**Лабораторная работа №20**

**Изучение систем контроля версий GIT, GITHUB, GitLAB и др.**

Контрольные вопросы:

1. Опишите назначение систем CVS

Система CVS (Concurrent Versions System) используется для управления версиями исходного кода программного обеспечения. Она позволяет нескольким разработчикам работать над одним проектом одновременно, отслеживать изменения в коде и возвращаться к предыдущим версиям. CVS также обеспечивает контроль доступа к репозиторию кода, что позволяет ограничить доступ к конфиденциальной информации или изменениям только определенных разработчиков. CVS является одной из первых систем контроля версий и по-прежнему используется в некоторых проектах. Однако, существуют более современные системы контроля версий, такие как Git и Subversion, которые имеют более продвинутые функции и возможности.

1. Какие виды CVS вам известны? Какими особенностями они обладают?

Существует только один вид системы CVS (Concurrent Versions System), который используется для управления версиями исходного кода программного обеспечения. Однако, существует несколько различных реализаций CVS, таких как CVSNT и CVS Suite, которые имеют некоторые дополнительные функции и возможности. Например, CVSNT поддерживает работу с Windows-системами и имеет расширенный функционал для работы с файловой системой NTFS. CVS Suite включает в себя дополнительные инструменты для управления проектами, такие как трекер ошибок и систему управления задачами. Однако, основные функции и принципы работы CVS остаются одинаковыми во всех его реализациях.

1. На примере системы контроля версиями GIT опишите особенности подхода

к работе с данными. Какими преимуществами данная система обладает?

GIT является распределенной системой контроля версий, что означает, что каждый пользователь имеет полную копию всего репозитория, а не только его части. Это позволяет работать с данными в офлайн-режиме, а также делать локальные изменения без необходимости подключения к центральному серверу. Одной из особенностей GIT является использование хэш-функций для идентификации каждого коммита и файла в репозитории. Это позволяет обеспечить уникальность каждого объекта в системе и быстрый доступ к ним. GIT также имеет мощный механизм ветвления и слияния, который позволяет эффективно управлять процессом разработки и совместной работы. Каждый пользователь может создавать свои ветки, делать изменения и сливать их с другими ветками. Другим преимуществом GIT является возможность создания локальных репозиториев на компьютере пользователя, что делает процесс работы с данными более быстрым и удобным. В целом, GIT обладает высокой скоростью работы, гибкостью и удобством использования, что делает его одной из наиболее популярных систем контроля версий в мире разработки программного обеспечения.

1. Опишите процесс установки И настройки GIT на свой ПК

1. Скачайте установочный файл GIT с официального сайта https://git-scm.com/downloads и запустите его.

2. В процессе установки выберите необходимые опции, например, добавление GIT в системную переменную PATH или выбор редактора по умолчанию.

3. После завершения установки откройте командную строку (Windows) или терминал (Mac/Linux) и проверьте, что GIT установлен, введя команду "git --version".

4. Настройте свое имя пользователя и email для использования в GIT, введя команды "git config --global user.name 'Ваше имя'" и "git config --global user.email 'Ваш email'".

5. Создайте локальный репозиторий на своем компьютере, выбрав папку и введя команду "git init".

6. Добавьте файлы в репозиторий, используя команду "git add <имя файла>" или "git add." для добавления всех файлов в текущей директории.

7. Создайте коммит с описанием изменений, используя команду "git commit -m 'Описание изменений'".

8. Связывайте локальный репозиторий с удаленным, например, на GitHub, используя команду "git remote add origin <URL репозитория>".

9. Отправляйте изменения на удаленный репозиторий с помощью команды "git push origin <название ветки>".

10. Для получения изменений из удаленного репозитория используйте команду "git pull origin <название ветки>".

5. Опишите назначение сервиса Github Desktop.

GitHub - это веб-сервис для хранения, управления и совместной работы над проектами, использующими систему контроля версий Git. На GitHub пользователи могут создавать репозитории для своих проектов, загружать код, отслеживать изменения, создавать ветки для экспериментов и совместной работы, а также вносить вклад в чужие проекты через pull-запросы. GitHub также предоставляет возможность управления задачами и ошибками, а также интеграцию с другими сервисами для автоматизации разработки. Благодаря своей популярности и общедоступности GitHub стал платформой для совместной разработки открытого программного обеспечения и многих других проектов.

1. Опишите назначение сервиса Github Desktop.

GitHub Desktop - это приложение, которое предназначено для удобной работы с GitHub на компьютере. Оно позволяет пользователям создавать и клонировать репозитории, загружать и синхронизировать код, отслеживать изменения, создавать ветки и выполнять другие операции без необходимости использования командной строки Git. GitHub Desktop также предоставляет удобный интерфейс для работы с pull-запросами, задачами и ошибками. Это приложение является полезным инструментом для разработчиков, которые хотят упростить работу с GitHub и улучшить эффективность своей работы.

1. Поясните термины git и github:

1. Репозиторий - это хранилище файлов и версий кода, которое используется для управления проектом и совместной работы над ним.

2. Рабочая область - это локальный набор файлов и изменений, сделанных в проекте, а хранилище - это удаленное место, где хранятся версии кода и другие данные проекта.

3. Форк - это копия репозитория, созданная пользователем для работы над проектом независимо от оригинального репозитория.

4. Ветка - это отдельная линия разработки, которая может содержать свои изменения и коммиты.

5. Мастер - это главная ветка разработки, которая содержит последнюю стабильную версию кода.

6. Коммит - это сохранение изменений в репозитории, которое может содержать описание изменений и ссылки на задачи или ошибки.

7. Пул - это запрос на объединение изменений из одной ветки в другую.

8. Пуш - это отправка изменений из локальной рабочей области в удаленное хранилище.

9. Пулреквест - это запрос на объединение изменений из форка в оригинальный репозиторий.

10. Мердж - это объединение изменений из одной ветки в другую.

11. Кодревью - это процесс проверки кода другими разработчиками, который помогает улучшить качество и безопасность кода.