Лабораторная работа №2

# Цель работы: Изучение системы контроля версий Mercurial.

Mercurial — это современная распределенная система контроля версий с открытым кодом. Эта система — замена для более ранних систем вроде Subversion. У нее два основных назначения:

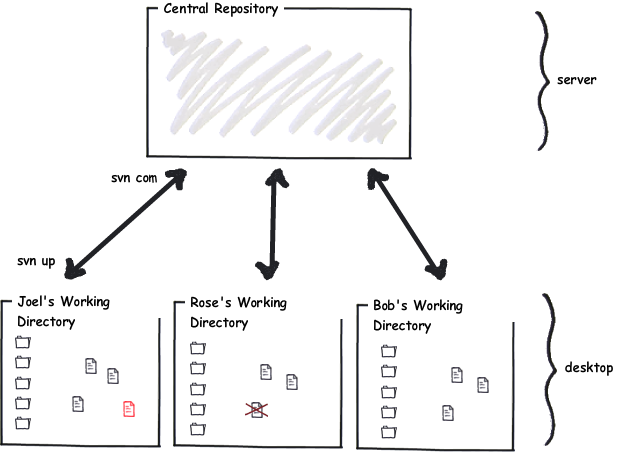
1. Она хранит все предыдущие версии каждого файла
2. Она может объединить разные версии вашего кода, то есть сотрудники могут независимо работать над кодом и затем объединять свои изменения.

Система Mercurial написана на Python, хотя чувствительные к производительности части (например, своя реализация diff) выполнены в качестве модулей-расширений на C. Mercurial первоначально была написана для Linux, позже портирована под Windows, Mac OS X и большинство Unix-систем. Репозитории Mercurial управляются при помощи утилиты командной строки hg, но есть и GUI–интерфейсы.

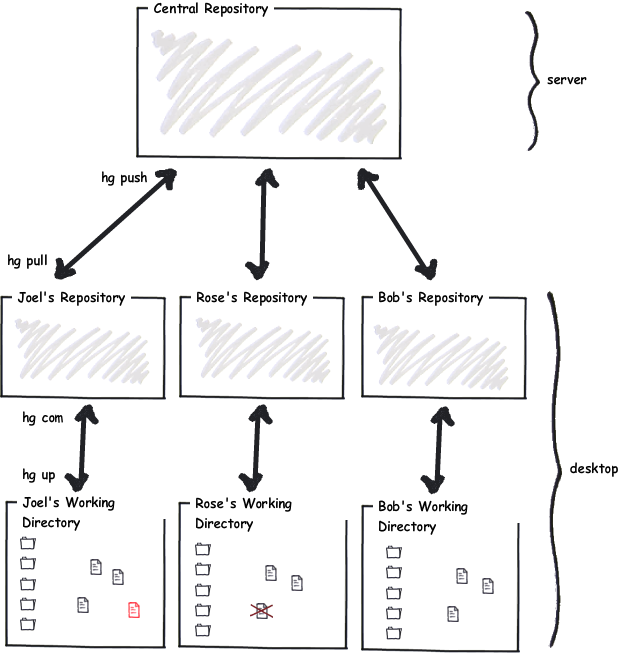
При работе с Subversion, когда вы вносите новый код в репозиторий, его получают все. Так как весь новый код, который пишет программист, содержит баги и/или недоделки, то разработчик встаёт перед следующим выбором:

* Вносить неработающий (не до конца рабочий) код (невозможно при работе с реальным заказчиком);
* Придерживать новый код до момента его окончательной отладки (всё время).

На практике, команда, использующая Subversion, часто днями или неделями ничего не добавляет в репозиторий. В таких командах новички боятся залить что-то в репозиторий из-за опасения поломать проект и возвращаются к старым методам передачи работы с проектом через архивы. Вот простая иллюстрация работы с Subversion:



При работе с Mercurial *у каждого разработчика свой собственный репозиторий, живущий у них на компьютере:*



*Mercurial разделяет момент внесения кода в репозиторий и момент получения этого кода всеми остальными.*

И это означает, что вы можете коммитить (hg com) (hg commit), но все остальные не получат ваши изменения. Когда у вас накопятся изменения, которые вас устраивают, вы проталкиваете (hg push) их в главный репозиторий.

В Subversion мыслят **ревизиями**. Ревизия — это то, как выглядит вся файловая система в определенный момент времени. В Mercurial вы мыслите наборами изменений (**changesets**). Набор изменений — это четкий список изменений между двумя соседними ревизиями. Subversion, по сути, система контроля изменений для файлов, а в Mercurial контроль изменений применяется ко всему каталогу, включая все подкаталоги.

Когда нам нужно сделать слияние, Subversion смотрит на обе ревизии —измененный собственно разработчика и измененный код, который уже находится на сервере — и пытается угадать, как слепить их вместе в одно целое (слияние). Обычно Subversion это не удается, и получаются длинные списки конфликтов («merge conflicts»), которые на самом деле не конфликты, а просто места, в которых система не смогла разобраться в наших изменениях. Такие конфликты разрешаются вручную.

## Основные команды Mercurial.

Большинство людей работают с Mercurial через интерфейс командной строки. Так можно работать в Windows, Unix, и Mac. Команда для Mercurial — это hg. Для того, чтобы выполнить эту команду, необходимо

sudo apt-get install mercurial

Если выполнить hg без параметров, то вы получите список наиболее часто используемых команд. Вы можете также попробовать выполнить hg help для получения полного списка команд.

Для того чтобы воспользоваться преимуществами контроля версий, вам нужен репозиторий. Репозиторий хранит все предыдущие версии всех ваших файлов. На самом деле, для экономии места на диске, все предыдущие версии не будут храниться — будет храниться только компактный список изменений.

hg init  
создает репозиторий.

появился новый каталог .hg:

Это и есть репозиторий! Это каталог со всем, что нужно Mercurial для работы. Настройки, предыдущие версии файлов, теги, лишняя пара носков на случай дождя, и прочее. Не лазьте туда. Вам почти всегда не стоит возиться с этим каталогом самостоятельно.  
  
Ну хорошо, раз у нас теперь есть свежеиспеченный репозиторий, то мы захотим добавить в него все исходные файлы. Это тоже просто: нужно лишь выполнить hg add.

hg add  
помечает файлы как запланированные для добавления в репозиторий. Файлы на самом деле не будут добавлены до тех пор, пока вы не зафиксируете изменения.

Остался еще один шаг… вам нужно зафиксировать ваши изменения. Какие изменения? Факт добавления всех этих файлов.  
  
Почему вы вынуждены фиксировать изменения? При работе с Mercurial фиксирование изменений означает: «Эй, вот как файлы сейчас выглядят, так их и запомни, пожалуйста». Это как создание копии всего каталога… каждый раз, когда у вас есть что-то измененное, и оно вам типа нравится, то вы фиксируете изменения.

hg commit  
сохраняет текущее состояние всех файлов в репозиторий.

В случае такой ошибки, которая вам вероятно всего встретится

error: abort: no username supplied (see "hg help config")

<http://stackoverflow.com/questions/2477781/mercurial-abort-no-username-supplied-see-hg-help-config>

Either put a hgrc in the .hg directory in your repo or put a .hgrc (hgrc for Windows) file in your home dir (then it covers all your projects) In the file you should write

[ui]

username = Your Name <your@mail>

Mercurial запустит редактор для того, чтобы вы могли напечатать комментарий к изменениям. Вам нужно просто написать что-нибудь, что напомнит вам о сделанных изменениях.

Вы можете выполнить hg log для просмотра истории изменений. Это как блог для вашего репозитория:

hg log  
отображает историю изменений, зафиксированных в репозитории.

Благодаря Mercurial, если вам не нравится то, что вы сделали, вы можете выполнить удобную команду hg revert, которая немедленно вернет ваш каталог к виду, в котором он был в момент последнего коммита.

hg revert  
возвращает измененные файлы к зафиксированному в репозитории виду.

Я использовал аргумент --all, потому что хотел вернуть все файлы к прежнему состоянию.

Таким образом, когда вы работаете над исходным кодом, используя Mercurial, вы:

1. Делаете изменения
2. Оцениваете, подходят ли они
3. Если подходят, то выполняете commit
4. Если не подходят, то выполняете revert
5. GOTO 1

Со временем вы можете забыть, на чем остановились и что поменяли с момента прошлого коммита. Mercurial отслеживает все это для вас. Все что вам нужно, это выполнить hg status и Mercurial даст вам список измененных файлов.

hg status  
отображает список измененных файлов.

hg status отображает список измененных файлов, добавляя значок в начале каждой строки. Этот значок сообщает о том, что же произошло. «M» означаете «Modified» — файл был изменен. "!" означает отсутствие — файл должен быть здесь, но куда-то делся. "?" означает что, состояние не определено — Mercurial ничего не знает про этот файл. Пока не знает.

А есть команда для этого: hg diff сообщит вам в точности, что произошло с файлом с прошлого коммита.

hg diff  
показывает, что изменилось в файле.

hg remove  
помечает файлы как запланированные для удаления из репозитория. Файлы на диске не будут удалены до тех пор, пока вы не зафиксируете изменения.

hg cat  
отображает содержимое любого файла для любой ревизии.

Для того чтобы увидеть как файл выглядел раньше, я могу просто указать нужный номер набора изменений (changeset из лога) при помощи аргумента -r («revision», то есть ревизия):

Если у файла сложное содержимое или большая длина, а изменился он лишь немного, то я могу использовать команду hg diff с аргументом r для вывода разницы между двумя ревизиями. К примеру, вот как посмотреть разницу между ревизиями 0 и 1:

hg update  
приводит рабочий каталог к состоянию, как у заданной ревизии.

Проще всего создать центральный репозиторий при помощи встроенного в Mercurial веб-сервера. Вам нужно лишь создать репозиторий при помощи hg init и открыть к нему доступ при помощи hg serve. По умолчанию репозиторий будет доступен на 8000-ом порту.

hg serve  
запускает веб-сервер и делает текущий репозиторий доступным в сети Интернет.

hg clone  
делает полную копию всего репозитория.

hg push  
проталкивает свежие изменения из текущего репозитория в центральный.

<http://habrahabr.ru/post/108443/>

<http://habrahabr.ru/post/108658/>

etc

## Краткие основы tcl.

Для выполнения задания вам понадобится tcl – скриптовый язык высокого уровня. Для того, чтобы запустить программу на tcl, её не нужно компиллировать – достаточно вызвать её при помощи утилиты командной строки tlcsh (соответвенно tcl и sh -- шелл):

dzmitry@dzmitry-VirtualBox:~/VCS-labs/Labs/tcl$ tclsh array\_sort.tcl

Для выполнения задания вам нужно знать основные команды языка:

|  |  |
| --- | --- |
| Пример функции | Краткое описание |
| set x "Hello, World!" | Присваивание строки переменной х |
| puts "A simple string: $x\n" | Вывод на экран |
| source [file join $where hello.tcl] | Добавить .tcl файл в другой .tcl файл |
| set together [concat $first $second] | Объединить массивы |
| array set copy [array get colors] | Скопировать значение |
| unset colors(red) | Удалить из массива |
| string compare s1 s2 | Сравнить строки |
| if {… } {…} | Если-тогда выражение |
| append result "World" | Объединение строк |
| format "%f" 43.5 | Форматирование |
| string index $s1 0 | Первое вхождение в строку |
| string last $s2 $s1 | Последнее вхождение в строку |
| string length $s1 | Длина строки |
| string match "\*@\*.com" $s1 | Проверка на паттерн |
| string match {tcl} $s1 | Проверка на присутствие строки |
| string wordend $s1 20 | Получить конец слова, начиная с текущей позиции |
| string wordstart $s1 20 | Получить начало слова, начиная с текущей позиции |
| string trimright $s1 $s2 | Убрать ненужные символы справа |
| string trimleft $s1 $s2 | Убрать ненужные символы слева |
| string trim $s1 $s2 | Убрать ненужные символы с двух сторон |
| foreach name [array names copy] {  puts "$name is $copy($name)"  } | Вывод массива на экран |

**Задание**:

Часть 1.

1. Установите tcl на рабочую машину при помощи команды

sudo apt-get install tk8.5

1. Склонируйте репозиторий, который находится по адресу, указанному преподавателем, к себе на локальную машину
2. Создайте в локальном репозитории файл со своим именем на языке tcl, который выводил бы в консоль вашу имя и фамилию. Используйте в качестве имени файла свою имя и фамилию (Ivan\_Ivanov.tcl).
3. Используя команды mercurial, добавьте этот файл в удалённый репозиторий
4. В файле all.tcl добавьте строку, которая вызывала бы ваш файл. В итоге должен получиться файл .tcl, вызвав который, преподаватель сможет увидеть имена всех, кто дошёл до этой части задания.

Часть 2.

1. Обратите внимание, что в папках array и string находятся .tcl файлы, которые реализуют некую функциональность для работы с массивами или строками на языке tcl. К сожалению, в результате аппаратной ошибки, файлы были повреждены. Ваша задача – преобразовать файлы таким образом, чтобы требуемая функциональность была восстановлена. Варианты:
   1. arrya\_concat (соединить два массива в один)
   2. array\_copy (сделать так, чтобы в copy была копия colors)
   3. array\_remove (убрать красный цвет из массива)
   4. array\_sort (отсортировать массив в порядке убывания (4 3 2 1)
   5. string\_compare (сделать правильное сравнение строк)
   6. string\_concat (слить три строки в одну)
   7. string\_format (сделать С-подобный вывод значений)
   8. string\_index (узнать номер в строке того или иного символа)
   9. string\_length (узнать длину строки)
   10. string\_match (в первом случае проверить, является ли строка адресом электронной почты, во втором – проверка на паттерн)
   11. string\_trim (убрать ненужные пробелы)
2. После выполнения задания добавьте изменения на сервер.