Основы работы с репозиторием

Преподаватели: Эпштейн Леонид Борисович

Репозиторий: основные понятия

Репозиторий (хранилище) — место, где хранятся и поддерживаются какие-либо данные.

Система управления версиями (Version Control System, VCS или Revision Control System) — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение и т. д.

Наиболее распространенные системы: **RCS, CVS, Subversion, Aegis, Monoton, Git, Bazaar, Arch, Perforce, Mercurial, TFS**

RCS - система управления пересмотрами версий (www.gnu.org/software/rcs/rcs.html)

RCS (Revision Control System – система управления пересмотрами версий), разработанной в 1985 году. RCS позволяет работать только с отдельными файлами, создавая для каждого историю изменений. Для текстовых файлов сохраняются не все версии файла, а только последняя версия и все изменение, внесенные в нее. RCS также может отслеживать изменения в бинарных файлах, но при этом каждое изменение хранится в виде отдельной версии файла.

Когда изменения в файл вносит один из пользователей, для всех остальных этот файл остается заблокированным. Они не могут запросить его из репозитория для редактирования, пока первый пользователь не закончит работу и не зафиксирует изменения.

RCS - система управления пересмотрами версий (www.gnu.org/software/rcs/rcs.html)

Преимущества:

- RCS - проста в использовании и хорошо подходит для ознакомления с принципами работы систем контроля версий.
- Хорошо подходит для резервного копирования отдельных файлов, не требующих частого изменения группой пользователей.
- Широко распространена и предустановлена в большинстве свободно распространяемых операционных системах.

Недостатки:

- Отслеживает изменения только отдельных файлов, что не позволяет использовать ее для управления версиями больших проектов.
- Не позволяет одновременно вносить изменения в один и тот же файл несколькими пользователями.
- Низкая функциональность, по сравнению с современными системами контроля версий.

Выводы:

Система контроля версий RCS предоставляет слишком слабый набор инструментов для управления разрабатываемыми проектами и подходит разве что для ознакомления с технологией контроля версий или ведения небольшой истории откатов отдельных файлов.

CVS - система управления параллельными версиями (www.nongnu.org/cvs)

Система управления параллельными версиями (Concurrent Versions System) – логическое развитие системы управления пересмотрами версий (RCS), использующая ее стандарты и алгоритмы по управлению версиями, но значительно более функциональная, и позволяющая работать не только с отдельными файлами, но и с целыми проектами. CVS основана на технологии клиент-сервер, взаимодействующих по сети. Клиент и сервер также могут располагаться на одной машине, если над проектом работает только один человек, или требуется вести локальный контроль версий.

Последняя версия и все сделанные изменения хранятся в репозитории сервера. Клиенты, подключаясь к серверу, проверяют отличия локальной версии от последней версии, сохраненной в репозитории, и, если есть отличия, загружают их в свой локальный проект. При необходимости решают конфликты и вносят требуемые изменения в разрабатываемый продукт. После этого все изменения загружаются в репозиторий сервера. CVS, при необходимости, позволяет откатываться на нужную версию разрабатываемого проекта и вести управление несколькими проектами одновременно.

CVS - система управления параллельными версиями (www.nongnu.org/cvs)

Достоинства:

- Несколько клиентов могут одновременно работать над одним и тем же проектом.
- Позволяет управлять не одним файлом, а целыми проектами.
- Обладает огромным количеством удобных графических интерфейсов, способных удовлетворить практически любой, даже самый требовательный вкус.
- При загрузке тестовых файлов из репозитория передаются только изменения, а не весь файл целиком.

Недостатки:

- При перемещении или переименовании файла или директории теряются все, привязанные к этому файлу или директории, изменения.
- Сложности при ведении нескольких параллельных веток одного и того же проекта.
- Ограниченная поддержка шрифтов.
- Для каждого изменения бинарного файла сохраняется вся версия файла, а не только внесенное изменение.
- С клиента на сервер измененный файл всегда передается полностью.
- Ресурсоемкие операции, так как требуют частого обращения к репозиторию, и сохраняемые копии имеют некоторую избыточность.

Выводы:

Несмотря на то, что CVS устарела и обладает серьезными недостатками, она все еще является одной из самых популярных систем контроля версий и отлично подходит для управления небольшими проектами, не требующих создания нескольких параллельных версий, которые надо периодически объединять. CVS можно порекомендовать, как промежуточный шаг в освоении работы систем контроля версий, ведущий к более мощным и современным видам таких программ.

Система управления версиями Subversion (www.subversion.tigris.org)

Subversion – эта централизованная система управления версиями, созданная в 2000 году и основанная на технологии клиент-сервер. Она обладает всеми достоинствами CVS и решает основные ее проблемы (переименование и перемещение файлов и каталогов, работа с двоичными файлами и т.д.). Часто ее называют по имени клиентской части – SVN.

Принцип работы с Subversion очень походит на работу с CVS. Клиенты копируют изменения из репозитория и объединяют их с локальным проектом пользователя. Если возникают конфликты локальных изменений и изменений, сохраненных в репозитории, то такие ситуации разрешаются вручную. Затем в локальный проект вносятся изменения, и полученный результат сохраняется в репозитории.

При работе с файлами, не позволяющими объединять изменения, может использоваться следующий принцип:

- Файл скачивается из репозитория и блокируется (запрещается его скачивание из репозитория).
- Вносятся необходимые изменения.
- Загружается файл в репозиторий и разблокируется (разрешается его скачивание из репозитория другим клиентам).

Во многом, из-за простаты и схожести в управлении с CVS, но в основном, из-за своей широкой функциональности, Subversion с успехом конкурирует с CVS и даже успешно ее вытесняет.

Система управления версиями Subversion (www.subversion.tigris.org)

Достоинства:

- Система команд, схожая с CVS.
- Поддерживается большинство возможностей CVS.
- Разнообразные графические интерфейсы и удобная работа из консоли.
- Отслеживается история изменения файлов и каталогов даже после их переименования и перемещения.
- Высокая эффективность работы, как с текстовыми, так и с бинарными файлами.
- Возможность создания зеркальных копий репозитория.
- Наличие удобного механизма создания меток и ветвей проектов.
- Можно с каждым файлом и директорией связать определенный набор свойств, облегчающий взаимодействие с системой контроля версии.
- Широкое распространение позволяет быстро решить большинство возникающих проблем, обратившись к данным, накопленным Интернет-сообществом.

Система управления версиями Subversion (www.subversion.tigris.org)

Недостатки:

- Полная копия репозитория хранится на локальном компьютере в скрытых файлах, что требует достаточно большого объема памяти.
- Существуют проблемы с переименованием файлов, если переименованный локально файл одним клиентом был в это же время изменен другим клиентом и загружен в репозиторий.
- Слабо поддерживаются операции слияния веток проекта.
- Сложности с полным удалением информации о файлах попавших в репозиторий, так как в нем всегда остается информация о предыдущих изменениях файла, и непредусмотрено никаких штатных средств для полного удаления данных о файле из репозитория.

Выводы:

Subversion – современная система контроля версий, обладающая широким набором инструментов, позволяющих удовлетворить любые нужды для управления версиями проекта с помощью централизованной системы контроля. В Интернете множество ресурсов посвящено особенностям Subversion, что позволяет быстро и качественно решать все возникающие в ходе работы проблемы. Простота установки, подготовки к работе и широкие возможности позволяют ставить subversion на одну из лидирующих позиций в конкурентной гонке систем контроля версий.

Система управления версиями Git. (www.git-scm.com)

С февраля 2002 года для разработки ядра Linux'а большинством программистов стала использоваться система контроля версий BitKeeper. Довольно долгое время с ней не возникало проблем, но в 2005 году Лари МакВоем (разработчик BitKeeper'а) отозвал бесплатную версию программы. Разрабатывать проект масштаба Linux без мощной и надежной системы контроля версий – невозможно. Одним из кандидатов и наиболее подходящим проектом оказалась система контроля версий Monotine, но Торвальдса Линуса не устроила ее скорость работы. Так как особенности организации Monatone не позволяли значительно увеличить скорость обработки данных, то 3 апреля 2005 года Линус приступил к разработке собственной системы контроля версий – Git. Практически одновременно с Линусом (на три дня позже), к разработке новой системы контроля версий приступил и Мэтт Макал. Свой проект Мэтт назвал Mercurial.

Git – это гибкая, распределенная (без единого сервера) система контроля версий, дающая массу возможностей для изменения, дополнения и ведения документации, и для многих других направлений, требующих управления историей изменений.

У каждого разработчика, использующего Git, есть свой локальный репозиторий, позволяющий локально управлять версиями. Затем, сохраненными в локальный репозиторий данными, можно обмениваться с другими пользователями. Часто при работе с Git создают центральный репозиторий, с которым остальные разработчики синхронизируются. В этом случае все участники проекта ведут свои локальны разработки и беспрепятственно скачивают обновления из центрального репозитория. Когда необходимые работы отдельными участниками проекта выполнены и отлажены, они, после удостоверения владельцем центрального репозитория в корректности и актуальности проделанной работы, загружают свои изменения в центральный репозиторий. Наличие локальных репозиторием также значительно повышает надежность хранения данных, так как, если один из репозиториев выйдет из строя, данные могут быть легко восстановлены из других репозиториев. Работа над версиями проекта в Git может вестись в нескольких ветках, которые затем могут с легкостью полностью или частично объединяться, уничтожаться, откатываться и разрастаться во все новые и новые ветки проекта.

Система управления версиями Git. (www.git-scm.com)

Достоинства:

- Надежная система сравнения ревизий и проверки корректности данных, основанные на алгоритме хеширования SHA1 (Secure Hash Algorithm 1).
- Гибкая система ветвления проектов и слияния веток между собой.
- Наличие локального репозитория, содержащего полную информацию обо всех изменениях, позволяет вести полноценный локальный контроль версий и заливать в главный репозиторий только полностью прошедшие проверку изменения.
- Высокая производительность и скорость работы.
- Удобный и интуитивно понятный набор команд.
- Множество графических оболочек, позволяющих быстро и качественно вести работы с Git'ом.
- Возможность делать контрольные точки, в которых данные сохраняются без дельта компрессии, а полностью. Это позволяет уменьшить скорость восстановления данных, так как за основу берется ближайшая контрольная точка, и восстановление идет от нее. Если бы контрольные точки отсутствовали, то восстановление больших проектов могло бы занимать часы.
- Широкая распространенность, легкая доступность и качественная документация.
- Гибкость системы позволяет удобно ее настраивать и даже создавать специализированные контроля системы или пользовательские интерфейсы на базе git.
- Универсальный сетевой доступ с использованием протоколов http, ftp, rsync, ssh и др.

Система управления версиями Git. (www.git-scm.com)

Недостатки:

- Unix – ориентированность. На данный момент отсутствует зрелая реализация Git, совместимая с другими операционными системами.
- Возможные (но чрезвычайно низкие) совпадения хеш - кода отличных по содержанию ревизий.
- Не отслеживается изменение отдельных файлов, а только всего проекта целиком, что может быть неудобно при работе с большими проектами, содержащими множество несвязных файлов.
- При начальном (первом) создании репозитория и синхронизации его с другими разработчиками, потребуется достаточно длительное время для скачивания данных, особенно, если проект большой, так как требуется скопировать на локальный компьютер весь репозиторий.

Выводы:

Git – гибкая, удобная и мощная система контроля версий, способная удовлетворить абсолютное большинство пользователей. Существующие недостатки постепенно удаляются и не приносят серьезных проблем пользователям. Если вы ведете большой проект, территориально удаленный, и тем более, если часто приходится разрабатывать программное обеспечение, не имея доступа к другим разработчикам (например, вы не хотите терять время при перелете из страны в страну или во время поездки на работу), можно делать любые изменения и сохранять их в локальном репозитории, откатываться, переключаться между ветками и т.д.). Git – один из лидеров систем контроля версий.

Система управления версиями Mercurial. (mercurial.selenic.com)

Распределенная система контроля версий Mercurial разрабатывалась Мэттом Макалом параллельно с системой контроля версий Git, созданной Торвальдсом Линусом.

Первоначально, она была создана для эффективного управления большими проектами под Linux'ом, а поэтому была ориентирована на быструю и надежную работу с большими репозиториями. На данный момент mercurial адаптирован для работы под Windows, Mac OS X и большинство Unix систем.

Большая часть системы контроля версий написана на языке Python, и только отдельные участки программы, требующие наибольшего быстродействия, написаны на языке Си.

Идентификация ревизий происходит на основе алгоритма хеширования SHA1 (Secure Hash Algorithm 1), однако, также предусмотрена возможность присвоения ревизиям индивидуальных номеров.

Так же, как и в git'е, поддерживается возможность создания веток проекта с последующим их слиянием.

Для взаимодействия между клиентами используются протоколы HTTP, HTTPS или SSH.

Набор команд - простой и интуитивно понятный, во многом схожий с командами subversion. Так же имеется ряд графических оболочек и доступ к репозиторию через веб-интерфейс. Немаловажным является и наличие утилит, позволяющих импортировать репозитории многих других систем контроля версий.

Система управления версиями Mercurial. (mercurial.selenic.com)

Достоинства:

- Быстрая обработка данных.
- Кросплатформенная поддержка.
- Возможность работы с несколькими ветками проекта.
- Простота в обращение.
- Возможность конвертирования репозиториев других систем поддержки версий, таких как CVS, Subversion, Git, Darcs, GNU Arch, Bazaar и др.

Недостатки:

- Возможные (но чрезвычайно низкие) совпадения хеш - кода отличных по содержанию ревизий.
- Ориентирован на работу в консоли.

Выводы:

Простой и отточенный интерфейс, и набор команд, возможность импортировать репозитории с других систем контроля версий, - сделают переход на Mercurial и обучение основным особенностями безболезненным и быстрым. Вряд ли это займет больше нескольких дней.

Надежность и скорость работы позволяют использовать его для контроля версий огромных проектов. Все это делает mercurial достойным конкурентом git’a.

Команды SVN

Типичная итерация рабочего цикла с Subversion включает следующие этапы.

- Обновление рабочей копии из хранилища (svn update) или её создание (svn checkout).
- Изменение рабочей копии. Изменения директорий и информации о файлах производится средствами Subversion, в *изменении* же (содержимого) файлов Subversion никак не задействован — изменения производятся программами, предназначенными для этого (текстовые редакторы, средства разработки и т. п.):
  - новые (еще не зафиксированные в хранилище) файлы и директории нужно *добавить* (команда svn add), то есть передать под управление версиями;
  - если файл или директорию в рабочей копии нужно *удалить*, *переименовать*, *переместить* или *скопировать*, необходимо использовать средства Subversion (svn mkdir, svn delete, svn move, svn copy);
  - просмотр состояния рабочей копии и локальных (ещё не зафиксированных) изменений (svn info, svn status, svn diff);
  - любые локальные изменения, если они признаны неудачными, можно *откатить* (svn revert).
- При необходимости — дополнительное обновление, для получения изменений, зафиксированных в хранилище другими пользователями и слияния этих изменений со своими (svn update).
- Фиксация своих изменений (и/или результатов слияния) в хранилище (svn commit).

tortoisesvn

http://svnbook.red-bean.com/index.ru.html - работа с Subversion на русском языке
http://tortoisesvn.net/docs/release/TortoiseSVN_ru/ - руководство пользователя на русском

(привыкаем курить мануалы;)

https://code.google.com/p/ifmo-game-1/wiki/UsingSVN - Как использовать Tortoise SVN