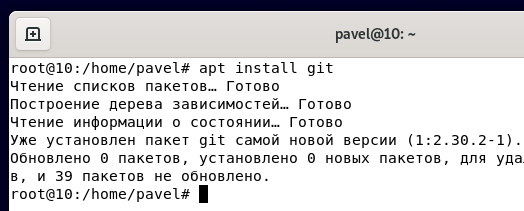
Лабораторная работа №9. Основы работы с системой управленияверсиями Git

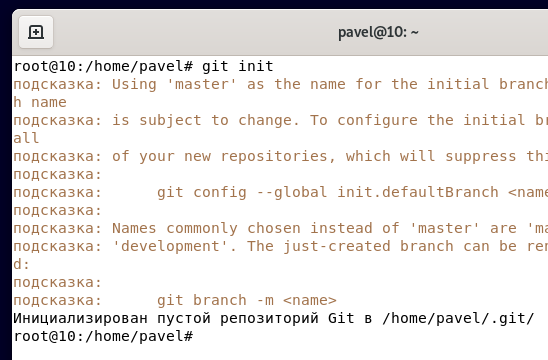
Цель: изучение системы контроля версий Git.  
ЗАДАНИЕ1. Изучите главу 11 теоретического курса «Системы управления версиями».

2. При необходимости установите пакет **git**.

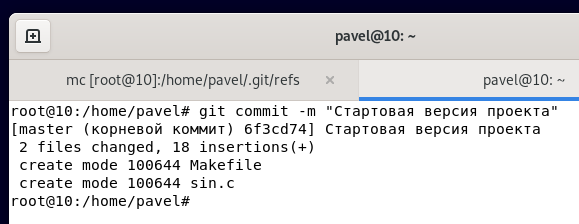


3. Выполните на рабочем компьютере все практические примеры из данной главы.

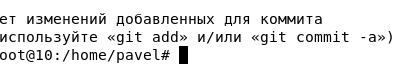
Пример 1:



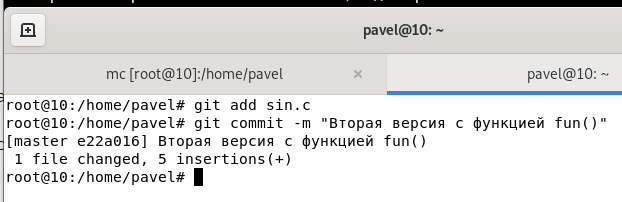
Пример 2



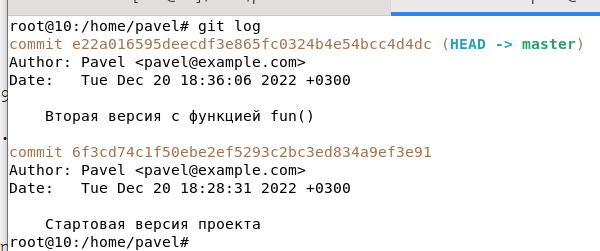
Пример 3 после изменения sin.c и команды git status:



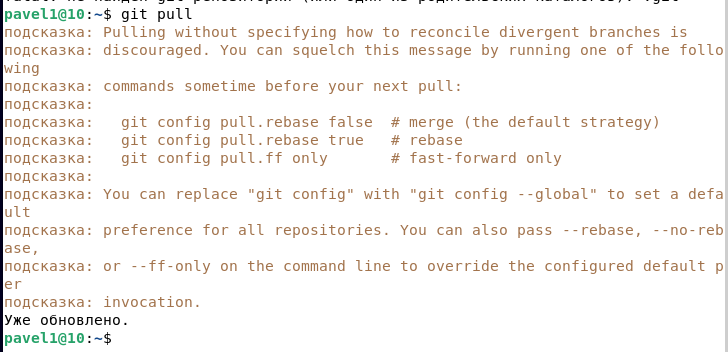
Пример 4:



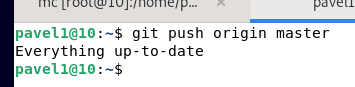
Пример 5:



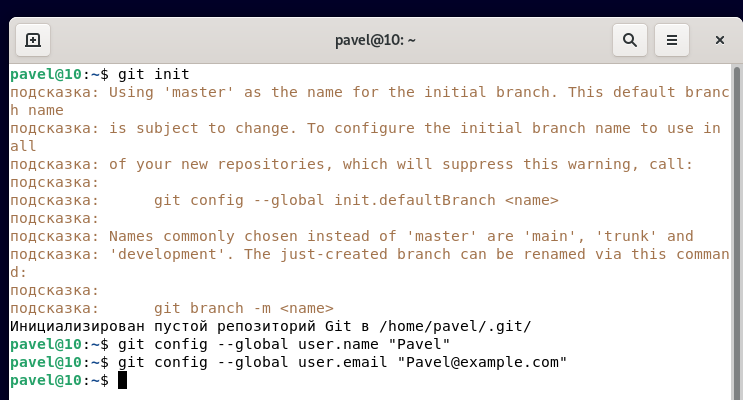
Пример 6:



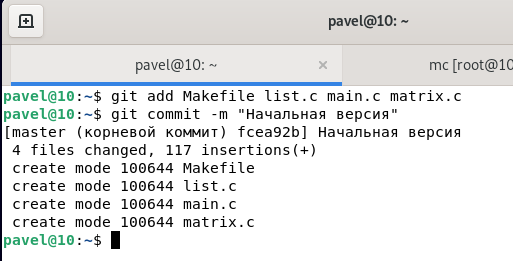
Пример 7:



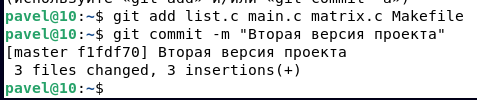
4. Создайте в рабочем каталоге пустой репозиторий Git. Настройте с помощью команд,приведенных ниже, осмысленное имя пользователя и адрес электронной почты:  
git config --global user.name "Your Name"  
git config --global user.email [you@example.com](mailto:you@example.com)



5. Скопируйте в рабочий каталог файлы исходного кода, созданные при выполнениилабораторной работы №6, и Make-файл, созданный при выполнении лабораторной работы №7.Укажите, что они включаются в репозиторий. Выполните коммит c пояснением «Начальная версияпроекта».

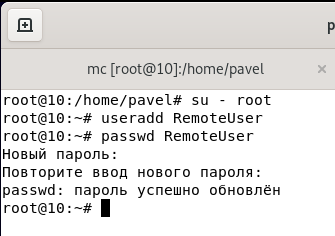


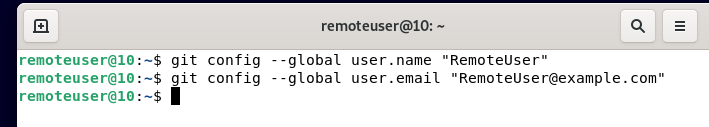
6. Внесите осмысленные изменения в файл исходного кода **list.c** и в Make-файл.Выполните сборку проекта. Изучите состояние файлов в рабочем каталоге командой **git status**. Добавьте измененные файлы к включению в очередной коммит. Выполните коммит спояснением «Вторая версия проекта».



7. Создайте пользователя RemoteUser и войдите от имени данного пользователя. Выполнитеклонирование репозитория в подкаталог рабочего каталога пользователя RemoteUser. Задайте дляданного пользователя осмысленное имя и адрес электронной почты (в системе Git).

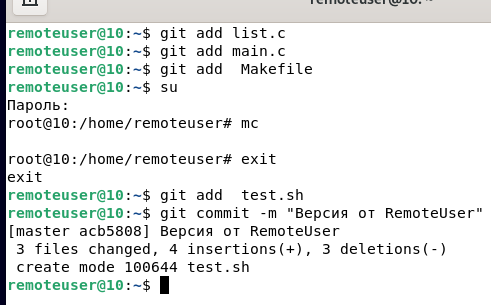
**Создал нового пользователя:**





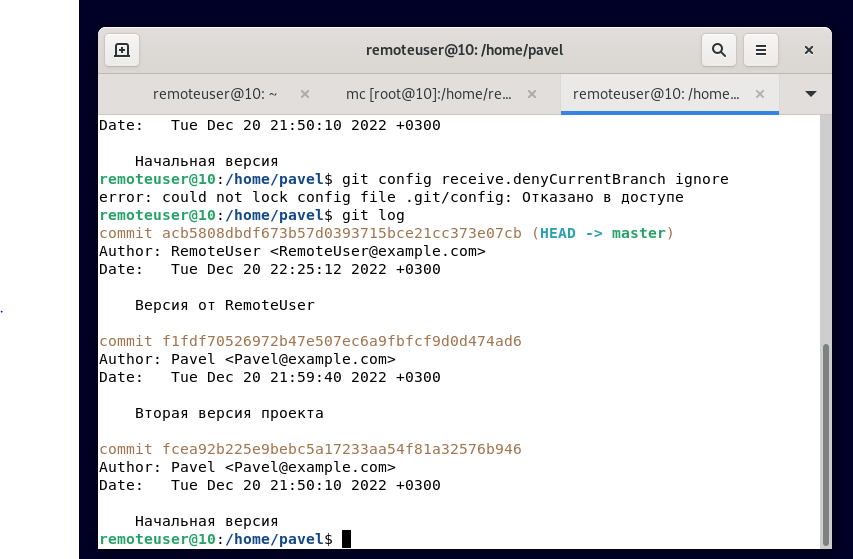
8. Внесите осмысленные изменения в файл исходного кода **list.c**, **main.c** и в Makeфайл. Выполните сборку проекта. Добавьте измененные файлы к включению в очередной коммит.Добавьте к проекту файл скрипта на языке Shell с несколькими осмысленными операторами иименем **test.sh**. Добавьте его для включения в коммит.

9. Выполните коммит с пояснением «Версия от RemoteUser».



10. Загрузите изменения, внесенные пользователем RemoteUser, обратно в исходныйрепозиторий.

11. Зайдите от имени основного пользователя User. Просмотрите историю версий проектакомандой **git log**. Проверьте текущее содержимое файлов исходного репозитория.



**КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Для чего предназначены системы управления версиями? Чем отличаются централизованные и распределенные системы управления версиями? Какие основные функции выполняет система управления версиями?

**Система управления версиями**(Version Control System, VCS) предназначена для отслеживания изменений в файлах исходного кода и других файлах проекта и управления этими изменениями. VCS организует хранение промежуточных версий файлов проекта, которые называются *версиями* или *ревизиями*.

В последнее время широкое распространение получили так называемые *распределенные* системы управления версиями. В таких системах отсутствует центральный репозиторий, а каждый пользователь имеет свою локальную версию репозитория. Текущая работа производится с локальным репозиторием, а крупные промежуточные версии синхронизируются с репозиториями других пользователей. Примерами распределенных систем контроля версий являются системы Git, Bazaar и Mercurial. В случае *бинарных файлов* все, что в большинстве случаев может сделать VSC — это заменить одну версию файла на другую версию файла (файл заменяется целиком). Объединение нескольких различных изменений в одном файле легко выполняется автоматически в том случае, если они затрагивают различные фрагменты данного файла. Если же несколько изменений затрагивают один и тот же фрагмент файла, то возникает *конфликт* (*conflict*). В случае конфликта, разработчик должен будет выполнить операцию объединения изменений (получить окончательную версию) вручную.

1. Поясните смысл терминов «update», «checkout», «commit», «merge» в контексте систем управления версиями.

Работа с системами управления версиями представляет собой следующую ежедневно повторяющуюся последовательность операций: загрузка изменений из репозитория, произведенных другими разработчиками, и объединение их с рабочими файлами. Для данной операции в большинстве систем используется термин «*update*». Если же мы хотим загрузить версию проекта из репозитория, полностью заменив рабочие файлы, то данная операция обычно называется «*checkout*»; внесение изменений в рабочую версию файлов на локальном компьютере; фиксация выполненных изменений в репозитории. Данная операция называется «*commit*». В русскоязычной литературе обычно пользуются русской калькой для данного слова – «*коммит*».

1. В каком состоянии может находится файл, включенный в проект в системе управления версиями Git?

В общем случае, файл, находящийся в рабочем каталоге, может находиться в одном из четырех состояний [11]:

* *untracked*— файл находится в рабочем каталоге, но не отслеживается системой управления версиями;
* *unmodified*— файл отслеживается системой управления версиями, но не изменялся с момента предыдущего коммита;
* *modified*— файл отслеживается системой управления версиями, изменен по сравнению с предыдущим коммитом, но не указан, как включаемый в следующий коммит;
* *staged*— файл отслеживается системой управления версиями, изменен по сравнению с предыдущим коммитом и *проиндексирован*– указан, как включаемый в следующий коммит.

1. Как производится переход из одного состояния файла в другое состояние в системе управления версиями Git?

