**3 РАЗРАБОТКА ИНФОРМАЦИОННОГО ОБЕСПЕЧЕНИЯ**

Для проведения лабораторного практикума были выбраны Subversion, Mercurial и Git в качестве систем контроля версий и tcl/tk, zenity и HTML в качестве примеров для написания программ или разметки.

**3.1 Обзор 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.

Subversion используется многими сообществами разработчиков открытого программного обеспечения (в том числе сообществами, ранее использовавшими CVS). В их числе такие известные проекты, как Apache, GCC, Free Pascal, Python, Ruby, FreeBSD, AROS, Blender, Boost, Tor. Subversion также широко используется в закрытых проектах и корпоративной сфере. Хостинг Subversion, в том числе для проектов с открытым кодом, также предоставляют популярные хостинг-проекты SourceForge.net, Tigris.org, Google Code и BountySource. Возможности Subversion:

− хранение полной истории изменений отслеживаемых объектов (файлов, каталогов, символьных ссылок) в централизованном хранилище (репозитории), в том числе при изменении атрибутов («метаданных»), перемещении, переименовании и удалении

− копирование объектов с разветвлением истории — при копировании в хранилище появляются два отдельных объекта с общей историей;

− поддержка переноса изменений между копиями объектов, в том числе полного слияния копий (в рабочей копии; без объединения истории);

− поддержка ветвления при помощи создания ветвей (копированием директорий) и работы с ними, или слияние ветвей (переносом изменений);

− поддержка меток (копированием директорий);

− история изменений и копии объектов (в том числе ветви и метки) хранятся в виде связанных разностных копий — «дешёвых» (не требующих больших временных и дисковых ресурсов) при создании и хранении;

− поддержка конкурентной (в том числе одновременной, с изоляцией транзакций) многопользовательской работы с хранилищем и, в большинстве случаев, автоматическим слиянием изменений различных разработчиков (в рабочей копии);

− фиксации изменений в хранилище (в том числе многообъектные) организуются в виде атомарных транзакций;

− сетевой обмен между сервером и клиентом предусматривает передачу только различий между рабочей копией и хранилищем;

− обеспечивается одинаково эффективная работа как с текстовыми, так и с двоичными файлами;

− различные варианты доступа к хранилищу, в том числе непосредственный доступ на локальной файловой системе, по собственному сетевому протоколу или через веб-сервер по протоколу WebDAV/DeltaV;

− вывод клиента командной строки одинаково удобен и для чтения, и для разбора программами;

− возможность зеркалирования хранилища;

− два возможных внутренних формата хранилища (англ. repository): база данных или набор обычных файлов;

− интернационализированные сообщения программы (используются настройки локали);

− библиотеки для языков PHP, Python, Perl, Java позволяют встроить функциональность клиента Subversion в программы, написанные на этих языках;

− многоуровневая архитектура библиотек, изначально рассчитанная на клиент-серверную модель.

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.

Общая схема работы с SVN выглядит следующим образом:

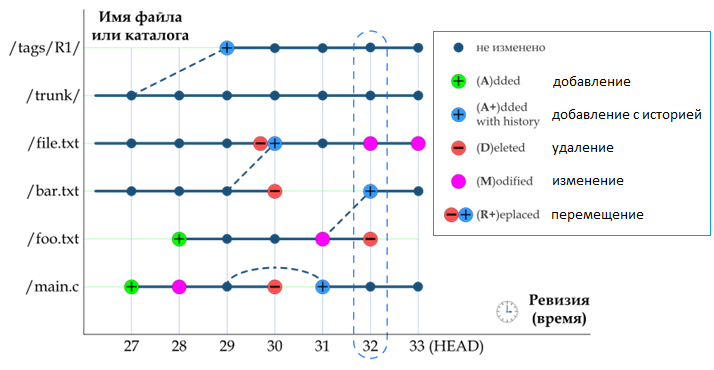


Схема 3.1 – Основные операции над файловой системой.

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 схеме 3.1 – файл /main.c был добавлен в ревизии 27.

− Модификация (M). Модификация объекта, например, изменение содержимого файла или изменение свойств файла или директории. Пример на схеме 3.1 –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 схеме 3.1 – 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 схеме 3.1 пунктирными связями). Примеры на схеме 3.1 ­– в ревизии 29 директория /tags/R1 была скопирована с директории /trunk@27 и 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 схеме 3.1 – 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.

**3.2 Обзор Mercurial**

Mercurial это кроссплатформенная распределенная система контроля версий с открытым кодом, разработанная для эффективной работы с очень большими репозиториями. Эта система является заменой для более ранних систем вроде Subversion. У нее два основных назначения:

− она хранит все предыдущие версии каждого файла;

− 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.

Mercurial разделяет момент внесения кода в репозиторий и момент получения этого кода всеми остальными. И это означает, что вы можете создавать наборы изменений (командами hg com или hg commit), но все остальные не получат ваши изменения. Когда у вас накопятся изменения, которые вас устраивают, вы записываете (hg push) их в главный репозиторий, находящийся на центральном сервере.

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.

Большинство людей работают с Mercurial через интерфейс командной строки. Так можно работать в Windows, Unix, и Mac. Команда для Mercurial — это hg.

**3.3 Обзор git**

Git – распределённая система управления версиями файлов. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux.

Система спроектирована как набор программ, специально разработанных с учётом их использования в скриптах. Это позволяет удобно создавать специализированные системы контроля версий на базе Git или пользовательские интерфейсы. Например, Cogito является именно таким примером оболочки к репозиториям Git, а StGit использует Git для управления коллекцией исправлений (патчей).

Git поддерживает быстрое разделение и слияние версий, включает инструменты для визуализации и навигации по нелинейной истории разработки. Как и Darcs, BitKeeper, Mercurial, Bazaar и Monotone, Git предоставляет каждому разработчику локальную копию всей истории разработки, изменения копируются из одного репозитория в другой.

Удалённый доступ к репозиториям Git обеспечивается git-daemon, SSH- или HTTP-сервером. TCP-сервис git-daemon входит в дистрибутив Git и является наряду с SSH наиболее распространённым и надёжным методом доступа. Метод доступа по HTTP, несмотря на ряд ограничений, очень популярен в контролируемых сетях, потому что позволяет использовать существующие конфигурации сетевых фильтров.

Ядро Git представляет собой набор утилит командной строки с параметрами. Все настройки хранятся в текстовых файлах конфигурации. Такая реализация делает Git легко портируемым на любую платформу и даёт возможность легко интегрировать Git в другие системы (в частности, создавать графические git-клиенты с любым желаемым интерфейсом).

Репозиторий Git представляет собой каталог файловой системы, в котором находятся файлы конфигурации репозитория, файлы журналов, хранящие операции, выполняемые над репозиторием, индекс, описывающий расположение файлов и хранилище, содержащее собственно файлы. Структура хранилища файлов не отражает реальную структуру хранящегося в репозитории файлового дерева, она ориентирована на повышение скорости выполнения операций с репозиторием. Когда ядро обрабатывает команду изменения (неважно, при локальных изменениях или при получении патча от другого узла), оно создаёт в хранилище новые файлы, соответствующие новым состояниям изменённых файлов. Существенно, что никакие операции не изменяют содержимого уже существующих в хранилище файлов.

По умолчанию репозиторий хранится в подкаталоге с названием «.git» в корневом каталоге рабочей копии дерева файлов, хранящегося в репозитории. Любое файловое дерево в системе можно превратить в репозиторий git, отдав команду создания репозитория из корневого каталога этого дерева (или указав корневой каталог в параметрах программы). Репозиторий может быть импортирован с другого узла, доступного по сети. При импорте нового репозитория автоматически создаётся рабочая копия, соответствующая последнему зафиксированному состоянию импортируемого репозитория (то есть не копируются изменения в рабочей копии исходного узла, для которых на том узле не была выполнена команда commit).

Для хранения бинарных файлов, например, электронной библиотеки, Git подходит лучше. По сравнению с Mercurial он не ориентирован на расчет дельты файлов, что для бинарного содержимого не очень эффективно. Сами файлы меняются редко, а основные операции с ними — это перемещение и добавление.

**3.4 Обзор 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.

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).

Установку zenity можно произвести следующей командой: sudo apt-get install zenity. Для проверки правильности установки можно выполнить команду zenity без параметров. В этом случае выведется следующая ошибка, говорящаю о том, что нужно выбрать тип окна.

Для того, чтобы создать простейшее диалоговое окно, необходимо выполнить команду: zenity --entry --title='addition' --text='enter first number'. Результат работы такой программы можно увидеть на рисунке 3.2:

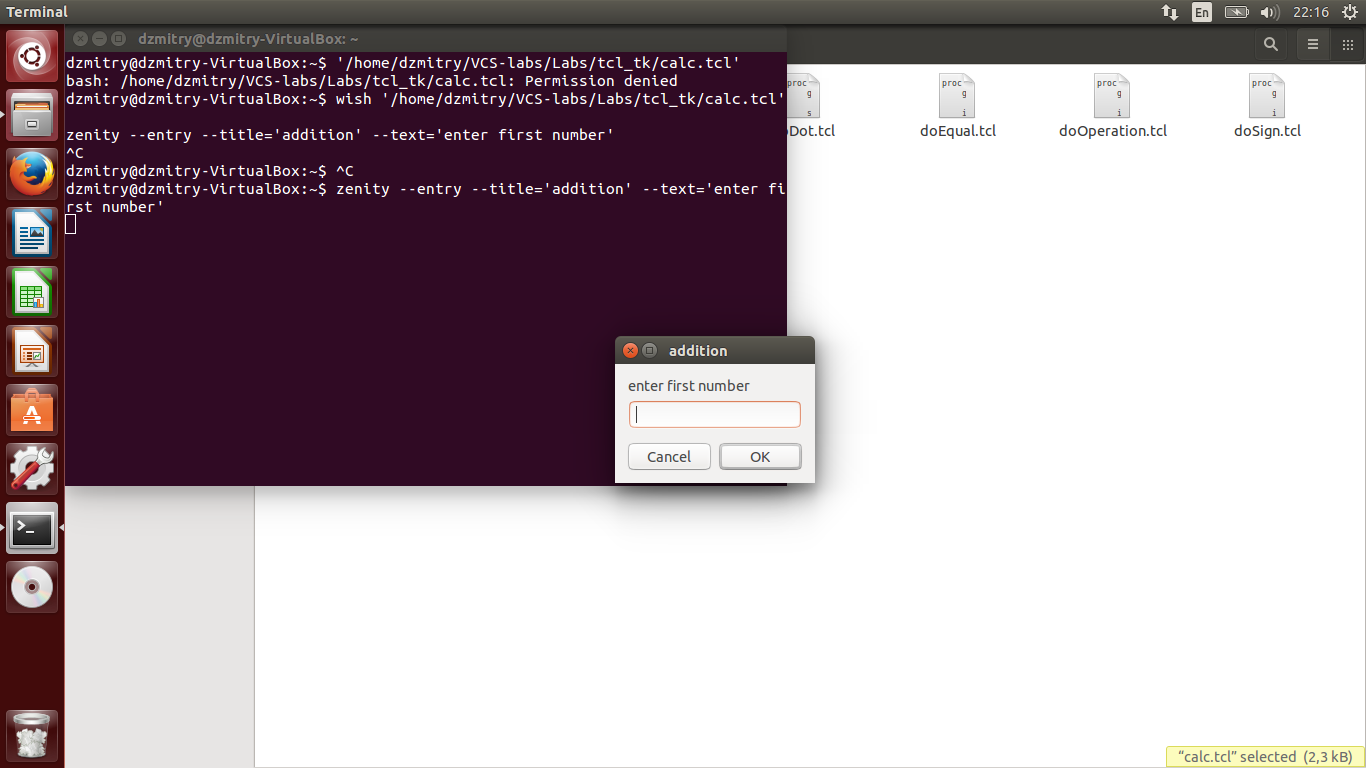


Рисунок 3.2 – Пример программы, созданной при помощи zenity

Таким образом, всего при помощи одной строки можно создать окно, которое выполнит какую-либо информационную функцию (или функцию ввода-вывода).

**3.5 Обзор tcl/tk**

Tcl (от англ. Tool Command Language — «командный язык инструментов») — скриптовый язык высокого уровня.

Tcl часто применяется совместно с графической библиотекой Tk (Tool Kit). Связку Tcl/Tk по-русски иногда называют «Так-тикль» (английский вариант — «тикль-ток»).

Tk (от англ. Toolkit — «набор инструментов», «инструментарий») — кроссплатформенная библиотека базовых элементов графического интерфейса, распространяемая с открытыми исходными текстами.

Tk был разработан Джоном Оустерхаутом как расширение для интерпретируемого языка программирования Tcl. Также, с использованием специальных библиотек, Tk может использоваться другими языками программирования, например, Perl, Python, Ruby. Большинство из этих языков используют Tcl как мост для Tk.

Области применения языка tcl – быстрое прототипирование, создание графических интерфейсов для консольных программ (пакетов программ), встраивание в прикладные программы, тестирование. Также Tcl применяется в веб-разработке. В Tcl данными всех типов, включая код программы, можно манипулировать как строками. Это делает его языком с естественной поддержкой метапрограммирования. Эта парадигма программирования учитывалась в ходе разработки и эволюции языка.

Программа на Tcl состоит из команд, разделённых символами новой строки или точками с запятой. Каждая команда состоит из набора полей, разделённых пробелами. Первое поле должно быть именем команды, а необязательные остальные поля — передаваемые этой команде аргументы. Команда возвращает значение, иногда пустое. То есть, как и в Лиспе, в Tcl используется префиксная нотация.

Ключевых слов как таковых нет – понятие команды в Tcl аналогично понятию процедуры или функции распространённых языков программирования. Это относится и к управляющим конструкциям языка. В сочетании с элементарным синтаксисом это обеспечивает хорошую расширяемость языка, в том числе и библиотеками, написанными на других языках, таких как C/C++ или Java.

В Tcl также качественно реализована модель управления программой на основе событий. События могут генерироваться таймером, при появлении данных в канале, изменении значения переменной, при завершении какой-либо внешней программы, или просто при работе пользователя с интерфейсом Tk. Можно задавать свои события и управлять ими.

Как и большинство современных скриптовых языков Tcl содержит развитые средства работы с регулярными выражениями, работает с ассоциативными массивами и другими динамическими структурами данных.

В базовом Tcl нет поддержки ООП, однако существует множество расширений Tcl объектно-ориентированными механизмами, реализованных в виде подключаемых библиотек на Си, или самом Tcl.

Tcl, наряду с Perl и Python, стал одним из трёх классических скриптовых языков общего назначения. Эта троица фигурирует не только в качестве списка свободных дистрибутивов, собираемых в ActiveState, но и, например, как языки, на которых (помимо диалекта PL/pgSQL) можно писать триггеры и хранимые процедуры популярного сервера БД PostgreSQL. При помощи tcl, в отличие от zenity, можно о=создавать довольно сложные программы (как на рисунке 3.3):

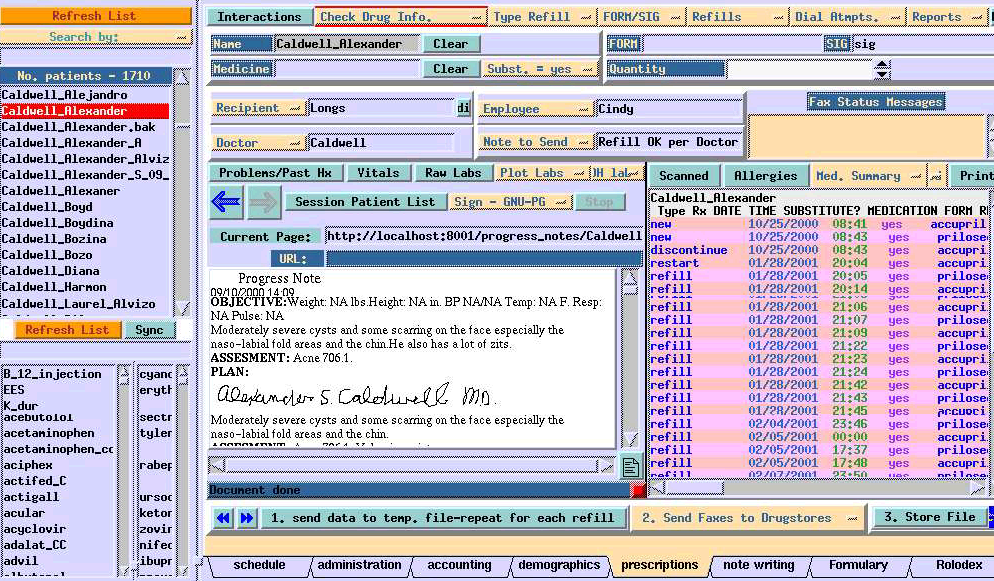


Рисунок 3.3 – Пример программы, созданной при помощи tcl/tk

Tcl, как встраиваемый язык, нашёл применение и в сфере САПР (CAD|CAM|CAE). Он, к примеру, используется как средство настройки баз данных, в постпроцессоре Unigraphics. Кроме того, он де-факто является языком автоматизации и интеграции во всех ведущих программных пакетах разработки микросхем (ПЛИС и ASIC).

**3.6 Обзор HTML**

HTML (от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) HyperText Markup Language — «язык [гипертекстовой](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%BF%D0%B5%D1%80%D1%82%D0%B5%D0%BA%D1%81%D1%82) разметки») — стандартный [язык разметки](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8) документов во [Всемирной паутине](https://ru.wikipedia.org/wiki/%D0%92%D1%81%D0%B5%D0%BC%D0%B8%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D1%83%D1%82%D0%B8%D0%BD%D0%B0). Большинство [веб-страниц](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0) содержат описание разметки на языке HTML (или [XHTML](https://ru.wikipedia.org/wiki/XHTML)). Язык HTML интерпретируется [браузерами](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80), полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

Язык HTML является приложением [SGML](https://ru.wikipedia.org/wiki/SGML) (стандартного обобщённого языка разметки) и соответствует международному стандарту [ISO](https://ru.wikipedia.org/wiki/ISO) 8879.

Язык [XHTML](https://ru.wikipedia.org/wiki/XHTML) является более строгим вариантом HTML, он следует всем ограничениям [XML](https://ru.wikipedia.org/wiki/XML) и, фактически, XHTML можно воспринимать как приложение языка XML к области разметки гипертекста.

Во всемирной паутине HTML-страницы, как правило, передаются браузерам от сервера по протоколам [HTTP](https://ru.wikipedia.org/wiki/HTTP) или [HTTPS](https://ru.wikipedia.org/wiki/HTTPS), в виде простого текста или с использованием [шифрования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

Изначально язык HTML был задуман и создан как средство структурирования и форматирования документов без их привязки к средствам воспроизведения (отображения). В идеале, текст с разметкой HTML должен был без стилистических и структурных искажений воспроизводиться на оборудовании с различной технической оснащённостью (цветной экран современного компьютера, монохромный экран органайзера, ограниченный по размерам экран мобильного телефона или устройства и программы голосового воспроизведения текстов). Однако современное применение HTML очень далеко от его изначальной задачи. Например, тег <TABLE> предназначен для создания в документах таблиц, но часто используется и для оформления размещения элементов на странице. С течением времени основная идея платформонезависимости языка HTML была принесена в жертву современным потребностям в мультимедийном и графическом оформлении. Пример простого документа на HTML показан на рисунке 3.4:

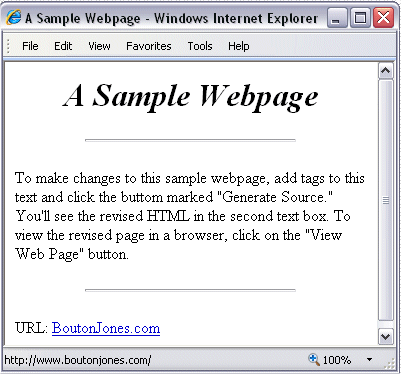


Рисунок 3.4 – Пример браузера, использующего разметку HTML

HTML — теговый язык разметки [документов](https://ru.wikipedia.org/wiki/HTML-%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82). Любой документ на языке HTML представляет собой набор [элементов](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D1%8B_HTML), причём начало и конец каждого элемента обозначается специальными пометками — [тегами](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%B3_(%D1%8F%D0%B7%D1%8B%D0%BA%D0%B8_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8)). Элементы могут быть пустыми, то есть не содержащими никакого текста и других данных (например, тег перевода строки <br>). В этом случае обычно не указывается закрывающий тег. Кроме того, элементы могут иметь атрибуты, определяющие какие-либо их свойства (например, размер шрифта для элемента font). Атрибуты указываются в открывающем теге. Вот примеры фрагментов HTML-документа для гиперссылок: <a href="http://www.example.com">Здесь элемент содержит атрибут href, то есть гиперссылку.</a>

Пример пустого элемента: <br>

Регистр, в котором набрано имя элемента и имена атрибутов, в HTML значения не имеет (в отличие от XHTML). Элементы могут быть вложенными. Кроме элементов, в HTML-документах есть и сущности ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) entities) — [«специальные символы»](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%B5%D0%BC%D0%BE%D0%BD%D0%B8%D0%BA%D0%B8_%D0%B2_HTML). Сущности начинаются с символа [амперсанда](https://ru.wikipedia.org/wiki/%D0%90%D0%BC%D0%BF%D0%B5%D1%80%D1%81%D0%B0%D0%BD%D0%B4) и имеют вид&имя; или &#NNNN;, где NNNN — код символа в [Юникоде](https://ru.wikipedia.org/wiki/%D0%AE%D0%BD%D0%B8%D0%BA%D0%BE%D0%B4) в десятичной системе счисления.

Например, &copy; — знак [авторского права](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D1%80%D1%81%D0%BA%D0%BE%D0%B5_%D0%BF%D1%80%D0%B0%D0%B2%D0%BE) (©). Как правило, сущности используются для представления символов, отсутствующих в кодировке документа, или же для представления «специальных» символов: &amp; — амперсанда (&), &lt; — символа «меньше» (<) и &gt; — символа «больше» (>), которые некорректно записывать «обычным» образом, из-за их особого значения в HTML.

Каждый HTML-документ, отвечающий спецификации HTML какой-либо версии, должен начинаться со строки объявления версии HTML <!DOCTYPE…>.

Далее обозначается начало и конец документа тегами <html> и </html> соответственно. Внутри этих тегов должны находиться теги заголовка (<head></head>) и тела (<body></body>) документа. В [HTML 5](https://ru.wikipedia.org/wiki/HTML_5) используется только один вариант DOCTYPE: <!DOCTYPE html>.