работа №4. Ветвление (branching)

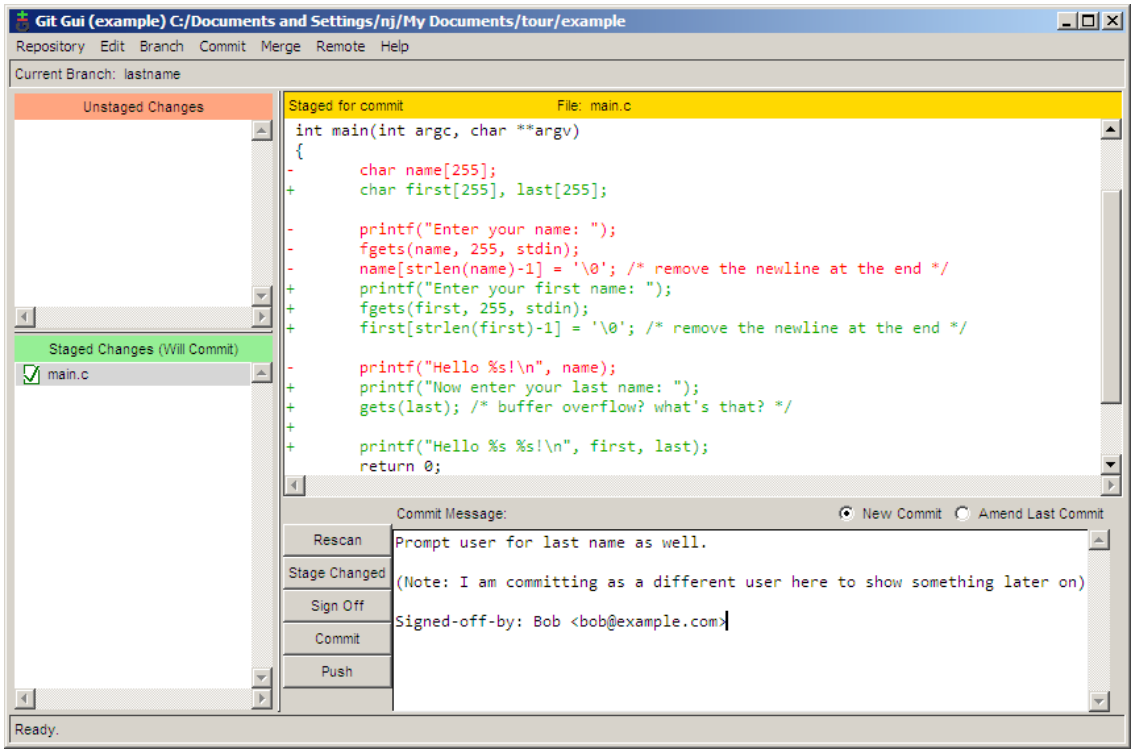
Теперь, давайте предположим, что мы хотим начать добавлять новые возможности в нашу следующую большую версию программы. Но мы так же хотим сохранить стабильную версию в которой исправлять ошибки. Что бы сделать это мы создадим ветку (branch) для наших новых разработок. Что бы создать новую ветку в git gui выберете *Branch → Create (Ветвь → Создать)*. Большая возможность какую я хочу добавить это возможность спросить пользователя его фамилию, поэтому я назову ветку lastname. Опции по умолчанию подходят без изменений, так что просто введите имя и щёлкните *Create*.



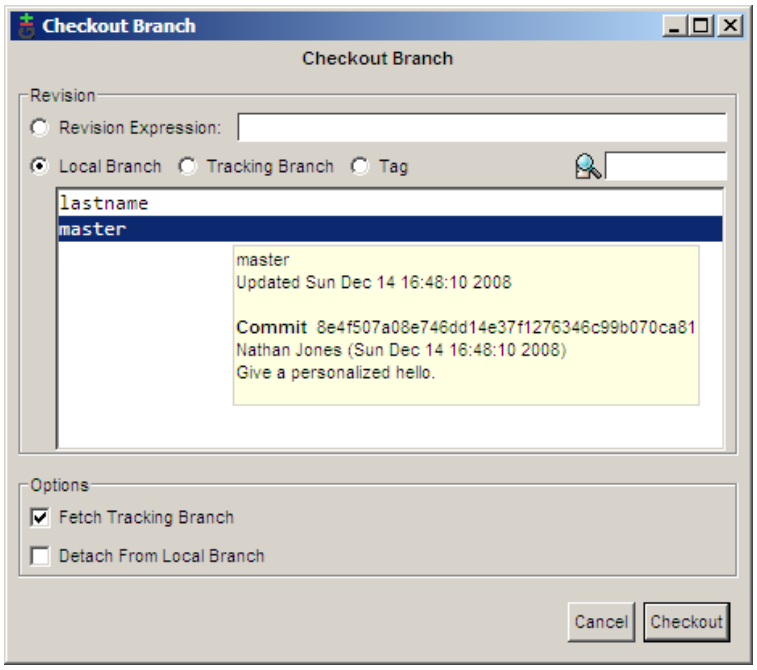
Теперь, когда я в lastname ветке, я могу делать мои новые модификации:

```
#include <stdio.h>
#include <string.h>

int main(int argc, char **argv)
{
    char first[255], last[255];

    printf("Enter your first name: ");
    fgets(first, 255, stdin);
    first[strlen(first)-1] = '\0'; /* remove the newline at the end */

    printf("Now enter your last name: ");
    gets(last); /* buffer overflow? what's that? */

    printf("Hello %s %s!\n", first, last);
    return 0;
}
```

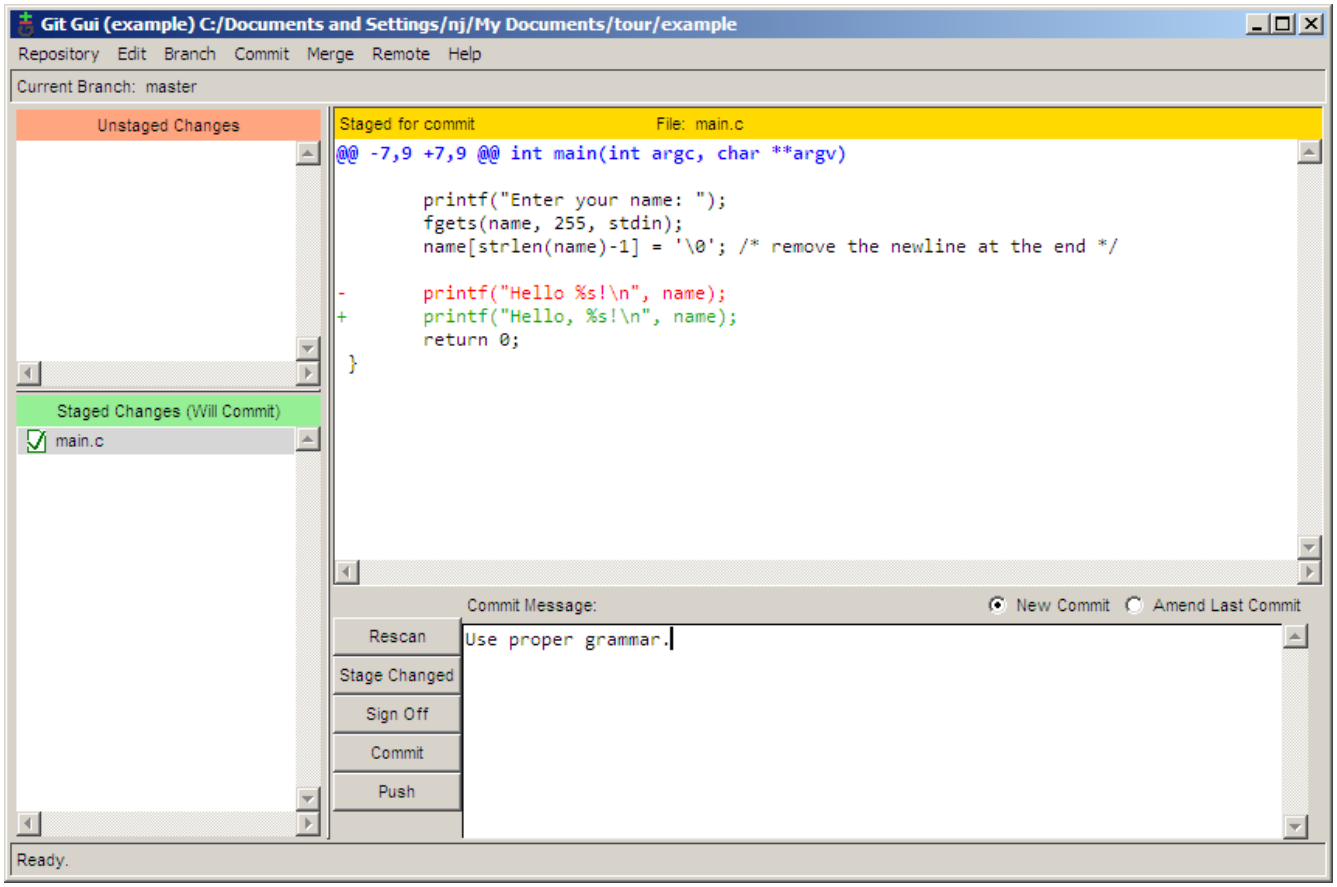
Теперь я могу зафиксировать изменения. Замете что я фиксирую изменения используя другое имя. Мы рассмотрим это позже. Обычно вы всегда будете использовать одно и тоже имя для фиксаций.



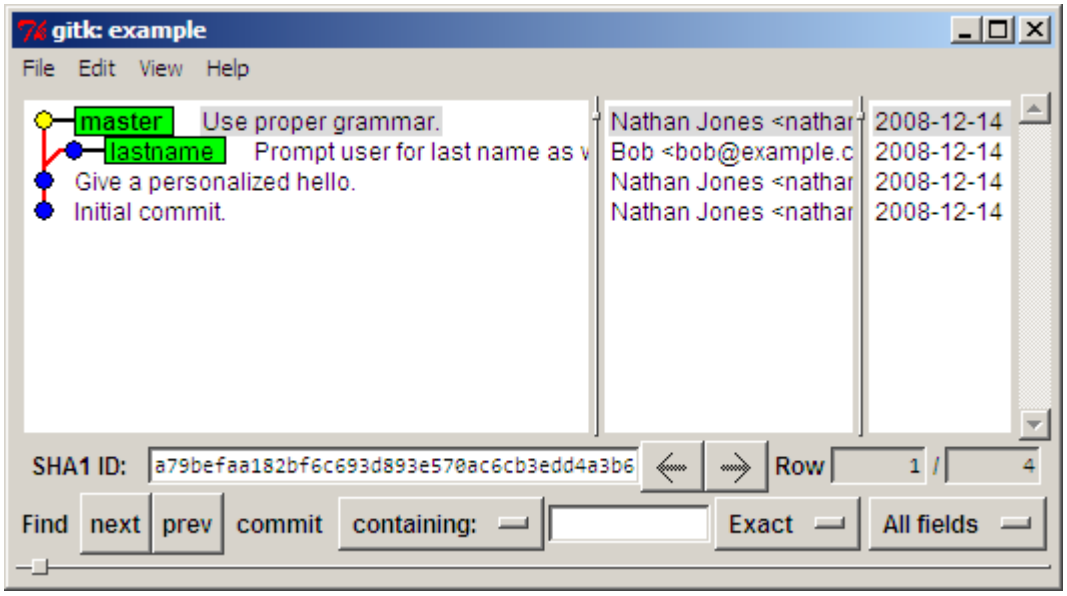
Между тем, пользователь проинформировал нас что не показ запятой после прямого обращения к кому-то это серьёзная ошибка. Что бы исправить её в нашей стабильной ветке, вы сначала должны переключится назад на неё. Это достигается, используя *Branch → Checkout (Ветвь → Перейти)*.



Теперь мы можем исправить нашу большую ошибку.

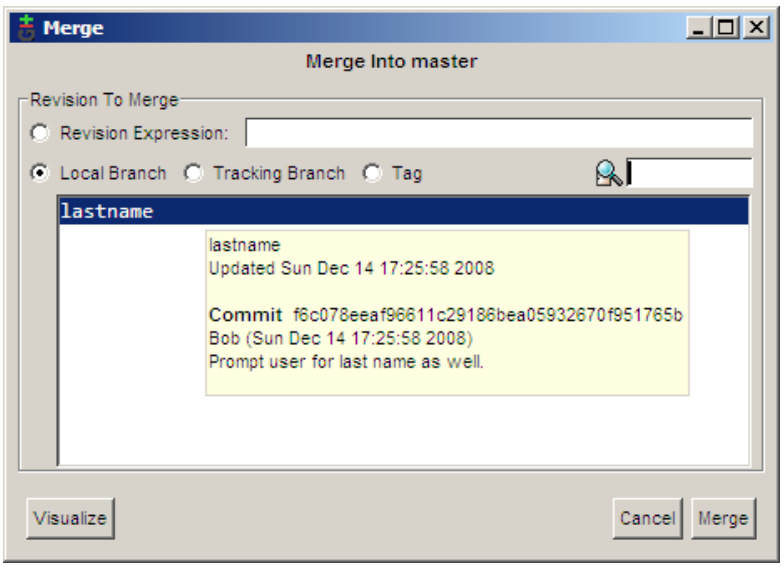


Если мы выберем *Repository → Visualize All Branch History (Репозиторий → Показать историю всех ветвей)*, мы увидим, как складывается наша история.

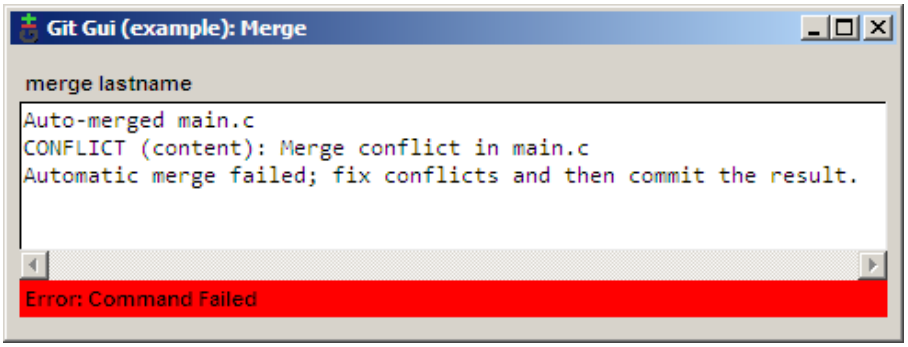


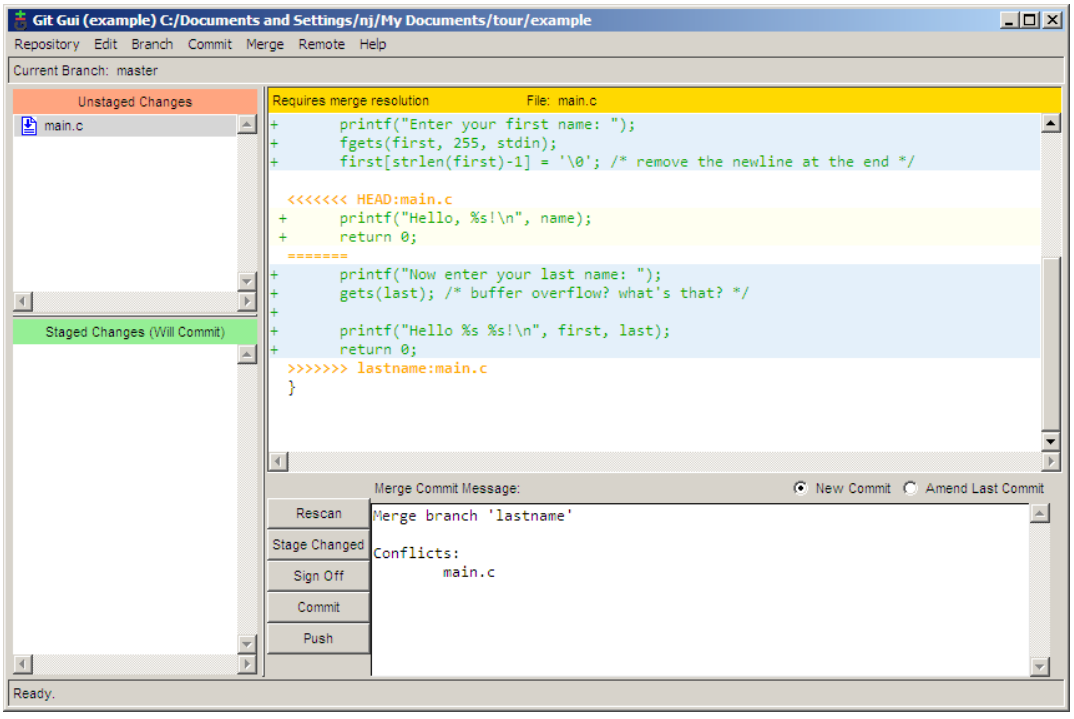
Лабораторная работа №5. Слияние (merging)

После напряжённой работы мы решили, что наша lastname ветка достаточно стабильна, чтобы влить её в master ветку. Что бы выполнить слияние, используйте *Merge → Local Merge (Слияние → Локальное слияние)*.



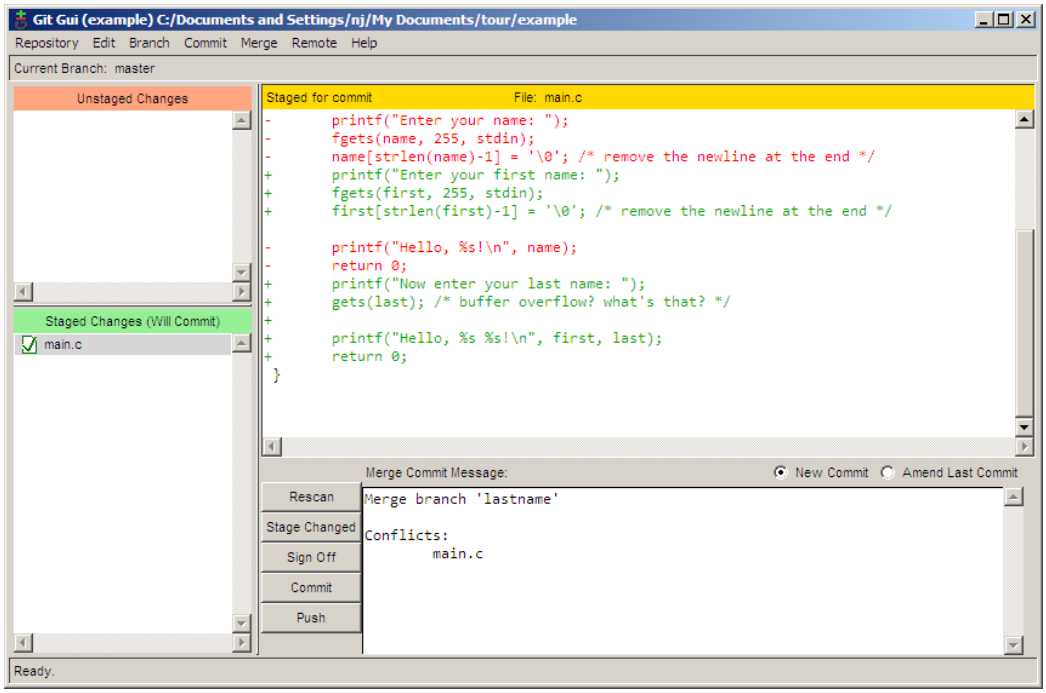
Так как две разных фиксации делали два разных изменения на одной и той же линии, происходит конфликт (conflict).





Конфликт может быть разрешён используя любой текстовый редактор (оставляете в тексте только нужный вариант).

После разрешения конфликта, подготовьте изменения щёлкнув на иконке файла и зафиксируйте слияние щёлкнув по *Commit* кнопке: в редакторе внести изменения в файл и сохранить его - Перечитать — Подготовить - Сохранить.



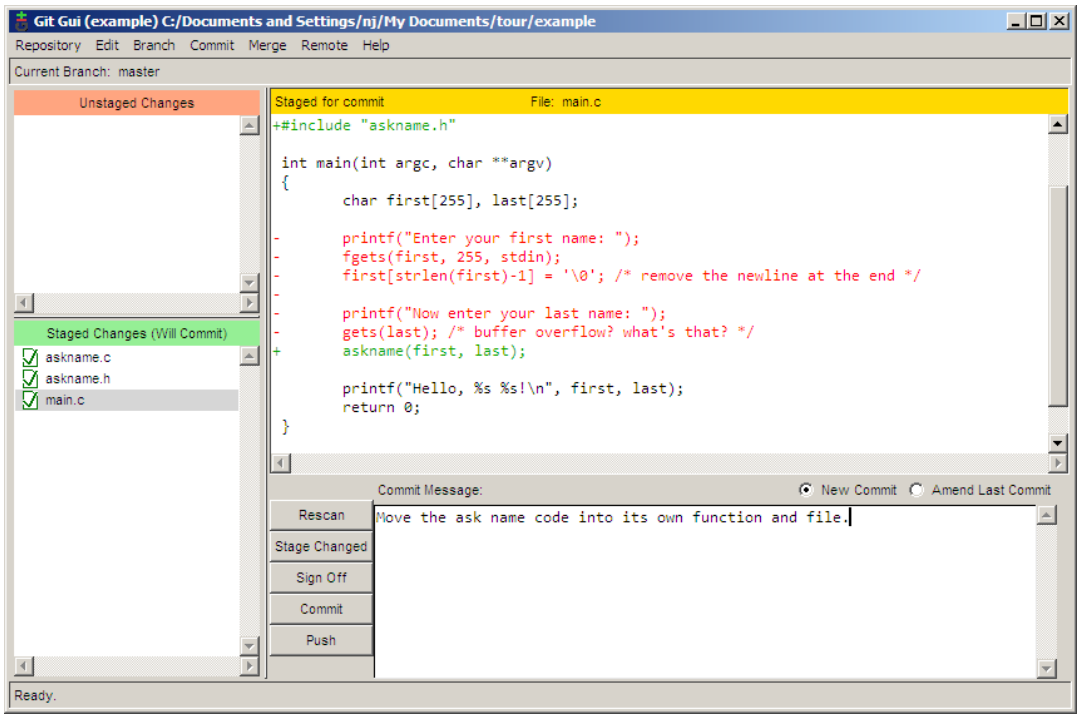
Лабораторная работа №6. Просмотр истории

Файл main.c становится немного большим, поэтому я решил вынести код, спрашивающий имя пользователя в отдельную функцию. Пока я это делал, я решил вынести функцию в отдельный файл. Хранилище теперь содержит файлы main.c, askname.c, и askname.h.

```
/* main.c */
#include <stdio.h>

#include "askname.h"

int main(int argc, char **argv)
{
    char first[255], last[255];

    askname(first, last);

    printf("Hello, %s %s!\n", first, last);
    return 0;
}

/* askname.c */
#include <stdio.h>
#include <string.h>

void askname(char *first, char *last)
{
    printf("Enter your first name: ");
    fgets(first, 255, stdin);
    first[strlen(first)-1] = '\0'; /* remove the newline at the end */

    printf("Now enter your last name: ");
    gets(last); /* buffer overflow? what's that? */
}
```

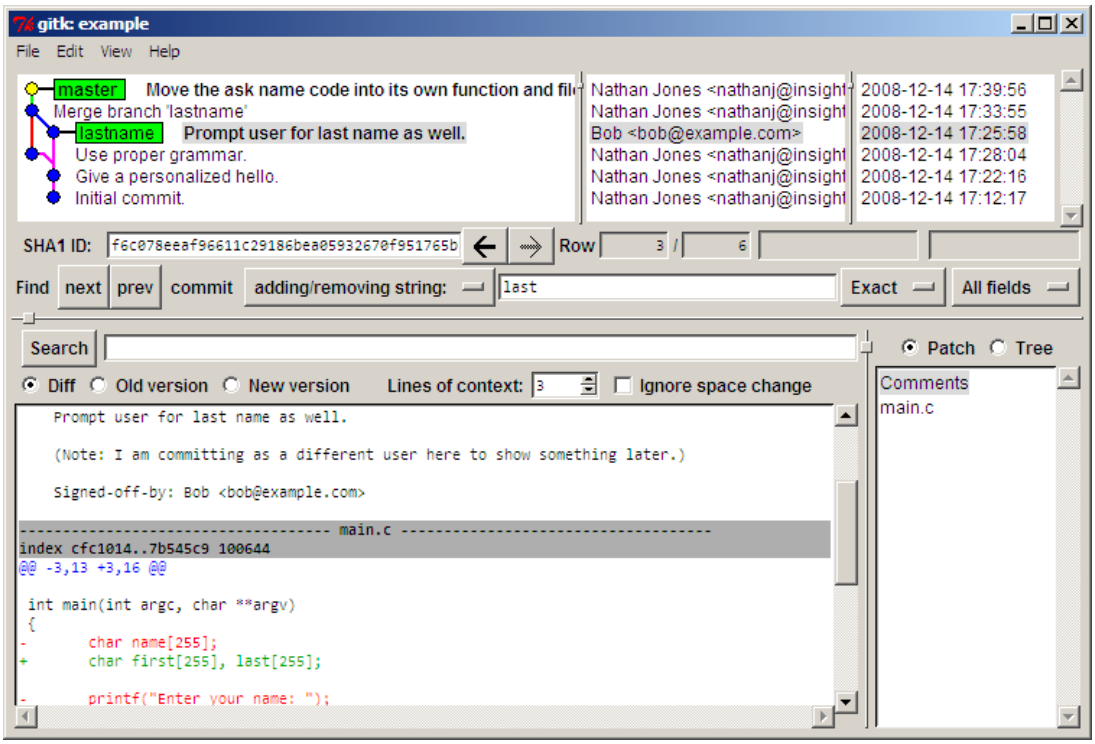
```
/* askname.h */
void askname(char *first, char *last);
```

Файлы создают в редакторе. Затем перечитать репозиторий, подготовить все и сохранить.



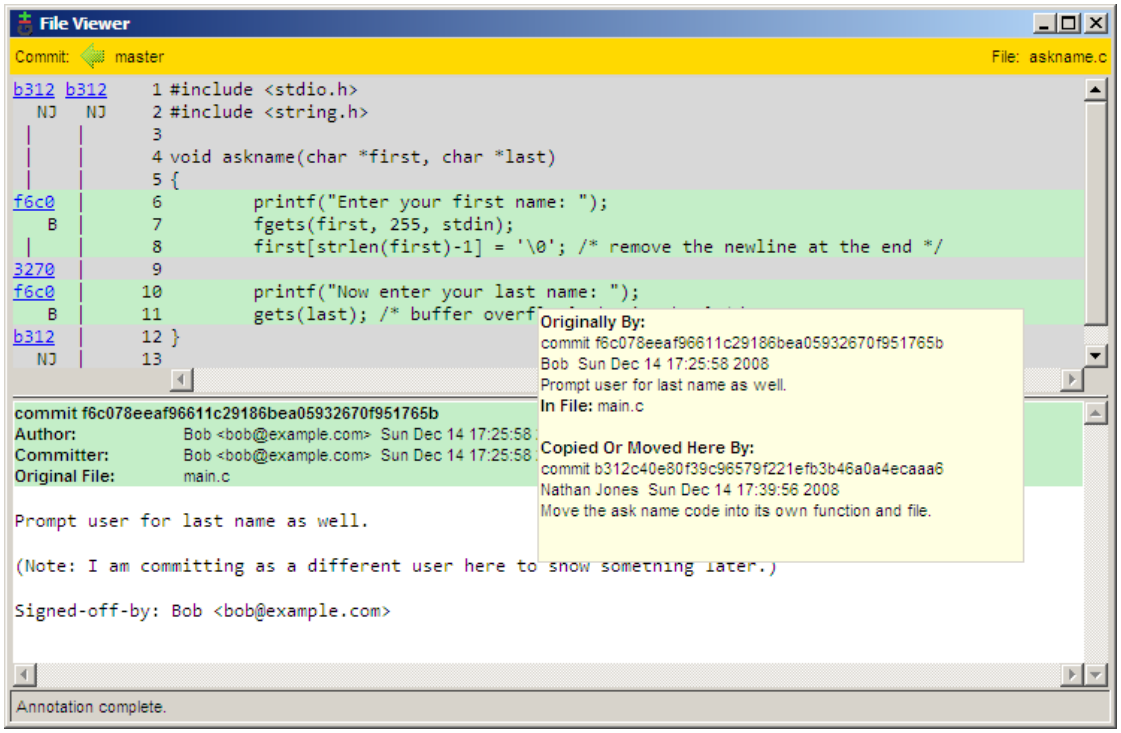
История хранилища может быть просмотрена и изучена выбрав *Repository → Visualize All Branch History*. На следующем скриншоте я пытаюсь найти в какой фиксации была добавлена last переменная, ища все фиксации, в которых было добавлено или убрано слово *last*.

Фиксации, которые подходят под условия поиска отмечены жирным шрифтом, чтобы быстро и легко обнаружить нужную фиксацию. Можно посмотреть старую и новую версии. Цветом выделены изменения.



Через пару дней, кто-то, просматривая наш код увидел, что gets функция может вызвать переполнение буфера. Будучи любителем показывать пальцем, этот человек решает запустить git blame что бы увидеть кто последний раз редактировал эту линию кода. Проблема в том, что Боб, тот кто зафиксировал эту линию в хранилище, а я последний кто трогал её, когда я переместил строку в другой файл. Очевидно, я не виноват (конечно же). Но так ли умен git что бы обнаружить это? Да, это так.

Что бы запустить blame, выберете ***Repository → Browse master's Files (Репозиторий → Показать файлы ветви master)***. Из дерева, которое появится, дважды щёлкните на файле с интересующей строкой, который в данном случае askname.c. Наведённая мышка на интересующую линию показывает нам подсказку, которая говорит нам всё что нам надо знать.



Здесь мы можем видеть, что эта линия была зафиксирована Бобом в фиксации f6c0, а затем я её переместил в её новое месторасположение в фиксации b312.

Лабораторная работа №7. Отмена изменений (revert или reset)

Для отмены внесенных изменений до состояния последней фиксации:

Меню Состояния – Отменить изменения.

Для отката к конкретной фиксации (**reset**):

Меню Репозиторий – История ветки – (выбрать нужное состояние и в контекстном меню выбрать установить для ветви это состояние). Возможны варианты (мягкий — изменение только индекса или жесткий — изменения в индексе и на диске).

Для отката через новую фиксацию — создается новая фиксация, содержащая изменения, обратные зафиксированным (**revert**):

Меню Репозиторий – История ветки – (выбрать нужное состояние и в контекстном меню выбрать revert this commit).

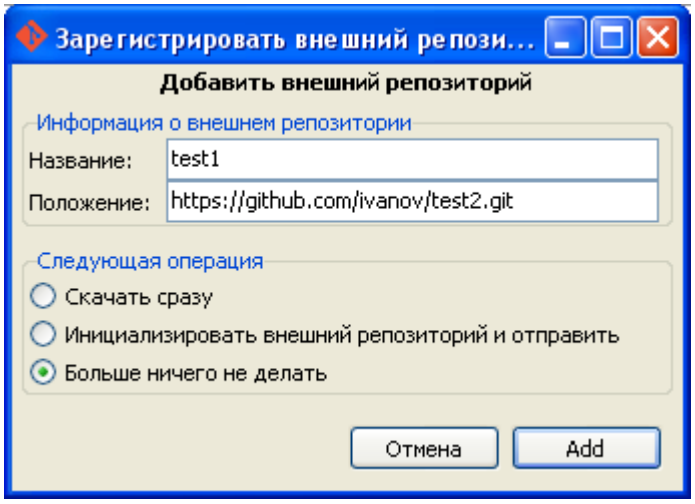
Лабораторная работа №8. Публикация изменений (pushing) на удалённом сервере

Зарегистрируйтесь на сайте https://github.com/ (укажите логин, почту и пароль, дальше выберите бесплатный доступ).

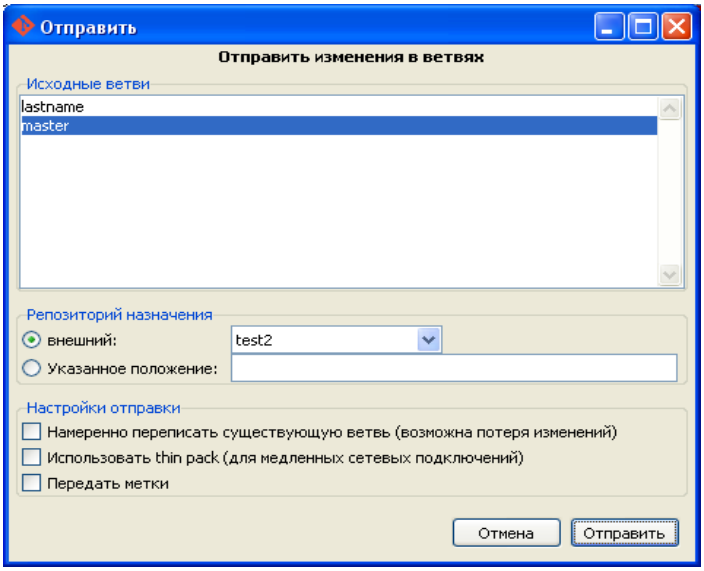
Создайте новый репозиторий: После входа на сайт выберите Create new repository, укажите его название, тип (публичный) и нажмите Create repository.

После создания репозитория будет отображено его имя - адрес (в формате HTTPS) и команды для работы с ним в режиме командной строки (для желающих).

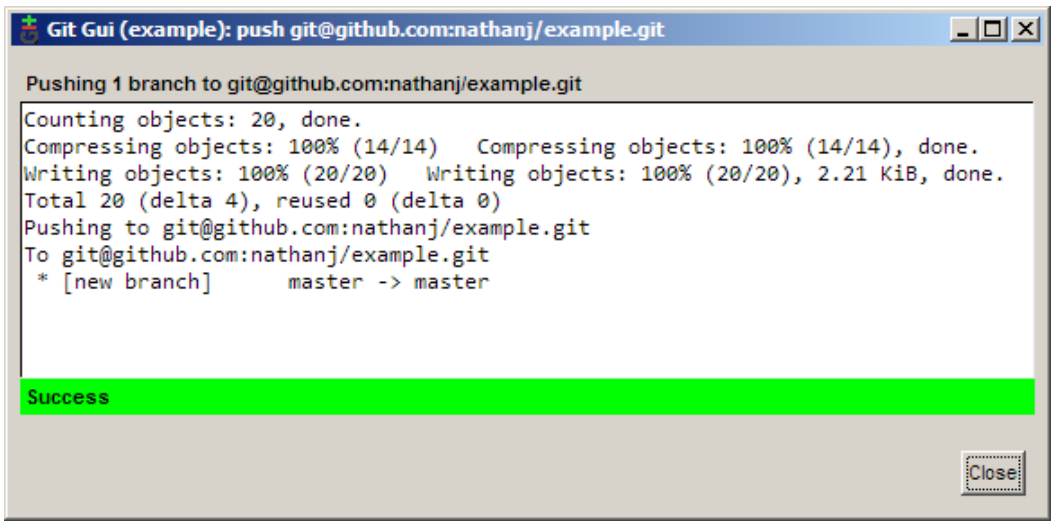
Далее из каталога репозитория на локальном ПК вызовите Git Gui, выберите меню ***Внешние репозитории*** → ***Добавить,*** в новом окне укажите псевдоним удаленного репозитория (любой, на рис. это Test1) и его адрес (https://github.com/ivanov/test2.git, где ivanov – логин пользователя сайта, test2 – название репозитория на сайте). Адрес рекомендую копировать с сайта.



Затем выбрать в меню ***Внешние репозитории*** → ***Отправить***, указать псевдоним удаленного репозитория и отправляемую ветку, нажать *Отправить*.



При запросе ввести логин и пароль, с которыми регистрировались на сайте.

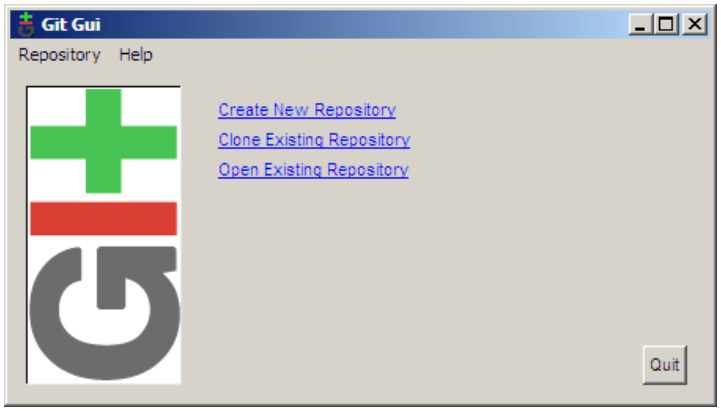


Если все успешно, то на сайте перейти в репозиторий и просмотреть его содержимое.

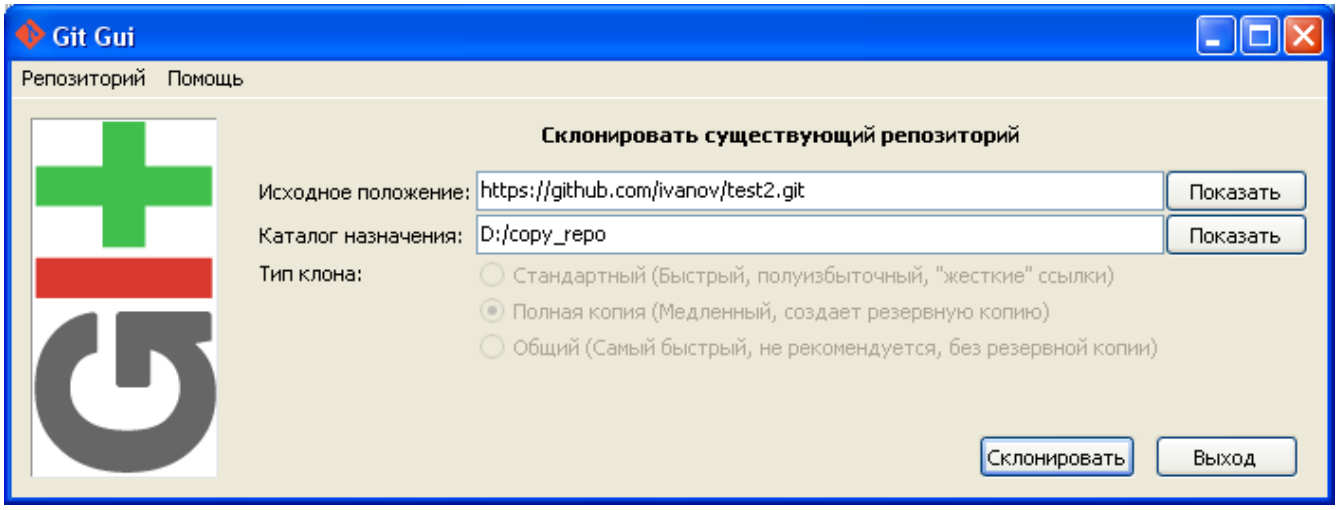
Можно просматривать историю изменений репозитория, содержимое фиксаций, изменения.

Лабораторная работа№9. Получение изменений (pulling) с удалённого сервера

Для получения копии удаленного репозитория открыть проводник, щелкнуть правой мышью и из контекстного меню выбрать *Git GUI*. Вам будет показан диалог создания.



Выбрать Clone (Клонировать существующий репозиторий). Будет открыт диалог клонирования. В качестве источника укажите удаленный репозиторий (его адрес), в качестве приемника — новый каталог.



Нажать склонировать. Если все успешно, то откроется среда Git Gui, в которой возможно посмотреть содержимое файлов и историю изменений.

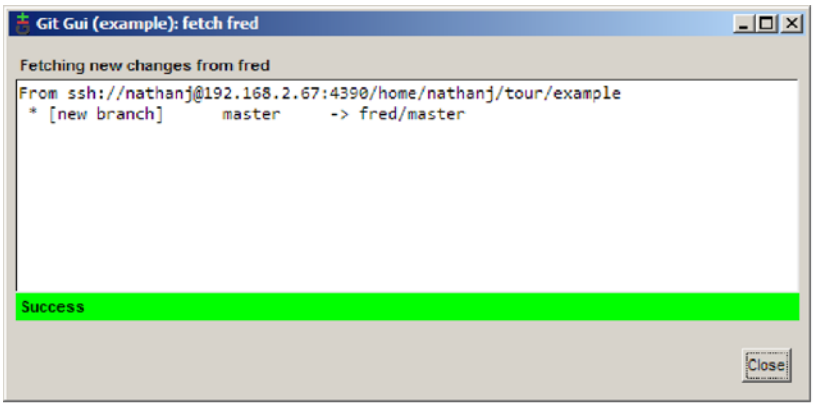
Дальше можно работать над проектами независимо. После внесения изменений и их фиксации отправим изменения обратно на сервер (меню ***Внешние репозитории → Отправить***, указать псевдоним удаленного репозитория и отправляемую ветку, нажать *отправить)*.

Предварительно необходимо разрешение владельца репозитория на внесение изменений. Для этого владелец проекта на сайте выбирает (сверху вверху) New *Collaborator,* откроется страница участников проектов. На ней указать имя добавляемого участника (он должен быть найден по имени или его части) и нажать *Add Collaborator.* Новый участник будет добавлен в список участников.

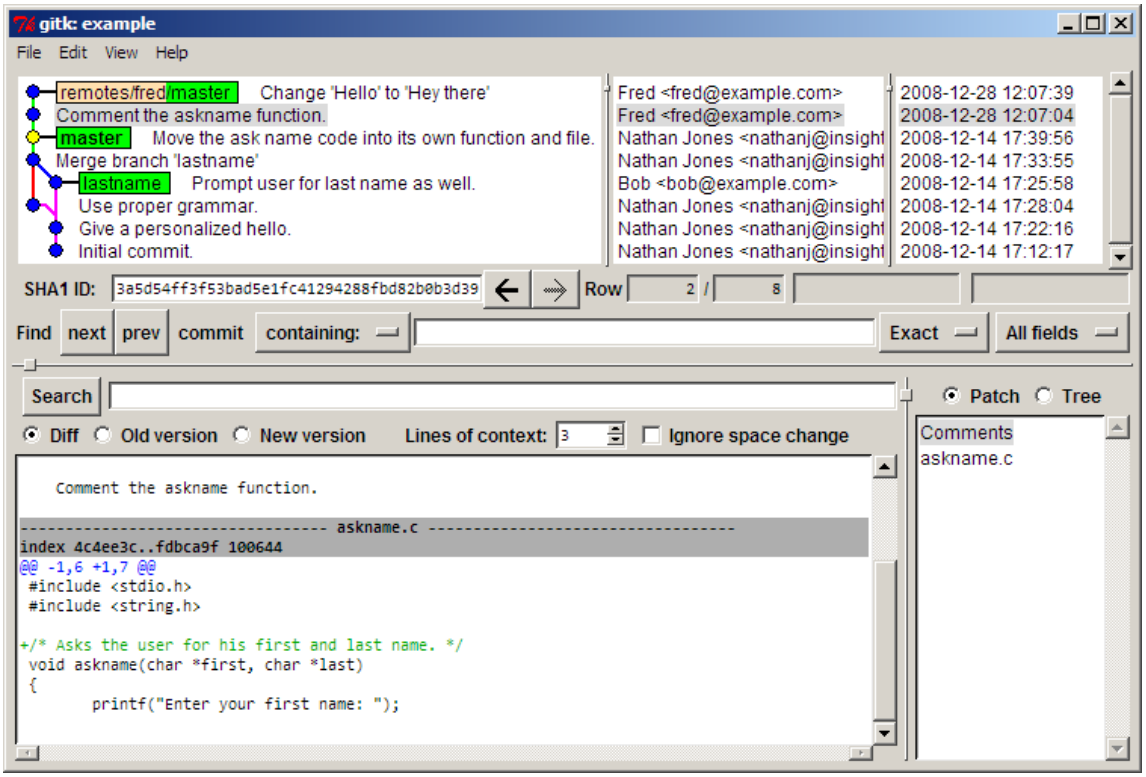
Все участники проекта могут вносить свои изменения, используя адрес репозитория, свои логин и пароль.

Из-за того, что наш код такой полезный, несколько человек скачали его и теперь используют нашу программу. А один человек, Фред, даже решил форкнуть его и добавить собственные изменения. Теперь, когда он добавил свой код, он хотел бы что бы мы перетянули его изменения в своё хранилище.

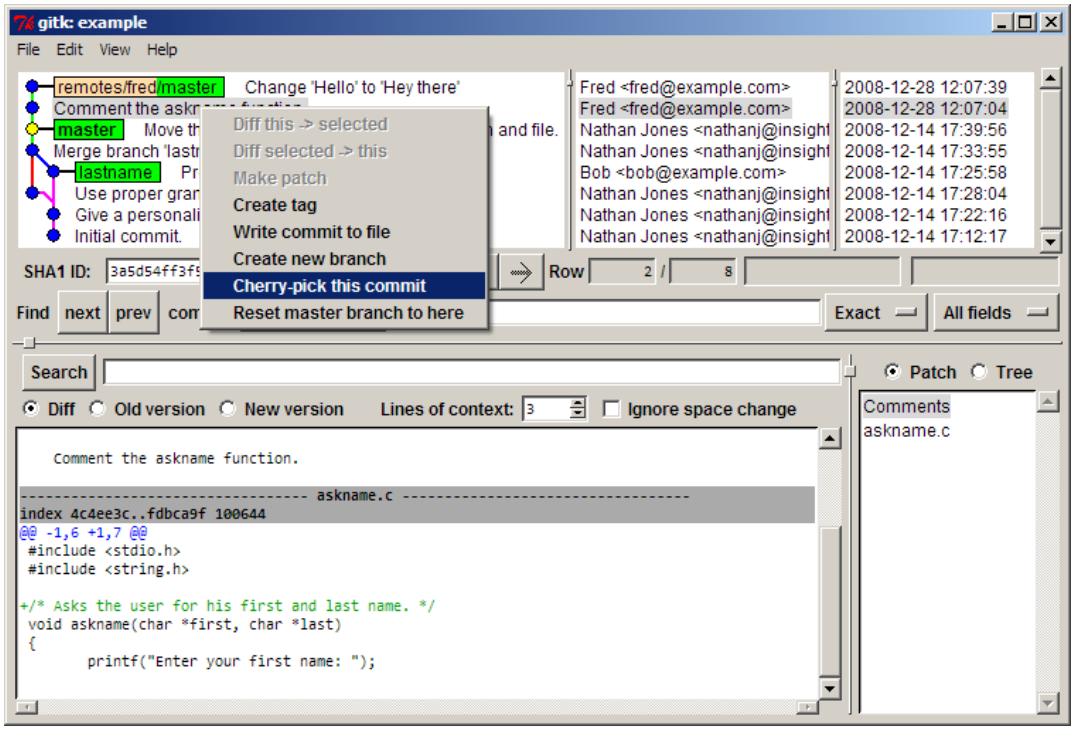
Мы можем получить изменения Фреда, используя ***Remote → Fetch from → fred (Внешние репозитории → Получить)***.



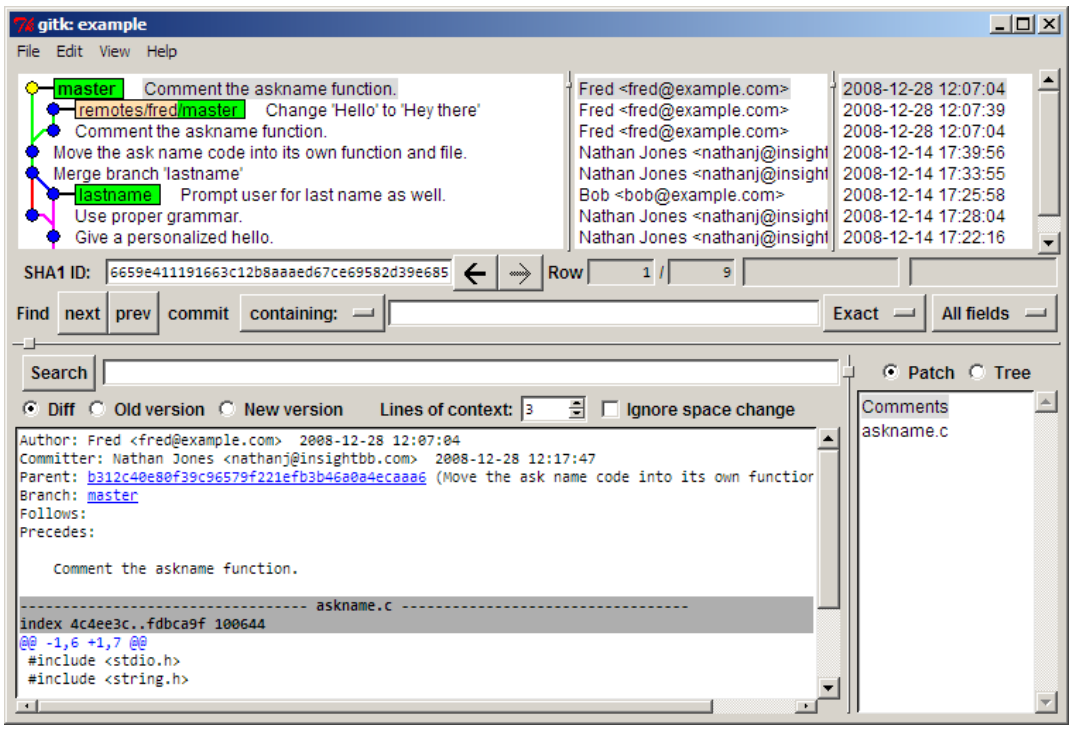
После скачивания, изменения Фреда были добавлены в наше локальное хранилище в remotes/fred/master ветку. Мы можем использовать gitk что бы визуализировать изменения, которые сделал Фред.



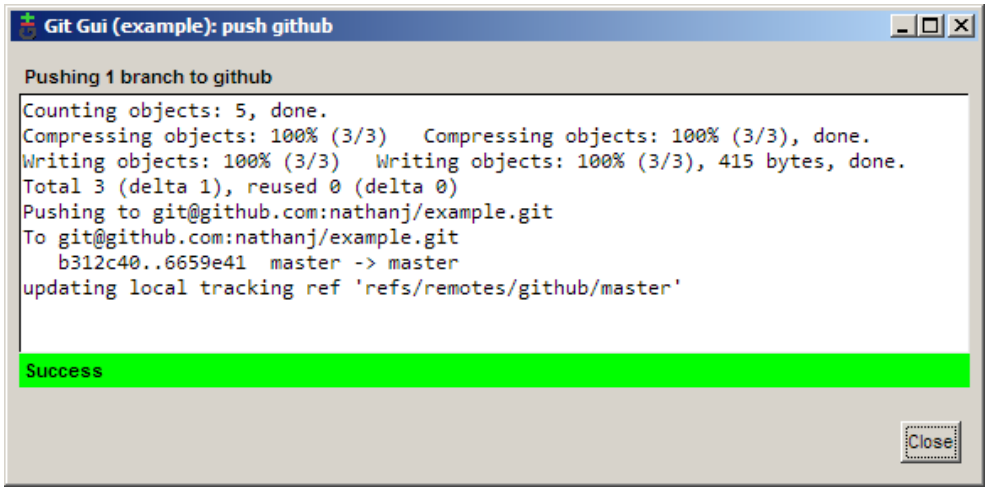
Если нам нравятся все изменения Фреда, мы можем сделать обычное слияние как было показано выше. Но всё же в этом случае, мне нравится только одно изменение Фреда. Что бы влить только одно из изменений Фреда, щёлкните правой кнопкой мыши на выбранной фиксации и выберите *Cherry-pick this commit (Скопировать это состояние)*. Фиксация будет влита в текущую ветку.



Результат:



Теперь мы можем опубликовать изменение Фреда в нашем хранилище на github-е, что бы все могли видеть и использовать его. При отправке надо проставить флаг *Перезаписать ветвь*.



Лабораторная работа№9. Последовательность работ с удаленным репозиторием

Предположим, что вы и несколько ваших напарников создали общественный репозиторий, чтобы заняться неким общим проектом. Как выглядит самая распространенная для git модель общей работы?

Первым делом создаем копию удаленного репозитория. Далее выполняем итерационно:

1. «Вытягиваем» последние обновления с удаленного репозитория;
2. смотрим, что же изменилось;
3. создаем новую ветвь и переключаемся в нее;
4. работаем в новой ветке, индексируем все изменения и создаем из них коммит;
5. переключаемся в главную ветвь,
6. обновляем ее из удаленного репозитория;
7. проводим слияние с веткой,
8. если есть конфликт, то разрешаем его и делаем коммит слияния.

9. закидываем изменения в удаленный репозиторий

4.1.2. Практическое задание

4.1.2.1. Порядок проведения.

Практические навыки проверяются путём выполнения обучающимися практических заданий в условиях, полностью или частично приближенных к условиям профессиональной деятельности. Проверяется знание теоретического материала, необходимое для правильного совершения необходимых действий, умение выстроить последовательность действий, практическое владение приёмами и методами решения профессиональных задач.

4.2.2.3. Критерии оценивания

Оценка «отлично» ставится, если обучающийся:

Задание выполнено полностью и правильно.

Оценка «хорошо» ставится, если обучающийся:

Задание выполнено полностью, но нет достаточного обоснования. Или при верном решении допущена ошибка или недочет, не влияющий на правильную последовательность рассуждений.

Оценка «удовлетворительно» ставится, если обучающийся:

Задание выполнено частично или с фактическими ошибками.

Оценка «неудовлетворительно» ставится, если обучающийся:

Задание не выполнено или выполнено с большим количеством фактических ошибок.

4.1.2.2. Содержание оценочного средства

Базовая часть:

- Запустить Git GUI или TortoiseGit (или консоль). Создать новый репозиторий (в папке по фамилии студента).
- Добавить в папку репозитория файлы. Зафиксировать состояние репозитория (выполнить commit).
- Внести изменения в файлы. Зафиксировать новое состояние репозитория.
- Создать новую ветку 1. Внести в нее изменения (добавить новый файл и изменить существующий файл: добавить, удалить и изменить строки) и зафиксировать их.
- Переключиться на ветку мастера. Внести в нее изменения (добавить новый файл; изменить существующие файлы: добавить, удалить и изменить строки первоначального файла) и зафиксировать их.
- Продемонстрировать слияние веток. Разрешить возникший конфликт.
- Просмотреть дерево изменений веток (историю).
- Продемонстрировать откат изменений в ветке 1.

Расширенная часть:

- Создать удаленный репозиторий (на github.com).
- Отправить данные на удаленный репозиторий (выполняется одним из студентов подгруппы). Добавить к удаленному репозиторию участников проекта.
- Получить данные из удаленного репозитория (выполняется прочими студентами).
- Изменить полученные данные.
- Зафиксировать изменения и отправить их на удаленный репозиторий (выполняется всеми студентами подгруппы).
- Получить данные из удаленного репозитория.
- Просмотреть историю изменений.

Дополнительно:

- Продемонстрировать работу revert и reset.
- Продемонстрировать сохранение изменений в stash с последующим восстановлением.
- Продемонстрировать создание и применение серии изменений (patch).
- Продемонстрировать создание и применение тегов.
- Продемонстрировать rebase.

4.2. Оценочные средства промежуточной аттестации

По дисциплине предусмотрен зачет в 6 семестре. Зачет проходит по билетам. В каждом билете два теоретических вопроса. Зачет проводится в устной / письменной и компьютерной форме. Оценивается владение материалом, его системное освоение, способность применять нужные знания, навыки и умения при анализе проблемных ситуаций.

4.2.1. Устный или письменный ответ на вопрос

4.2.1.1. Порядок проведения.

Устный или письменный ответ на вопрос направлен на проверку знаний основных разделов по дисциплине «Системы контроля версий программного обеспечения».

4.2.1.2. Критерии оценивания.

«Оценка «отлично» ставится, если обучающийся:

В ответе качественно раскрыл содержание темы. Ответ хорошо структурирован. Прекрасно освоен понятийный аппарат. Продемонстрирован высокий уровень понимания материала. Превосходное умение формулировать свои мысли, обсуждать дискуссионные положения.

Оценка «хорошо» ставится, если обучающийся:

Основные вопросы темы раскрыл. Структура ответа в целом адекватна теме. Хорошо освоен понятийный аппарат. Продемонстрирован хороший уровень понимания материала. Хорошее умение формулировать свои мысли, обсуждать дискуссионные положения.

Оценка «удовлетворительно» ставится, если обучающийся:

Тему частично раскрыл. Ответ слабо структурирован. Понятийный аппарат освоен частично. Понимание отдельных положений из материала по теме. Удовлетворительное умение формулировать свои мысли, обсуждать дискуссионные положения.

Оценка «неудовлетворительно» ставится, если обучающийся:

Тему не раскрыл. Понятийный аппарат освоен неудовлетворительно. Понимание материала фрагментарное или отсутствует. Неумение формулировать свои мысли, обсуждать дискуссионные положения.

4.2.1.3. Оценочные средства.

Вопросы для устного или письменного ответа

6 семестр

1. Что такое система управления версиями?
2. Как создать репозиторий?
3. Как создать ветку?
4. Как провести слияние? Как разрешить конфликт и что это такое?
5. Как зафиксировать изменения?
6. Как провести откат? Различия в reset и revert, мягкий и жесткий reset.
7. Какова последовательность действий при работе с локальным репозиторием?
8. Какова последовательность действий при работе с удаленным репозиторием?
9. Каковы возможности при работе с удаленным репозиторием? Как его клонировать, получать и отправлять данные?
10. Основные команды git.
11. Как работают команды reset, revert, rebase, stash, tag?
12. Как создавать и применять изменений (patch)? Что это такое?
13. Как можно отменить коммит в Git, если он уже был опубликован?
14. Что такое «staging area» или «index» в Git?
15. За что отвечает команда Git stash?
16. Как найти список файлов, которые изменились в определенном коммите?
17. За что отвечает команда «git config»?
18. Из чего состоит коммит в Git?
19. Как объединить несколько отдельных коммитов в один цельный коммит?
20. За что отвечает команда Git bisect? Как ее можно использовать для определения источника бага (регрессии)?
21. Как настроить Git-репозиторий для запуска инструментов проверки работоспособности кода непосредственно перед выполнением коммитов и предотвращения их в случае сбоя теста?
22. Какие вы знаете модели ветвления в Git? Опишите их.

Приложение 2
к рабочей программе дисциплины (модуля)
Б1.В.04.03 Системы контроля версий программного обеспечения

Перечень литературы, необходимой для освоения дисциплины (модуля)

Направление подготовки: 23.03.01 Технология транспортных процессов
Профиль подготовки: Проектирование и управление интеллектуальными транспортными системами
Квалификация выпускника: бакалавр
Форма обучения: заочное
Язык обучения: русский
Год начала обучения по образовательной программе: 2024

Основная литература:

1. Риз, Р. Обработка естественного языка на Java / Р. Риз ; пер. с англ. А.В. Снастина. - Москва : ДМК Пресс, 2016. - 264 с. - ISBN 978-5-97060-331-4. - Текст : электронный. - URL: https://znanium.com/catalog/product/1028050
2. Васюткина, И. А. Технология разработки объектно-ориентированных программ на JAVA / Васюткина И.А. - Новосибирск :НГТУ, 2012. - 152 с.: ISBN 978-5-7782-1973-1. - Текст : электронный. - URL: https://znanium.com/catalog/product/55711
3. Шеррингтон, М. Осваиваем язык Julia / М. Шеррингтон ; пер. с анг. А.В. Логунова. - Москва : ДМК Пресс, 2017. - 416 с. - ISBN 978-5-97060-370-3. - Текст : электронный. - URL: https://znanium.com/catalog/product/1028117
4. Плаксин, М. А. Тестирование и отладка программ для профессионалов будущих и настоящих / М. А. Плаксин. — 4-е изд., электрон. — Москва : Лаборатория знаний, 2020. — 170 с. - ISBN 978-5-00101-810-0. - Текст : электронный. - URL: https://znanium.com/catalog/product/1093870
5. Гагарина, Л. Г. Технология разработки программного обеспечения : учебное пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Сидорова-Виснадул ; под ред. Л.Г. Гагариной. — Москва : ФОРУМ : ИНФРА-М, 2022. — 400 с. — (Среднее профессиональное образование). - ISBN 978-5-8199-0812-9. - Текст : электронный. - URL: https://znanium.com/catalog/product/1794453

Приложение 3
к рабочей программе дисциплины (модуля)
Б1.В.04.03 Системы контроля версий программного обеспечения

Перечень информационных технологий, используемых для освоения дисциплины (модуля), включая перечень программного обеспечения и информационных справочных систем

Направление подготовки: 23.03.01 Технология транспортных процессов
Профиль подготовки: Проектирование и управление интеллектуальными транспортными системами
Квалификация выпускника: бакалавр
Форма обучения: заочное
Язык обучения: русский
Год начала обучения по образовательной программе: 2024

Освоение дисциплины (модуля) предполагает использование следующего программного обеспечения и информационно-справочных систем:
1. Операционная система Microsoft office professional plus 2010, или Microsoft Windows 7 Профессиональная, или Windows XP (Volume License)
2. Пакет офисного программного обеспечения Microsoft Office 365, или Microsoft office professional plus 2010
3. Adobe Reader XI или Adobe Acrobat Reader DC
4. Браузер Mozilla Firefox
5. Браузер Google Chrome
6. Kaspersky Endpoint Security для Windows
7. Программная система для обнаружения текстовых заимствований в учебных и научных работах. АО «Антиплагиат»
8. Электронная библиотечная система «ZNANIUM.COM»
9. Электронная библиотечная система Издательства «Лань»
10. Электронная библиотечная система «Консультант студента»