Лабораторная работа №1

# Изучение системы контроля версий Subversion.

**Subversion**, часто сокращаемая до SVN, из-за основной команды svn – **система контроля версий**, которая была выпущена для замены устаревшей CVS ещё в 2004 году. Subversion реализует все основные функции CVS и свободна от ряда недостатков последней.

Subversion — **централизованная** система (в отличие от распределённых систем, таких как Git или Mercurial), то есть данные хранятся в едином хранилище. Хранилище может располагаться на локальном диске или на сетевом сервере.

Работа в Subversion построена следующим образом: клиенты копируют файлы из хранилища, создавая локальные рабочие копии, затем вносят изменения в рабочие копии и фиксируют эти изменения в хранилище. Несколько клиентов могут одновременно обращаться к хранилищу. Для совместной работы над файлами в Subversion преимущественно используется модель копирование — изменение — слияние. Кроме того, для файлов, не допускающих слияние (различные бинарные форматы файлов), можно использовать модель блокирование — изменение — разблокирование.

При сохранении новых версий используется дельта-компрессия: система находит отличия новой версии от предыдущей и записывает только их, избегая дублирования данных.

При использовании доступа с помощью WebDAV также поддерживается прозрачное управление версиями — если любой клиент WebDAV открывает для записи и затем сохраняет файл, хранящийся на сетевом ресурсе, то автоматически создаётся новая версия. WebDAV (Web Distributed Authoring and Versioning) или просто DAV — набор расширений и дополнений к протоколу HTTP, поддерживающих совместную работу пользователей над редактированием файлов и управление файлами на удаленных веб-серверах.

## Основные операции над репозиторием в SVN.

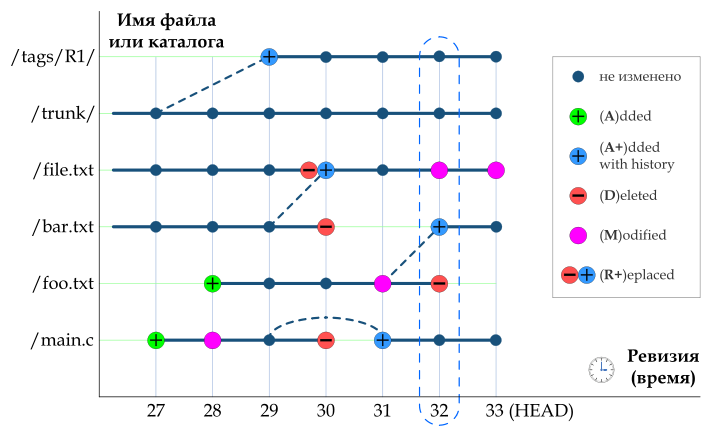


Рисунок 1 – Основные операции над файловой системой.

Над объектами файловой системы в хранилище Subversion (см. рис. 1) могут быть произведены перечисленные ниже операции. В скобках указано краткое именование операции в обозначениях команды svn status.

**Добавление** (A). Добавление объекта в файловую систему. Добавленный объект не имеет истории ревизий. Пример на рисунке:

файл /main.c был добавлен в ревизии 27.

**Модификация** (M). Модификация объекта, например, изменение содержимого файла или изменение свойств файла или директории. Пример на рисунке:

файл /main.c был модифицирован в ревизии 28.

**Удаление** (D). Удаление файла из головной и последующих ревизий. При этом файл остаётся в предыдущих ревизиях. Пример на рисунке:

файл /main.c был удалён в ревизии 30.

**Добавление с историей** (A+). Представляет собой копирование объекта внутри файловой системы хранилища, то есть объектимя\_источника@ревизия\_источника копируется в имя\_копии@HEAD. Скопированный объект наследует от источника историю ревизий до момента копирования (наследование истории показано на рисунке пунктирными связями). Примеры на рисунке:

в ревизии 29 директория /tags/R1 была скопирована с директории /trunk@27;

в ревизии 31 файл /main.c был скопирован с /main.c@29, то есть с более ранней ревизии самого себя, таким образом, произведено восстановление ранее удалённого (в ревизии 30) файла с сохранением истории ревизий.

**Замена** (R+). Имеет место в случае, когда в одной ревизии произведено и удаление объекта (D), и добавление с историей (A+) объекта с тем же самым именем. Хотя имя при операции замены остаётся неизменным, Subversion рассматривает объект до и после замены как два различных объекта с различными историями ревизий (история старого заканчивается в точке замены, история нового наследуется от источника копирования и продолжается далее). Пример на рисунке:

в ревизии 30 файл /file.txt был заменён: старый файл /file.txt удалён, а новый файл с тем же именем скопирован с файла /bar.txt@29

## Команды Subversion:

Для начала работы с SVN необходимо забрать данные из репозитория (в этой работе это должно быть сделано при помощи командной строки). Чаще всего, «репозиторий» на начальном этапе – это просто ссылка; в данной лабораторной работе ­– полученная от преподавателя.

Для того, чтобы забрать данные из репозитория, необходимо воспользоваться командой svn co (помощи по команде: snv help co), которая является сокращением от svn checkout. Например,

svn co http://example.com/svn/name

Скорее всего, первый раз вам будет ввести имя пользоватля и пароль, которые в дальнейшем можно сохранить у себя на компьютере, чтобы не вводить их при каждом запросе к репозиторию. После этого папка с директорией появится у вас на рабочей машине.

Далее вносятся необходимые изменения в требуемые файлы. После их сохранения и выполнения команды svn status требуемые файлы покажутся в консоли. Если всё верно и изменения правильные, то можно начинать процесс заливки этой версии репозитория на сервер. Если файлы по какой-то причине не отображаются – попробуйте команду svn add . (с точкой на конце – она является признаком того, что вы хотите добавить все файлы); если и это не сработало – попробуйте выполнить svn add . --force

Однако сначала необходимо получить все остальные изменения. Это делается при помощи команды svn up (сокращенно от svn update).

Для начала нужно ввести svn ci –m "здесь сообщение о том, что вы сделали", сокращение от svn commit –m "message". Сообщение нужно для того, чтобы при дальнейшей разработке знать, кто что сделал.

В дальнейшем, когда задание будет закончено, при помощи таких сообщений и команды svn log можно будет узнать, кем и когда были сделаны коммиты, а самое главное, что в них было (т.е. информация, указанная в сообщении). В svn log можно указать флаг -l , который подскажет Subversion,какое количество сообщений необходимо взять. Например,

svn log -l 10

покажет первые 10 сообщений.

## Краткий справочник команд Subversion:

**svn checkout http://repository.url/svn/name** — извлекаем файлы проекта из репозитория, сокращение: **svn co**;

**svn update** — получаем обновления из репозитория, сокращение: **svn up**;

**svn update -r rev\_num ./file\_name** — извлекаем ревизию файла с номером rev\_num;

**svn add ./file\_name** — добавляем файл в репозиторий (не важно текстовый или бинарный);

**svn commit ./file\_name** — заливаем файл в репозиторий (не важно текстовый или бинарный);

**svn rename ./old\_file\_name ./new\_file\_name** — переименовываем файл в репозитории;

**svn remove ./file\_name** — удаляем файл/директорию из репозитория;

**svn status** — просматриваем локально измененные файлы, сокращение: **svn st**;

**svn status -u** — просматриваем локально измененные и изменившиеся в репозитории файлы, сокращение: **svn st -u**;

**svn diff ./file\_name** — показывает локальные изменения в файле построчно;

**svn diff -r rev\_num1:rev\_num2 ./file\_name** — показывает различия между ревизией rev\_num1 и rev\_num2 файла;

**svn revert ./file\_name** — откатывает локальные изменения файла (выгружает из репозитория последнюю закоммиченную ревизию);

**svn revert -R ./** — откатывает все локальные изменения файлов;

**svn log ./file\_name** — список ревизий с комментариями;

**svn blame ./file\_name** — показывает авторов изменений файла построчно, синоним: **svn annotate**;

**svn propset svn:ignore ./file\_name .** — добавляем файл в список игнорируемых файлов;

**svn propset svn:keywords "Id Author Date" ./file\_name** — установка атрибутов файла;

**svn cleanup** — снимает блокировки с файлов;

**svn unlock http://repository.url/svn/file\_name** — снять блокировку файла (URL можно узнать с помощью команды **svn info ./file\_name | grep URL**, его и нужно передавать в svn unlock);

**svnadmin setlog --bypass-hooks /path/to/repository -r rev\_num ./commit\_text\_file** — заменяет текстовое описание коммита, где rev\_num — номер ревизии, commit\_text\_file — путь к файлу, содержащему новый комментарий к коммиту;

**svn help command\_name** — выводит помощь по команде command\_name, например, «svn help update»;

**svn merge -r rev\_to\_rollback:rev\_good ./file\_name** — откатываем ревизию номер rev\_to\_rollback до ревизии rev\_good, причем все изменения старше rev\_to\_rollback сохраняются (Например, у файла есть ревизии 11,12 и 13. Хотим откатить 12-ую ревизию, но так, что бы изменения 13-ой остались в силе. Делаем тогда так: svn merge -r 12:11 ./file\_name);

**svn copy http://repository.url/svn/name/trunk/ http://repository.url/svn/name/branches/new\_branch\_name/** — создаем ветку с названием new\_branch\_name из главной линии разработки;

**svn merge --dry-run -r rev\_num1:rev\_num2 http://repository.url/svn/name/trunk/** — проверяем, что будет изменено при объединении веток, где rev\_num1 — номер ревизии, когда ваша ветка была «открыта», или это м.б. номер предыдущего объединения (слияния), rev\_num2 — версия главной линии разработки, с которой производим объединение. Необходимо отметить, что все изменения будут применены для директории, в которой выполнялась эта команда;

**svn merge -r rev\_num1:rev\_num2 http://repository.url/svn/name/trunk/** — синхронизирует вашу ветку с главной линией разработки с учетом ревизий: rev\_num1 — номер ревизии, когда ваша ветка была «открыта», или это м.б. номер предыдущего объединения (слияния), rev\_num2 — версия главной линии разработки, с которой производим объединение. Необходимо отметить, что все изменения будут применены для директории, в которой выполнялась эта команда.

# Zenity

Утилита **zenity** – это средство создания диалоговых окон в режиме командной строки. Следует отметить, что на самом деле диалоговые окна создаются средствами Gtk+, поэтому в системе должны быть установлены соответствующие библиотеки. От аналогичных программ, таких как **dialog** и **whiptail**, **zenity** отличают средства реализации GUI-элементов.

Чтобы начать использовать **zenity** на практике не требуется обладать особыми знаниями или умениями, достаточно просто познакомиться с различными опциями (ключами), позволяющими в полной мере использовать возможности этой программы.

В **zenity** определены четыре типа диалоговых окон для вывода сообщений:

* ошибка (ключ --**error**);
* информация (ключ **--info**);
* вопрос (ключ **--question**);
* предупреждение (ключ **--warning**).

Установка zenity:

sudo apt-get install zenity

Для того, чтобы создать простейшее диалоговое окно,

# Задание:

1. Изучить базовую функциональность SVN и zenity.
2. Склонировать репозиторий, находящийся по адресу, указанный преподавателем.
3. Используя zenity, реализовать следующую функциональность (исходя из своего варианта):

|  |  |  |  |
| --- | --- | --- | --- |
| № Варианта | Операция | Описание | Файл |
| 1 | Add | Сложение | add.sh |
| 2 | Sub | Умножение | sub.sh |
| 3 | Mul | Вычитание | mul.sh |
| 4 | Div | Деление | div.sh |
| 5 | Mod | Остаток от деления | mod.sh |
| 6 | Not | Отрицание | neg.sh |
| 7 | And | Конъюнкция | conj.sh |
| 8 | Or | Дизъюнкция | dis.sh |
| 9 | Pow | Возведение в квадрат | pow.sh |
| 10 \* | Sqrt | Извлечение корня | sqrt.sh |
| 11 \* | Exp | Возведение в степень | exp.sh |
| 12 \* | Log | Взятие логарифма | log.sh |
| 13 \* | Cos | Взятие косинуса от числа (радианы) | bc | cos.sh |
| 14 \* | Sin | Взятие синуса от числа (радианы) | bc | sin.sh |
| 15 \*\* | Main dialog | Создание файла-точки входа | main.sh |

1. Создать новую ветку, где название ветки формируются по типу задание-операция (для варианта 1: 1-add).
2. Внести изменения в свою ветку и занести их результаты на сервер.
3. Результат выполнения работы предъявить преподавателю.

# Контрольные вопросы:

1. Что такое Система Контроля Версий?
2. Перечислите основные команды git.
3. Расскажите подробнее о следующей команде:
   1. clone
   2. checkout
   3. branch
   4. commit
   5. push
   6. merge
4. Что такое zenity? Какие возможности он имеет?
5. \* Можно ли переименовать файл conj.sh в файл con.sh в Windows? Объяснить, если да, то какие последствия будет иметь такая операция, если нет, то почему.