**Лабораторная работа №20**

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

1. Опишите назначения систем CVS

Система управления версиями (Система контроля версий, CVS) — это программное обеспечение, которое используется для отслеживания изменений в исходном коде и координации работы между несколькими разработчиками. CVS является одной из первых и наиболее широко распространенных систем управления версиями.

Основное назначение системы CVS заключается в контроле версий файлов исходного кода. Она позволяет разработчикам отслеживать изменения, вносимые в код, и сохранять историю изменений. Это полезно при работе над проектами, в которых несколько разработчиков одновременно вносят изменения в код. CVS помогает предотвратить конфликты изменений и обеспечить согласованность кодовой базы.

Некоторые основные назначения системы CVS:

1. Контроль версий: CVS позволяет разработчикам отслеживать все изменения, внесенные в файлы, и восстанавливать предыдущие версии при необходимости. Это позволяет вам вернуться к предыдущей работоспособной версии кода в случае возникновения проблем.
2. Совместная разработка: CVS поддерживает совместную работу нескольких разработчиков над одним проектом. Разработчики могут работать над разными частями кода, а CVS будет автоматически объединять их изменения.
3. Управление ветвлением: CVS позволяет создавать и управлять ветками в кодовой базе. Ветвление полезно, когда вы хотите разрабатывать новую функциональность или исправлять ошибки, не затрагивая основную рабочую версию проекта.
4. Слежение за изменениями: CVS сохраняет историю изменений, включая информацию о том, кто и когда внес изменения. Это полезно для отслеживания прогресса разработки, а также для определения, кто ответственен за определенные изменения или возникновение проблем.
5. Восстановление исходного кода: CVS позволяет восстановить исходный код в любой момент времени, даже если он был удален или испорчен. Это обеспечивает защиту от потери данных и позволяет вернуться к предыдущим версиям кода.
6. Какие виды CVS вам известны? Какими особенностями они обладают?

Существует несколько видов систем управления версиями (Систем контроля версий, CVS). Некоторые из наиболее широко используемых включают:

* 1. CVS (Concurrent Versions System): CVS является одним из первых и наиболее распространенных инструментов для контроля версий. Он поддерживает совместную разработку, ветвление и слияние изменений. Однако CVS имеет некоторые ограничения, такие как отсутствие поддержки распределенной работы и сложности работы с бинарными файлами.
  2. SVN (Subversion): SVN является более современной альтернативой CVS. Он сохраняет историю изменений в централизованном хранилище и поддерживает совместную разработку, ветвление и слияние. SVN также имеет более надежную поддержку для работы с бинарными файлами и обладает некоторыми расширенными функциями, такими как атомарные коммиты.
  3. Git: Git является распределенной системой управления версиями и стала очень популярной среди разработчиков. Она предлагает высокую производительность, мощные возможности ветвления и слияния, а также локальное хранение истории изменений. Git также обладает функциями, такими как возможность работы в автономном режиме и возможность легкого переключения между разными ветками.
  4. Mercurial: Mercurial также является распределенной системой управления версиями и предоставляет схожий набор функций с Git. Она обладает простым и понятным интерфейсом, а также поддерживает совместную разработку, ветвление и слияние.

1. На примере системы контроля версиями GIT опишите особенности подхода к работе с данными. Какими преимуществами данная система обладает?

GIT является одной из наиболее популярных распределенных систем управления версиями. Вот некоторые особенности и преимущества работы с данными в системе GIT:

* 1. Распределенная модель: В отличие от централизованных систем управления версиями, в GIT каждый разработчик имеет полную копию репозитория. Это позволяет работать с данными и вносить изменения в код в автономном режиме, без необходимости подключения к центральному серверу. Каждая копия репозитория содержит всю историю изменений, что обеспечивает надежность и удобство работы.
  2. Ветвление и слияние: GIT предоставляет мощные возможности ветвления и слияния. Разработчики могут создавать новые ветки, чтобы работать над различными функциональностями или исправлениями ошибок, не затрагивая основную ветку разработки. После завершения работы ветки могут быть безопасно слиты в основную ветку. Это позволяет эффективно организовывать и согласовывать работу нескольких разработчиков.
  3. История изменений: GIT сохраняет полную историю изменений в репозитории. Каждое изменение, коммит или ветка фиксируются и могут быть легко просмотрены и отслежены. Это позволяет разработчикам возвращаться к предыдущим версиям кода, отменять изменения и анализировать историю разработки.
  4. Легкость переключения между ветками: GIT обладает быстрым и эффективным механизмом переключения между ветками. Разработчики могут быстро переключаться между различными ветками, чтобы проверить различные версии кода, вносить исправления и тестировать функциональность. Это обеспечивает гибкость и удобство при работе с разными ветками разработки.
  5. Резервное копирование и восстановление: GIT предлагает надежное резервное копирование данных и возможность восстановления репозитория в случае потери или повреждения данных. Каждая копия репозитория содержит полную историю изменений, что позволяет восстановить код на любом этапе разработки.
  6. Широкая поддержка и экосистема: GIT является широко распространенной системой управления версиями и имеет большое сообщество разработчиков. Это обеспечивает доступ к обширной документации, ресурсам и инструментам, которые упрощают работу с GIT и расширяют его возможности.

1. Опишите процесс установки и настройки GIT на свой ПК.

Установка и настройка Git на вашем ПК включает несколько шагов. Вот пошаговая инструкция:

Шаг 1: Загрузка Git

Первым делом вам нужно загрузить Git для вашей операционной системы. Вы можете скачать его с официального сайта Git (https://git-scm.com/downloads). Просто выберите версию для своей операционной системы и следуйте инструкциям для загрузки.

Шаг 2: Установка Git

После загрузки установочного файла запустите его и следуйте инструкциям мастера установки. Обычно вам будет предложено выбрать путь установки, компоненты для установки и другие параметры. Если вы не уверены, какие параметры выбрать, оставьте значения по умолчанию.

Шаг 3: Проверка установки

После завершения установки Git вы можете проверить, работает ли он на вашем ПК. Откройте командную строку (в Windows это может быть командная строка или PowerShell, в Linux или macOS - терминал) и введите команду:

git --version

Если Git успешно установлен, вы увидите вывод текущей установленной версии Git.

Шаг 4: Настройка Git

Теперь вам нужно настроить некоторые параметры в Git, такие как ваше имя пользователя и адрес электронной почты. Введите следующие команды в командной строке, заменив "Your Name" на ваше имя и "your.email@example.com" на ваш адрес электронной почты:

git config --global user.name "Your Name"

git config --global user.email "your.email@example.com"

Эти параметры будут использоваться для идентификации вас в коммитах, которые вы будете делать в Git.

Шаг 5: Генерация SSH-ключа (необязательно)

Если вы планируете использовать SSH для аутентификации при работе с удаленными репозиториями Git, вы можете сгенерировать SSH-ключ. Вам понадобится утилита, которая поставляется с установкой Git и называется "Git Bash" (в Windows) или просто терминал (в Linux или macOS). Запустите Git Bash или терминал и выполните команду:

ssh-keygen -t rsa -b 4096 -C "your.email@example.com"

Затем следуйте инструкциям по созданию SSH-ключа. Это сгенерирует два файла, приватный ключ (обычно с именем "id\_rsa") и публичный ключ (с расширением ".pub"). Публичный ключ вам потребуется для настройки доступа к удаленным репозиториям.

Шаг 6: Готово!

Поздравляю, у вас теперь установлен и настроен Git на вашем ПК! Вы можете начать использовать Git для управления версиями вашего кода и сотрудничества с другими разработчиками.

**Контрольные вопросы:**

1. GitHub - это веб-сервис, предоставляющий хостинг для Git-репозиториев и дополнительные инструменты для совместной работы над проектами. Он позволяет разработчикам хранить, управлять и совместно работать над своим кодом.

Основное назначение GitHub включает:

* 1. Хостинг репозиториев: GitHub предоставляет облачное хранилище для Git-репозиториев, где разработчики могут хранить свой код. Это позволяет им иметь резервные копии своих проектов, а также легко обмениваться кодом с другими разработчиками.
  2. Совместная работа: GitHub облегчает совместную работу над проектами. Разработчики могут приглашать других пользователей к сотрудничеству над своими репозиториями, делиться кодом, обсуждать изменения и вносить свои вклады через запросы на включение изменений (pull requests).
  3. Отслеживание задач: GitHub предоставляет инструменты для отслеживания задач и управления проектами. С помощью функций, таких как задачи (issues) и проектные доски (project boards), команды могут создавать, отслеживать и решать задачи, планировать работы и управлять прогрессом разработки.
  4. Ветвление и слияние кода: GitHub упрощает ветвление и слияние кода. Разработчики могут создавать отдельные ветки для работы над определенными функциями или исправлениями, а затем объединять их с основной веткой с помощью запросов на включение изменений. Это позволяет эффективно управлять изменениями и контролировать кодовую базу.
  5. Интеграции и автоматизация: GitHub предлагает множество интеграций с другими сервисами и инструментами разработки, такими как системы непрерывной интеграции (Continuous Integration) и системы управления задачами. Это позволяет автоматизировать различные процессы разработки и улучшить эффективность команды.
  6. Контроль версий и история изменений: GitHub базируется на системе контроля версий Git, поэтому он предоставляет мощные возможности для отслеживания изменений в коде, восстановления предыдущих версий и анализа истории изменений. Разработчики могут просматривать коммиты, вносить правки и анализировать различные ветви.

1. Опишите назначение сервиса Github Desktop

GitHub Desktop - это графический клиент для работы с Git и сервисом GitHub. Он предоставляет простой и интуитивно понятный интерфейс, который облегчает работу с Git-репозиториями и управление проектами на GitHub.

Назначение GitHub Desktop включает:

* 1. Удобный интерфейс: GitHub Desktop предоставляет графический интерфейс, который делает работу с Git более доступной и интуитивно понятной, особенно для новичков. Вы можете просматривать изменения в коде, создавать новые ветки, коммитить изменения и выполнять другие операции Git с помощью простых визуальных элементов.
  2. Простота использования: GitHub Desktop разработан с упором на простоту использования, что позволяет разработчикам сфокусироваться на своей работе, а не на сложностях работы с Git командной строкой. Он предоставляет прямой доступ к основным функциям Git, таким как создание веток, коммиты, слияния и синхронизация с удаленными репозиториями.
  3. Интеграция с GitHub: GitHub Desktop непосредственно интегрируется с платформой GitHub, что обеспечивает простое взаимодействие с вашими репозиториями на GitHub. Вы можете легко клонировать репозитории, созданные на GitHub, и делать коммиты, отправлять запросы на включение изменений и синхронизировать свои изменения с удаленными репозиториями.

1. Контроль версий и история изменений: GitHub Desktop позволяет просматривать историю изменений, вносить правки и восстанавливать предыдущие версии файлов. Вы можете легко перемещаться по коммитам, сравнивать изменения и откатывать нежелательные изменения.
2. Уведомления и совместная работа: GitHub Desktop предоставляет уведомления о новых коммитах, запросах на включение изменений и других событиях ваших репозиториев. Это упрощает совместную работу с другими разработчиками и позволяет отслеживать активность в ваших проектах.
3. Поясните термины git и github

Git - это распределенная система контроля версий (Version Control System, VCS), разработанная Линусом Торвальдсом. Она предназначена для управления версиями кода и отслеживания изменений в проектах различной сложности. Git позволяет разработчикам эффективно сотрудничать над проектами, отслеживать изменения, вносить правки и восстанавливать предыдущие версии файлов.

GitHub - это веб-сервис и хостинг-платформа для Git-репозиториев. Он предоставляет разработчикам возможность хранить, управлять и совместно работать над исходным кодом. GitHub предлагает дополнительные функции и инструменты, такие как система отслеживания задач, запросы на включение изменений, интеграции и другие.

1. Репозиторий

Репозиторий (Repository) - это основная структура данных в системе контроля версий (Version Control System, VCS), такой как Git или GitHub. Он представляет собой хранилище для файлов и папок, связанных с проектом, а также сохраняет историю изменений этих файлов.

1. Рабочая область и хранилище

Рабочая область (Working Directory): Рабочая область, также известная как рабочий каталог или рабочее дерево, представляет собой локальное пространство файловой системы, где вы работаете над проектом. Она содержит все файлы и папки проекта, над которыми вы вносите изменения. Когда вы клонируете репозиторий (repository) или создаете новый, Git копирует все файлы в рабочую область, и вы можете вносить изменения в эти файлы.

Хранилище (Repository): Хранилище, в контексте системы контроля версий, такой как Git или GitHub, представляет собой централизованное место, где хранится история изменений вашего проекта. Оно содержит все файлы и папки проекта, а также метаданные, связанные с контролем версий, такие как коммиты, ветки, теги и другие. Хранилище Git сохраняет историю изменений, позволяет вам возвращаться к предыдущим версиям кода, отслеживать изменения, сливать (merge) изменения из разных веток и совместно работать с другими разработчиками.

1. Форк

Форк (Fork) - это процесс создания независимой копии репозитория (repository) в системе контроля версий, такой как Git. Форк позволяет вам создать свою собственную копию репозитория, чтобы работать над проектом независимо от оригинального репозитория.

1. Ветка

Ветка (Branch) - это механизм в системе контроля версий (Version Control System, VCS), такой как Git, который позволяет создавать параллельные линии разработки внутри репозитория. Ветка представляет собой независимую линию разработки, которая может содержать свою собственную последовательность коммитов (commits) и изменений файлов.

1. Мастер

В контексте системы контроля версий (Version Control System, VCS), такой как Git, термин "master" используется для обозначения основной ветки (main branch) в репозитории. Основная ветка обычно называется "master" или "main" и представляет собой главную линию разработки проекта.

1. Коммит

Коммит (Commit) - это основная операция в системе контроля версий (Version Control System, VCS), такой как Git. Коммит представляет собой сохранение изменений в репозитории и создание точки в истории проекта.

1. Пул

Пул (Pull) - это операция в системе контроля версий, которая позволяет получить (получить) и объединить (сливать) изменения из удаленного репозитория в локальный репозиторий.

1. Пуш

Пуш (Push) - это операция в системе контроля версий (Version Control System, VCS), такой как Git, которая позволяет отправить (запушить) локальные изменения ветки в удаленный репозиторий.

1. Пулрекквест

Pull Request (пулреквест) - это механизм в системах контроля версий, таких как Git и GitHub, который позволяет разработчикам предлагать изменения в основной репозиторий проекта.

1. Мердж

Мердж (Merge) - это операция в системе контроля версий, которая позволяет объединить (слияние) изменения из одной ветки в другую.

1. Кодревью

Кодревью (Code Review) - это процесс, в рамках которого другие разработчики или члены команды проверяют код, написанный одним из участников проекта. Цель кодревью состоит в том, чтобы оценить качество кода, обнаружить потенциальные проблемы, предложить улучшения и обеспечить соответствие стандартам разработки.