М. Гуссенс
С. Рацц

Библиотека издательских ТЕХнологий

XML
PDF
@
CSS
DSSL

LaTeX2HTM
TeX4ht

ПУТЕВОДИТЕЛЬ ПО ПАКЕТУ LaTeX

и его Web-приложениям

Издательство "Мир"

The LaTeX Web Companion

Integrating TeX, HTML, and XML

Michel Goossens
CERN, Geneva, Switzerland

Sebastian Rahtz
Elsevier Science Ltd., Oxford, United Kingdom

with Eitan M. Gurari, Ross Moore, and Robert S. Sutor

Corrected Second Printing, December 1999

ADDISON-WESLEY

An Imprint of Addison Wesley Longman, Inc.
Reading, Massachusetts • Harlow, England • Menlo Park, California
Berkeley, California • Don Mills, Ontario
Sydney • Bonn • Amsterdam • Tokyo • Mexico City

Библиотека издательских TEXнологий

М. Гуссенс, С. Ратц
при участии
Э. М. Гурари, Р. Мура и С. Сьютора

Путеводитель по пакету LaTeX и его Web-приложениям

Перевод с английского
Ю. В. Тюменцева и А. В. Чернышева

под редакцией
Б. В. Тоботраса

Москва «Мир» 2001

УДК 681.322
ББК 32.97
Г96

Гуссенс М., Ратц С.

Г96 Путеводитель по пакету LaTeX и его Web-приложениям: Пер. с англ. — М.: Мир, 2001. — 604 с., ил.

ISBN 5-03-003387-4

Подробный справочник, отражающий современное состояние программных продуктов для представления в сети Internet научных публикаций, подготовленных в LaTeX 2_ε, и для обратного преобразования Web-документов в LaTeX-формат. Дается полное описание таких средств, как HTML, XML, MathML, и конвертеров для них, например LaTeX2HTML, TeX4ht и др.

Большинство упомянутых в книге пакетов имеется в свободном доступе в TeX-архивах в сети Internet. Книга снабжена полным аннотированным каталогом Web-сайтов, которые предоставляют исходные тексты программ и документацию к ним.

Представляет интерес для всех TeX-пользователей: научных работников, самостоятельно готовящих свои работы к изданию, профессиональных наборщиков, специалистов по издательским системам, преподавателей и студентов соответствующих специальностей.

ББК 32.97

Редакция автоматизации издательских процессов

ISBN 5-03-003387-4 (русск.)
ISBN 0-201-43311-7 (англ.)

Предисловие редактора перевода

Вы держите в руках перевод *The LaTeX Web Companion* — одного из томов широко известного трехтомника, написанного М. Гуссенсом совместно с разработчиками LaTeX'овских продуктов[1]. Книга посвящена теме, чрезвычайно актуальной для TeX-сообщества: представлению (LA)TeX'овских документов на страницах Всемирной паутины.

Несмотря на отставание нашей страны от более развитых по уровню распространения Интернета, его популярность уже достаточна для того, чтобы многие пользователи TeX'а ощутили потребность в Web-публикациях TeX'овских файлов. Необходимые для этого знания и инструментарий в большой мере нашли отражение в этой книге. Особую ценность ей придает то, что авторы сами принимали участие в разработке части упоминаемых в книге инструментальных средств.

В данном издании рассматриваются различные способы Web-публикации TeX'овских документов: непосредственно в формате TeX при помощи дополнительных модулей (plugins) Web-навигаторов; с преобразованием в HTML, XML или PDF при помощи различных конвертеров и стилевых программ. Разнообразие представленных способов работы с TeX'ом в Сети объясняется широким спектром задач, для которых применяется TeX. В одних задачах может понадобиться точное представление математических формул в Web-навигаторе, в других — простой способ создания гипертекста из TeX-файлов. Часто бывает нужен только удобный метод генерации презентаций для просмотра на экране. В зависимости от своих потребностей большинство читателей смогут найти наиболее приемлемый для себя способ интеграции TeX'а и WWW.

Подготовленной TeX'овской аудитории окажутся полезными сведения об XML и его применении в создании электронных документов. Для пользователей LaTeX'а концепция независимой от способа представления разметки электронного документа будет хорошо знакомой и понятной. Широкое распространение XML в современных информационных системах позволит TeX'овской аудитории вдохнуть вторую жизнь в существующие TeX-документы.

[1] Перевод первой книги *Путеводитель по пакету LaTeX и его расширению LaTeX 2ε* вышел в издательстве «Мир» в 1999 г., перевод третьей — *Путеводитель по пакету LaTeX и его графическим расширениям* — готовится к изданию. — *Прим. ред.*

Большой интерес читателя наверняка вызовут главы, посвященные работе с файлами формата PDF и, в частности, с программой pdfTEX. Практически повсеместная распространенность программы Acrobat Reader фирмы Adobe — разработчика и владельца формата PDF — превратила этот формат в один из стандартов *de facto* для обмена электронными документами. Благодаря автору pdfTEX пользователи TEX'а могут практически безо всяких усилий готовить свои публикации в формате PDF, пригодном для размещения на WWW. Однако между «классической» статичной TEX'овской верстkой и интерактивной Web-страницей существует множество крупных и мелких различий. Преодолеть возникающие при этом крупные и мелкие трудности вам поможет эта книга.

Сто́ит отметить, что для русского читателя на пути от TEX'а к WWW встречается еще одна проблема, помимо описанных в книге: проблема русификации. К счастью, на сегодня бо́льшая часть задач, связанных с русификацией, имеет более или менее простые решения. Так, в современные дистрибутивы (LA)TEX'а входит все необходимое для работы с русским языком и кириллическими шрифтами. В частности, в TEX-архивах CTAN в директории `ftp://ctan.tug.org/tex-archive/macros/latex/required/cyrillic/` лежит русификация `Cyrillic`, основанная на кодировке T2.

Пожалуй, самая типичная неприятность на сегодня заключается в том, что для подготовки PDF-документов необходимо включать в документы кириллические шрифты в формате Type 1. Для этого есть два пути: приобрести набор шрифтов у одного из коммерческих поставщиков или воспользоваться одним из существующих бесплатных наборов. Таким набором, например, специально предназначенным для работы с TEX'ом и pdfTEX'ом, является пакет `PsCyr`, поддерживаемый в настоящее время А. И. Лебедевым. Этот пакет можно найти в Интернете по адресу `ftp://mch5.chem.msu.su/pub/russian/psfonts/`.

Несмотря на то, что книга изобилует ссылками на электронные адреса сайтов в Сети, где находится тот или иной пакет, не все эти сайты оказались «живыми». Кроме того, не все читатели имеют свободный доступ в Интернет и способны скачивать большие объемы информации. Поэтому мы сочли полезным для читателя иметь описываемые в книге пакеты «под рукой». С любезного разрешения международной TEX-группы TUG и Себастиана Ратца — автора CD-собрания TEX-пакетов под разные платформы «TEX Live 6» — перевод книги снабжен вариантом этого CD ROM'а, несколько модернизированным под нужды российского TEX-пользователя. В него добавлены документация на русском языке и пакет `PsCyr`. Работу по модернизации диска выполнили О. Г. Лапко и Б. В. Тоботрас.

Хочется надеяться, что и книга, и диск принесут пользу всем, кто работает в пакете TEX (или собирается начать работать): авторам, желающим публиковаться в Сети; разработчикам сайтов, размещающим научные работы, подготовленные в TEX'е; издателям электронных публикаций и просто любознательным пользователям Интернета.

Б. В. Тоботрас

Предисловие

Эта книга была написана для того, чтобы помочь авторам, прежде всего, ученым, которые хотели бы разместить свою информацию во Всемирной паутине (Сети) или какой либо другой системе воспроизведения гипертекста, но недостаточно свободно чувствуют себя в общении с такими программами, как Microsoft Word или QuarkXpress. Вложив средства в приобретение системы разметки текста наподобие LaTeX'а, они рассчитывают удовлетворить свои особые потребности, например в таких областях, как математика, неевропейские языки или алгоритмическая графика. Эта книга расскажет им о том, как:

- работая в LaTeX'е, полностью использовать возможности формата Adobe Acrobat;
- преобразовать традиционные документы в формат HTML или XML;
- использовать в Web-приложениях привычную математическую символику;
- использовать LaTeX для подготовки Web-страниц;
- прочитать и написать простой документ на языке XML/SGML;
- используя TeX или PDF, получить высококачественную печать размещенных в Сети страниц формата XML или HTML.

LaTeX как хранилище документов для Интернета

Всемирная паутина (The World Wide Web — WWW) уже опутала все стороны жизни общества, и наука не представляет собой исключения из этого правила. В этом нет ничего удивительного, особенно если учесть, что сама парадигма Сети зародилась в Европейском центре ядерных исследований (CERN) — одной из крупнейших в мире научных лабораторий.

Тот интерфейс к Сети, который в настоящее время стал поистине повсеместным, появился в результате фундаментальных исследований, проводившихся в CERN'е в начале 90-х гг. XX века. До этого времени использование

Интернета было почти исключительно прерогативой специалистов. Лишь благодаря таланту и интуиции Тима Бернерс-Ли и его сотрудников, физики, живущие в разных концах света, но совместно участвующие в программе CERN'a по физике высоких энергий, получили инструмент, который позволил им обмениваться через Интернет данными и информацией интуитивно понятным и «дружественным», т. е. удобным для пользователя, способом. Эта работа непосредственно привела к созданию языка разметки HTML, протокола HTTP и схемы адресации URL — трех китов, на которых стоит Сеть. Разработчики сразу же приняли дальновидное решение — предоставить всему Интернет-сообществу свободный доступ к своей работе. К тому же, благодаря появлению обладающего графическим интерфейсом навигатора *Mosaic*, парадигма Сети была с одинаковым энтузиазмом воспринята как разработчиками, так и пользователями. Рост числа Web-сайтов и пользователей Сети принял экспоненциальный характер, и кульминацией этого процесса стал состоявшийся в мае 1994 г. в CERN'е конгресс «Woodstock of the Web». CERN как научная лаборатория, предназначенная для проведения фундаментальных исследований, не имела возможностей для того, чтобы и дальше координировать развитие Сети, и поэтому эти обязанности были переданы международному консорциуму World Wide Web Consortium [↪W3C], тремя основными организациями которого в настоящее время являются: Лаборатория вычислительной техники (информатики) Массачусетского технологического института MIT [↪MIT], США, Национальный институт исследований в области компьютерной обработки данных и автоматики INRIA [↪INRIA], Франция, и Университет Киото [↪KEIO], Япония. Поддержку консорциуму также оказывают Управление перспективных исследовательских программ DARPA [↪DARPA], США, и Европейская Комиссия [↪EC].

Вся история возникновения Сети является наглядным свидетельством того, что фундаментальные научные исследования могут иметь совершенно неожиданные, но очень важные и широкомасштабные последствия для общества.

Хотя большинство этого и не осознает, но без всякого сомнения основным языком разметки электронных документов в настоящее время является SGML — стандартный обобщенный язык разметки (в форме HTML — общеупотребительного лингва-франка Сети). Аналогичным образом, LaTeX уже более десяти лет используется для разметки научных документов. Даже сегодня LaTeX'у нет реальной альтернативы, когда речь идет о печати текстов, содержащих множество математических формул. Поэтому представляется разумным поиск способов создания (по возможности) автоматических процедур, которые преобразовывали бы LaTeX'овские документы в форму, пригодную для использования в Сети. С другой стороны, желательно, чтобы при работе с документами, размеченными в формате XML или HTML, можно было использовать высокие типографские характеристики TeX'овского процессора.

Поэтому в этой книге мы объясняем, как можно использовать LaTeX для организации работы с электронными документами в Сети. Мы показываем, как уже существующим LaTeX'овским документам можно дать новую жизнь в Сети, преобразуя их в формат HTML, и как можно более полно использовать

гипертекстовые возможности HTML с помощью некоторых пакетов расширения LaTeX'а. На сегодняшний день HTML и Web-навигаторы не в состоянии достаточно хорошо оперировать с такими нетекстовыми компонентами документа, как рисунки (которые преобразуются в растровое изображение) или математические формулы. Мы также обращаемся к таким темам, как преобразование LaTeX'овских документов в формат PDF и возможности интерпретации LaTeX'овских команд непосредственно расширениями какого-либо навигатора.

Кроме того, мы посвящаем вас в секреты XML — расширяемого языка разметки, который использует подмножество языка SGML и предназначен для замены языка HTML, поскольку допускает возможность расширений, зависящих от приложения. В частности, мы рассматриваем MathML— язык математической разметки — его синтаксис, способы генерации и области возможного использования.

Как самостоятельную тему мы обсуждаем различные подходы к преобразованию исходных Web-документов, размеченных с помощью XML или HTML, в LaTeX или PDF с целью их наилучшей печати, в частности с использованием стилей DSSSL и XSL.

За прошедшие годы было разработано множество средств для преобразования TeX'овских исходных файлов в формат HTML. Те программы, которые описаны в нашей книге — это типичные образцы, отобранные главным образом потому, что мы сами их использовали и поэтому хорошо с ними знакомы. То, что другие средства не описаны в этой книге, ни в коем случае не означает, что мы считаем их менее полезными или худшего качества.

Логическая схема книги

Мы рекомендуем всем читателям начать с гл. 1, потому что в ней объясняется, что Сеть — это не угроза, а наоборот, благо для LaTeX'а, и почему следует или же не следует продолжать пользоваться LaTeX'ом. Здесь же мы коротко рассказываем о Сети с точки зрения пользователя LaTeX'а.

В главе 2 рассматривается вопрос о том, как с помощью языка описания страниц PDF обеспечить высококачественное воспроизведение страниц гипердокументов.

Главы 3 и 4 посвящены преобразованию LaTeX'овских документов в формат HTML. В главе 3 мы обсуждаем LaTeX2HTML, в котором для интерпретации LaTeX'овских исходных документов и генерации HTML-кода используется язык Perl. Существует возможность легко добавлять пакеты расширений в виде программ на Perl, а благодаря разнообразным расширениям LaTeX'овского языка LaTeX2HTML становится действительно высокоэффективным инструментом для создания гипертекстовых документов.

Другой подход представлен в гл. 4, в которой рассматривается, как в TeX4ht для генерации HTML или XML переопределяются LaTeX'овские макро TeX'а, а для изображения математических формул по возможности используется также приложение MathML.

В последнее время появились навигаторы (с динамически подключаемыми модулями), способные непосредственно интерпретировать математическую разметку. В главе 5 рассматриваются те из них, которые могут непосредственно воспринимать широкие подмножества «родного» LaTeX'овского кода без его предварительного преобразования в формат HTML. В частности, `techexplorer` — разработанный фирмой IBM динамически подключаемый модуль для Netscape и Internet Explorer'a и WebEQ — апплет языка Java, предназначенный для изображения математических формул.

Более широкий взгляд на проблему характерен для гл. 6, которая постепенно подводит нас к стандартному обобщенному языку разметки SGML (Standard Generalized Markup Language). В ней объясняется, почему расширяемый язык разметки XML (Extensible Markup Language) — более простой и более «дружественный к Интернету и пользователю» вариант языка SGML — со временем должен стать важным элементом любой будущей стратегии работы с документами в Интернете. Можно ожидать, что объединение языка XML с объектными базами данных и другими современными объектно-ориентированными технологиями коренным образом изменит управление документооборотом на всех уровнях. Будут описаны средства для создания и интерпретации XML, а также некоторое время будет уделено вопросу разработки LaTeX'оподобного языка разметки XML.

Первоначально TeX был разработан Доном Кнутом для печати его математических книг в соответствии с лучшими полиграфическими традициями. Поэтому неудивительно, что именно TeX был предложен в качестве средства набора текстов для материалов Сети (Web-документов). Средства для преобразования исходных документов в формате XML в различные выходные форматы описываются в гл. 7. Детально будет рассмотрено использование каскадных стилей CSS (Cascading Style Sheets), языка семантики и определения стиля документа DSSSL (Document Style Semantics and Specification Language), а также расширяемого языка стилей XSL (Extensible Stylesheet Language)для управления процессом трансляции.

В главе 8 самое пристальное внимание уделяется «горячей» теме максимально полного использования оптимальной LaTeX'овской системы математических обозначений для преобразования LaTeX'овской разметки в XML и MathML (Mathematical Markup Language — язык математической разметки), который совместно с XML используется для отображения и работы с математическими формулами в Сети (обзор различных способов работы с математическими формулами в Сети можно найти в Foster (1999)).

Книга заканчивается приложениями, которые содержат техническую информацию, дополняющую главы книги. Мы знакомим читателей с пространством имен Сети, обсуждаем проблемы интернационализации и даем обзор некоторых правил описания типа документа (DTD) в языке XML.

История и авторство

В начале работы над книгой *The LaTeX Graphics Companion* Себастиан Ратц и Мишель Гуссенс намеревались включить в нее вопросы, связанные с PDF, SGML и Сетью (Web). Однако стало ясно, что, для того чтобы осветить материал, посвященный гипертексту и языку SGML, им понадобится написать отдельную книгу. Поэтому сразу же по завершении работы над *Graphics Companion* началась работа над книгой *Web Companion*, которую вы держите в руках. По сравнению с большинством работ по TeX'у пакеты и программы, предназначенные для работы с TeX'ом в Сети, сменялись очень быстро. Поэтому, для того чтобы приведенная в книге информация была точной и свежей, было решено обратиться с просьбой о сотрудничестве к авторам трех важнейших пакетов.

Основным автором главы о LaTeX2HTML является Росс Мур, глава о TeX4ht написана Эйтаном М. Гурари, а глава о `techexplorer`'е фирмы IBM и WebEq — Робертом Сьютором. Остальные главы поделили между собой М. Гуссенс и С. Ратц при существенном участии Э. Гурари, Р. Мура и Р. Сьютора, которое выразилось в комментариях к материалу, добавлении разделов и участии в обсуждении затрагиваемых вопросов.

Возможно, что именно благодаря Интернету за все время работы над книгой пять авторов ни разу не собирались все вместе. Ближе всего они сошлись на приятном обеде во время Европейской встречи по TeX'у в Сен-Мало (Saint-Malo) в 1998 г., где присутствовали все, кроме Эйтана М. Гурари.

Как найти и использовать все эти пакеты и программы

Все описываемые в этой книге пакеты и программы, кроме тех, о которых явно сказано что-либо другое, можно свободно получить в архивах общедоступного программного обеспечения. Одни из них являются общественным достоянием, тогда как другие защищены законом об авторских правах. Некоторые программы распространяются только в виде исходного текста либо предназначены только для определенных компьютерных платформ. Поэтому нужно быть готовым к тому, что в некоторых случаях вам придется немного «испачкать руки». Мы не можем также гарантировать, что результаты, полученные при использовании более новых версий рассматриваемых пакетов и программ, будут такими же, как и приведенные в нашей книге. Многие из этих программ находятся в стадии интенсивного совершенствования, и их новые или модифицированные версии выходят несколько раз в год. Мы готовили эту книгу к печати осенью 1999 г. и демонстрировали в ней работу распространенных в то время версий.

Тем, кто регулярно пользуется Всемирной паутиной, известно, что сайты все время исчезают или меняют свою структуру, и поэтому бывает трудно

уследить за изменениями унифицированных указателей ресурсов (URL), и к тому же при этом легко ошибиться. По этой причине в тексте нашей книги мы не приводим самих формальных адресов URL, а вместо этого даем ссылки (набранные следующим образом: [↪W3C]) на каталог унифицированных указателей ресурсов (начинающийся на с. 563). Этот постоянно обновляемый каталог будет доступен в директории `info/lwc` на CTAN, где можно будет также найти исходные коды для большинства приведенных нами примеров. Пытаясь сделать некоторые акронимы более понятными, мы включили в книгу словарь специальных терминов (начинающийся на с. 552).

Сведения об издании

Эта книга была подготовлена в LaTeX'е. Текст в основном набран шрифтом Adobe Janson, в качестве рубленого шрифта использован European Modern Sans фирмы Y&Y, математические выражения набраны шрифтом Y&Y MathTime Plus, а для буквального воспроизведения текста пишущей машинки использован шрифт European Modern typewriter фирмы Y&Y[1].

Франк Миттельбах усовершенствовал и обобщил LaTeX'овский стиль, взяв за основу стиль, разработанный им вместе с Себастианом Ратцем для книги *The LaTeX Graphics Companion*. А тот стиль в свою очередь был получен из разработанного Франком Миттельбахом и Мишелем Гуссенсом стиля для книги *The LaTeX Companion*[2].

Благодарности

Мы признательны Нельсону Бийби (Университет Юты), Тиму Брею (Textuality), Мими Бэрбанк (Университет штата Флорида), Дэвиду Карлайлу (NAG), Гансу Хагену (Pragma), Тану (Университет Масарика, Брно), Т. В. Раману (Adobe Systems), Д. П. Стори (Акронский университет), Майклу Даунзу (Американское математическое общество), Питеру Флинну (Университетский колледж, Корк), Крису Мэйдену (O'Reilly), Томасу Мерцу (Мюнхен) и Крису Роули (Открытый университет) за консультации, содействие и замечания по первоначальному тексту книги.

Себастиан Ратц хотел бы воспользоваться случаем, чтобы поблагодарить Тэнмоя Бхаттачариа, Дэвида Карлайла, Патрика Дейли, Янниса Хараламбуса и многих других за их помощь в работе над пакетом `hyperref`, а также Бертольда Хорна (Y&Y) за финансирование части работы.

[1] Речь идет, разумеется, об английском оригинале. Русский перевод набран шрифтом Computer Modern с кириллическим расширением LH (авторы О. Г. Лапко, А. Б. Ходулев).— *Прим. ред.*

[2] Стилевой файл русского издания переработан на основе любезно предоставленных фирмой Pearson исходных стилевых файлов.— *Прим. ред.*

Эйтан М. Гурари очень благодарен Гертьяну Кляйну и Себастиану Ратцу за их вклад в разработку TEX4ht. На начальных этапах работы над проектом Гертьян помогал советами и написал кусок программы, обеспечивающий мобильность TEX4ht. Кроме того, он сделал много обстоятельных замечаний и предложений, касающихся конфигурации создаваемых документов. Себастиан, присоединившийся к проекту на более поздних этапах, был чрезвычайно отзывчив, ставил требующие решения задачи, участвовал в разработке интересных файлов конфигурации, настойчиво рекламировал разрабатываемую систему и очень серьезно редактировал данный раздел книги. Помимо профессиональных аспектов, Гертьян и Себастиан были горячими сторонниками Сети.

Роберт Сьютор выражает свою благодарность Биллу Паллибланку, Маршаллу Шору и Дику Дженксу из научно-исследовательского отдела фирмы IBM за оказанную ими поддержку при разработке `techexplorer`'а.

Росс Мур хотел бы прежде всего поблагодарить Никоса Дракоса за его дальновидность, проявленную при разработке транслятора LATEX2HTML и при проектировании основных принципов его дизайна. Здесь не хватит места, чтобы перечислить всех, кто внес существенный вклад в эту работу; мы благодарим их всех. Среди них мы хотим выразить особую благодарность Маркусу Хеннеке и Хербу Свану, которые оказали самую большую помощь, когда Никос не смог больше участвовать в работе. Мы также хотим выразить признательность Йенсу Липману, Скотту Нельсону и Мареку Рушалю, которые продолжают оказывать необходимую помощь в разработке, обслуживании и распространении последних версий программы LATEX2HTML. Во-вторых, Росс хочет поблагодарить Мишеля Гуссенса, Мими Джетт, Джерольда Марсдена, Роберта Майнера и Кристофера Роуза за финансирование поездок по всему свету — в разные места, где обсуждались и/или развивались планы создания расширений для LATEX2HTML. Некоторые из этих поездок оказали непосредственное влияние на содержание этой книги.

Со стороны издательства, Франк Миттельбах (редактор серии) великолепно справился с задачей удержать нас на прямом и узком пути, а Питер Гордон (Addison Wesley Longman, Inc.) обеспечил авторов всем, что они только могли пожелать: ободрением, поддержкой, шутками и помощью. Когда же дело дошло до производства, то Джон Фулер, Хелен Голдстайн и Маурин Уиллард были очень терпеливы к нашим идиосинкразиям, бережно вели нас к завершению работы и превращали нашу сомнительную прозу в настоящий английский язык.

Обратная связь

Мы хотели бы обратиться к вам, дорогие читатели, с просьбой о сотрудничестве. Мы искренне призываем вас присылать свои суждения, предложения или замечания любому из авторов. Мы будем рады исправить в следующем издании любые ошибки или упущения, а также готовы рассмотреть предло-

жения, касающиеся различных усовершенствований, или включения важных разработок, которые мы могли упустить из виду. Мы будем поддерживать список опечаток, который будет находиться в файле `webcomp.err` дистрибутива LaTeX'а. В этом же файле будут приведены и действующие адреса авторов.

Многие из Web-приложений, которые мы описываем в этой книге, продолжают быстро развиваться. Исходные программы для приведенных в книге примеров, хранящиеся в директории `info/lwc` на узлах CTAN, будут постоянно обновляться, чтобы гарантировать их работоспособность при использовании будущих версий спецификаций W3C. Это относится, в частности, к примерам, приведенным в разд. 7.6.6, поскольку словарь форматирующих объектов языка XSL пока еще полностью не сформирован.

Сеть, ее документы и LaTeX

В этой главе мы познакомимся с зарождением Всемирной паутины (World Wide Web) и с ее основными компонентами, благодаря которым она превратилась в поистине глобальное средство общения, не зависящее от культурных и языковых особенностей. Будет дано краткое введение в HTML — язык разметки, который в настоящее время используется в большинстве документов Сети (Web-документов). В большей части рассматриваемых в книге средств применяется для разметки язык HTML или, в более общем случае, XML, поэтому основные сведения об их синтаксисе окажутся впоследствии весьма кстати.

Затем мы в общих чертах рассмотрим различные подходы к работе с содержательными научными документами, обращая особое внимание на ту роль, которую LaTeX играет в этом окружении в качестве формата входных или выходных данных. Это позволит сделать вывод о том, что развитие Сети и новый облик электронных документов предоставляют ученым богатые возможности для обмена результатами и данными.

* * *

Интернет и, в особенности, Всемирная паутина достигли пика своей общественной известности, когда Президент Билл Клинтон упомянул о них в своем докладе о положении США в американском Конгрессе 27 января 1998 г. [↪SOTU98]:

> Мы должны дать всем людям на свете возможность исследовать самые дальние уголки киберпространства. Подумайте об этом: когда я первый раз докладывал вам о положении Соединенных Штатов, только несколько физиков пользовались Всемирной паутиной. В самом прямом смысле всего лишь кучка людей. Теперь же в школах и библиотеках, дома и на работе миллионы и миллионы американцев путешествуют по Сети каждый день... Но мы также должны быть уверены в том, что защищаем растущий коммерческий потенциал Интернета....
> В частности, я прошу Конгресс усилить поддержку создания нового поколения Интернета. Вы знаете, что он засоряется. А новое поколение Интернета будет работать на скоростях в тысячи раз больших, чем сегодня.

А 19 января 1999 г. президент Клинтон с гордостью объявил [↪SOTU99]:

> ... Мы успешно идем к своей цели: подключить к Интернету каждую классную комнату и библиотеку.

В наши дни Белый Дом — не единственный, кто серьезно относится к Сети (приведенная выше цитата взята из одного из тысяч правительственных документов, доступных на его Web-сайте). Любая фирма, школа, общественная или коммерческая организация, а вскоре и каждый индивидуум захотят присоединиться к Сети. В научных (и околонаучных) публикациях стало обычным использование URL-адресов для ссылок на дополнительную информацию, а во многих случаях научная работа появляется сначала на Web-сайте, причем задолго до опубликования в «бумажном» виде в каком-либо общепризнанном журнале.

В научном мире мы наблюдаем исключительно стремительную эволюцию от некоторых колебаний в применении новой Web-технологии до энтузиазма в использовании баз данных препринтов для ускорения распространения информации. Основная проблема, которую осталось решить, это *качество* как формы, так и содержания. Качество содержания можно обеспечить, используя уже сформировавшуюся систему перекрестного рецензирования, основанную на экспертизе многими существующими издательствами, которые начинают играть новую роль — «агентов по проверке информации». Качество воспроизведения — это проблема, которая до сих пор полностью не решена. Активно обсуждался вопрос о том, могут ли экраны современных компьютеров обеспечить необходимую детализацию для точного воспроизведения многомерной визуальной информации, характерной для математических или химических формул. До настоящего времени было предпринято несколько попыток определить способы преодоления низкой разрешающей способности компьютерного экрана (в лучшем случае несколько пикселов на миллиметр) при сохранении возможностей гибкого интерактивного поиска по гипертексту.

Следует также принять во внимание ситуацию в таких регионах, как Россия, Юго-Восточная Азия и Африка, где из-за серьезных финансовых трудностей часто даже не поднимается вопрос о печати узкоспециализированного документа большим тиражом. Поэтому единственным способом публикации оказывается распространение по Сети в электронном виде. Таким образом, Сеть — это не просто дополнительный канал распространения традиционной печатной продукции, но для большей части света это еще и необходимость совместного доступа к информации и извлечения пользы из создаваемых ею ценностей и прогресса.

Какую форму примет в будущем информация в Сети? Консорциум производителей программного обеспечения для WWW (*World Wide Web Consortium* — W3C) был образован для того, чтобы содействовать совместной работе компаний — производителей программного обеспечения, направленной на достижение таких решений, которые гарантировали бы возможность взаимодействия и отсутствие необходимости в дополнительном контексте для интерпретации электронных документов и данных в Интернете.

Глядя на свою собственную систему, мы видим, что Сеть не представляет угрозы LaTeX'у, а наоборот, открывает для него дополнительные возможности. Мы думаем, что ученые при создании своих документов и впредь будут пользоваться наиболее удобными для себя инструментами (будь то LaTeX или иная система), но они получат новые благоприятные возможности для того, чтобы сделать результаты общедоступными.

Мы надеемся с помощью этой книги убедить вас в том, что благодаря описываемым в последующих главах средствам у вас не возникнет необходимости выбирать между типографским качеством TeX'а и глобальностью Сети. У вас будет и то, и другое.

1.1 Сеть: окно в Интернет

В 90-е годы популярность Интернета стремительно росла, пересекая национальные и междисциплинарные границы, и все это в значительной степени благодаря появлению Всемирной паутины, разработанной в начале 90-х в Европейской лаборатории физики частиц (CERN) (правильно — Европейская организация ядерных исследований) [↪CERN].

Прежде чем обратиться непосредственно к Сети, поговорим немного об Интернете. Потребность военного комплекса США в надежных каналах связи, охватывающих всю страну, привела к разработке сети с пакетной коммутацией. В сети *Arpanet* (Advanced Research Project Agency network — сеть Управления перспективными исследовательскими программами) сообщения могут достигать пункта назначения по нескольким маршрутам, благодаря чему обеспечивается доставка при неблагоприятных условиях. Работа над Arpanet послужила основой для протокола управления передачей/Интернет-протокола (TCP/IP) — метода, который в настоящее время используется в сетях почти повсеместно для разделения сообщений на отдельные пакеты данных (дейтаграммы), каждому из которых присваивается уникальный порядковый номер. Эти сообщения, двигаясь по множеству маршрутов, достигают пункта своего назначения, где они собираются воедино, для того чтобы сформировать текст исходного сообщения (Tanenbaum (1996)). Поскольку Arpanet, а впоследствии и заменивший его примерно в 1988 г. Internet объединяют все виды компьютеров и физических сетей, TCP/IP превратился в лингва-франка протоколов связи.

В 80-е годы самым популярным способом общения пользователей в сети была электронная почта (*e-mail*), а при необходимости в качестве средств непосредственного соединения использовались протокол передачи файлов (*ftp*) и эмуляция терминала (*telnet*). Однако не ограниченные государственными границами и простые в употреблении системы распределения информации, в которых не нужно сохранять соединение включенным, не возникали еще долго.

К 1989 г. TCP/IP окончательно утвердился в CERN'е, а Интернет заменил большинство закрытых доморощенных протоколов связи. Именно в том году Тим Бернерс-Ли сформулировал свои первые идеи того, что позже превратилось во Всемирную паутину. При этом он мог основываться на своем опыте, полученном в области «распределенных вычислений» и, в особенности, при рабо-

те над вызовами удаленных процедур. Полезным также было знакомство с Mac и NeXT — двумя компьютерными системами, которые отличались богатым и дружественным интерфейсом для (распределенного) программирования. Эти системы в то время (а Mac и поныне) пользовались большой популярностью у многих физиков. К середине 90-х Тим и один из его коллег — Робер Кайо[1] — придали окончательную форму трем основным программным протоколам Всемирной паутины:[2]

1. *Hypertext Transport Protocol — Протокол Передачи Гипертекста* (HTTP), метод, который позволяет различным WWW-серверам обмениваться информацией (см. разд. 1.1.1);
2. *Universal Resource Locator — Универсальный Локатор Ресурса* (URL), универсальная схема адресации, позволяющая определить местонахождение любой информации в Интернете (см. разд. 1.1.2);
3. *Hypertext Markup Language — Язык Разметки Гипертекста* (HTML), язык для разметки информации (см. разд. 1.1.3).

К концу 1990 г. появилась демонстрационная версия Web-клиента для NeXT. Вскоре за ним последовал строковый навигатор, который можно было легко настроить на работу со многими используемыми в CERN'е форматами представления информации. Таким образом открытая (благодаря URL) архитектура позволила повторно использовать тысячи существующих страниц документов и элегантно встроить их в новую парадигму WWW.

Хотя в июне 1990 г. на семинаре было сделано сообщение о Сети, а из компьютерного информационного бюллетеня CERN *Computer Newsletter* 204 о ней узнали все, работающие в области физики высоких энергий, к декабрю 1991 г. во всем мире насчитывалось не более десяти Web-серверов, а физики считали WWW всего лишь еще одной из тех компьютерных игрушек, которые отвлекают их от занятий физикой. Дело оставалось только за программой-«приманкой», которая на практике показала бы все преимущества Сети как интерфейса к всемирному Интернету. Хотя в начале 1993 г. уже существовало несколько навигаторов с графическим пользовательским интерфейсом, все по-настоящему началось с программы Mosaic Марка Андриссена. Впервые пользователи смогли просто щелкать мышью, чтобы перемещаться между документами, составляющими информационную паутину — просто и наглядно.

В мае 1994 г. в CERN'е состоялась первая конференция по Всемирной паутине, которая впоследствии стала известна как Woodstock of the Web. Действительно, 1994 г. можно считать поистине поворотным, когда главенствующая роль в развитии WWW перешла от CERN'а к WWW-консорциуму [↪W3C], а кроме того, была образована весьма влиятельная компания Netscape. С тех пор Microsoft, а к 1998 г. и все остальные гиганты индустрии программных средств заинтересовались теми выгодами, которые сулит всемирная Сеть. В наши дни вряд ли найдется хоть одна компания или правительственная организация, ко-

[1] Робер был специалистом по `Hypercard` — программе работы с гипертекстом фирмы Macintosh, в которой использовался язык подготовки сценариев `hypertalk`.

[2] Более подробно с ранней историей Сети можно познакомиться в [↪WEBHIST] и [↪RAGHIST].

торая на визитных карточках или печатных бланках вместе с номером факса и адресом электронной почты не указала бы с гордостью URL своей домашней страницы.

1.1.1 Протокол передачи гипертекста

HTTP — это язык, на котором WWW-серверы общаются друг с другом. Говоря техническим языком, это протокол прикладного уровня для распределенных, гипермедийных информационных систем. Его первая версия (почему-то получившая известность как HTTP/0.9) была разработана Тимом Бернерс-Ли с сотрудниками в 1990 г. в рамках инициативы CERN'а в области WWW. Это был простой протокол для передачи через Интернет двоичных данных. Появившаяся в 1996 г. версия HTTP/1.0 поддерживала сообщения, использующие формат многоцелевых расширений электронной почты в Интернете — Multipurpose Internet Mail Extensions (MIME) [↪RFC2045]. Благодаря наличию элементарной поддержки метаданных появилась возможность некоторой семантической интерпретации содержания. Оставались, однако, некоторые проблемы, и это привело к тому, что в январе 1997 г. была описана версия HTTP/1.1 [↪HTTPRFC]. В ней были введены «поиск», «обновление клиента» и «комментарий», а также предусматривалась возможность использования универсальных идентификаторов ресурса Uniform Resource Identifiers (URIs). Более того, теперь HTTP мог использоваться в качестве универсального протокола для взаимодействия агентов пользователя и «посредников»/маршрутизаторов с другими системами Интернета. Тем самым общий стандартный гипермедийный доступ к ресурсам, стал возможен из разнообразных приложений.

Так как Всемирная паутина продолжает чрезвычайно быстро расширяться, и ее дальнейшее развитие очень сильно зависит от эффективности HTTP, возник ряд вопросов монолитной структуры HTTP. Поэтому проблемная группа проектирования Интернета — Internet Engineering Task Force (IETF) [↪IETF] в настоящее время готовит Новое Поколение протокола HTTP. В нем будет использоваться модульный подход, предусматривающий эффективное и маленькое ядро, дополненное передачей сообщений, универсальным дистанционным вызовом методов и многоуровневой обработкой документов. Это должно гарантировать глобальную расширяемость, сетевую эффективность и гибкость транспорта (более подробную информацию см. в [↪HTTPNG]).

1.1.2 Универсальные указатели и идентификаторы ресурсов

URL — это «почтовый адрес» документа в Сети. Синтаксис и семантика унифицированного указателя ресурса (URL), предназначенного для определения местонахождения и доступа к ресурсу через Интернет, впервые были разработаны в CERN'е в 1990 г., а впоследствии формализованы [↪RFC1738].

Вскоре стало ясно, что исходный синтаксис URL слегка жестковат для работы с перемещающимися ресурсами, как, например, в случае, когда изменяется

Интернет-адрес сервера, на котором хранится документ. Поэтому была выдвинута идея унифицировать синтаксис выражений для имен и адресов объектов в Интернете [↪RFC1630]. В Сети должна быть обеспечена возможность работы с объектами, доступ к которым осуществляется с использованием все большего числа протоколов, как уже существующих, так и могущих появиться в будущем, в которых команды доступа кодируются в виде строки адреса объекта.

Универсальный идентификатор ресурса (URI) [↪RFC2396] представляет собой элемент этого универсального множества имен из зарегистрированных пространств имен и адресов в соответствии с зарегистрированными протоколами. URL является частным случаем URI, который отображается на алгоритм доступа с использованием сетевых протоколов. Унифицированное имя ресурса — Uniform Resource Name (URN) должно быть неизменным, не зависящим от местоположения идентификатором ресурса, способным отображать другие пространства имен, обладающие сходными свойствами, в пространство URN [↪RFC2141].

В общем случае структура URI имеет вид:

схема `://`*имя_хоста* [`:`*порт*] `/`*путь*`/`*имя_файла* [`#`*фрагмент*]

При использовании протокола HTTP можно получить, например,

```
http://info.internet.isi.edu:80/in-notes/rfc/files/rfc1630.txt
http://www.oasis-open.org/cover/topics.html#entities
```

а вот примеры, относящиеся к локальной файловой системе (Microsoft Windows или UNIX):

```
file:///D:\lark\lark.html
file:/afs/cern.ch/user/g/goossens/hagel.html
```

Следующим образом можно задать протоколы, ориентированные на соединения, например, ftp или telnet:

```
ftp://ftp.jclark.com/pub/xml/xt.zip
telnet://mycomputer.cern.ch
```

1.1.3 Язык разметки гипертекста

HTML — это язык элементарной разметки, предназначенный для создания документов, которые будут публиковаться в Сети. До 1995 г. он не имел формального описания («Определение Типа Документа» SGML — см. разд. 6.3.3) [↪RFC1866]. В этой книге будет рассмотрено подмножество второй версии HTML (еще одно краткое введение можно найти в [↪RAGHTML]). Современная четвертая версия HTML (см. разд. 6.2) обладает существенно более широкими функциональными возможностями, хотя основные принципы остались неизменными. Более того, HTML постепенно становится совместимым с XML (см. приложение B.5 и [↪HTMLINXML]). Обзор некоторых из основных элементов языка HTML приведен в табл. 1.1. Ниже мы рассмотрим некоторые из них

Таблица 1.1. Основные метки языка HTML.

Общая структура HTML-документа		
`<html>`		
`<head>`		
`<title>...</title>` название документа		
`<meta>...</meta>` общая мета-информация		
`</head>`		
`<body>...</body>`		
`</html>`		
Элементы, используемые в теле документа		
Элементы текста		
`<p>` начало нового абзаца		
`<pre>...</pre>` предварительно форматированный текст (допускается наличие вложенных элементов, но не всех)		
Заголовки		
`<h1>...</h1>` заголовок первого уровня	`<h2>...</h2>` заголовок второго уровня	
`<h3>...</h3>` заголовок третьего уровня	`<h4>...</h4>` заголовок четвертого уровня	
`<h5>...</h5>` заголовок пятого уровня	`<h6>...</h6>` заголовок шестого уровня	
Логическая разметка		
`<em>...</em>` выделение	`<code>...</code>` машинный код	
`<samp>...</samp>` пример вывода	`<kbd>...</kbd>` текст, вводимый с клавиатуры	
`<var>...</var>` определение переменной	`<cite>...</cite>` ссылка	
Шрифтовое оформление		
`<b>...</b>` **полужирный текст**	`<i>...</i>` *курсивный текст*	
`<u>...</u>` подчеркнутый текст	`<tt>...</tt>` `шрифт пишущей машинки`	
Списки		
`<dl>` начало перечня обозначений	`<ul>` начало ненумерованного списка	`<ol>` начало нумерованного списка
`<dt>` *термин* `<dd>` *описание*	`<li>` пункты списка	`<li>` пункты списка
`</dl>` конец перечня обозначений	`</ul>` конец ненумерованного списка	`</ol>` конец нумерованного списка
Гиперссылки и анкеры		
`<a name="имя_анкера">...</a>` определение анкера		
`<a href="#имя_анкера">...</a>` ссылка на анкер в том же самом файле		
`<a href="URI#имя_анкера">...</a>` ссылка на заданное место в другом файле		
Изображения		
`<img src="URI">...</img>` вставка графического объекта		

более подробно. Это позволит вам разобраться в структуре исходного HTML-документа, если вы когда-нибудь его увидите. Большинство навигаторов позволяет увидеть исходный текст отображаемого HTML-документа с помощью пункта меню наподобие «Посмотреть исходный текст».

1.1.3.1 Простейший HTML-документ

Вот простейший пример записи на языке HTML:

```
<TITLE>Простейший пример HTML</TITLE>
<H1>Заголовок первого уровня</H1>
<P>Добро пожаловать в мир HTML!
<P>Начинается второй абзац.
```

В языке HTML *метки* используются главным образом для того, чтобы сообщить Web-навигатору информацию о способе отображения текста. В приведенном примере используются:

- метка <TITLE> (которому соответствует завершающая метка </TITLE>) для описания названия документа,
- метка заголовка <H1> (с соответствующим окончанием </H1>) и
- метка начала абзаца <P>.

Метки языка HTML[1] начинаются с левой угловой скобки (<), за которой следует имя *типа элемента* с несколькими необязательными атрибутами, и заканчиваются правой угловой скобкой (>). Обычно метки встречаются парами, например, <H1> и </H1>, в которых закрывающая метка выглядит точно так же, как открывающая, но начинается с косой черты (/). В рассмотренном примере <H1> указывает на начало заголовка высшего уровня, а </H1> — на его конец.

Многие метки необязательно использовать парами «начало/конец» (как, например, метка начала параграфа <P> в рассмотренном примере), поскольку начало следующей метки часто закрывает предыдущую. Тем не менее очень неплохо, чтобы каждой открывающей метке всегда соответствовала закрывающая (при соблюдении правильности вложения — см. гл. 6).

Заметим также, что в метках (и атрибутах) языка HTML *не различаются* строчные и прописные буквы, т. е. записи <title>, <TITLE> и даже <TiTlE> полностью эквивалентны.

1.1.3.2 Названия и заголовки

Каждый HTML-документ должен иметь название, которое обычно отображается отдельно от самого документа и служит главным образом для его идентификации (в большинстве графических навигаторов название документа показывается в заголовке окна). Название желательно выбирать со смыслом, потому что обычно именно оно является результатом поиска в Интернете. Название должно располагаться в начале документа и должно быть заключено между метками <title>...</title>.

В языке HTML предусмотрено шесть уровней заголовков, для которых используются шрифты большего размера и/или более жирные, чем шрифт обыч-

[1] Синтаксис разметки, основанный на стандарте языка SGML, будет подробно рассмотрен в гл. 6.

ного текста в основной части документа. Самый первый заголовок в каждом документе должен начинаться с метки <H1>. Синтаксис метки заголовка имеет следующий вид:

```
<Hy>Текст заголовка</Hy>
```

где y — число от 1 до 6, определяющее уровень заголовка.

1.1.3.3 Списки

В языке HTML обеспечивается поддержка нумерованных и ненумерованных списков, а также перечней определений. При перечислении элементов списка не требуется использование разделителей абзацев. Метки элементов списка неявным способом завершают предыдущий элемент списка.

Ненумерованные списки

Ненумерованные или неупорядоченные списки начинаются с открывающей список метки <UL>. Затем вводится метка <LI> со следующим за ней текстом пункта (закрывающая метка может быть опущена), и весь список завершается закрывающей меткой </UL>.

Исходный текст на языке HTML, содержащий два пункта

```
<UL>
  <LI>яблоки
  <LI>бананы
</UL>
```

Генерируемый текст

- яблоки
- бананы

Навигаторы могут изображать ненумерованный или неупорядоченный список различными способами, например для указания элементов списка могут использоваться маркеры абзаца, закрашенные круги или тире. Как объясняется в гл. 7, для управления визуализацией элементов HTML следует использовать таблицы стилей.

Нумерованные списки

Нумерованный или упорядоченный список по своей структуре очень похож на неупорядоченный список, за исключением того, что вместо метки <UL> используется метка <OL>. Маркировка пунктов выполняется с помощью той же самой метки <LI>. При выводе эти пункты будут пронумерованы.

Исходный текст на языке HTML, содержащий три пункта

```
<OL>
  <LI>апельсины
  <LI>персики
  <LI>виноград
</OL>
```

Генерируемый текст

1. апельсины
2. персики
3. виноград

Еще раз отметим, что с помощью таблицы стилей можно определять, какие именно символы будут использоваться для упорядочения пунктов (арабские или римские цифры, буквы и т. д.).

Перечни обозначений

Перечень обозначений состоит из набора *сокращенных обозначений* (маркируемых меткой <DT>) и *обозначаемых данных* (маркируемых меткой <DD>).

Исходный текст на языке HTML, содержащий три пункта

```
<DL>
<DT>URI
<DD>Универсальный
    Идентификатор Ресурса
<DT>URL
<DD>Универсальный
    Локатор Ресурса
<DT>URN</DT>
<DD>Универсальное
    Имя Ресурса</DD>
</DL>
```

Генерируемый текст

URI Универсальный Идентификатор Ресурса

URL Универсальный Локатор Ресурса

URN Универсальное Имя Ресурса

Вводимые вслед за метками <DT> и <DD> данные могут содержать несколько абзацев (разделяемых метками абзаца), списки или другую описательную информацию. Обратите внимание на использование завершающей метки в последнем блоке данных.

1.1.3.4 Гипертекстовые ссылки

Основные возможности языка HTML обусловлены его способностью связывать области текста (или изображения) с «анкером» (местом) в том же самом или каком-либо внешнем документе. Обычно навигатор выделяет эти области, для того чтобы показать, что они являются гипертекстовыми ссылками. Единственной директивой языка HTML, относящейся к гипертексту, является элемент <A>.

Анкеры (объекты привязки) можно использовать для перехода к конкретному разделу документа. Предположим, что вы хотите сделать ссылку из «документа А» на определенный раздел «документа В». Прежде всего, вам необходимо установить в «документе В» *именованный анкер* следующим способом:

```
<A NAME="мое_имя">заданный текст</A>...
```

Теперь, для того чтобы установить связь с предварительно помеченной заданной областью текста, создайте ссылку в «документе А». Вставьте имя требуемого документа вместе с именем самой заданной области следующим образом:

```
Моя ссылка на цель находится <A HREF="documentB.html#мое_имя">здесь </A>.
```

Когда вы щелкнете мышкой на слове «здесь» в «документе А», навигатор загрузит «документ В» и поместит начало абзаца «заданный текст» в верхней части окна.

1.1.3.5 Изображения

Навигаторы могут воспроизводить в документах растровые изображения (обычно в форматах GIF, PNG или JPEG). Вставить такое изображение можно с помощью метки `<IMG>` (или, в более общем случае, метки `<OBJECT>` версии HTML4; см. разд. 6.2.1). Например, изображение формата GIF из файла `mypict.gif` можно вставить (в документ) следующими двумя эквивалентными способами:

```
<IMG SRC="mypict.gif">
```

и

```
<OBJECT DATA="mypict.gif" TYPE="image/gif">
```

Такие изображения являются внутритекстовыми компонентами, поэтому для того, чтобы разместить их в нужном месте экрана, вам придется подробно описать способы выравнивания или же отвести для них отдельный абзац.

В общем случае в полях `SRC` или `DATA` можно указывать URI, и таким образом можно вставить (в документ) любой рисунок, независимо от того, в каком месте Интернета он находится. Существует также возможность указать URI рисунка непосредственно в самой ссылке, предоставляя пользователю возможность самому решать, вызывать рисунок или нет (и экономя, если рисунок имеет большой размер, объем передаваемых данных). Это можно сделать примерно так:

```
Мой рисунок находится <A HREF="http://host/path/mypict.gif">здесь</A>.
```

1.1.3.6 Специальные символы

Некоторые символы ASCII в языке HTML трактуются особым способом и поэтому не могут непосредственно использоваться в тексте. К ним относятся <, > и &. Для того чтобы вставить их в обычный текст, необходимо воспользоваться так называемой ссылкой на объект[1]. Первым символом ссылки является амперсанд, за ним следует имя объекта, а потом точка с запятой как признак конца ссылки. Ссылки на три упомянутых выше символа выглядят следующим образом: `<`, `>` и `&` соответственно.

Существует несколько наборов объектов. Примером может служить набор символов Latin 1 (ISO8859-1), в котором наряду со многими другими определены следующие объекты:

`&ouml;`	создает строчную букву o с умляутом: ö,
`&ntilde;`	создает строчную букву n с тильдой: ñ,
`&Egrave;`	создает прописную букву E со знаком тупого ударения: È.

1.2 LaTeX в среде Web

В предыдущих разделах говорилось о революционных изменениях, произошедших в области всеобщего доступа к электронным документам, распространяемым в Интернете, в особенности посредством Всемирной паутины, и о тех технических принципах, которые лежат в основе достигнутых успехов.

Но в какой степени эта новая ситуация согласуется с проблемами создания и распространения научных документов наивысшего возможного типографского качества – той области, в которой LaTeX до сих пор остается практически безупречен?

Прежде всего отметим, что в HTML и LaTeX'е используются исходные подходы к структурному представлению информации. В табл. 1.2 представлено сравнение разметки в этих двух системах. На основном структурном уровне не так уж трудно выполнить преобразование между LaTeX'ом и HTML. И таким образом усилия, затраченные на создание LaTeX'овских документов, вновь принесут пользу при воспроизведении этих документов в Сети.

Современные навигаторы не всегда воспроизводят сложную информацию (таблицы, математические формулы, рисунки) точным (аккуратным) и приемлемым способом. Как будет показано ниже, в подобных случаях информацию можно представить различными способами, например в виде встроенных растровых рисунков либо внешних файлов в форматах PDF или PostScript. Чтобы возбудить интерес к тому, чего можно достичь простейшими средствами, рассмотрим проект Национального института стандартов и технологий (National Institute of Standards and Technology — NIST) по созданию *Цифровой библиотеки математических функций*. Планируется, что

[1] О механизме действия ссылки на объект в языке SGML более подробно говорится в разд. 6.5.4.3

Таблица 1.2. Сравнение языка HTML с LaTeX'ом.

Элемент	**HTML**	**LaTeX'овский эквивалент**
	Команды секционирования документа	
уровень 1	`<H1>`*текст*`</H1>`	`\chapter{`*текст*`}`
уровень 2	`<H2>`*текст*`</H2>`	`\section{`*текст*`}`
уровень 3	`<H3>`*текст*`</H3>`	`\subsection{`*текст*`}`
уровень 4	`<H4>`*текст*`</H4>`	`\subsubsection{`*text*`}`
уровень 5	`<H5>`*текст*`</H5>`	`\paragraph{`*текст*`}`
уровень 6	`<H6>`*текст*`</H6>`	`\subparagraph{`*текст*`}`
новый абзац	`<P>`	`\par`
	Выделение текста	
выделение	`<EM>`*текст*`</EM>`	`\emph{`*текст*`}`
полужирный шрифт	`<B>`*текст*`</B>`	`\textbf{`*текст*`}` (`\mathbf{`*текст*`}`)
шрифт пишущей машинки	`<TT>`*текст*`</TT>`	`\texttt{`*текст*`}` (`\mathtt{`*текст*`}`)
	Списки	
упорядоченный список	`<OL>...</OL>`	`\begin{enumerate}...\end{enumerate}`
неупорядоченный список	`<UL>...</UL>`	`\begin{itemize}...\end{itemize}`
элемент списка	`<LI>`*текст*	`\item` *текст*
перечень обозначений	`<DL>...</DL>`	`\begin{description}...\end{description}`
сокращение	`<DT>`*элемент*	`\item[`*элемент*`]`
обозначаемые данные	`<DD>`*текст*	*текст*
	Специальные символы	
знаки ударения (например, é)	`&eacute;`	`\'e`
умляуты (например, ü)	`&uuml;`	`\"u`
новая строка	` `	`\newline`

все содержание *Справочника по математическим функциям* (Abramowitz and Stegun (1972)) будет представлено в электронной форме [↪NISTHMF] с интерактивными VRML-графиками[1], ссылками на программы вычисления функций и т. д. Примеры из главы о функциях Айри показаны на рис. 1.1.

[1] *Virtual Reality Modeling Language* — язык моделирования виртуальной реальности — это одна из идей, родившихся во время неформальной дискуссии на конференции Woodstock of the Web в CERN'е в мае 1994 г. В нем предусматривается создание анимаций в трехмерном виртуальном мире, отличающихся полной интерактивностью на уровне зрителя. См. [↪WEB3D] и стандарт ISO/IEC 14772:1997ISO/IEC:14772-1 (1998).

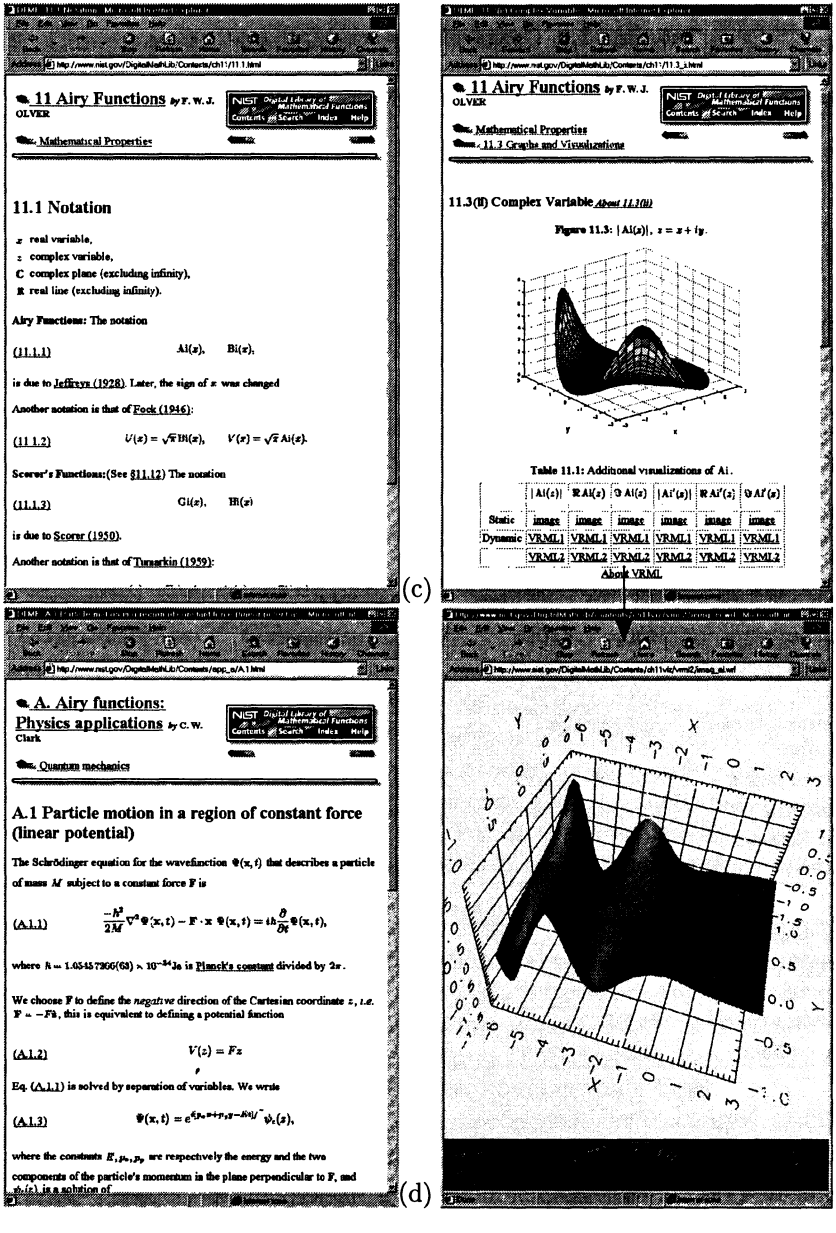


Рис. 1.1. Представление математического справочника в Сети.

(a) обозначения, используемые для записи функций Айри; (b) пример использования функций Айри в физической прикладной задаче из области квантовой механики; (c) трехмерное статическое изображение функции Айри с комплексным аргументом; (d) VRML-визуализация (реализуемая с помощью динамически подключаемой программы-вставки), запускаемая путем активации ссылки, которая обозначена стрелкой на HTML-странице, показанной на (c).

1.2.1 Обзор форматов документов и стратегий

На рис. 1.2 показаны различные форматы, в которых могут быть подготовлены электронные документы для Сети, и способы их взаимного преобразования. В левом верхнем углу показан LaTeX'овский документ. Стандартная и хорошо известная процедура состоит в том, что LaTeX'овский файл компилируется с помощью TeX'а, в результате чего получается DVI-файл, а из него с помощью программы `dvips` создается файл формата PostScript. Последний файл при необходимости может быть отпечатан на принтере. Другой вариант состоит в том, что можно создать файл формата PDF либо непосредственно из LaTeX'а с помощью pdfTeX, либо кружным путем, конвертируя в формат PDF PostScript'овский файл (используя, например, Acrobat Distiller фирмы Adobe) или DVI-файл (например, с помощью `dvipdfm`). Файл формата PDF также при необходимости может быть отпечатан, но он одинаково хорош и для того, чтобы открыть к нему доступ в Интернете, поскольку он может содержать гиперссылки и является редактируемым.

Как отмечалось в начале данного раздела, можно также использовать сходство схем общей разметки в LaTeX'е и HTML (или в более общем случае, как объясняется в гл. 6, какой-либо вариант описания типа документа в языке XML),

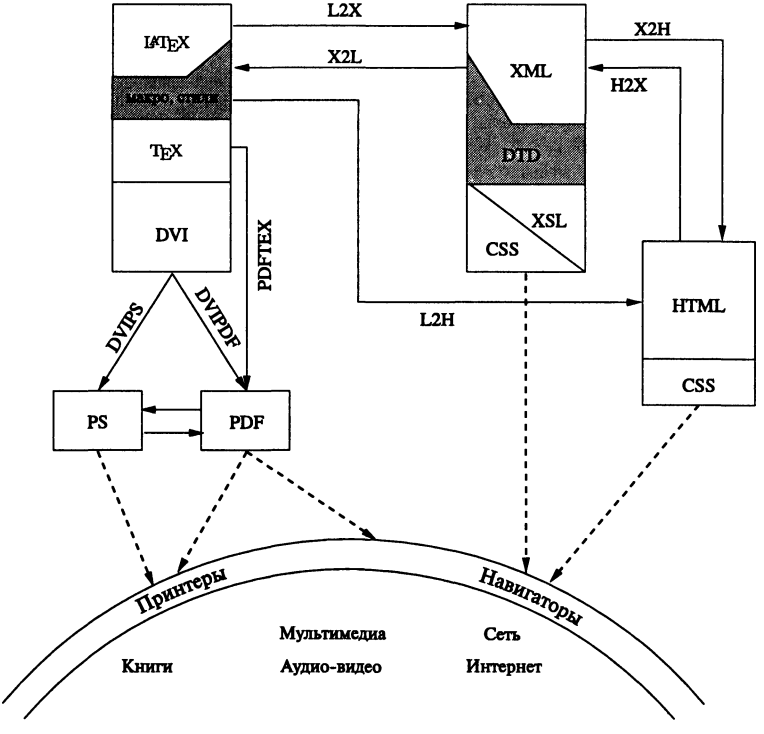


Рис. 1.2. Электронные документы и Сеть.

для того чтобы конвертировать LaTeX в HTML (или некоторый вариант XML). На рис. 1.2 это преобразование обозначено как «L2H» («L2X»). С помощью программы X2H можно также конвертировать XML в HTML. Отметим, что для воспроизведения подобных файлов в Сети можно использовать язык стилей (таких, как CSS, DSSSL или XSL; см. гл. 7).

Можно также пойти в другом направлении и преобразовать файлы XML (HTML) в LaTeX, и таким образом использовать богатые типографские возможности TeX'а для получения высококачественных печатных документов (соответствующее программное обеспечение на рис. 1.2 обозначено «X2L»). Мы предполагаем, что в обозримом будущем XML будет все более и более широко использоваться как формат для хранения электронных документов, а LaTeX превратится в популярную серверную программу, используемую для печати.

В следующих разделах будет дан краткий обзор различных способов воспроизведения одного и того же LaTeX'овского документа, начиная с эталонного — стандартного LaTeX'овского вывода, показанного на «обычном» рис. 1.3. В тексте, взятом из документа по физической тематике, содержится несколько простых внутритекстовых и выключных математических формул, на примере которых можно будет продемонстрировать основные характеристики предложенных решений. Ссылка на текст документа дана в приложении A.1.

3 Vavilov theory

Vavilov[5] derived a more accurate straggling distribution by introducing the kinematic limit on the maximum transferable energy in a single collision, rather than using $E_{\mathbf{max}} = \infty$. Now we can write[2]:

$$f(\epsilon, \delta s) = \frac{1}{\xi}\phi_v\left(\lambda_v, \kappa, \beta^2\right)$$

where

$$\begin{aligned}
\phi_v\left(\lambda_v, \kappa, \beta^2\right) &= \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} \phi(s)\, e^{\lambda s} ds \qquad c \geq 0 \\
\phi(s) &= \exp\left[\kappa(1+\beta^2\gamma)\right] \exp\left[\psi(s)\right], \\
\psi(s) &= s\ln\kappa + (s+\beta^2\kappa)\left[\ln(s/\kappa) + E_1(s/\kappa)\right] - \kappa e^{-s/\kappa},
\end{aligned}$$

and

$$\begin{aligned}
E_1(z) &= \int_z^\infty t^{-1}e^{-t}dt \qquad \text{(the exponential integral)} \\
\lambda_v &= \kappa\left[\frac{\epsilon-\bar{\epsilon}}{\xi} - \gamma' - \beta^2\right]
\end{aligned}$$

The Vavilov parameters are simply related to the Landau parameter by $\lambda_L = \lambda_v/\kappa - \ln\kappa$. It can be shown that as $\kappa \to 0$, the distribution of the variable λ_L approaches that of Landau. For $\kappa \leq 0.01$ the two distributions are already practically identical. Contrary to what many textbooks report, the Vavilov distribution *does not* approximate the Landau distribution for small κ, but rather the distribution of λ_L defined above tends to the distribution of the true λ from the Landau density function. Thus the routine **GVAVIV** samples the variable λ_L rather than λ_v. For $\kappa \geq 10$ the Vavilov distribution tends to a Gaussian distribution (see next section).

Рис. 1.3. Стандартный LaTeX'овский вывод.

1.2.2 Остаемся с DVI

TeX'овское сообщество уже на протяжении некоторого времени имеет дело с гипертекстом (см. Carr et al. (1991)). Благодаря открытой архитектуре команды `\special` можно относительно просто вставлять в DVI-файлы, например, команды переходов по гипертексту. Затем можно расширить возможности программ просмотра в части использования этих команд ссылок и применять их в качестве простых гипертекстовых навигаторов. Пакет `hyperref` (см. разд. 2.3) содержит драйверы для команд `\special`, поддерживаемых разными драйверами (см. приложение B.1). На рис. 1.4, 1.5 и 1.6 показано, как три раз-

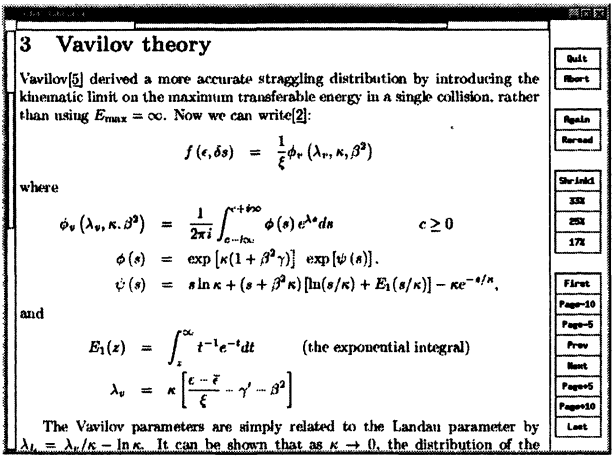


Рис. 1.4. Визуализация гипертекста в формате DVI с помощью `xdvi`.

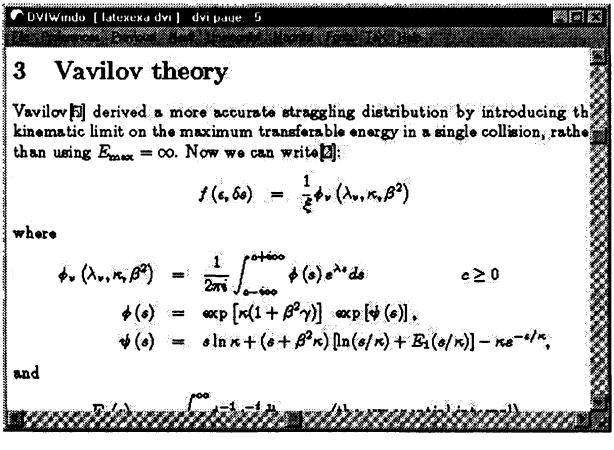
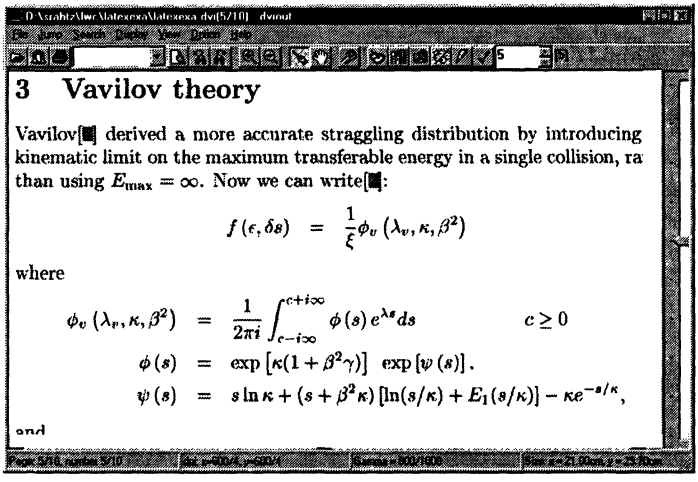


Рис. 1.5. Визуализация гипертекста в формате DVI с помощью `dviwindo`.



3 Vavilov theory

Vavilov[■] derived a more accurate straggling distribution by introducing kinematic limit on the maximum transferable energy in a single collision, ra than using $E_{\max} = \infty$. Now we can write[■]:

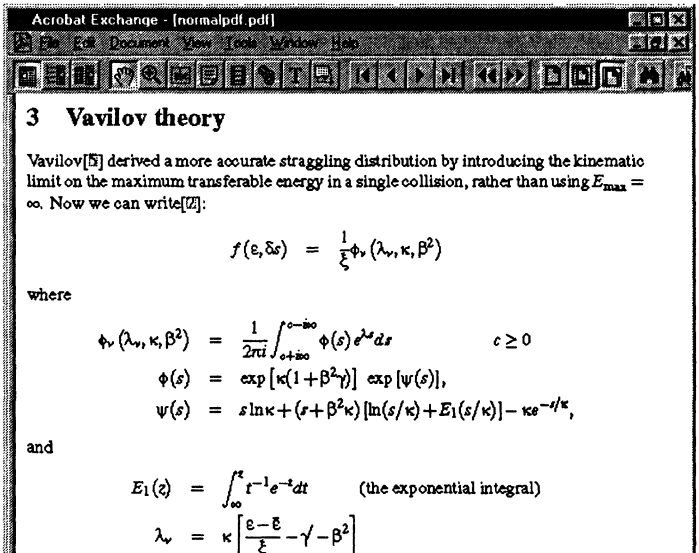
$$f(\epsilon, \delta s) = \frac{1}{\xi}\phi_v(\lambda_v, \kappa, \beta^2)$$

where

$$\phi_v(\lambda_v, \kappa, \beta^2) = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty}\phi(s)\,e^{\lambda s}ds \qquad c \geq 0$$
$$\phi(s) = \exp\left[\kappa(1+\beta^2\gamma)\right]\ \exp\left[\psi(s)\right].$$
$$\psi(s) = s\ln\kappa + (s+\beta^2\kappa)\left[\ln(s/\kappa) + E_1(s/\kappa)\right] - \kappa e^{-s/\kappa},$$

and

Рис. 1.6. Визуализация гипертекста в формате DVI с помощью `dviout`.



3 Vavilov theory

Vavilov[5] derived a more accurate straggling distribution by introducing the kinematic limit on the maximum transferable energy in a single collision, rather than using $E_{\max} = \infty$. Now we can write[7]:

$$f(\epsilon, \delta s) = \frac{1}{\xi}\phi_v(\lambda_v, \kappa, \beta^2)$$

where

$$\phi_v(\lambda_v, \kappa, \beta^2) = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty}\phi(s)\,e^{\lambda s}ds \qquad c \geq 0$$
$$\phi(s) = \exp\left[\kappa(1+\beta^2\gamma)\right]\ \exp\left[\psi(s)\right],$$
$$\psi(s) = s\ln\kappa + (s+\beta^2\kappa)\left[\ln(s/\kappa) + E_1(s/\kappa)\right] - \kappa e^{-s/\kappa},$$

and

$$E_1(z) = \int_z^\infty t^{-1}e^{-t}dt \qquad \text{(the exponential integral)}$$
$$\lambda_v = \kappa\left[\frac{\epsilon - \bar{\epsilon}}{\xi} - \gamma - \beta^2\right]$$

The Vavilov parameters are simply related to the Landau parameter by $\lambda_L = \lambda_v/\kappa - \ln\kappa$. It can be shown that as $\kappa \to 0$, the distribution of the variable λ_L approaches that of Landau. For $\kappa \leq 0.01$ the two distributions are already practically identical. Contrary to what many textbooks report, the Vavilov distribution *does not* approximate the Landau distribution for small κ, but rather the distribution of λ_L defined above tends to the distribution of the true λ from the Landau density function. Thus the routine GVAVIV samples the variable λ_v rather than λ_L. For $\kappa > 10$ the Vavilov distribution tends to a

Рис. 1.7. Простое воспроизведение PDF (с использованием Acrobat).

личных программы просмотра DVI-файлов (`xdvi` [↪XDVI] для системы Linux, `dviwindo` [↪YANDY] и `dviout` [↪DVIOUT] для системы Windows) изображают наш файл и выделяют библиографические ссылки в первом абзаце как ссылки.

Эти приложения поддерживают также загрузку Web-навигатора при ссылке на URL, но интегрировать их в Web-навигатор невозможно. В разделе 1.2.6 мы упомянем одну программу просмотра DVI-файлов, которая *действительно* интегрируется с навигаторами.

1.2.3 PDF для типографского качества

Другой простой подход, который гарантирует типографское качество такого же уровня, как и при использовании DVI-файла, заключается в создании файла формата PDF либо с помощью `dvipdfm` или pdfTeX, либо с использованием Acrobat Distiller, исходным материалом для которого является PostScript'овский файл. В этом случае с помощью PDF-навигатора, такого как Acrobat (см. рис 1.7), можно легко передвигаться по документу и использовать гипертекстовую информацию, которая может присутствовать в исходном LaTeX'овском тексте (LaTeX'овские перекрестные ссылки и библиографические ссылки могут быть автоматически преобразованы в гиперссылки). Web-навигаторы могут быть сконфигурированы таким образом, чтобы при обнаружении PDF-файлов в качестве вспомогательного приложения загружался Acrobat, и тогда ваш документ будет отображаться в окне навигатора. Acrobat может также передавать навигатору для разрешения ссылки на URL, обеспечивая тем самым полную интеграцию. Этот простой подход объясняется в гл. 2, где, кроме того, обсуждаются макеты страницы, оптимальные для чтения с экрана (рис. 1.8).

1.2.4 Преобразование в HTML

Существует способ непосредственной трансляции LaTeX'овского исходного документа в формат HTML: с использованием LaTeX2HTML (см. гл. 3) или TeX4th (см. гл. 4). Поскольку обычно используемые в настоящее время навигаторы не обладают встроенными средствами для правильного изображения математических формул, программы конвертации преобразуют большинство нестандартных символов (греческие буквы, математические знаки и т. д.) в растровые рисунки. На рис. 1.9 показан типичный пример такого подхода.

Информация отображена правильно, но необходимо аккуратно настраивать параметры визуального представления специальных символов (размер шрифта, стиль и т. д.), чтобы их растровые образы в потоке текста и выключные математические формулы соответствовали друг другу. Однако из-за того, что пользователь какого-либо конкретного навигатора может произвольно изменять выбираемые по умолчанию шрифт и размер обычного текста, к сожалению, невозможно гарантировать, что каждый из них увидит «правильную» картинку. Кроме того, поскольку уравнения являются рисунками фиксирован-

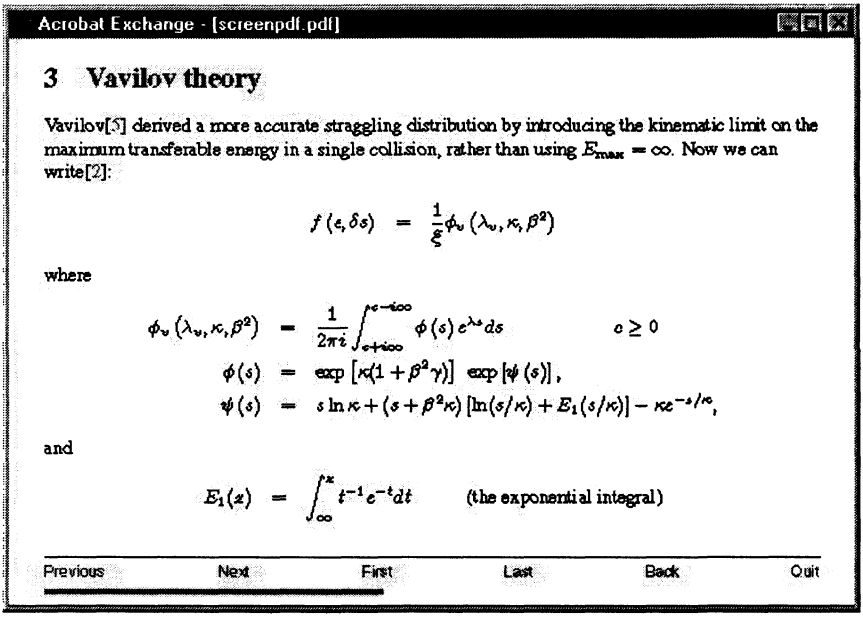


Рис. 1.8. Изображение PDF, предназначенное для просмотра на экране.

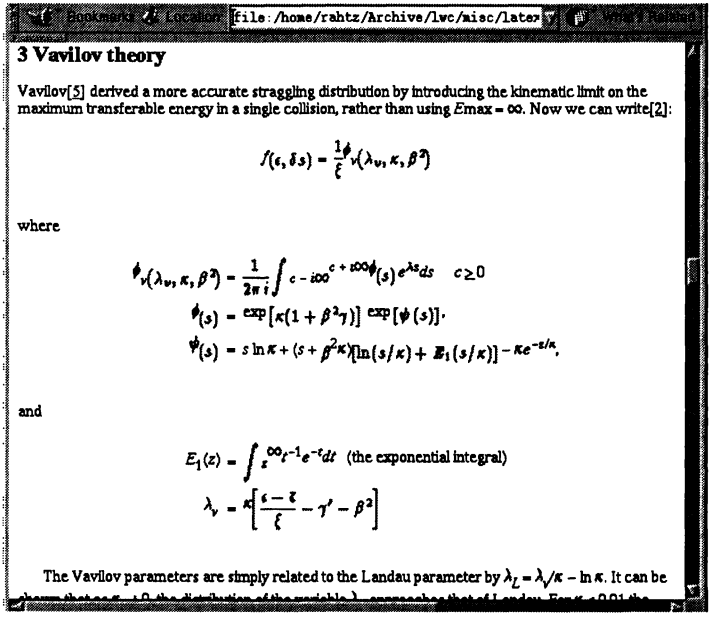


Рис. 1.9. Преобразование в HTML с математическими формулами в виде рисунков.

ного размера, они, конечно же, не могут «переформатироваться» при изменении размеров окна.

Преимущество этих систем заключается в том, что путем тонкой настройки можно получить вполне приемлемое качество при использовании стандартных навигаторов. К тому же перекрестные ссылки, библиографические ссылки и иерархическая структура LaTeX'овского исходного файла могут быть конвертированы в функции гипертекстовых ссылок языка HTML с учетом оптимальной навигации и интеграции в Сеть. При этом использование командных опций и файлов расширений обеспечивает богатые возможности для настройки.

С другой стороны, из-за того, что большинство нелатинских символов конвертируется в битовые образы, для научного документа, содержащего больше, чем несколько страниц, может понадобиться много тысяч растровых изображений формата GIF или PNG (их число будет зависеть, естественно, от сложности встречающихся в документе математических выражений и от того, допускается ли многократное использование образов). Поскольку все эти маленькие файлы должны быть загружены вместе с исходным HTML-текстом страницы, то время, необходимое для изображения страницы навигатором, может оказаться весьма большим. Кроме того, эти рисунки занимают много места на диске, а при изменении исходного LaTeX'овского кода часто бывает необходимо перегенерировать весь набор образов. И наконец, инсталляция транслятора LaTeX-HTML иногда вызывает затруднения.

Более простой альтернативный подход реализован в программе Яна Хатчинсона TtH [↪TTH], которая доступна также и в коммерческой версии, с бóльшими возможностями и предоставлением технической поддержки ([↪TEX2HTML]). Благодаря наличию в большинстве навигаторов шрифта Symbol, при преобразовании LaTeX'овского исходного кода в формат HTML «специальные» символы изображаются с помощью встроенных шрифтов. На рис. 1.10 показано, как выглядит наш демонстрационный документ при использовании этого способа. Обратите внимание, что знак интеграла состоит из меньших линейных сегментов.

Большим преимуществом программы TtH является то, что она работает очень быстро и поэтому может быть использована для оперативного получения HTML-документов. Этот способ вполне пригоден для документов, содержащих достаточно простые математические формулы (т. е. содержащие только те символы, которые присутствуют в шрифте Symbol). А поскольку необходим только LaTeX'овский исходный код, стоимость поддержки документов минимальна. Правильность изображения документов не нарушается, даже когда пользователь изменяет шрифт или размеры окна.

Недостаток рассматриваемого подхода заключается в том, что качество не настолько хорошо, как при воспроизведении формул в виде рисунков. Кроме всего прочего, невозможно уследить за появлением в первоисточнике символов, которые не отображаются доступными в навигаторе шрифтами. Естественно, что этот подход окажется бесполезным, когда все навигаторы будут поддерживать стандарт Unicode, и благодаря этому можно будет использовать все дополнительные математические символы, обсуждаемые в гл. 8. Тем не менее

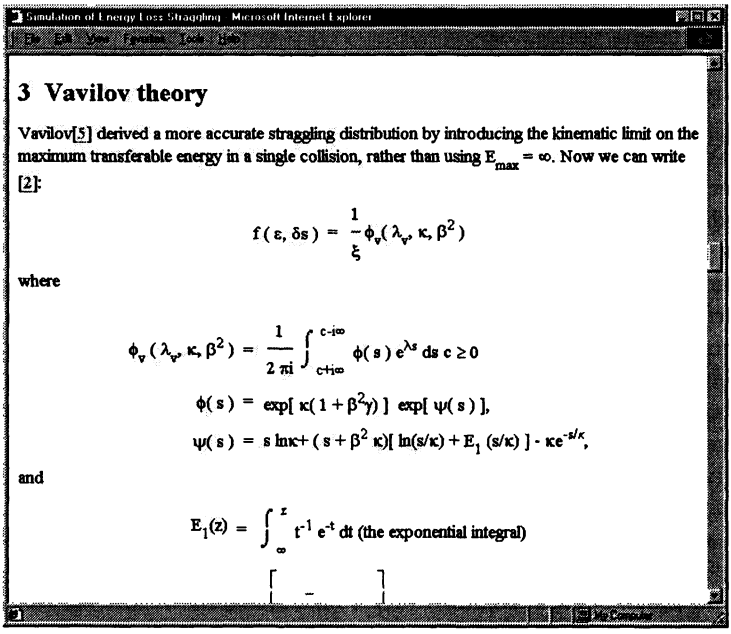


Рис. 1.10. Преобразование в HTML с использованием шрифта Symbol для воспроизведения математики.

остается еще проблема точного расположения этих символов в математической формуле, которая может быть решена только тогда, когда навигаторы будут располагать типографскими средствами, обладающими определенным уровнем «математических знаний».

Удобный способ конвертирования LaTeX'а в HTML предоставляет модифицированный TeX-процессор, который входит в состав коммерческой версии VTEX фирмы MicroPress [↪TEXPIDER]. Поскольку при этом HTML-документ создается непосредственно, то никаких проблем с TeX'овскими макро не возникает, а вот математические формулы по-прежнему изображаются в виде растровых рисунков.

1.2.5 Язык Java и динамически подключаемые к навигаторам модули

Динамически подключаемый к навигатору модуль `techexplorer` понимает широкое подмножество языка LaTeX и отображает исходный LaTeX'овский документ *непосредственно* в навигаторе. Результат показан на рис. 1.11, а соответствующее программное обеспечение обсуждается в гл. 5. Хотя этот способ действительно работает быстро, у него есть серьезный недостаток, который

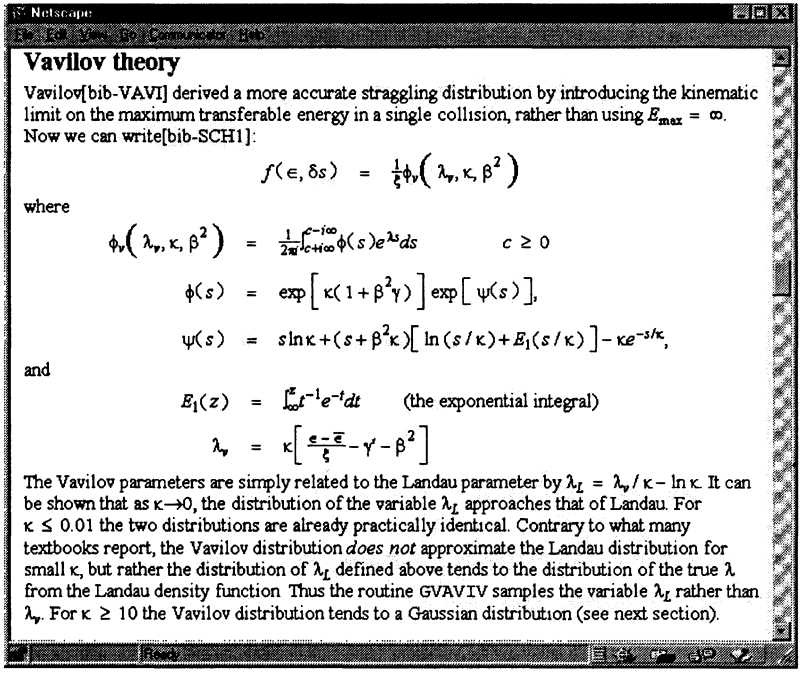


Рис. 1.11. Воспроизведение с помощью динамически подключаемого к навигатору модуля `techexplorer`.

заключается в том, что `techexplorer` необходимо загружать и устанавливать на все используемые навигаторы. Более того, для каждой компьютерной платформы необходима специальная версия.

И наконец, существует возможность трансляции математического содержания на язык MathML, но в этом случае необходимо иметь доступ к некоторому динамически подключаемому к навигатору модулю или какой-либо другой процедуре, которая может анализировать и отображать MathML (см. рис. 1.12). Примером может служить основанный на Java-апплетах WebEQ [↪WEBEQ], который дает высококачественное изображение. При этом его основное преимущество состоит в том, что он может приспосабливаться к изменению размеров шрифта и окна навигатора. Однако это решение может быть весьма медленным. Во-первых, необходимо дождаться завершения начальной загрузки Java-апплетов, используемых для отображения кода на языке MathML, а это может занять существенно больше, чем полминуты. Затем еще несколько секунд уходит на отображение всех уравнений, за исключением самых тривиальных. В конечном счете, приходится достаточно долго ждать, пока страница будет готова к просмотру. Альтернативный подход состоит в том, что Java-программу можно установить на вашей собственной машине, так что она всегда будет доступной, но даже в этом случае придется ждать, пока

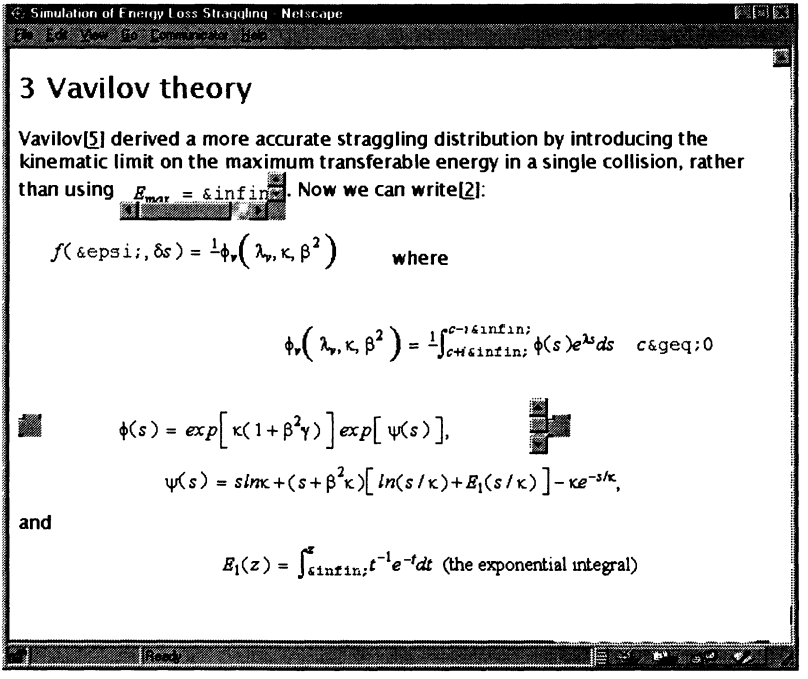


Рис. 1.12. Преобразование в HTML со встроенным MathML. Визуализация с помощью динамически подключаемого модуля.

будут обработаны уравнения. Еще одна проблема, связанная с WebEQ, заключается в том, что он обрабатывает не сам LaTeX, а один из его вариантов — WebTeX.

1.2.6 Другие связанные с LaTeX'ом подходы к Сети

Основные подходы к представлению TeX'овских документов в Сети рассмотрены в предыдущих разделах. Однако было бы полезно ознакомиться еще с несколькими вариантами, хотя их применение в настоящее время довольно ограничено. Некоторые из них основаны на стандарте использования `\special` из проекта HyperTeX, который подробно описан в приложении В.1.

1. Hyperlatex [↪HYPERLTX] Отфрида Чеонга — это конвертер с LaTeX'оподобного языка в HTML. В нем не поддерживается отображение математических выражений, но есть собственные команды, которые расширяют LaTeX, делая его развитым языком для создания продвинутых HTML-документов.
2. Гарт Дикки написал на языке Java программу просмотра DVI-файлов `idvi`, в которой поддерживаются HyperTeX'овские команды `\special`. Она представляет собой динамически подключаемый к Web-навигатору модуль, ко-

торый отображает обычный DVI-файл в окне навигатора и использует Сеть для доступа к необходимым ресурсам, например шрифтам. К сожалению, несмотря на прекрасную идею, работа над этой программой, кажется, прекращена, и ее производительность невелика.

3. Программа `nDVI` [↪NDVI] Каспера Питера — это динамически подключаемый к навигатору модуль, который непосредственно отображает DVI-версии HyperTEX'овского документа.
4. `ltoh` — это конвертер LATEX—HTML, написанный Расселом Квоунгом на языке Perl, но без какой-либо поддержки математики.
5. Хотя работа Т. В. Рамана над системой Aster [↪ASTER] и не имеет непосредственного отношения к Сети, с ней следовало бы познакомиться каждому, кто интересуется отображением математики не только на бумаге. Это система для визуализации документов, содержащих размеченные в TEX'е математические выражения, использующая синтезатор речи. Будучи разработанной главным образом для людей с ослабленным зрением, эта система тем не менее является прекрасным примером нетрадиционных способов анализа и визуализации хорошо размеченных математических выражений.

1.3 Существует ли оптимальный подход?

При решении вопроса о том, какой из многих рассмотренных выше инструментов наиболее предпочтителен в какой-либо конкретной ситуации, необходимо принимать во внимание следующие обстоятельства, касающиеся вашего документа:

- Является ли он чисто текстовым и простым LATEX'овским документом? Если да, то конвертируйте его непосредственно в HTML (например, с помощью `TtH`) или начинайте готовиться к работе с XML.
- Содержит ли он большое количество низкоуровневой математики с самодельным макро, позволяющим выстроить свою собственную нотацию? Тогда вам, вероятно, придется применить TEX4th или LATEX2HTML (в последнем случае будьте готовы к тому, что придется написать несколько программ на языке Perl, реализующих ваши расширения), а если вы предпочитаете коммерческое решение, используйте VTEX фирмы MicroPress.
- Содержится ли в нем преимущественно «обычная» математика? В этом случае у вас в перспективе — трансляция в MathML и один из динамически подключаемых к навигатору модулей.
- Много ли нелатинских символов используется в документе? Возможно, вы захотите воспользоваться конвертером LATEX-XML и преобразовать нелатинские символы в Unicode.
- Имеет ли документ сложную компоновку (возможно, таблицы) или оформление, и существенно ли это для читателя? Используйте PDF.

- Является ли документ в достаточной степени самодостаточным? Является ли взаимодействие с другими материалами Сети незначительным? Целесообразно рассмотреть подход, основанный на DVI.

Реальный документ, вероятно, можно будет отнести более чем к одной из перечисленных выше категорий, поэтому выбор оптимальной стратегии может оказаться непростым. В последующих главах подробно описываются различные программы, имеющиеся в вашем распоряжении. Изучение этих материалов поможет вам сделать выбор и в каждом конкретном случае решить, какой подход является наилучшим.

Рассматривая вопросы сетевой публикации, нельзя забывать о людях, которые либо полностью слепы, либо страдают нарушениями зрения наподобие дальтонизма. Деятельность «Инициативы по доступу к Сети» [↪WAI] направлена на анализ различных способов представления Web-документов, и она может оказать влияние на способ представления вашей работы.

При создании новых LaTeX'овских документов автор сталкивается с очень серьезной проблемой: как разметить документ, чтобы извлечь пользу из таких общих удобств Сети, как ссылки, цвет, формы, всплывающие элементы и Java-апплеты. В каждом пакете для этих расширений вводится новый синтаксис. LaTeX2HTML, `techexplorer` или `hyperref` — какой бы из этих пакетов ни использовать — потребуется своя разметка документа, и, к сожалению, мир LaTeX'а до сих пор не в состоянии остановить развитие этой опасной ситуации. Мы настоятельно рекомендуем принять следующую стратегию, пользуясь LaTeX'ом:

- При любой возможности использовать «родные» LaTeX'овские синтаксические структуры наподобие `\ref` и `\cite`, которые допускают автоматическую трансляцию.
- Пакет `hyperref` обладает множеством необходимых функциональных возможностей и включает в себя встроенные драйверы для широкого множества основанных на TeX'е гипертекстовых систем. Имеет смысл укрепить свои позиции за счет использования синтаксиса этого пакета и получения доступа к различным системам.
- Если вы хотите получить мобильный DVI-файл, то придерживайтесь HyperTeX'овских соглашений, как это по умолчанию делает пакет `hyperref`, и обратитесь за поддержкой к авторам ваших любимых DVI-драйверов.
- Если вам необходим синтаксис специфического пакета, то попробуйте изолировать его в LaTeX'овских макро (если система их поддерживает), чтобы их легко можно было найти при необходимости перейти на другую систему.

Не забывайте, что LaTeX и Сеть все еще приноравливаются друг к другу, и поэтому сейчас, наверное, не следует слишком привыкать к какой-либо одной системе.

1.4 Заключение

В то время как появление и легкодоступность персональных компьютеров чрезвычайно сильно снизили стоимость создания электронных документов, создание Сети сделало распространение этих документов по всему миру намного более дешевым, простым и быстрым. Эти два события, взятые вместе, существенно изменили факторы, влияющие на создание, поддержку и распространение электронных документов. Кроме того, благодаря развитию семейства стандартов XML и мобильности не зависящего от платформы языка Java, сейчас стали возможны унифицированный подход к огромному объему информации, хранящейся в базах данных, и управление их представлением в различных настраиваемых формах.

Мы убеждены в том, что LaTeX'у, как инструменту для ввода текста и отображения вывода, принадлежит важная роль в этом новом и интегрированном киберпространстве, особенно в области математики. LaTeX может быть должным образом дополнен представлением на языке MathML, обладающем более широкими семантическими возможностями. Однако наибольшее влияние LaTeX сохранит в области верстки, а TeX при этом превратится в важный промежуточный формат для создания выходных документов в формате PDF, допускающих высококачественную печать. Изучение различных средств, описанных в этой книге, гарантирует не только то, что вы будете готовы к революции, производимой языком XML, но и то, что вы сможете внести позитивный вклад в богатство и возрастающее процветание научной гиперкультуры, если, используя один из предложенных способов, конвертируете свои LaTeX'овские документы в формат PDF, HTML или XML и сделаете их доступными по Сети. Вам не нужно выбирать между использованием LaTeX'а или другого способа разметки в Сети; вы можете использовать наиболее подходящий в конкретной ситуации и извлекать пользу из достоинств обоих способов.

Глава 2

PDF: мобильный формат документов

Для ряда применений методы, используемые для распространения информации во Всемирной паутине, нельзя признать удовлетворительными. Причина этого состоит в том, что формирование и воспроизведение «страницы» пересылаемого документа возлагается на программное обеспечение читателя, а не автора. Даже если используются «чистый» HTML и каскадные стили (Cascading Style Sheets), автор не имеет представления о том, где произойдет разбиение строки; нет, разумеется, также и концепции «разбиения на страницы». Графический материал часто передается в виде растровых изображений с низкой разрешающей способностью и некорректной цветопередачей; вывод таблиц может быть существенно разным для передающей и принимающей стороны. Автор не может быть даже уверен в том, какой шрифт увидит читатель и, в частности, не возникнет ли ситуация, когда не окажется каких-либо символов, например при выводе математических выражений. Наконец, и это пожалуй наиболее существенное обстоятельство, существующее поколение Web-навигаторов не слишком хорошо справляется с задачей формирования страниц типографского качества; по этой причине нажатие кнопки Print в навигаторе отнюдь не приводит к получению высококачественного результата.

Кого может обеспокоить такого рода ситуация? С одной стороны, это может оказаться существенным для юристов, которые могли бы заявить, что *точное* соответствие получаемого документа его печатному варианту жизненно важно. А с другой стороны, дело может быть просто в том, что некий автор потратил уйму времени, чтобы получить привлекательную по внешнему виду страницу, и он хочет, разумеется, чтобы и читатель увидел ее такой же. Между этими крайностями — такие приложения, в которых внешний вид выводимого HTML-документа просто «не очень хорош». Это, в частности, относится к документам, содержащим очень сложные диаграммы, таблицы, материалы математического характера.

Эта глава, состоящая из трех частей, описывает возможное решение упомянутой выше проблемы — формат PDF (Portable Document Format), предназначенный для получения мобильных документов. В первой части данной главы

дается общее представление о PDF, а также о том, как документы в этом формате можно создавать с помощью TEX'овских средств. Во второй части описывается специальный LATEX'овский пакет (`hyperref`), позволяющий получать улучшенные документы в формате PDF из исходного LATEX'овского текста, используя набор драйверов и высокоуровневый интерфейс к гипертекстовым командам.

Завершающая часть данной главы содержит достаточно подробное описание специальной версии TEX'а, порождающего выходной текст непосредственно в формате PDF. Для многих из читателей, если не для большинства из них, нет никакой необходимости понимать новые PDF-примитивы, добавленные в этой версии TEX'а, поскольку пакеты типа `hyperref` обеспечивают интерфейс к ним, подобный LATEX'овскому. Однако pdfTEX предоставляет огромные возможности по формированию высококачественных интерактивных электронных документов, и опытные TEX'овские программисты захотят, по-видимому, понять, каким образом этими возможностями воспользоваться.

2.1 Что такое PDF?

PDF является потомком языка PostScript, созданного фирмой Adobe Systems. Хотя примерно в течение десятилетия PostScript был лучшим языком описания страниц для издательских приложений, его возраст и сложность являются недостатками. Некоторые важнейшие средства (такие, как метаданные, довольно грубо реализованные с использованием соглашений о комментариях, а также полноценная допечатная обработка цвета) были добавлены в PostScript только лишь в поздних его редакциях; существует, кроме того, множество мелких различий в реализациях этого языка. Более важно, однако, то обстоятельство, что PostScript представляет собой полномасштабный язык программирования. Весьма трудно писать интерпретаторы для него, достаточно быстрые, чтобы формировать изображение на экране монитора за очень ограниченное время, трудно также гарантировать, что каждая страница текста может быть показана независимо друг от друга. В версии Display PostScript часть этих проблем была решена, добавлены средства обеспечения интерактивности. В то же время при разработке фирмой Adobe программы Illustrator было создано функциональное подмножество языка PostScript в качестве внутреннего языка представления данных. Судя по всему, именно основываясь на опыте, полученном в ходе выполнения этих проектов, фирма Adobe разработала язык описания страниц второго поколения, названный Portable Document Format (PDF) — мобильный формат документов. Различия между языками PDF и PostScript очень значительны:

- В PDF нет встроенного языка программирования (в версии 3.5 программы Acrobat Forms фирмы Adobe добавлена поддержка языка JavaScript, осуществляемая вызовом динамически подключаемого модуля, однако это

расширение относится скорее к реализации программы-визуализатора, чем к собственно языку PDF).

- Предложенный формат гарантирует независимость страниц, четко разделяя ресурсы и объекты страниц.
- В язык введены гипертекстовые средства и средства обеспечения безопасности, что дает возможность формирования достаточно изощренного интерфейса.
- Средства управления шрифтами позволяют не включать требуемый шрифт непосредственно в файл, а давать лишь сопровождающую информацию о них, достаточную для настройки программ отображения PDF-файлов, подобных программе Adobe Acrobat. Программа Acrobat осуществляет такую настройку, используя шрифтовую технологию Multiple Master Fonts.
- Значительные усилия при создании PDF были приложены к тому, чтобы снизить средствами сжатия, насколько возможно, размер PDF-файлов.

В неприкосновенности осталась значительная часть преимуществ PostScript-подхода: PDF гарантирует точность отображения страницы, вплоть до мельчайшего знака или пробела на ней, причем мобильность обеспечивает сохранение этой точности при переносе с одной компьютерной платформы на другую. Формат PDF все шире используется в профессиональном издательском мире в качестве замены языка PostScript; в настоящее время существует уже третья редакция PDF (это версия 1.2 данного формата). После 1996 г. появилась возможность отображать PDF-документы внутри Web-навигаторов вместе с HTML посредством технологии динамического вызова программных модулей (plug-in).

Надо четко представлять себе разницу между PDF и программой Acrobat. Формат PDF представляет собой один из видов открытых языков, спецификация для которого опубликована (хотя фирма Adobe и сохраняет за собой контроль за данным форматом); Acrobat — это семейство программ коммерческого характера, созданных фирмой Adobe, предназначенных для формирования и отображения текстов в формате PDF, а также для выполнения различных манипуляций с этими текстами. Основными элементами данного семейства программ являются:

- Reader — свободно распространяемая программа для просмотра PDF-файлов и их печати;
- Exchange — программа, обладающая всеми функциональными возможностями программы Reader и дополнительно позволяющая вносить изменения в PDF-файл;
- Distiller — программа для преобразования файлов из формата PostScript в формат PDF;

- PDFWriter — программа, включающая драйверы принтеров для платформ Windows и Macintosh для вывода результатов («печати») любого приложения в файл сразу в PDF-формате;
- Catalog — программа подготовки указателей для наборов, состоящих из документов в формате PDF;
- Capture — программа формирования PDF-файлов из растровых изображений путем оптического распознавания символов (OCR — Optical Character Recognition).

Хотя программа Acrobat Reader является свободно распространяемой и доступна для многих (но не для всех) платформ, она не является единственной в своем роде, существуют еще и программы Ghostscript [↪GSHOME], позволяющая также получать PDF-файл из PostScript-файла, и `Xpdf` [↪XPDF].

Как формат PDF, так и семейство программ Acrobat отлично документированы[1]. Кроме текста спецификации для PDF [↪PDFSPEC] и документации по программам семейства Acrobat, есть еще целый ряд книг, распространяемых на коммерческой основе. Читатели, привыкшие к стилю технической документации, обнаружат, что книга *Web Publishing with Acrobat/PDF* (Merz, 1998) представляет собой неоценимый источник сведений обо всех аспектах PDF, которые не нашли отражения в нашей книге. К числу тем, знакомство с которыми необходимо для серьезных пользователей PDF и Сети, относятся:

- оптимизация PDF-файлов для их постраничной загрузки;
- встраивание PDF в страницы HTML;
- программирование на JavaScript и VBScript в сочетании с PDF;
- обработка данных-форм (forms data) на Web-серверах;
- динамическое создание PDF-файлов.

2.2 Формирование PDF-файлов из TEX'а

Пользователи TEX'а открыли для себя PDF уже довольно давно, так что к настоящему времени достигнут высокий уровень понимания проблемы формирования PDF-текстов из TEX'а. Ниже будут рассмотрены три группы вопросов, относящихся к вышеупомянутой проблеме: как создавать документ в формате PDF, как обеспечить высококачественное отображение шрифтов и как добавить в документ дополнительную информацию наподобие гипертекстовых ссылок и «кнопок» для навигации по тексту документа. Эти группы вопросов рассматриваются в следующих трех разделах.

[1] Заметным исключением является лишь описание аппарата форм (Forms), остающееся пока запутанным.

2.2.1 Формирование PDF-документов и выполнение манипуляций с ними

Документы в формате PDF могут быть получены одним из четырех способов:

1. Преобразование имеющихся PostScript-файлов в PDF-файлы с помощью «программы-дистиллятора». Программа Adobe Acrobat Distiller является наиболее мощной и совершенной из программ такого рода. С подобной задачей преобразования, однако, хорошо справляются также программы `Ghostscript` и NikNak [↪NIKNAK]. Данный подход означает, что можно получить PDF-файл, основываясь на результатах работы некоторой другой программы, позволяющей выдавать документ в PostScript-формате (и таких приложений сейчас довольно много).
2. Использование принтерной программы-драйвера Adobe PDFWriter для платформ Windows и Macintosh, позволяющей переводить в PDF-формат выдачу любого нормального приложения (прикладной программы), например текстового процессора или электронной таблицы.
3. Использование программы Acrobat Capture фирмы Adobe, с помощью которой можно сканировать уже отпечатанные страницы документа, пропускать их через систему OCR и сохранять в формате PDF. Интересной особенностью этой программы является то, что слова, которые не удалось распознать, сохраняются в виде растровых фрагментов. Это ведет к тому, что повышается надежность воспроизведения внешнего вида документа, но он может представлять собой в итоге смесь из реального текста и небольших графических фрагментов.
4. Использование таких прикладных программ, которые сами непосредственно порождают документ в формате PDF. В TEX'овском мире к подобного рода программам относятся pdfTEX (см. разд. 2.4 на с. 91), VTEX фирмы MicroPress [↪MICROPRESS], а также DVI-драйвер `dvipdfm` Марка Уикса.

Первый из описанных методов является наиболее общим и превосходит все остальные, поскольку сейчас почти все программы форматирования текста могут выдавать хорошие PostScript-тексты. Второй подход из перечисленных выше не может быть рекомендован для серьезной работы, поскольку отсутствуют возможность автоматического добавления гипертекстовых ссылок, а также средства управления такими характеристиками документа, как сжатие и растеризация оригинала. Он, однако, полезен для быстрого получения материалов чернового характера. Третий метод весьма специфичен и на самом деле он подходит только для крупномасштабных проектов типа преобразования в электронный вид юридических документов, когда имеется обученный персонал, умеющий контролировать качество работы. Последний из перечисленных методов будет, естественно, наилучшим, однако примеров его реализации совсем немного. Причину этого найти нетрудно — Acrobat Distiller представляет собой отличную программу, очень гибкую в применении, и потому нет осо-

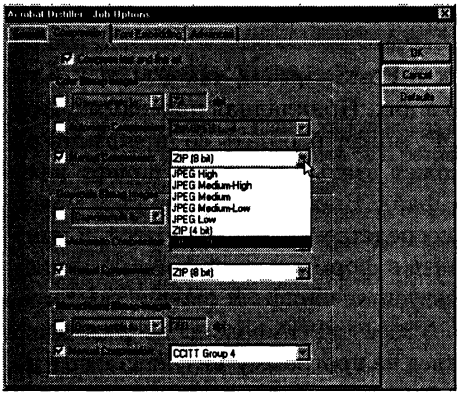


Рис. 2.1. Графические параметры задания для программы Distiller.

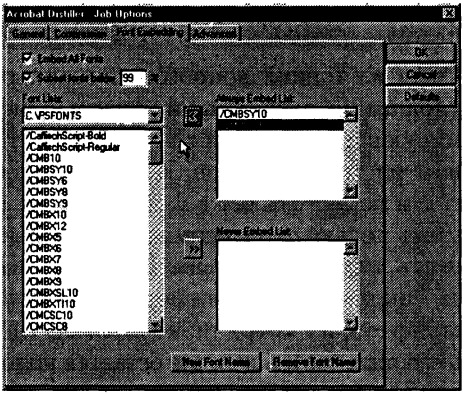


Рис. 2.2. Шрифтовые параметры задания для программы Distiller.

бых стимулов разрабатывать еще какие-то постпроцессоры для прикладных программ.

Пользователи программы Acrobat Distiller должны тщательно проверять настройки приложения. В частности, важное значение имеют две панели «Job Options» (опции задания)[1]: одна из них устанавливает значения параметров сжатия и работы с графикой (рис. 2.1), вторая определяет, какие шрифты должны встраиваться (рис. 2.2). О шрифтах мы поговорим подробнее в следующем разделе, но надо хорошо понимать, что Acrobat Distiller модифицирует используемые растровые рисунки. У вас должна быть уверенность в том, что программа будет делать именно то, чего вы от нее ожидаете. По умолчанию она нацелена на получение возможно более компактных файлов, а это совсем не обязательно может совпадать с вашими потребностями.

Работа с PDF-файлами после их создания — тема, выходящая за рамки данной главы. Достаточно сказать здесь, что существует обширный и богатый по составу выбор динамически подключаемых программных модулей для семейства программ Acrobat. Они нацелены на выполнение всевозможных видов действий, включая функции поддержки допечатных процессов (prepress functions), обеспечения повышенного уровня защиты, внесения разметочных комментариев (marking-up comments). Полезный перечень такого рода средств содержится в Merz (1998), с. 53–55, существует также исчерпывающий каталог их, поддерживаемый фирмой Adobe [↪ADOBE], а также независимыми разработчиками (см., например, [↪PDFZONE]).

[1] Здесь мы описываем и иллюстрируем применение версии Acrobat Distiller для Windows, версия для Macintosh очень похожа на нее. Пользователи варианта данной программы для ОС UNIX должны обратиться к соответствующей документации, чтобы понять, как правильно написать конфигурационные файлы или воспользоваться параметрами командной строки.

2.2.2 Настройка шрифтов

Один из тонких вопросов, связанных как с PostScript'ом, так и с PDF — это управление различными типами шрифтов в них. Прикладная программа, формирующая выходной текст в формате PDF, может работать со шрифтами одним из трех способов. Во-первых, она может взять требуемый шрифт целиком и поместить его в формируемый PDF-файл. Во-вторых, из шрифта можно взять только те его символы, которые непосредственно используются, и поместить в файл только их. В третьем варианте в формируемый файл помещается лишь некоторая информация об используемых шрифтах (такая, как имена шрифтов, их метрики, кодировка, тип — без засечек, символьный — и сведения о его дизайне), а дальше остается положиться на программу-визуализатор в надежде, что она воспроизведет что-либо подходящее. Последняя из перечисленных стратегий лучше всего соответствует документам, распространяемым через Сеть, поскольку при таком подходе получаем наименьший размер файлов. Программа отображения опять же может действовать согласно одному из нескольких вариантов. Она может попытаться отыскать шрифты, перечисленные в отображаемом тексте, в своей локальной среде. Второй вариант действий — заменить перечисленные шрифты какими-то другими настолько аккуратно, насколько это возможно. И наконец, третий вариант состоит в использовании шрифтов согласно технологии Multiple Master Fonts, чтобы воспроизвести вид шрифтов, употреблявшихся в исходном документе.

К несчастью для пользователей TEX'а, их системы традиционно были связаны с растровыми шрифтами фиксированных размеров (а именно, с .pk-шрифтами), поскольку TEX задумывался и создавался еще в то время, когда масштабируемые шрифты не стали повседневной реальностью. Такого рода TEX'овские шрифты встраиваются в PostScript'овский выход как «шрифты типа 3» (Type 3 fonts) (для получения более полной информации о типах шрифтов, используемых в PostScript'е, см. разд. 10.3 в Goossens et al. (1997)). Программа Acrobat Distiller не в состоянии сколько-нибудь аккуратно работать со шрифтами этого типа, поскольку ей недоступны дескрипторы шрифтов. Она оставляет эти шрифты встроенными в PDF-файл, причем программы Acrobat воспроизводят их очень плохо. Печататься нормально они будут лишь в том случае, когда первоначальное разрешение было установлено достаточно высоким.

Насколько различаются результаты, получаемые в каждом из этих трех случаев, можно видеть из рис. 2.3, 2.4 и 2.5. В первых двух рисунках использован шрифт Monotype Baskerville, но в случае, показанном на рис. 2.4, данный шрифт не встроен в PDF-файл, здесь применена технология Multiple Master, с помощью которой программа Acrobat попыталась построить шрифт, возможно более похожий на исходный. На рис. 2.5 применен TEX'овский шрифт Computer Modern в виде растровых PK-шрифтов: результат практически нечитаемый.

Решение проблем, порождаемых использованием растровых шрифтов, жизненно важно для TEX'а. Если вы намерены получить PDF-файл, обеспечивающий хорошее качество печати, надо отыскать варианты типа Type 1

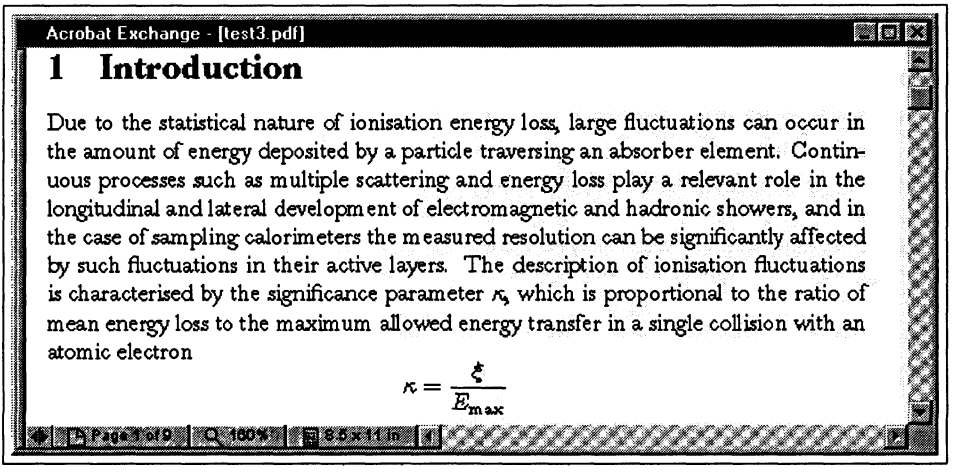


Рис. 2.3. Отображение PDF-файла с использованием встроенных шрифтов.

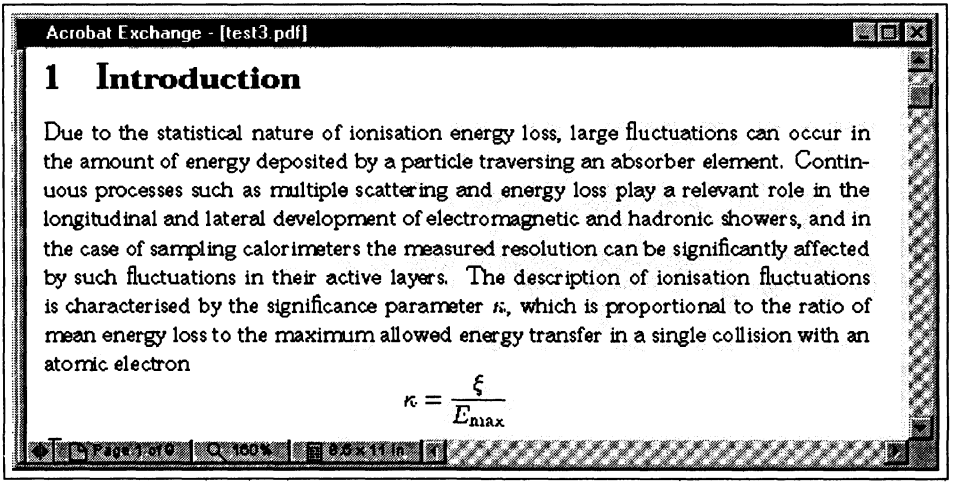


Рис. 2.4. Отображение PDF-файла с использованием подстановки шрифтов согласно технологии Multiple Master.

(или TrueType, хотя этот формат большинством DVI-драйверов поддерживается менее удовлетворительно) для всех шрифтов, которые предполагается задействовать, а затем проинформировать драйвер о том, что он должен воспользоваться именно этими шрифтами. Способ, которым выполняются эти действия, зависит от используемого DVI-драйвера. Например, драйверы `dviwindo` и `dvipsone` фирмы Y&Y поддерживают (за исключением *некоторых крайних случаев*) *только* масштабируемые шрифты Type 1, установленные под Adobe Type Manager. Для широко распространенного драйвера `dvips` (см. Goossens

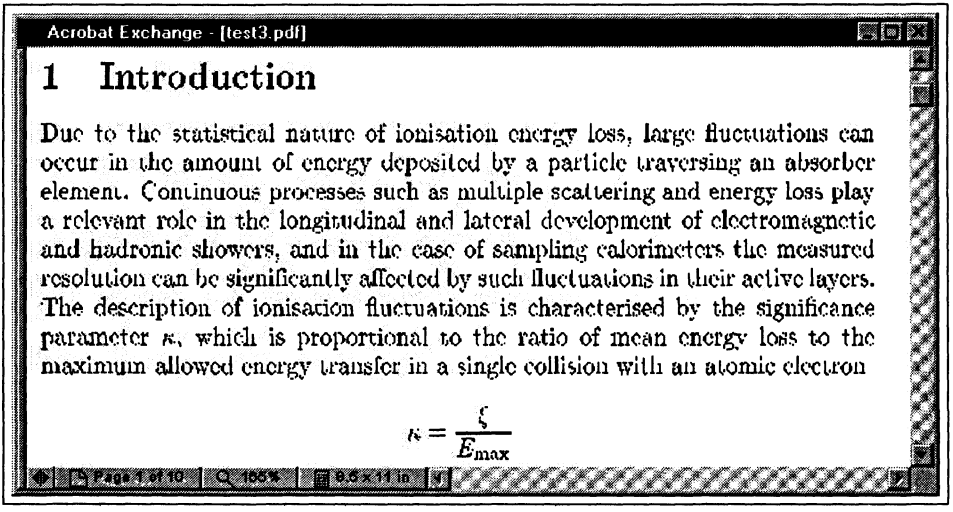


Рис. 2.5. Отображение PDF-файла с использованием растровых шрифтов.

et al. (1997), разд. 11 для более полной информации) надо обеспечить, чтобы требуемые шрифты были перечислены в файле `psfonts.map` или же в `map`-файле, имя которого указано в конфигурационном файле. Например, чтобы добиться корректной обработки шрифтов семейства Computer Modern, в дистрибутив TEX Live [↪TEXLIVE] включен конфигурационный файл `config.cms` программы `dvips`, загружающий файл `cms.map`, который содержит следующие строки:

```
cmb10 CMB10 <cmb10.pfb
cmbsy10 CMBSY10 <cmbsy10.pfb
cmbx10 CMBX10 <cmbx10.pfb
...
cmr10 CMR10 <cmr10.pfb
cmr12 CMR12 <cmr12.pfb
cmr17 CMR17 <cmr17.pfb
...
logo10 logo10 <logo10.pfb
logo8 logo8 <logo8.pfb
```

Использование его будет выглядеть примерно следующим образом:

```
latex myfile
dvips myfile -Pcms -o myfile.ps
```

что в данном случае обеспечивает подготовку PostScript-файла (`myfile.ps`), который затем можно пропустить через программу Acrobat Distiller, чтобы получить PDF-файл.

Отметим, что в последних строках map-файла имена соответствующих шрифтов набираются строчными (малыми) (logo8) буквами, что отличается от формата, принятого для шрифтов семейства Computer Modern (CMR10). Это существенный момент и делается так потому, что шрифты приходят из разных источников. Большинство шрифтов семейства Computer Modern первоначально были конвертированы в формат Type 1 фирмой Blue Sky Research в 1980-х гг., улучшены затем фирмой Y&Y, после чего при посредничестве Американского математического общества с 1996 г. стали свободно доступными; у всех этих шрифтов имена пишутся прописными (большими) буквами. Ряд элементов данного семейства, например шрифты, добавленные Тейко Хокуотером, обладает именами, записанными строчными буквами. К сожалению, некоторая путаница возникает из-за того, что есть еще один набор шрифтов Computer Modern Type 1, подготовленный Василием Малышевым в начале 90-х гг. (первая версия этого набора именовалась Paradissa, затем это имя было изменено на ВаКоМа). В данном шрифтовом наборе его элементам (отдельным шрифтам) присвоены имена, состоящие из *строчных* букв; этот набор имеется в составе ряда TEX'овских дистрибутивов. Если у вас нет уверенности в том, какая версия шрифтов в вашем распоряжении, посмотрите текст одного из pfb-файлов и найдите в нем сведения, относящиеся к авторским правам.

Наиболее широко используемые технические шрифты, относящиеся к категории public domain (т. е. к категории общедоступных) *уже* конвертированы в формат Type 1; среди них в TEX'овских архивах можно найти следующие:

- все шрифты семейства Computer Modern (включая и расширения их, сделанные для LATEX'а);
- шрифты Американского математического общества;
- символьные шрифты St. Mary's Road;
- рукописный шрифт RSFS;
- фонетические шрифты TIPA;
- шрифты XY-pic.

Варианты Type 1 и TrueType шрифтового семейства European Computer Modern (ec) можно получить на фирме MicroPress по адресу [↪MICROPRESS]. Есть, однако, еще две альтернативы:

1. Пакет ae, который обеспечивает формирование виртуальных шрифтов, отвечающих шрифтам ec, насколько это вообще возможно, и использующих рисунок исходных шрифтов Computer Modern. Несколько символов при этом оказались пропущенными, но для удовлетворения потребностей большинства пользователей этого пакета вполне достаточно.
2. Набор шрифтов European Modern (коммерческий) фирмы Y&Y [↪YANDY], который представляет собой семейство высококачественных, хинтованных шрифтов, полностью заменяющих семейство ec.[1]

[1] Оригинал этой книги на английском языке использует из семейства European Modern два шрифта — шрифт без засечек (sans serif) и шрифт, имитирующий пишущую машинку (typewriter font).

И наконец, рассмотрим еще один очень важный вопрос. Если вы используете коммерческие шрифты (например, фирм Adobe или Monotype, шрифт Lucida Bright фирмы Y&Y или же шрифты European Modern), то не можете включить целиком такой шрифт в PDF-файл и затем сделать этот файл доступным через Интернет. Это явно нарушит условия лицензионного соглашения, поскольку кто-то другой получает возможность извлечь шрифты из данного файла. Вам придется, как минимум, выделить *подмножество* упомянутых шрифтов и, вполне возможно (например, в случае небольших фирм-поставщиков вроде Y&Y, для которых «шрифтовое пиратство» представляет серьезную угрозу), дополнительно заплатить за такое использование шрифта. При этом фирма Y&Y настаивает еще и на том, чтобы были внесены изменения в механизм выделения подмножеств программы Acrobat Distiller. По умолчанию, эта программа *не* формирует шрифтового подмножества, если используется более 35% глифов, входящих в данный шрифт. Вам необходимо изменить значение этого параметра на 99% (см. рис. 2.2), чтобы быть уверенным, что Distiller всегда будет формировать подмножество до тех пор, пока не возникнет ситуация, когда задействован *каждый* глиф.[1] У этого требования есть положительная сторона — обеспечивается сокращение размера файла при сохранении приемлемого уровня качества документа.

2.2.3 Совершенствование PDF-файла

Создать PDF-отображение имеющейся у вас обычной печатной страницы — это одно, а подготовка электронного документа, использующего все те преимущества, которыми обладает PDF, — совсем другое. Как минимум, перекрестные ссылки требуют формирования PDF-документа с гипертекстовыми ссылками. Многие также рассчитывают обнаружить в этом документе возможность автоматического формирования закладок (необязательное в PDF «оглавление» в левой части документа, отображаемого на экране монитора), ожидается также, что URL будут активными ссылками и что можно будет устанавливать новые произвольные ссылки. Перечисленные средства могут устанавливаться одним из четырех способов (упорядочены по возрастанию предпочтительности):

1. Трудоемкой ручной работой по добавлению в документ ссылок с использованием программы Acrobat Exchange. Этот вариант подвержен ошибкам, а работу надо повторять всякий раз, когда в документ внесены изменения.
2. Исполнением прикладной программы, которая пытается определить ссылки на основе информации, имеющейся в файле, содержащем документ. Это не очень надежный вариант.

[1] В общем случае *невозможно* стопроцентно использовать большинство текстовых шрифтов, поскольку вектор кодирования обычно не дает доступа ко всем глифам, содержащимся в шрифте.

3. Встраиванием в вывод вашей программы специального PostScript'овского кода, который будет затем распознан программой Acrobat Distiller и преобразован, например, в связи.
4. Путем наделения вашей прикладной программы способностью формировать PDF-код непосредственно. Информация в нем будет отвечать информации о перекрестных ссылках в исходном файле.

Самым популярным из перечисленных является третий подход, который поддерживается семейством программ фирмы Adobe. Программа Acrobat Distiller распознает специальную команду PostScript'а `pdfmark`, которая потом используется в качестве «зацепки», чтобы вставить требуемые функциональные средства в PDF-файл. Так, например,

```
[   /Color [1 0 0] /H /I /Border [0 0 0] /Subtype /Link
/Action << /S /GoTo /D (figure.1) >> /Rect [100 254 125 266]
/ANN pdfmark end
```

формирует гипертекстовую ссылку из прямоугольника, определяемого командой `Rect`, на место в документе, именуемое `figure.1`, а текст

```
[   /Count 0 /Action << /S /GoTo /D (section.2) >> /Title
(Introduction) /OUT pdfmark end
```

создает закладку с текстом `Introduction`, указывающую на объект `section.2`. В наши намерения не входит рассказывать в этой книге о команде `pdfmark`, поскольку большинству пользователей применять ее напрямую не придется. Однако фирмой Adobe подготовлена хорошая документация по этой команде [↪PDFMARKD], Томас Мерц [↪PDFMARKP] написал прекрасное доступное введение (глава в Merz (1998)), а на Web-сайте Д. П. Стори [↪ACROTEX] есть детальный и очень доходчивый учебный материал по использованию команды `pdfmark` непосредственно в TEX'е.

Здесь, однако, возникает такой вопрос: как команды `pdfmark` попадают в PostScript-файл? Для пользователей TEX'а это относительно простая задача, поскольку почти все драйверы-конвертеры DVI-файлов в PostScript позволяют включать «чистый» PostScript-текст в выходной поток[1]. Используя, например, программу `dvips`, можно применять конструкции `\special{ps:: ....}`, чтобы вставить в текст требуемый PostScript-код. Маловероятно, однако, чтобы пользователю пришлось записывать эти команды в LATEX'овский документ, так как проще сделать следующее:

1. Использовать команды `\special` из HyperTEX'а (см. приложение B.1 на с. 460), чтобы вставить высокоуровневые команды, которые какой-либо драйвер преобразует затем в команды `pdfmark`.

[1] В работе Merz (1998), гл. 6, описаны методы, которые можно использовать для других приложений, наподобие Microsoft Word и FrameMaker.

2. Использовать команды `\special`, специфические для некоторого драйвера, как это делается, в частности, в программе `dviwindo` или в VTEX.
3. Лучше все же будет, если имеется возможность работать на уровне обобщенных LaTeX'овских команд с помощью какого-нибудь макропакета, преобразующего эти обобщенные команды в механизм, работающий в вашей среде.

Последний из перечисленных подходов соответствует тому, что принято в пакете `hyperref` и описывается в разд. 2.3. Аналогичным образом пользователи программ, наподобие Microsoft Word, FrameMaker и PageMaker, могут использовать существующий в этих программах механизм образования перекрестных ссылок и записывать команды `pdfmark` в выходной PostScript-файл. Полностью открытая и программируемая природа TeX'а позволяет сделать нашу программу вполне пригодной для использования такого подхода.

Кроме `hyperref`, следующие пакеты также «знают» о PDF:

- Пакеты, которые порождают команды `\special` в соответствии с соглашениями, принятыми в HyperTeX. Таков, в частности, пакет `hyper` Майкла Мелиха. Результирующий DVI-файл может быть обработан с помощью программы `dvips` при включенной опции `-z`, что дает возможность получать качественный PostScript-файл для обработки его программой Acrobat Distiller.
- Макропакет ConTeXT Ганса Хагена, обеспечивающий полную поддержку формата PDF в своих обобщенных гипертекстовых средствах.
- Полноценный PDF-файл может быть получен из документов `texinfo` с помощью программы `pdftexinfo.tex`, представляющей собой незначительно модифицированный стандартный макропакет `texinfo`. Она входит как составная часть в дистрибутив pdfTeX'а и может работать только совместно с этой системой.
- Аналогичная модификация пакета `webmac`, именуемая `pdfwebmac.tex`, позволяет формировать гипертекстовые PDF-версии программ, написанных в среде WEB. Данная программа также входит составной частью в дистрибутив pdfTeX'а.

Не надо забывать, наконец, что во многих случаях лучшее решение — работать с прикладными программами, которые сразу формируют текст в формате PDF. Один из вариантов такого рода программ (pdfTeX) будет рассмотрен в разд.2.4. Она дает возможность использовать все средства формата PDF. Программа pdfTeX добавляет к языку TeX некоторое число примитивов, которые можно использовать непосредственно. На практике, однако, многие найдут более удобным и простым оставаться в привычной среде LaTeX'а, используя пакет `hyperref`, драйвер которого отображает соответствующие команды в новые примитивы pdfTeX'а.

2.3 PDF, улучшенный LaTeX'ом: пакет hyperref

Себастианом Ратцем и Хейко Обердиком разработан пакет **hyperref**[1] Этот пакет создан в рамках проекта HyperTeX и базируется на его результатах (см. приложение B.1 на с. 460 и [↪HYPERTEX]). Он расширяет диапазон функциональных возможностей всех LaTeX'овских команд для работы с перекрестными ссылками (включая команды формирования оглавления, библиографии и т. п.) таким образом, что они начинают использовать команды `\special`, которые драйвер может преобразовать в гипертекстовые ссылки. Введен также ряд новых команд, позволяющих писать специализированные гипертекстовые ссылки, в том числе ссылки на внешние документы, а также на URL.

Данный пакет позволяет работать с несколькими DVI-драйверами, которые могут использовать либо команды `\special` из HyperTeX'а, либо команды `\special` из PostScript, если формируется *только* PDF. Еще один вариант — использование команд-примитивов pdfTeX'а. Требуемые команды определяются для различных драйверов в конфигурационных файлах, один из которых выбирается путем задания соответствующих управляющих параметров пакета. Пакет включает следующие драйверы:

hypertex Для DVI-процессоров, отвечающих требованиям со стороны HyperTeX'а (т. е., `xdvi`, `dvips` с опцией `-z`, OzTeX и Textures).

dvips Командами `\special` выводит PostScript, ориентированный на `dvips`.

dvipsone Командами `\special` выводит PostScript, ориентированный на `dvipsone`.

pdftex Выводит команды для варианта TeX'а, разработанного Таном, непосредственно порождает вывод в формате PDF (см. разд. 2.4).

dvipdfm Выводит команды `\special` для драйвера-конвертера `dvipdfm`, разработанного Марком Уиксом, преобразующего файлы из формата DVI в формат PDF.

dviwindo Выводит команды `\special`, которые программа-визуализатор фирмы Y&Y для Windows интерпретирует как гипертекстовые переходы внутри документа, отображаемого этой программой.

vtex Выводит команды `\special`, которые TeX в варианте фирмы MicroPress, порождая вывод в формате HTML и PDF, интерпретирует как гипертекстовые переходы.

Вывод программ `dvips` или `dvipsone`[2] необходимо обработать программой Acrobat Distiller для получения на его основе PDF-файла. Этот результат в об-

[1] Существенную помощь в разработке этого пакета оказали многие специалисты, в особенности Дэвид Карлайл.

[2] Оба эти драйвера поддерживают частичную загрузку шрифтов; рекомендуется отключить ее при подготовке PostScript-файла для последующей обработки его программой Acrobat Distiller, поскольку она располагает своими собственными средствами формирования шрифтовых подмножеств.

щем случае будет более предпочтительным, чем в случае использования драйвера `hypertex` с последующей обработкой командой `dvips -z`. Преимущество DVI-файла, сформированного с помощью HyperTEX'а с использованием команд `\special`, состоит в том, что полученный таким образом файл можно использовать также с гипертекстовыми программами просмотра типа `xdvi`.

2.3.1 Прозрачная работа пакета hyperref

Этот пакет, указав соответственно его в преамбуле, можно использовать практически в рамках любого нормального LATEX'овского документа. Надо только сделать так, чтобы данный пакет был *последним* среди всех загружаемых пакетов, чтобы дать ему возможность переопределить необходимые команды LATEX'а. Есть надежда, что все перекрестные ссылки будут правильно работать в качестве гипертекстовых ссылок, кроме случая, когда параметр-опция `implicit` установлен в значение `false`, при котором обрабатываются только явные команды установления гиперссылок. С помощью опций можно управлять способом визуального представления ссылок, а также обеспечить дополнительные возможности управления выходным текстом в PDF-формате.

На рис. 2.6 показан результат обработки нашего тестового файла (см. приложение А.1) при преобразовании его в формат PDF, причем управляющим параметрам `hyperref` здесь присваиваются значения по умолчанию. На рис. 2.7 показано отображение того же самого файла с помощью программы `xdvi`.

Две общеупотребительные опции пакета `hyperref` — это

- `colorlinks`, меняющая цвет текста гипертекстовых ссылок вместо помещения этого текста в рамку вокруг него (см. рис. 2.8, где вместо цвета используются оттенки серого);
- `backref`, которая вставляет добавочные «обратные» ссылки в библиографию для каждого элемента. На рис. 2.9 и 2.10 показано, что получается, когда эти параметры включены или выключены. *Примечание:* опции `backref` и `pagebackref` смогут работать соответствующим образом, только если после каждого `\bibitem` имеется пустая строка (как если бы этот элемент был создан программой BIBTEX).

2.3.2 Настройка пакета hyperref

Все настройки в пакете `hyperref` осуществляются по простой схеме «ключ-значение» (с использованием макропакета `keyval` с ключом `Hyp`). Требуемые опции можно задать либо с помощью необязательного аргумента при вызове пакета командой `\usepackage`, либо с помощью команды

`\hypersetup{`*keyvalue pairs*`}`

Следует отметить, что необязательный аргумент данного пакета использует экспериментальное расширение синтаксиса LATEX'а. LATEX налагает некоторые ограничения на точное содержание опций макропакета, так что этот метод

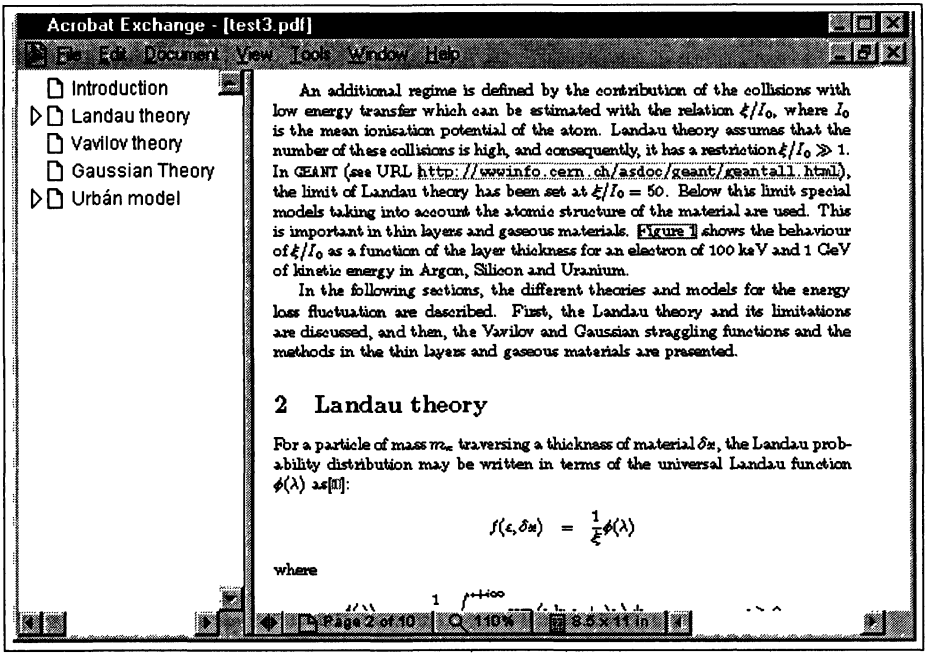


Рис. 2.6. Внешний вид тестового документа в формате PDF, улучшенного с помощью `hyperref` и выведенного со значениями управляющих параметров по умолчанию.

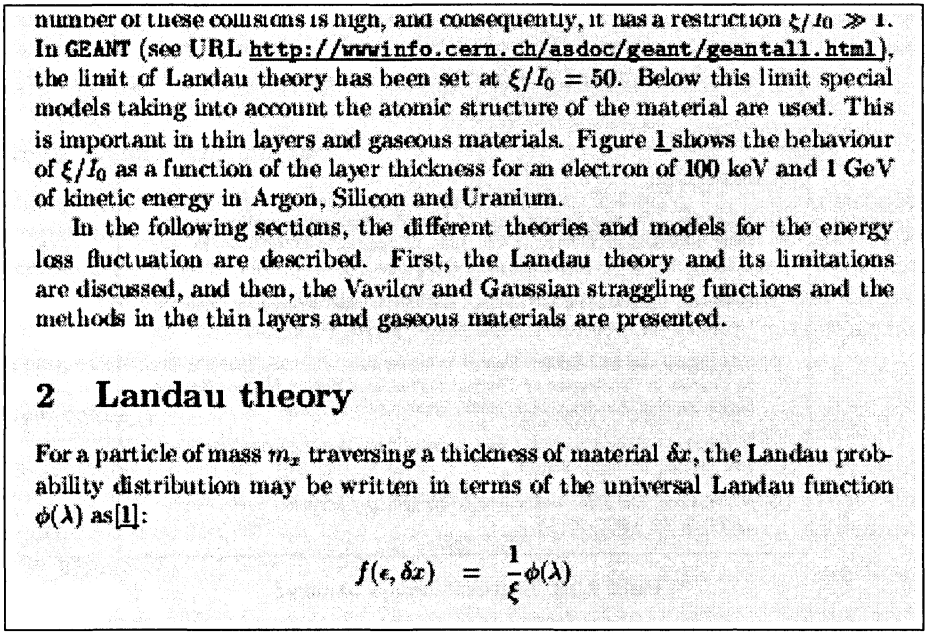


Рис. 2.7. Тестовый документ, отображаемый с помощью программы `xdvi`.

where $c \geq 4$. From the equations (13), (16) and (18) and from the conditions (22) and (23) the following limits can be derived:

$$\alpha_{\min} = \frac{(n_3 + c^2)(E_{\max} + I)}{n_3(E_{\max} + I) + c^2 I} \leq \alpha \leq \alpha_{\max} = \frac{(n_3 + c^2)(E_{\max} + I)}{c^2(E_{\max} + I) + n_3 I} \quad (25)$$

This conditions gives a lower limit to number of the ionisations n_3 for which the fast sampling can be done:

$$n_3 \geq c^2 \quad (26)$$

As in the conditions (22), (23) and (24) the value of c is as minimum 4, one gets $n_3 \geq 16$. In order to speed the simulation, the maximum value is used for α.

Рис. 2.8. Тестовый документ, показывающий использование опции `colorlinks`.

References

[1] L.Landau. On the Energy Loss of Fast Particles by Ionisation. Originally published in *J. Phys.*, 8:201, 1944. Rerpinted in D.ter Haar, Editor, *L.D.Landau, Collected papers*, page 417. Pergamon Press, Oxford, 1965.

[2] B.Schorr. Programs for the Landau and the Vavilov distributions and the corresponding random numbers. *Comp. Phys. Comm.*, 7:216, 1974.

[3] S.M.Seltzer and M.J.Berger. Energy loss straggling of protons and mesons. In *Studies in Penetration of Charged Particles in Matter*, Nuclear Science Series 39, Nat. Academy of Sciences, Washington DC, 1964.

[4] R.Talman. On the statistics of particle identification using ionization. *Nucl. Inst. Meth.*, 159:189, 1979.

[5] P.V.Vavilov. Ionisation losses of high energy heavy particles. *Soviet Physics JETP*, 5:749, 1957.

Рис. 2.9. Обычный список литературы.

References

[1] L.Landau. On the Energy Loss of Fast Particles by Ionisation. Originally published in *J. Phys.*, 8:201, 1944. Rerpinted in D.ter Haar, Editor, *L.D.Landau, Collected papers*, page 417. Pergamon Press, Oxford, 1965. 2

[2] B.Schorr. Programs for the Landau and the Vavilov distributions and the corresponding random numbers. *Comp. Phys. Comm.*, 7:216, 1974. 3

[3] S.M.Seltzer and M.J.Berger. Energy loss straggling of protons and mesons. In *Studies in Penetration of Charged Particles in Matter*, Nuclear Science Series 39, Nat. Academy of Sciences, Washington DC, 1964. 4

[4] R.Talman. On the statistics of particle identification using ionization. *Nucl. Inst. Meth.*, 159:189, 1979. 2

[5] P.V.Vavilov. Ionisation losses of high energy heavy particles. *Soviet Physics JETP*, 5:749, 1957. 3

Рис. 2.10. Влияние опции `backref`.

может работать не всегда; в общем случае для надежной работы опции могут включать только буквы, цифры и знаки пунктуации.

Кроме того, когда происходит загрузка рассматриваемого макропакета, читается файл `hyperref.cfg`, если его удается найти; этот файл — удобное место для того, чтобы задать постоянные значения всех требуемых опций. Таким образом, поведение макропакета `hyperref` при обработке конкретного файла определяют следующие элементы:

- Значения опций в файле `hyperref.cfg`, задающие вид ссылок, обратные ссылки и значения по умолчанию для опций отображения PDF-файла:

```
\hypersetup{backref,
    pdfpagemode=FullScreen,
    colorlinks=true}
```

- Глобальная опция данного документа, которая передается в `hyperref`:

```
\documentclass[dvips]{article}
\usepackage{hyperref}
```

- Опции в командах `\usepackage`, зависящие от конкретного обрабатываемого файла, *перекрывающие* те, что заданы в файле `hyperref.cfg`:

```
\usepackage[pdftitle={A Perfect Day},colorlinks=false]{hyperref}
```

Подробное описание значений всех опций данного пакета приводится в разд. 2.3.8 на с. 83. Для многих из них значения задавать необязательно, поскольку по умолчанию им присвоено значение `true`. Это те параметры, которые классифицируются как булевы. Значения `true` и `false` можно, однако, задать всегда.

Опции общего назначения

Используя одну из опций, перечисленных в табл. 2.4 на с. 83, можно выбрать конкретный драйвер. Если ни один из допустимых драйверов не задан, по умолчанию загружается драйвер `hypertex`. Большинство драйверов обеспечивает требуемый режим работы сразу после загрузки без дополнительной подстройки, однако если используется программа `dviwindo`, может потребоваться переопределение следующей команды:

```
\wwwbrowser
```

Эта команда сообщает драйверу `dviwindo`, какую программу требуется запустить, по умолчанию это будет программа `c:\netscape\netscape.exe`. Это значит, что пользователям программы Internet Explorer придется добавить в файл `hyperref.cfg` что-то вроде:

```
\renewcommand{\wwwbrowser}{C:\string\Program\space
  Files\string\Plus!\string\Microsoft\space
  Internet\string\iexplore.exe}
```

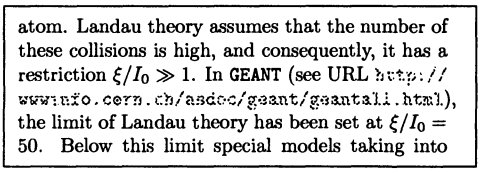

atom. Landau theory assumes that the number of these collisions is high, and consequently, it has a restriction $\xi/l_0 \gg 1$. In GEANT (see URL http://wwwinfo.cern.ch/asdoc/geant/geantall.html), the limit of Landau theory has been set at $\xi/l_0 = 50$. Below this limit special models taking into

Рис. 2.11. Разделение текста длинной ссылки на строки.

Часть опций общего назначения, перечисленных в табл. 2.5 на с. 83, применима ко всем драйверам. Важность одной из таких опций, `breaklinks`, видна из рис. 2.11; этот пример получен обработкой документа с помощью pdfTEX'а, позволяющего разделять URL на несколько частей и устанавливать для каждой из них свою ссылку. Другие драйверы не позволяют проделывать этот «фокус» и URL «заезжает» за поле страницы.

В табл. 2.6 на с. 84 перечислены опции, также применимые ко всем драйверам, но обеспечивающие расширение функциональных возможностей. Есть несколько таких параметров, которые определяют цвет текста в ссылках (гипертекстовых). Все имена цветов должны быть определены до их использования, следуя обычной системе соглашений, установленных в стандартном LATEX'овском пакете `color`. Надо сознавать при этом, что выделенные цветом гипертекстовые ссылки являются одновременно и составной частью текста документа, например, если для URL выбран зеленый цвет, то именно таким цветом и будет отпечатан этот фрагмент текста из программы Acrobat (или же некоторым оттенком серого цвета при печати на черно-белом принтере).

Использование пакета `xr` совместно с `hyperref`. С использованием макропакета `xr` можно автоматически создать набор взаимодействующих файлов. Однако, поскольку может использоваться как формат `dvi`, так и `pdf`, пакет `hyperref` не обязательно знает, с чем он имеет дело. Рассмотрим следующий файл:

```
\documentclass{article}
\usepackage{xr}
\usepackage{hyperref}
\externaldocument{other}
\begin{document}
См. разд.~\ref{facts} в другом файле
\end{document}
```

Метка `facts` определена в файле `other.tex`; когда обрабатывается текущий файл, читается файл `other.aux` и все метки в нем становятся доступными для команды `\ref` в рассматриваемом текущем файле (это функция пакета `xr`). Поскольку уже загружен пакет `hyperref`, команда `\ref{facts}` создает гипертекстовую ссылку, указывающую на файл `other`. Но должна ли эта ссылка вести к файлу `other.dvi` (это то, чего мы хотим, если используем программу `dviwindo`, например) или к файлу `other.pdf` (который программа

Acrobat может открыть непосредственно)? Пакет `hyperref` старается угадать это, однако пользователь может потребовать заменить значение, определенное пакетом, задавая суффикс файла опцией `extension`.

Опции, связанные с формированием PDF-файлов

Если выходным файлом является PDF-файл, то формированием его можно управлять, используя довольно большое количество опций. Они перечислены в табл. 2.7 на с. 85.

Установка режима отображения ссылок. Установка режимов отображения ссылок в программе Acrobat может быть довольно сложной — в отличие от многих других гипертекстовых систем, Acrobat с каждой ссылкой связывает возможность изменения масштаба. В табл. 2.1 на следующей странице показаны соответствующие возможности, т. е. набор ключей с переменным числом параметров. К сожалению, для пользователя пакета `hyperref` слишком сложно напрямую задавать значения этих параметров, они должны быть выражены в специальном координатном пространстве, по умолчанию связываемом с форматом PDF, которое совсем не обязательно будет совпадать с тем, что принято в TeX'е. Хорошая новость состоит в том, что pdfTeX пытается выполнить за пользователя эту работу, присваивая значения по умолчанию общеупотребительным ключам, `XYZ` и `FitBH`; плохая же новость — драйверы, использующие систему `pdfmark`, не поддерживают значений по умолчанию. Так, если записать

```
\usepackage[dvips, pdfview=FitBH]{hyperref}
```

то это, вообще говоря, приведет к катастрофическому результату, поскольку за `FitBH` *должно* следовать число. Чтобы несколько упростить себе жизнь, `hyperref` устанавливает в качестве значения данного параметра число -32768[1] для команды отображения, если в явном виде требуемое значение не задано. Драйвером pdfTeX это значение будет проигнорировано.

Значение *по умолчанию* — это всегда `XYZ`, за которым следует соответствующее значение для драйвера, т. е. увеличение не меняется, когда происходит переход по ссылке. Типичным можно считать такое изменение

```
pdfview=FitBH
```

при котором переход по ссылке приводит к заполнению окна неким достаточно рациональным образом с шириной соответствующей текстовой области на текущей странице.

Изменение цвета текста в гипертекстовых ссылках. Цвет ссылок в программах Acrobat можно задать *только* в виде тройки чисел в диапазоне $0 \div 1$, представляющих собой значения основных цветов в RGB-кодировке.

[1] Это бессмысленное значение заставляет программу Acrobat Distiller установить обычно ожидаемые значения по умолчанию для некоторых из ключей.

Таблица 2.1. Возможные значения для спецификаций отображения ссылок в PDF.

Ключ	*Параметры*	*Описание*
`XYZ`	*left top zoom*	Задать координаты верхнего левого угла фрагмента страницы для помещения его в окно и определения коэффициента масштабирования. Если параметры *left*, *top* или *zoom* имеют нулевые значения, используется текущее значение. Таким образом, значения «`null null null`» задают те же самые значения для *top*, *left* и *zoom*, какие они имеют на текущей странице. Значение 0 это то же самое, что и `null`.
`Fit`		Подогнать страницу под размер окна.
`FitH`	*top*	Подогнать ширину страницы под размер окна; параметр *top* задает координату y верхнего угла окна.
`FitV`	*left*	Подогнать высоту страницы под размер окна; параметр *left* задает координату x левого угла окна.
`FitR`	*left bottom right top*	Вписать прямоугольник, задаваемый параметрами *left bottom right top* в требуемое окно.
`FitB`		Согласовать рамку, ограничивающую страницу, с заданным окном.
`FitBH`	*top*	Согласовать ширину рамки, ограничивающей страницу, с заданным окном; параметр *top* задает координату y верхнего угла окна.
`FitBV`	*left*	Согласовать высоту рамки, ограничивающей страницу, с заданным окном; параметр *left* задает координату x левого угла окна.

Цвета, определяемые в TEX'е, использовать нельзя. Эти цвета *не* влияют на изменение характеристик текста ссылки и при печати никак не скажутся.

Настройка параметров окна. Опции, управляющие отображением окна, продемонстрированы на рис. 2.12; этот пример порождается следующим набором команд:

```
\usepackage[pdftoolbar=false,
 pdfmenubar=false,
 pdfwindowui=false,
 pdffitwindow=true,
 pdfpagelayout=TwoColumnLeft
]{hyperref}
```

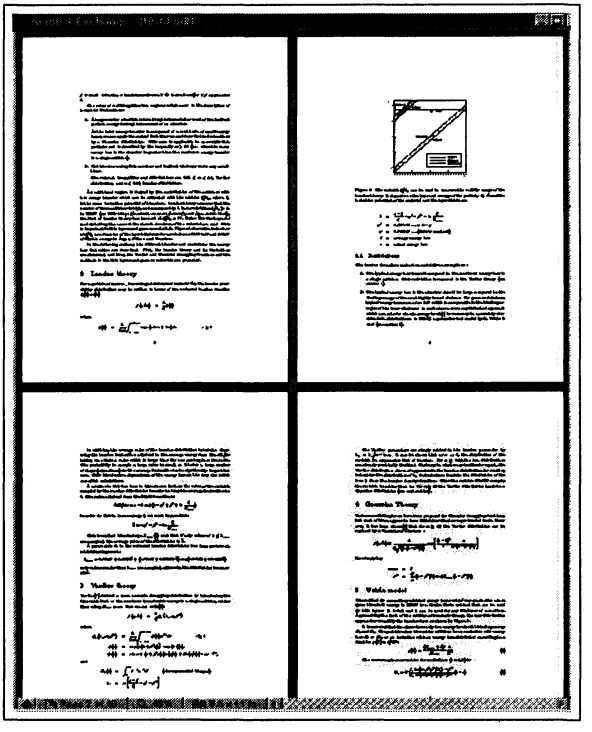


Рис. 2.12. PDF-документ, отображаемый в многостраничном режиме в окне без полос прокрутки и других элементов управления.

Поскольку панели инструментов и меню удалены, любое взаимодействие пользователя с документом должно осуществляться через сам документ (включая такие базовые функции, как Quit (Выход)); здесь может быть очень полезна команда `\Acrobatmenu` (см. разд. 2.3.4 на с. 67).

Команды установки закладок в программах Acrobat

Команды установки закладок требуют дополнительных пояснений. Они запоминаются в файле с именем *jobname*`.out`; этот файл можно затем подвергнуть постпроцессорной обработке, если из него надо удалить LaTeX'овские коды. Пакет `hyperref` старается преобразовать внутреннюю кодировку в PDFDocEncoding или в Unicode, используемые в закладках, но при этом текст закладок LaTeX'ом *не обрабатывается*, так что любая разметка проходит через него в неизменном виде. На рис. 2.13 показаны влияние опции `bookmarksopen`, а также ограничения кодировки PDFDocEncoding, поскольку математическая символика отображаться здесь не может. Чтобы справиться с некоторыми из этих проблем, была разработана макрокоманда

`\texorpdfstring{`*TeX string*`}{`*PDF string*`}`

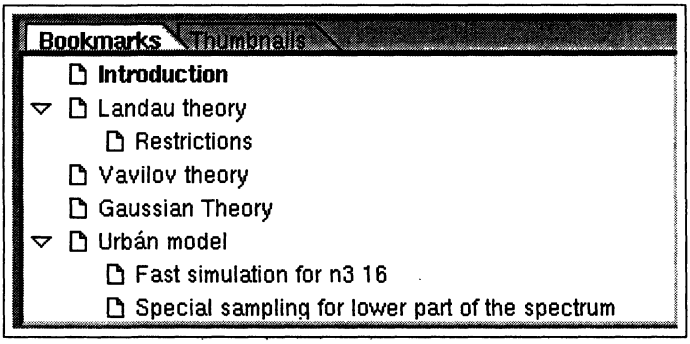


Рис. 2.13. PDF-закладки открыты.

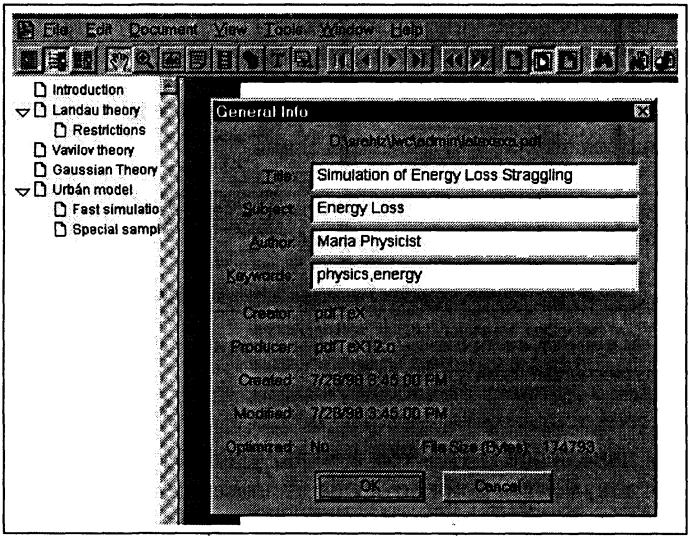


Рис. 2.14. Отображение информации о PDF-документе.

Когда она выполняется, вместо цепочки литер *TEX string*, которая не подходит для создания закладки, будет использоваться цепочка литер *PDF string*. Например, можно было бы написать

```
\section{Роль \texorpdfstring{H$_2$O}{воды}}
```

и увидеть текст «Роль H_2O» в обычном заголовке и, соответственно, «Роль воды» — в закладке.

Информационные поля PDF-документа. Опции, перечисленные в табл. 2.8 на с. 88, дают возможность помещать текст в информационные поля PDF. На рис. 2.14 показано отображение информации о документе в

программе Acrobat; рассматриваемый документ был создан с применением следующей команды в преамбуле документа:

```
\usepackage{hyperref}
\hypersetup{%
 pdfauthor={Maria Physicist},
 pdftitle={Simulation of Energy Loss Straggling},
 pdfcreator={pdfTeX},
 pdfsubject={Energy Loss},
 pdfkeywords={physics,energy}}
```

2.3.3 Дополнительные пользовательские макро для гиперссылок

Если имеется необходимость дать ссылку на URL или задать в явном виде точные ссылки, это может быть выполнено с помощью специального набора макрокоманд. Следует отметить, что имеется возможность различать связи и ссылки по категориям, однако это свойство серьезным образом в `hyperref` не используется.

`\hyperbaseurl{`*url*`}`

Этой командой вводится базовый *url*, присоединяемый спереди к другим задаваемым URL, для того чтобы облегчить создание мобильных документов.

`\href{`*url*`}{`*text*`}`

Данной командой параметр *text* становится гиперссылкой, определяемой параметром *url*, который должен представлять собой полный адрес URL (или относительный, если определен базовый URL). Специальные литеры # и ~ экранировать *не* требуется. Например,

```
\href{http://www.tug.org/~rahtz/nonsense.html#fun}{Some fun}
```

`\hyperimage{`*image url*`}`

Этой командой вставляется изображение, на которое ссылается параметр *image url*.

`\hyperdef{`*category*`}{`*name*`}{`*text*`}`

Этой командой маркируется фрагмент документа (параметр *text*), которому дается имя, определяемое параметром *category.name*.

`\hyperref{`*url*`}{`*category*`}{`*name*`}{`*text*`}`

Текст, задаваемый параметром *text*, становится ссылкой на *url#category.name*.

1. The typical energy loss is small compared to the maximum energy loss in a single collision. This restriction is removed in the Vavilov theory (see section 3).

2. The typical energy loss in the absorber should be large compared to the binding energy of the most tightly bound electron. For gaseous detectors, typical energy losses are a few keV which is comparable to the binding energies of the inner electrons. In such cases a more sophisticated approach which accounts for atomic energy levels[4] is necessary to accurately simulate data distributions. In GEANT, a parameterised model by L. Urbán is used (see section 3).

Рис. 2.15. Сопоставление действия команд `\ref` и `\autoref`.

`\hyperref[`*label*`]{`*text*`}`

Текст, задаваемый параметром *text*, становится ссылкой на место появления обычной LaTeX'овской команды `\label` с символьным именем *label* (см. далее описание команды `\ref*` для получения справки по ее синтаксису).

`\hyperlink{`*name*`}{`*text*`}`
`\hypertarget{`*name*`}{`*text*`}`

Командой `\hypertarget` создается простая внутренняя ссылка; здесь два параметра — имя ссылки *name* и текст ссылки *text*. Команда `\hyperlink` также имеет два аргумента: имя гипертекстового объекта, определенного в документе командой `\hypertarget`, и параметр *text*, используемый в качестве ссылки на данной странице.

Говоря в терминах HTML, команда `\hyperlink` вставляет # перед каждой ссылкой, что делает ее относительной в рамках данного документа; команда `\href` требует полного URL.

`\autoref{`*label*`}`

Эта команда представляет собой замену для обычной команды `\ref`, помещающей *контекстную* метку перед ссылкой. Разница в использовании этих команд показана на рис. 2.15, где первая ссылка на раздел установлена с помощью команды `\autoref{...}`, а вторая — с помощью команды `\ref{...}`. Первая из упомянутых команд включает слово «section» (раздел) как часть ссылки, тогда как последняя — только номер. Первый из этих двух вариантов чаще удобнее для пользователя, чем второй.

Метка извлекается пакетом `hyperref` из контекста первоначальной команды `\label` с использованием макро, перечисленных в табл. 2.2. Эти макро могут быть переопределены в документе с помощью команды `\renewcommand`; следует отметить, что часть из этих макро уже определена в стандартных классах документов. Использование прописных и строчных букв произвольно и соответствует авторской практике.

Таблица 2.2. Имена команды `\autoref` пакета `hyperref`.

Макро	*Значение по умолчанию*
`\figurename`	Figure (рисунок)
`\tablename`	Table (таблица)
`\partname`	Part (часть)
`\appendixname`	Appendix (приложение)
`\equationname`	Equation (уравнение)
`\Itemname`	item (элемент)
`\chaptername`	chapter (глава)
`\sectionname`	section (раздел)
`\subsectionname`	subsection (подраздел)
`\subsubsectionname`	subsubsection (подподраздел)
`\paragraphname`	paragraph (пункт)
`\Hfootnotename`	footnote (примечание)
`\AMSname`	Equation (уравнение)
`\theoremname`	Theorem (теорема)

Иногда возникает необходимость задать текст ссылки целиком самостоятельно, не используя команды `\ref` и `\pageref` для формирования ссылок. Для этой цели есть два варианта команд:

`\ref*{`*label*`}`
`\pageref*{`*label*`}`

В типичном случае использования этих команд можно было бы записать

```
\hyperref[other]{этот замечательный раздел (\ref*{other}), который мы
                 читали ранее}
```

где мы хотели бы использовать команду `\ref*{other}` для формирования правильного номера, но не для формирования ссылки. Мы выполним это самостоятельно с помощью команды `\hyperref`.

2.3.4 Специальные команды программ Acrobat

Если требуется получить доступ к командам из меню программ Acrobat Reader или Acrobat Exchange, можно воспользоваться следующей командой:

`\Acrobatmenu{`*menuoption*`}{`*text*`}`

Параметр *text* используется здесь для создания кнопки, активизирующей элемент меню, определяемый значением параметра *menuoption*. В табл. 2.3[1] перечисляются имена, которые можно использовать для параметра

[1] Эта таблица — результат кропотливого труда Томаса Мерца, который экспериментировал с программой Acrobat Exchange и опубликовал полученные результаты в Merz (1998), поскольку соответствующие имена в документации фирмы Adobe не приводятся.

Таблица 2.3. Значения имен ссылок из меню программ Acrobat.

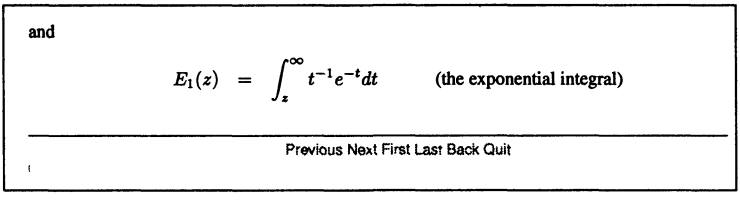
Меню программы Acrobat	*Допустимые опции для команды* `\Acrobatmenu`
File	Open, Close, Scan, Save, SaveAs, Optimizer:SaveAsOpt, Print, PageSetup, Quit
File→Import	ImportImage, ImportNotes, AcroForm:ImportFDF
File→Export	ExportNotes, AcroForm:ExportFDF
File→DocumentInfo	GeneralInfo, OpenInfo, FontsInfo, SecurityInfo, Weblink:Base, AutoIndex:DocInfo
File→Preferences	GeneralPrefs, NotePrefs, FullScreenPrefs, Weblink:Prefs, AcroSearch:Preferences (Windows) or, AcroSearch:Prefs (Mac), Cpt:Capture
Edit	Undo, Cut, Copy, Paste, Clear, SelectAll, Ole:CopyFile, TouchUp:TextAttributes, TouchUp:FitTextToSelection, TouchUp:ShowLineMarkers, TouchUp:ShowCaptureSuspects, TouchUp:FindSuspect, Properties
Edit→Fields	AcroForm:Duplicate, AcroForm:TabOrder
Document	Cpt:CapturePages, AcroForm:Actions, CropPages, RotatePages, InsertPages, ExtractPages, ReplacePages, DeletePages, NewBookmark, SetBookmarkDest, CreateAllThumbs, DeleteAllThumbs
View	ActualSize, FitVisible, FitWidth, FitPage, ZoomTo, FullScreen, FirstPage, PrevPage, NextPage, LastPage, GoToPage, GoBack, GoForward, SinglePage, OneColumn, TwoColumns, ArticleThreads, PageOnly, ShowBookmarks, ShowThumbs
Tools	Hand, ZoomIn, ZoomOut, SelectText, SelectGraphics, Note, Link, Thread, AcroForm:Tool, Acro_Movie:MoviePlayer, TouchUp:TextTool, Find, FindAgain, FindNextNote, CreateNotesFile
Tools→Search	AcroSrch:Query, AcroSrch:Indexes, AcroSrch:Results, AcroSrch:Assist, AcroSrch:PrevDoc, AcroSrch:PrevHit, AcroSrch:NextHit, AcroSrch:NextDoc
Window	ShowHideToolBar, ShowHideMenuBar, ShowHideClipboard, Cascade, TileHorizontal, TileVertical, CloseAll
Help	HelpUserGuide, HelpTutorial, HelpExchange, HelpScan, HelpCapture, HelpPDFWriter, HelpDistiller, HelpSearch, HelpCatalog, HelpReader, Weblink:Home
Help(Windows)	About

menuoption. Сравнение этого списка с меню в программах семейства Acrobat показывает способ их использования. Часть из них, очевидно, имеет смысл только в пределах программы Exchange.

Например, пусть требуется добавить меню в нижнем колонтитуле нашего документа посредством `fancyhdr`:

```
\usepackage{fancyhdr}
\usepackage[colorlinks]{hyperref}
\pagestyle{fancy}
\cfoot{\NavigationBar}
\newcommand{\NavigationBar}{%
  \Acrobatmenu{PrevPage}{Previous}~
  \Acrobatmenu{NextPage}{Next}~
  \Acrobatmenu{FirstPage}{First}~
  \Acrobatmenu{LastPage}{Last}~
  \Acrobatmenu{GoBack}{Back}~
  \Acrobatmenu{Quit}{Quit}%
}
```

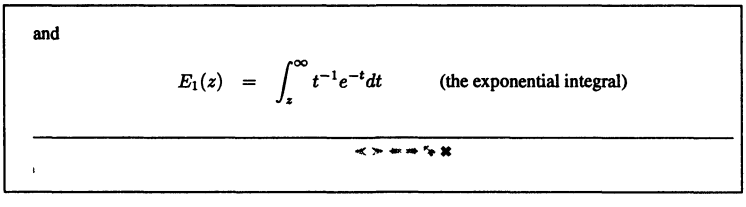
Получаемый результат можно видеть на следующем рисунке:



Текст здесь, разумеется, может быть любым, соответствующим правилам LaTeX'а. В следующем ниже варианте используются символы из шрифта ZapfDingbats (загружаемого с помощью пакета `pifont`) при том же самом наборе элементов меню:

```
\usepackage{pifont}
\usepackage{graphics}
\newcommand{\NavigationBar}{{\Large
  \Acrobatmenu{PrevPage}{\reflectbox{\ding{227}}}
  \Acrobatmenu{NextPage}{\ding{227}}
  \Acrobatmenu{FirstPage}{\reflectbox{\ding{224}}}
  \Acrobatmenu{LastPage}{\ding{224}}
  \Acrobatmenu{GoBack}{\reflectbox{\ding{249}}}
  \Acrobatmenu{Quit}{\ding{54}}%
}}
```

что приводит к следующему результату:



Вместо шрифта Dingbat можно было бы использовать некоторую картинку, с кодом наподобие

```
\Acrobatmenu{Back}{\includegraphics{backpic}}
```

2.3.5 Специализированные средства поддержки других пакетов

Пакет `hyperref` разрабатывался таким образом, чтобы обеспечить, насколько это возможно, его работу во взаимодействии с другими макропакетами. Иногда это похвальное стремление приводит к непрактичным результатам. Причинами конфликтов являются:

- Макропакеты, предназначенные для манипулирования библиографическими средствами. Обеспечивается, в частности, поддержка пакета `harvard`, разработанного Питером Уильямсом. Однако рекомендуемым для этих целей пакетом является `natbib` Патрика Дейли, оснащенный средствами организации взаимодействия с пакетом `hyperref`. Этот пакет охватывает широкую совокупность способов представления библиографии и стилей цитирования, причем все они нормально работают совместно с `hyperref`.
- Пакеты, распечатывающие содержимое макрокоманд `\label` и `\ref`, например пакет `showkeys`. Пакет `hyperref` переопределяет все такие команды, если только не используется опция `implicit=false`; при этом эти пакеты не работают соответствующим образом.
- Пакеты, проделывающие что-нибудь серьезное с предметным указателем.

Пакет `hyperref` распространяется в совокупности с двумя другими полезными пакетами, специально доработанными для взаимодействия с ним. Это пакет `xr` и пакет `minitoc`, первый из которых обеспечивает установление перекрестных ссылок между документами с использованием обычного LaTeX'овского механизма `\label`/`\ref`, а второй — формирование содержания документов по главам.

2.3.6 Создание форм в PDF и HTML

Сейчас быстро становится общеупотребительным (можно даже сказать — необходимым) умение преобразовывать «бумажные» формы в их электронные эквиваленты. Во многих Web-страницах уже используются HTML-формы, позволяющие собирать данные достаточно сложным образом, однако еще не все понимают, что формат PDF имеет те же самые функциональные возможности. В данном разделе будут рассмотрены средства, имеющиеся в пакете `hyperref` для создания полноценных форм в PDF (и в HTML).

Тем, кто интересуется созданием форм, полезно иметь в виду следующее.

1. Заполняемые формы — не единственный способ использования объектов типа «форма» в PDF. Д. П. Стори [↪ACROTEX] и Ганс Хаген [↪CONTEXT]

используют их для создания довольно изощренных интерактивных приложений и реализации сложных методов навигации в гипертексте. На рис. 2.16 показан калькулятор, разработанный Гансом Хагеном в среде TeX'а (на основе его пакета CONTeX) со встроенными средствами JavaScript и графикой, реализуемой средствами METAPOST и преобразованной в PDF.

2. Весьма развитые формы, используемые в настоящее время, представляют собой относительно недавнее добавление к PDF; чтобы использовать их, требуются программа Acrobat версии 3.01 или старше и дополнительная компонента Forms 3.5 к ней.
3. Немногие из PDF-приложений, сформированных вне TeX'а, реально поддерживают создание PDF-форм, базируясь на разметке, а документация большей частью описывает создание этих форм вручную, с использованием программы Acrobat Exchange. К пакету `hyperref` эти средства также были добавлены сравнительно недавно, хотя интерфейс и может еще измениться. *Примечание:* формы поддерживают только драйверы `pdftex`, `dvips` и `tex4ht`.

Прекрасная книга Томаса Мерца дает основу для понимания того, как PDF работает с формами (см. Merz (1998), гл. 7 и 10), а на Web-сайте П. Д. Стори [↪ACROTEX] содержится как набор отлично подобранных примеров, так и подробное учебное руководство по использованию `pdfmark` (см. разд. 2.2 на с. 45) для создания форм. Надо также иметь в виду, что бóльшая часть функциональных возможностей здесь опирается на использование языка JavaScript, который также придется изучить. Вместе с динамически подключаемым модулем Forms фирма Adobe поставляет руководство под названием *Acrobat Forms JavaScript Object Specification*; полезен также справочник *JavaScript Reference Manual* фирмы Netscape. PDF-формы — это потенциально очень мощное средство, и лишь весьма небольшая часть его возможностей может быть задействована при помощи пакета `hyperref`.

Поддержка форм в `hyperref` организуется таким образом, чтобы выдерживался стиль, принятый в HTML. При таком подходе требуется, чтобы все поля формы находились внутри окружения `Form`; в каждом файле допускается наличие только одного такого окружения.

```
\begin{Form}[parameters]
... поле ...
\end{Form}
```

Параметры *parameters* здесь — пары «ключ-значение», перечисленные в табл. 2.9 на с. 88.

В формах поддерживается четыре типа полей:

1. Текстовые поля, обеспечивающие свободный ввод текста.
2. Управляющие элементы типа checkbox.

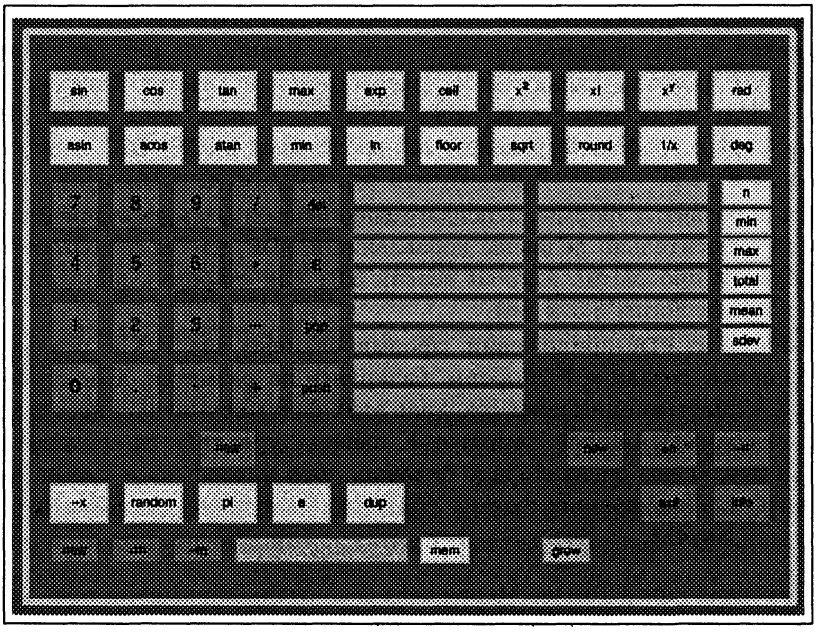


Рис. 2.16. Калькулятор, написанный на PDF Гансом Хагеном.

3. Поля выбора, позволяющие пользователю выбрать один из заданного набора вариантов.
4. Кнопки, запускающие (инициирующие) некоторые действия.

Кроме того, есть еще два типа полей: Запустить (Submit) и Сбросить (Reset). Для подготовки полей используются следующие шесть макро:

```
\TextField[options]{label}
\CheckBox[options]{label}
\ChoiceMenu[options]{label}{choices}
\PushButton[options]{label}
\Submit[options]{label}
\Reset[options]{label}
```

Отметим, что на верхнем уровне нет разницы между

- простым меню, где перечислены все варианты выбора;
- всплывающим меню, где показывается значение по умолчанию, а остальные появляются только при активизации поля;
- комбинированным меню, где дается список альтернатив, но также предоставляется возможность ввести новое значение[1];

[1] Этот тип не разрешен в HTML.

- «радиокнопкой», где пользователь может выбрать лишь одну из перечисляемых альтернатив.

Имеется обширный набор опций (см. табл. 2.10 на с. 89), которые влияют на характер реакции программы при обработке полей.

Процесс подготовки полей включает формирование текстовой метки, возможно, списка альтернатив для выбора одной из них, значения по умолчанию (необязательная возможность), а также начальный выбор. Способ, которым будет отображаться список возможностей для выбора, зависит от используемых опций; элементы в списке просто разделяются запятыми. Для каждого элемента списка можно задать раздельно «видимый» его вариант и реально возвращаемый вариант, когда выбирается этот элемент. Эти варианты задаются в виде двух строк, разделяемых знаком = для выбираемого элемента. Тогда первая часть в данной паре будет отображаемым элементом, а вторая — возвращаемым значением.

Каждое из трех основных видов полей (текстовое, поле типа checkbox, поле выбора) состоит из двух частей — *метки* и отвечающего ему *поля*. Положение метки относительно поля определяется тремя макро, которые можно переопределять:

```
\LayoutTextField{label}{field}
\LayoutChoiceField{label}{field}
\LayoutCheckboxField{label}{field}
```

Эти макро по умолчанию имеют формат `#1 #2`, т. е. метка здесь пишется слева от поля. Типичное переопределение может иметь такой вид:

```
\renewcommand{\LayoutTextField}[2]{\makebox[2in]{#1}#2}
```

согласно которому все метки будут вписываться в прямоугольник фиксированной ширины, равной 2 дюйма.

То, что на самом деле появится в области набора текста для данного поля, определяется следующим образом:

```
\MakeRadioField{width}{height}
\MakeCheckField{width}{height}
\MakeTextField{width}{height}
\MakeChoiceField{width}{height}
\MakeButtonField{text}
```

Эти макро по умолчанию формируют поле в виде прямоугольника шириной *width* и высотой *height*, причем содержимое поля будет центрироваться внутри данного прямоугольника. Параметры *width* и *height* имеют значения по умолчанию, определяемые размером отображаемого поля, но могут быть перекрыты с помощью опций (см. табл. 2.10 на с. 89). Исключением здесь будет макро для полей типа «кнопка», для которых по умолчанию принят формат `#1`; так обстоит дело для кнопок, а также для макро `\Submit` и `\Reset`.

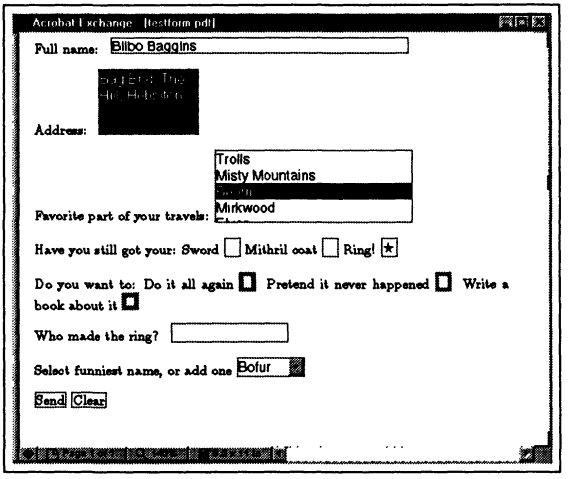


Рис. 2.17. Простая форма в программе Acrobat.

Может потребоваться также переопределить следующие макро, которые используются для генерации размеров в отсутствие другой информации:

Макро	*Значение по умолчанию*
`\DefaultHeightofSubmit`	12pt
`\DefaultWidthofSubmit`	2cm
`\DefaultHeightofReset`	12pt
`\DefaultWidthofReset`	2cm
`\DefaultHeightofCheckBox`	0.8`\baselineskip`
`\DefaultWidthofCheckBox`	0.8`\baselineskip`
`\DefaultHeightofChoiceMenu`	0.8`\baselineskip`
`\DefaultWidthofChoiceMenu`	0.8`\baselineskip`
`\DefaultHeightofText`	`\baselineskip`
`\DefaultWidthofText`	3cm

Отметим, что все цвета должны быть выражены в кодировке RGB в виде троек чисел из диапазона $0 \div 1$ (например, `color=0 0 0.5`). В общем случае эти опции просто обеспечивают интерфейс с соответствующими кодами PDF, полностью документированными в Bienz et al. (1996). Знакомство с этим документом весьма существенно, если вы предполагаете выйти за пределы, описываемые значениями по умолчанию и простыми их вариациями.

Так, все требуемые средства описаны, не пора ли привести хотя бы простой пример? На рис. 2.17 показана типичная форма, на которой представлены поля почти всех используемых типов. Рассмотрим подробно, что за код требуется для получения такой формы. Во-первых, в преамбуле документа надо

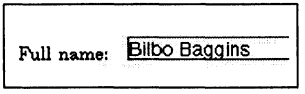
загрузить пакет **hyperref** и затем открыть окружение **Form** с соответствующим URL (простая почтовая отсылка).

```
\documentclass{article}
\usepackage[bookmarks=false]{hyperref}
\setlength{\parindent}{0pt}\setlength{\parskip}{10pt}
\begin{document}
\begin{Form}[action=mailto:srahtz,method=post]
```

Ниже дается перечень полей различных типов:

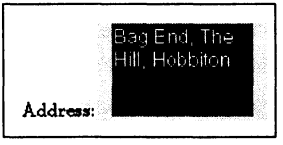
1. Текстовое поле. Здесь мы задаем значение для ширины поля, по умолчанию оно будет равно величине, принятой для значения умолчания:

```
\TextField[width=3in,name=xname,value={Bilbo Baggins}]{Full name: }
```



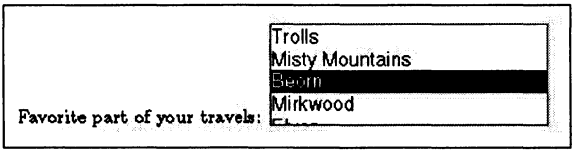
2. Текстовое поле, многострочное. Цвет текста и цвет рамки изменяются, вокруг поля будет нарисована пунктирная рамка:

```
\TextField[multiline,width=1in,name=address,borderstyle=D,
    color=1 1 1,backgroundcolor=0 0 .5,
    value={Bag End, The Hill, Hobbiton}]{Address: }
```



3. Поле выбора. По умолчанию, высота данного поля должна быть достаточной для одновременного отображения всех вариантов выбора, но мы ограничим значение этого параметра таким образом, чтобы одновременно отображалось только три варианта:

```
\ChoiceMenu[default=Home,menulength=3,width=2in,name=travel,default=Beorn]
    {Favorite part of your travels:}
    {Trolls,Misty Mountains,Beorn,Mirkwood,Elves,Laketown,%
    Smaug,The Battle}
```



4. Элементы типа checkbox. В момент запуска выбрано только одно из этих полей:

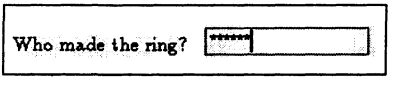
```
Have you still got your:
\CheckBox[]{Sword}
\CheckBox[name=coat]{Mithril coat}
\CheckBox[name=ring,checked]{\textbf{Ring!}}
```

5. Элемент выбора альтернативы типа «радиокнопка». Обратите внимание, что показаны различные варианты значений из тех, что будут возвращаться при выборе соответствующего элемента:

```
\ChoiceMenu[radio,default=Again,name=next,
     borderwidth=3,bordercolor=0 1 0]{Do you want to:}
    {Do it all again=Again,
     Pretend it never happened=Forget,
     Write a book about it=Write}
```

6. Текстовое поле в стиле «пароль». Когда вводится соответствующий текст, то вместо букв на экране отображаются звездочки:

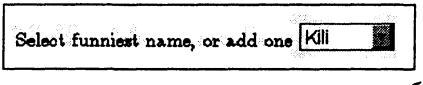
```
\TextField[password,name=made]{Who made the ring? }
```



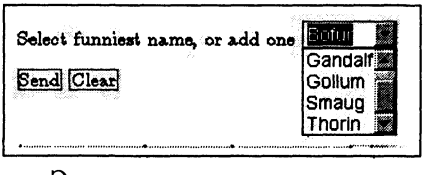
7. Элемент выбора альтернативы типа «комбинированный список». Элемент такого вида дает возможность либо выбрать необходимое значение из списка, либо непосредственно ввести такое значение:

```
\ChoiceMenu[combo,default=Bofur,name=whatdwarf]
     {Select funniest name, or add one}
     {Bofur,Thorin,Gollum,Smaug,Gandalf}
```

Ниже показано состояние, в котором будет находиться данное поле, пока оно не активизировано:



А здесь показано, как в нем открыт список для выбора:



8. Поле типа «кнопка». Это поле предназначено для запуска на исполнение некоторого кода на JavaScript при нажатии на кнопку:

```
\PushButton[name=xxx,onclick={app.beep(0)}]{Make a horrid beep}
```

9. Поля «Послать» (Send) и «Очистить» (Clear). Они обеспечивают передачу значений полей, либо очистку их, соответственно:

```
\Submit{Send}          \Reset{Clear}
```

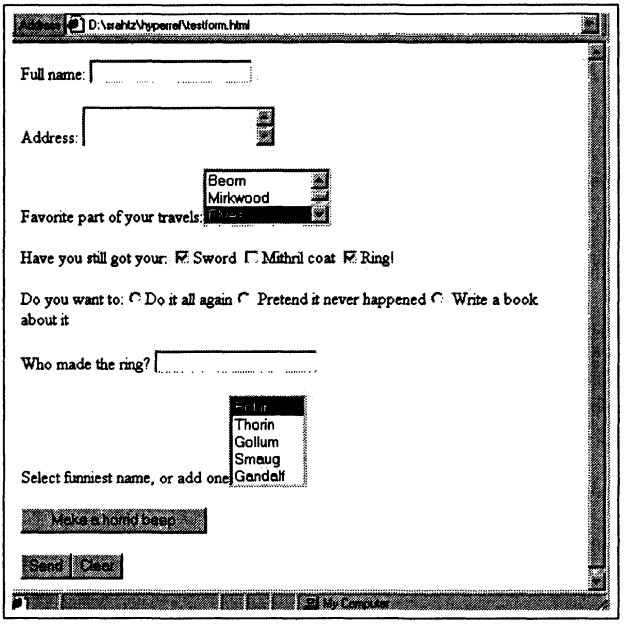


Рис. 2.18. Простая форма, представленная в HTML.

Наконец, завершим окружение `Form` и запишем команду завершения документа.

```
\end{Form}
\end{document}
```

Точно такой же LaTeX'овский файл может быть использован для получения почти идентичной HTML-формы (рис. 2.18), если указать опцию `tex4ht` при загрузке пакета `hyperref`. В гл. 4 обсуждается, как запустить программу `tex4ht`, чтобы получить HTML-файл.

Теперь, когда в нашем распоряжении имеется интерактивная форма, каким образом ввести с ее помощью данные? В гл. 10 книги Merz (1998) этот вопрос обсуждается достаточно подробно, здесь же мы только перечислим имеющиеся возможности. Есть два существенно различающихся способа ввода данных в форму и обработки их:

1. Если PDF-форма просматривается Web-навигатором и имеется соответствующий URL, нажатием кнопки Send можно отправить данные по адресу, указываемому посредством URL.
2. В меню программы Acrobat имеется позиция (File → Export → Form Data), где запрашивается имя FDF-файла, в котором надо сохранить данные из полей формы.

Файловый формат FDF подробно описан в Bienz et al. (1996), приложение H. Фирма Adobe предоставляет инструментальный пакет, в котором содержатся средства для написания программ обработки таких файлов. Это упрощенная

форма PDF, а следующий пример (связанный с нашей простой формой) показывает несложную структуру данных:

```
%FDF-1.2
1 0 obj
<<
/FDF << /Fields [
 << /V (Bag End, The Hill, Hobbiton)/T (address)>>
 << /V /Off /T (coat)>>
 << /V /next1 /T (next)>>
 << /V /Yes /T (ring)>>
 << /V /Yes /T (Sword)>>
 << /V (Mirkwood)/T (travel)>>
 << /V (Kili)/T (whatdwarf)>>
 << /V (Bilbo Baggins)/T (xname)>>
]
>>
endobj
trailer
<< /Root 1 0 R >>
%%EOF
```

2.3.6.1 Проверка значений полей форм

Можно написать JavaScript-код для осуществления тщательной проверки содержимого полей формы (в Merz (1998), с. 141–144, показано, например, как проверить значение кода ISBN), однако для выполнения несложных проверок программа Acrobat непосредственно содержит набор встроенных JavaScript-функций. Их можно использовать, задавая имена требуемых функций и соответствующих аргументов в опциях `keystroke`, `format`, `validate` и `calculate`. **Предупреждение:** Эти функции являются недокументированными и не сопровождаются фирмой Adobe! Вполне может оказаться, что в какой-то из последующих версий программы Acrobat их уже не окажется.

Перечень доступных JavaScript-функций дается ниже; следует вводить код в точности так, как это показано здесь, например

```
\TextField[name=ndwarves,
   validate={AFRange_Validate\string\(true, 3, true, 10\string\);}]
   {How many dwarves came along:  }
```

Следует подчеркнуть разницу между функциями типа `Keystroke`, которые определяют, что пользователю позволено вводить, и функциями типа `format`, задающими вид отображения вводимых данных. Обычно для одного и того же поля определены оба этих вида функций.

Обеспечить ввод значения поля и его форматирование в виде процентов или числа

```
AFPercent_Keystroke\string\(places , 0\string\);
AFPercent_Format\string\(value , 0\string\);
AFNumber_Keystroke\string\(places , 0, 0, 0, "", true\string\);
AFNumber_Format\string\(places , 0, 0, 0, "", true\string\);
```

где параметр *places* — число десятичных разрядов.

Обеспечить ввод значения поля и его форматирование в виде даты

```
AFDate_Keystroke\string\(type\string\);
AFDate_Format\string\(type\string\);
```

где параметр *type* — число из следующей таблицы:

0	1/3	5	3-Jan-81	10	Jan 3, 1981
1	1/3/81	6	03-Jan-81	11	January 3, 1981
2	01/03/81	7	81-01-03	12	1/3/81 2:30pm
3	01/81	8	Jan-81	13	1/3/81 14:30
4	3-Jan	9	January-81		

Обеспечить ввод значения поля и его форматирование в виде времени

```
AFTime_Keystroke\string\(type\string\);
AFTime_Format\string\(type\string\);
```

где параметр *type* — число из следующей таблицы:

0 14:30
1 2.30 pm
2 14:30:15
3 2:30:15pm

Отформатировать ввод заданным образом

```
AFSpecial_Format\string\(type\string\);
```

где параметр *type* — число из следующей таблицы:

0 Zip Code
1 Zip Code + 4
2 Phone Number
3 Social Security Number[1]

Проверить числа на принадлежность их заданному диапазону

```
AFRange_Validate\string\(true, minimum, true, maximum\string\);
```

Указать, что это поле является производным от других

```
AFSimple_Calculate\string\("function", "list of field names"\string\);
```

Возможные значения для параметра *function* — это `SUM`, `PRODUCT`, `AVERAGE`, `MINIMUM` и `MAXIMUM` (`СУММА`, `ПРОИЗВЕДЕНИЕ`, `СРЕДНЕЕ`, `МИНИМУМ` и `МАКСИМУМ` соответственно); элементы списка полей в параметре *list of fields* должны разделяться запятыми.

```
AFRange_Validate\string\(true, minimum, true, maximum\string\);
```

2.3.7 Дизайн PDF-документов для экрана

Для многих пользователей в качестве экранного варианта вполне достаточно иметь простую PDF-версию печатного документа, в которой активизированы ссылки. В других случаях, однако, формат PDF может употребляться для со-

[1] Social Security Number — уникален для каждого гражданина США и часто используется для его идентификации при обработке данных в некоторых госучреждениях.

здания «чисто электронных» документов, т. е. таких, которые предназначаются только для отображения на экране монитора.

В разд. 2.3.4 на с. 67 мы видели, как организовать доступ ко всем позициям меню программы Acrobat из LaTeX'а; это дает возможность строить полноценный пользовательский интерфейс, используя всю мощь TeX'а.

Рассмотрим некоторые из способов разработки документа с учетом особенностей отображения его на экране монитора (см. рис. 2.19, 2.20 и 2.21).

- Будем использовать ориентацию страницы типа landscape, для которой соотношение сторон то же самое, что и для компьютерного экрана; выберем размер страницы 6 × 4 дюйма; подстроим также значения полей. Надо помнить, что LaTeX по умолчанию устанавливает величину полей, равную одному дюйму; надо расширить отображаемую часть страницы так, чтобы поля были минимальны по величине. Следует использовать LaTeX'овскую опцию `oneside`, поскольку понятие четной и нечетной страницы здесь теряет смысл. Если ширина текста невелика, надо использовать установку `\raggedright`.
- Установим шрифтовое семейство таким, чтобы текст хорошо читался на экране, этому требованию удовлетворяет, в частности, семейство шрифтов Lucida Bright;
- Поскольку нет особых причин требовать полной заполненности страниц, примем, что каждый раздел будет начинаться с новой страницы.
- Будем использовать цвет: заголовки разделов будем выделять синим цветом, гипертекстовые ссылки окрашиваются, строки таблиц оттеняются и т. п.
- Команды `\ref` заменим командами `\autoref` (или командами `\hyperref`, если требуется более детальное управление гипертекстовыми ссылками); это дает возможность для цветных ссылок иметь более видимый контекст.
- Отключим меню программы Acrobat и инструментальную линейку, панель навигации и функциональных кнопок поместим в нижней части каждой страницы.
- Чтобы обеспечить видимый указатель на текущую страницу в пределах статьи, в нижней части каждой страницы поместим прогресс-индикатор: он работает на основе сравнения текущего номера страницы с номером последней страницы и формирования цветной полоски с длиной, соответствующей найденному выше числу.

Потребуется также определить, как быть с заметками на полях и подстрочными примечаниями, поскольку эти элементы в экранном варианте выглядят неестественно.

На рис. 2.22 показан другой возможный вариант исполнения формы; в нем присутствует навигационная панель в правой части каждой из страниц, включая и страницу с оглавлением. Пакет `pdfscreen`, построенный на основе пакета `hyperref` Ч. В. Радхакришнаном, реализует эту схему. Существуют опции, позволяющие формировать боковое меню или меню в нижней части страницы, чтобы задавать логотипы и адреса на каждой из страниц.

1 Introduction

Due to the statistical nature of ionisation energy loss, large fluctuations can occur in the amount of energy deposited by a particle traversing an absorber element. Continuous processes such as multiple scattering and energy loss play a relevant role in the longitudinal and lateral development of electromagnetic and hadronic showers, and in the case of sampling calorimeters the measured resolution can be significantly affected by such fluctuations in their active layers. The description of ionisation fluctuations is characterised by the significance parameter κ, which is proportional to the ratio of mean energy loss to the maximum allowed energy transfer in a single collision with an atomic electron

$$\kappa = \frac{\xi}{E_{\max}}$$

$E_{\max}$ is the maximum transferable energy in a single collision with an atomic electron.

$$E_{\max} = \frac{2m_e\beta^2\gamma^2}{1 + 2\gamma m_e/m_x + (m_e/m_x)^2},$$

Previous Next First Last Back Quit

Рис. 2.19. PDF-файл I, разработанный для отображения на экране.

2.1 Restrictions

The Landau formalism makes two restrictive assumptions :

1. The typical energy loss is small compared to the maximum energy loss in a single collision. This restriction is removed in the Vavilov theory (see section 3).
2. The typical energy loss in the absorber should be large compared to the binding energy of the most tightly bound electron. For gaseous detectors, typical energy losses are a few keV which is comparable to the binding energies of the inner electrons. In such cases a more sophisticated approach which accounts for atomic energy levels[4] is necessary to accurately simulate data distributions. In GEANT, a parameterised model by L. Urbán is used (see section 5).

In addition, the average value of the Landau distribution is infinite. Summing the Landau fluctuation obtained to the average energy from the dE/dx tables, we obtain a value which is larger than the one coming from the table. The probability to sample a large value is small, so it takes a large number of steps (extractions) for the average fluctuation to be significantly larger than zero. This introduces a dependence of the energy loss on the step size which can affect calculations. A solution to this has been to introduce a limit on the value of the variable sampled by the Landau distribution in order to keep the average fluctuation to 0. The value obtained from the GLANDO

Previous Next First Last Back Quit

Рис. 2.20. PDF-файл II, разработанный для отображения на экране.

E_{max} is the GEANT cut for δ-production, or the maximum energy transfer minus mean ionisation energy, if it is smaller than this cut-off value. The following notation is used:

r, C	parameters of the model
E_i	atomic energy levels
I	mean ionisation energy
f_i	oscillator strengths

The model has the parameters f_i, E_i, C and r $(0 \leq r \leq 1)$. The oscillator strengths f_i and the atomic level energies E_i should satisfy the constraints

$$f_1 + f_2 = 1 \quad (4)$$

$$f_1 \ln E_1 + f_2 \ln E_2 = \ln I \quad (5)$$

The parameter C can be defined with the help of the mean energy loss dE/dx in the following way: The numbers of collisions (n_i, i = 1,2 for the excitation and 3 for the ionisation) follow the Poisson distribution with a mean number $\langle n_i \rangle$. In a step Δx the mean number of collisions is

$$\langle n_i \rangle = \Sigma_i \Delta x \quad (6)$$

Previous Next First Last Back Quit

Рис. 2.21. PDF-файл III, разработанный для отображения на экране.

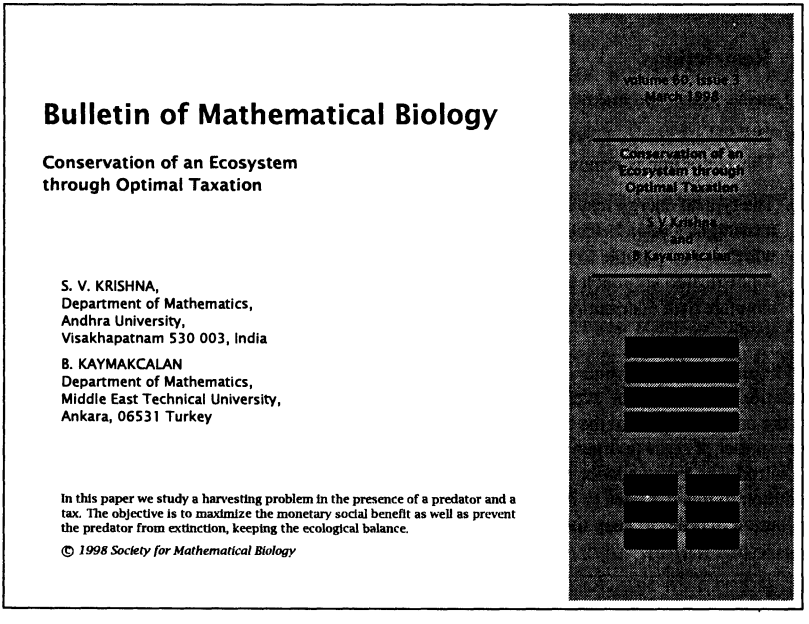


Рис. 2.22. Альтернативный вариант представления PDF-файла, подготовленного специально для отображения на экране (с разрешения фирмы River Valley Technologies and Focal Image Ltd.).

2.3.8 Каталог опций пакета

Таблица 2.4. Опции пакета `hyperref`, задающие вид драйвера.

Опция	*Значение*	*Значение по умолчанию*	*Описание*
draft	`boolean`	*false*	Все гипертекстовые опции выключены.
pdftex	`boolean`		Настраивает `hyperref` для работы с программой pdfTeX.
dvipdfm	`boolean`		Настраивает `hyperref` для работы с драйвером `dvipdfm`.
nativepdf	`boolean`		Псевдоним для `dvips`.
pdfmark	`boolean`		Псевдоним для `dvips`.
dvips	`boolean`		Настраивает `hyperref` для работы с драйвером `dvips`.
hypertex	`boolean`		Настраивает `hyperref` для работы с HyperTeX-совместимыми драйверами.
dviwindo	`boolean`		Настраивает `hyperref` для работы с `dviwindo` — программой предварительного просмотра для Windows.
dvipsone	`boolean`		Настраивает `hyperref` для работы с драйвером `dvipsone`.
vtex	`boolean`		Настраивает `hyperref` для работы с VTEX фирмы MicroPress; постпроцессоры PDF и HTML распознаются автоматически.
latex2html	`boolean`		Переопределяет несколько макро для совместимости с LaTeX2HTML.
tex4ht	`boolean`		Настраивает `hyperref` для работы с TeX4ht (см. гл. 4).

Таблица 2.5. Опции настройки конфигурации.

Опция	*Значение*	*Значение по умолчанию*	*Описание*
breaklinks	`boolean`	*false*	Разрешает разрывать текст ссылки между строками. Поскольку в PDF это не разрешается, по умолчанию устанавливается в значение true только в том случае, если используются драйверы `pdftex` или `dvipdfm`, приписывающие строкам многострочной ссылки соответствующее число PDF-ссылок, указывающих на один и тот же объект.
extension	`text`		Устанавливает расширение файла (например, `dvi`), приписываемое файловым ссылкам, которые будут формироваться в случае использования пакета `xr`.

Таблица 2.5. *Продолжение.*

Опция	*Значение*	*Значение по умолчанию*	*Описание*
hypertexnames	`boolean`	*true*	Формирует имена ссылок согласно спецификациям TEX'а. Если значение этого параметра равно `false`, то ссылкам присваиваются произвольные уникальные имена.
implicit	`boolean`	*true*	Внутренние переменные LATEX'а переопределяются для формирования гипертекстовых ссылок.
linktocpage	`boolean`	*false*	Устанавливает гиперссылки в оглавлении таким образом, чтобы они указывали на номера страниц, а не на текст соответствующего раздела.
naturalnames	`boolean`	*false*	Формирует имена ссылки в соответствии с правилами LATEX'а (даже если они при этом окажутся некорректными с точки зрения PDF).
nesting	`boolean`	*false*	Разрешает формировать вложенные (иерархические) ссылки; ни один из драйверов пока не поддерживает эту возможность, поскольку ни HTML, ни PDF не разрешают их.
pageanchor	`boolean`	*true*	Формирует ссылку с верхним левым углом каждой страницы. Если значение этого параметра есть `false`, то оглавление `\tableofcontents` не содержит гиперссылок.
plainpages	`boolean`	*true*	Заставляет отсылки к страницам нумеровать арабскими цифрами, а не использовать форматированное представление.
raiselinks	`boolean`	*true*	В драйвере `hypertex` заставляет ссылки учитывать реальную высоту литер в тексте ссылки вместо того, чтобы использовать базовую линию.

Таблица 2.6. Опции расширения.

Опция	*Значение*	*Значение по умолчанию*	*Описание*
backref	`boolean`	*false*	Добавляет в конце каждого элемента библиографического списка текст обратной ссылки в виде списка номеров разделов.
pagebackref	`boolean`	*false*	Добавляет в конце каждого элемента библиографического списка текст обратной ссылки в виде списка номеров страниц.
hyperindex	`boolean`	*false*	Превращает текст элементов указателя в гиперссылки. Это очень хрупкая операция, поэтому в любом серьезном проекте обычно потребуется реализовать свою собственную схему для выполнения этих действий.

Таблица 2.6. *Продолжение.*

Опция	*Значение*	*Значение по умолчанию*	*Описание*
colorlinks	`boolean`	*false*	Выделяет цветом текст ссылок и анкеров. Используемые цвета зависят от вида ссылки. В настоящее время принято выделять такие виды ссылок, как ссылки на литературу в библиографическом списке, ссылки на страницы, адреса URL, отсылки к локальным файлам и другие ссылки.
linkcolor	`color`	*red*	Цвет для простой внутренней ссылки.
anchorcolor	`color`	*black*	Цвет для текста ссылки.
citecolor	`color`	*green*	Цвет для библиографического цитирования в тексте.
filecolor	`color`	*magenta*	Цвет для URL, открывающих *локальные* файлы.
menucolor	`color`	*red*	Цвет для элементов меню программ Acrobat.
pagecolor	`color`	*red*	Цвет ссылок на другие страницы.
urlcolor	`color`	*cyan*	Цвет для сетевых ссылок URL.
frenchlinks	`boolean`	*false*	Записывать текст ссылок капителью вместо выделения их цветом.

Таблица 2.7. Опции отображения, специфические для PDF.

Опция	*Значение*	*Значение по умолчанию*	*Описание*
a4paper	`boolean`	*true*	Установить размер бумаги 210 мм × 297 мм.
a5paper	`boolean`	*false*	Установить размер бумаги 148 мм × 210 мм.
b5paper	`boolean`	*false*	Установить размер бумаги 176 мм × 250 мм.
letterpaper	`boolean`	*false*	Установить размер бумаги 8.5 дюйм × 11 дюйм.
legalpaper	`boolean`	*false*	Установить размер бумаги 8.5 дюйм × 14 дюйм.
executivepaper	`boolean`	*false*	Установить размер бумаги 7.25 дюйм × 10.5 дюйм.
bookmarks	`boolean`	*false*	Записать набор закладок Acrobat'a как для оглавления, требует двух запусков LaTeX'a.
bookmarksopen	`boolean`	*false*	Если задано формирование закладок Acrobat'a, показывать их со всеми раскрытыми поддеревьями.

Таблица 2.7. *Продолжение.*

Опция	*Значение*	*Значение по умолчанию*	*Описание*
bookmarksnumbered	`boolean`	*false*	Если задано формирование закладок Acrobat'а, включать в них номера разделов.
bookmarksopenlevel	`number`	*all*	Уровень, до которого показывать закладки раскрытыми.
unicode	`boolean`	*false*	Текст закладок записывать в кодировке Unicode.
citebordercolor	`RGB color`	*0 1 0*	Цвет рамки вокруг ссылки на объект.
filebordercolor	`RGB color`	*0 .5 .5*	Цвет рамки вокруг ссылок на файлы.
linkbordercolor	`RGB color`	*1 0 0*	Цвет рамки вокруг простых ссылок.
menubordercolor	`RGB color`	*1 0 0*	Цвет рамки вокруг элементов меню программ Acrobat в качестве ссылок.
pagebordercolor	`RGB color`	*1 1 0*	Цвет рамки вокруг ссылок на страницы.
urlbordercolor	`RGB color`	*0 1 1*	Цвет рамки вокруг ссылок на URL.
pdfpagemode	`name`	*None*	Определяет, каким образом открывать файл в программе Acrobat; возможные варианты здесь `None`, `UseThumbs` (показывать пиктографические кнопки (thumbnails)), `UseOutlines` (показывать закладки), и `FullScreen` (развернуть на весь экран). Если ни один из вариантов не был выбран, но установлена опция `bookmarks`, то используется значение `UseOutlines`.
pdfborder		*0 0 1*	Стиль отображения рамки вокруг ссылок. По умолчанию используется прямоугольник с линиями шириной 1bp, однако если используется опция `colorlinks`, то линия рисоваться не будет (устанавливается нулевая толщина линий).
pdfcenterwindow	`boolean`	*false*	Определяет, должна ли программа просмотра центрировать на экране дисплея окно, где будет отображаться документ.

Таблица 2.7. *Продолжение.*

Опция	*Значение*	*Значение по умолчанию*	*Описание*
pdffitwindow	`boolean`	*false*	Определяет, должна ли программа просмотра изменять окно отображения документа, чтобы согласовать его размер с размером первой из отображаемых страниц документа.
pdfhighlight	`name`	*/I*	Определяет, какое поведение должны реализовывать кнопки ссылок при их нажатии. `/I` — для инверсии (по умолчанию); другие возможности: `/N` (никакого эффекта), `/O` (контурное отображение) и `/P` (подсветка).
pdfmenubar	`boolean`	*false*	Определяет, будет ли видимым меню программы просмотра.
pdfnewwindow	`boolean`	*false*	Определяет, будут ли другие PDF-файлы стартовать в новом окне или же заменять содержимое уже открытого текущего окна.[1]
pdfpagelayout	`name`	*SinglePage*	Формат страницы при открытии документа. Возможные варианты этого параметра приводятся в табл. 2.11 на с. 90.
pdfpagelabels	`boolean`	*false*	Настроить Acrobat 4 для отображения логических номеров страниц вместо физических номеров страниц.
pdfpagescrop	`n n n n`		Установить значение по умолчанию для бокового прямоугольника (PDF crop box) для страниц. Это значение должно представлять собой набор из четырех чисел, аналогично тому, как это имеет место для PostScript BoundingBox.
pdfpagetransition	`name`		Эффект, используемый при переходе на новую страницу; возможные варианты значений этого параметра приведены в табл. 2.12 на с. 91.

[1] При работе в среде Windows вновь открываемое окно помещается в точности на то же самое место, что и ранее открытое, уже существующее окно. Чтобы сделать их видимыми одновременно, надо использовать позицию меню Window→Tile...; после чего подстроить окна вручную с помощью мышки или используя клавиши Ctrl-Tab.

Таблица 2.7. *Продолжение.*

Опция	*Значение*	*Значение по умолчанию*	*Описание*
pdfstartpage	`name`	*1*	Установить номер страницы, с которой будет начинаться документ в открываемом PDF-файле.
pdfstartview	`name`	*Fit*	Установить вид начальной страницы.
pdftoolbar	`boolean`	*false*	Определить, будет ли видимой панель инструментов программы просмотра при открытии документа.
pdfview	`name` *parameters*	*XYZ*	Установить PDF-вид для каждой из ссылок.
pdfwindowui	`boolean`	*false*	Определить, будут ли видимыми элементы пользовательского интерфейса.

Таблица 2.8. Информационные опции PDF.

Опция	*Значение*	*Значение по умолчанию*	*Описание*
baseurl	`URL`		Установить базовый URL для PDF-документа.
pdftitle	`text`		Задать значение информационного поля Title (Заглавие) для документа.
pdfauthor	`text`		Задать значение информационного поля Author (Автор) для документа.
pdfsubject	`text`		Задать значение информационного поля Subject (Тема) для документа.
pdfcreator	`text`		Задать значение информационного поля Creator (Программа-создатель) для документа.
pdfproducer	`text`		Задать значение информационного поля Producer (Исполнитель) для документа.
pdfkeywords	`text`		Задать значение информационного поля Keywords (Ключевые слова) для документа.

Таблица 2.9. Опции окружения `Form`.

Опция	*Значение*	*Значение по умолчанию*	*Описание*
action	`URL`		URL для приема данных, введенных в поля формы, если в форму включена кнопка Submit.
encoding	`name`		Способ, которым кодируется строка для URL; в норме это FDF-кодировка и единственно верное значение здесь `html`.

Таблица 2.9. *Продолжение.*

Опция	*Значение*	*Значение по умолчанию*	*Описание*
method	`name`		Значениями этой опции могут быть `post` или `get`; используется только при формировании HTML (см. [↪HTML4], разд. 17.3).

Таблица 2.10. Опции формы.

Опция	*Значение*	*Значение по умолчанию*	*Описание*
accesskey	`key`		(Как в HTML)
align	`number`	*0*	Выравнивание для текстового поля: 0 — выравнивание влево, 1 — центрирование, 2 — выравнивание вправо.
backgroundcolor	`RGB color`	*1 1 1*	Цвет фона.
bordercolor	`RGB color`	*1 0 0*	Цвет границы.
bordersep	`dimen`	*1pt*	Зазор между содержимым поля и его границей.
borderstyle	`name`	*S*	Стиль отображения границы. Возможные варианты: `S` (сплошная линия), `D` (пунктирная линия), `B` (с имитацией рельефности границы), `I` (врезка) и `U` (подчеркнутый текст). Более подробно см. Bienz et al. (1996), разд. 6.6.1.
borderwidth	`number`	*1*	Ширина границы поля (в типографских пунктах).
calculate			Код JavaScript для вычисления значения поля.
charsize	`dimen`	*10pt*	Размер шрифта для текста в поле.
checked	`boolean`	*false*	Будет ли данное поле активизировано по умолчанию.
color	`RGB color`	*0 0 0*	Цвет текста в поле.
combo	`boolean`	*false*	Будет ли использоваться выбор в стиле комбинированного списка.
default			Значение по умолчанию для поля
disabled	`boolean`	*false*	Отключено ли поле.
format			Код JavaScript для форматирования введенного элемента.
height	`dimen`		Высота поля.
hidden	`boolean`	*false*	Будет ли поле упрятываться.
keystroke			Код JavaScript для контроля нажатий клавиш при вводе.
maxlen	`number`	*0*	Число литер, допускаемое в текстовом поле (0 означает отсутствие ограничения).
menulength	`number`	*4*	Число элементов, показываемых в списке.
multiline	`boolean`	*false*	Будет ли текстовый блок многострочным.

Таблица 2.10. *Продолжение.*

Опция	*Значение*	*Значение по умолчанию*	*Описание*
name	`name`		Имя поля (по умолчанию — содержимое метки).
onblur			Код JavaScript.
onchange			Код JavaScript.
onclick			Код JavaScript.
ondblclick			Код JavaScript.
onfocus			Код JavaScript.
onkeydown			Код JavaScript.
onkeypress			Код JavaScript.
onkeyup			Код JavaScript.
onmousedown			Код JavaScript.
onmousemove			Код JavaScript.
onmouseout			Код JavaScript.
onmouseover			Код JavaScript.
onmouseup			Код JavaScript.
onselect			Код JavaScript.
password	`boolean`	*false*	Будет ли содержимое текстового поля отображаться в стиле «пароль».
popdown	`boolean`	*false*	Будет ли список выбора отображаться в стиле ниспадающего меню.
radio	`boolean`	*false*	Будет ли список выбора отображаться в стиле набора радиокнопок.
readonly	`boolean`	*false*	Будет ли для поля разрешено только чтение.
tabkey			(как в HTML)
validate			Код JavaScript для проверки корректности ввода.
value			Начальное значение для поля, не совпадающее со значением по умолчанию.
width	`dimen`		Ширина поля.

Таблица 2.11. Опции формата страницы для программы Acrobat.

Имя	*Описание*
`SinglePage`	Отображение единичной страницы.
`OneColumn`	Отображение страниц в одну колонку.
`TwoColumnLeft`	Отображение страниц в две колонки, причем левые страницы разворотов будут нечетными.
`TwoColumnRight`	Отображение страниц в две колонки, причем нечетными будут правые страницы разворотов.

Таблица 2.12. Опции межстраничных переходов программы Acrobat

Имя	*Ключ(и)*	*Описание*
`Split`	/Dm, /M	Чтобы показать начало новой страницы, поперек страницы проводятся две линии; эти линии могут быть как горизонтальными, так и вертикальными, они могут перемещаться от центра или от углов.
`Blinds`	/Dm	Появляется несколько линий, распределенных по поверхности экрана, они синхронно разворачиваются в одном и том же направлении, чтобы открыть новую страницу. Эти линии могут быть как горизонтальными, так и вертикальными.
`Box`	/M	Прямоугольник, раскрывающийся из центра к углам.
`Wipe`	/Di	Единичная линия, раскрывающаяся на экране от одного угла к другому, открывая изображение новой страницы.
`Dissolve`		Изображение страницы рассыпается на кусочки, символизируя появление новой страницы.
`Glitter`	/Di	Аналогично предыдущей позиции (Dissolve), за исключением эффекта прохождения через изображение в широкой полосе, передвигаясь от одной стороны экрана к другой.

Ключ(и) задают эффекты, например `pdfpagetransition={Blinds /Dm /V}`

/Di (Направление) Направление движения, в градусах (против часовой стрелки). Значения этого параметра обычно изменяются с шагом 90°.

/Dm (Размер) Если допускается выбор между горизонтальным и вертикальным, то значение этого параметра будет `/H` (горизонтальный) или `/V` (вертикальный).

/M (Движение) Если задается эффект движения из центра к углам, то значение данного параметра будет `/O` (наружу), а если наоборот — то `/I` (внутрь).

2.4 Формирование документов в формате PDF непосредственно из TEX'а

Целью проекта pdfTEX, реализованного Таном[1], было создание расширенного варианта TEX'а, который давал бы возможность формировать файлы в формате PDF непосредственно из исходного TEX'овского текста, а также позволял бы улучшить качество TEX'овского набора, используя средства PDF. Программа pdfTEX (она основана на исходных текстах TEX'а в его первоначальном варианте, а также на `web2c`) содержит TEX в качестве подмножества: если пользователем не задается формирование выходного текста в PDF, то pdfTEX выдает обычный DVI-файл в качестве результата работы, в противном случае генерируется PDF-файл, порождающий в точности такой же по виду документ, что и соответствующий DVI-файл. Следующий этап проекта pdfTEX связан с изучением альтернативных алгоритмов выравнивания, возможно использова-

[1] Мы признательны Тану за существенную помощь, оказанную при подготовке данного раздела.

ние технологии multiple master fonts. В нашей книге, однако, мы эти вопросы обсуждать не будем.

2.4.1 Установка и настройка программы pdfTEX

Программа pdfTEX [↪PDFTEXS] входит в состав целого ряда свободно распространяемых дистрибутивов TEX'a[1], включая MikTeX и fpTEX для Windows 32, teTEX для UNIX, CMacTeX для Macintosh и общую систему `web2c`, на основе которой построено большинство этих пакетов.

Помимо обычных TEX'овских шрифтов и макро, в состав дистрибутива pdfTEX входят также следующие компоненты:

`pdftex.pool` буферный file, необходимый для получения форматных файлов;

`ttf2afm` внешняя программа для формирования AFM-файлов из шрифтов TrueType, необходимая для создания TEX'овских файлов метрик шрифтов;

`pdftex.cfg` конфигурационный файл для pdfTEX (см. разд. 2.4.1.1);

`map` карты определения шрифтов типов PostScript и TrueType (см. разд. 2.4.1.3).

При работе pdfTEX'a кроме обычно используемых TEX'ом путей для поиска файлов используются еще несколько:

`VFFONTS` — путь, где pdfTEX будет искать виртуальные шрифты;

`T1FONTS` — путь, где pdfTEX будет искать шрифты типа Type1;

`TTFONTS` — путь, где pdfTEX будет искать шрифты типа TrueType;

`PKFONTS` — путь, где pdfTEX будет искать растровые PK-шрифты;

`TEXINPUTS` — путь, где pdfTEX будет искать конфигурационный файл (`pdftex.cfg`) и графические файлы (см. разд. 2.4.2.3);

`TEXPSHEADERS` — путь, где pdfTEX будет искать карты шрифтов (`map`), файлы с кодировками (`enc`) и шрифты.

2.4.1.1 Конфигурационный файл для pdfTEX

При запуске программы pdfTEX читается конфигурационный файл с именем `pdftex.cfg`, поиск которого осуществляется в каталоге, определяемом параметром `TEXINPUTS`. Поскольку системы, построенные на основе `web2c`, задают обычно собственное дерево каталогов для pdfTEX, где размещаются файлы конфигурации и описания шрифтов, это дает возможность пользователям или проектам располагать специализированными версиями этих файлов. Из этого следует также, что индивидуальным TEX'овским входным файлам нет необходимости устанавливать какие-либо макро, специфические для pdfTEX.

[1] Ко времени написания данной книги ни один из разработчиков коммерческих вариантов TEX'a не включил в них pdfTEX, хотя в состав пакета VTEX фирмы MicroPress входит их собственный конвертер для прямого преобразования TEX'a в PDF.

Конфигурационные файлы используются для того, чтобы задать значения по умолчанию следующих параметров, любой из которых может быть перекрыт в исходном TEX'овском файле:

output_format Целочисленный параметр, определяющий вид вывода. Если его значение равно нулю, то документ будет создаваться в формате PDF, в противном случае — в формате DVI.

compress_level Целочисленный параметр, определяющий уровень сжатия текста (с использованием библиотеки `zlib`). При нулевом значении сжатие проводиться не будет; если он равен 1, то сжатие нацелено на создание наиболее быстро работающего документа; при значении 9 — наиболее компактного документа; значения от 2 до 8 задают некоторые промежуточные положения.

decimal_digits Целочисленный параметр, определяющий точность представления действительных чисел в PDF-тексте. Корректное значение этого параметра лежит в диапазоне $0 \div 5$. Большее значение данного параметра соответствует более высокой точности представления чисел, но одновременно это и бóльшие требования к памяти и времени вывода на печать. В большинстве случаев оптимальным значением будет 2.

image_resolution Целочисленный параметр, определяющий значение по умолчанию для разрешения, с которым надо отображать растровые рисунки, если их файлы не содержат в явном виде такой информации.

page_width, page_height Параметры, определяющие ширину и высоту страницы, на которую будет выводиться PDF-документ. Если значения этих параметров не заданы, то ширина страницы вычисляется на основе ширины выводимого бокса прибавлением к нему $2 \times$ (`horigin` + `\hoffset`). Аналогичным образом вычисляется и высота страницы.

horigin, vorigin Параметры-размеры, определяющие смещение выходного TEX'овского бокса относительно верхнего левого угла «бумажного листа».

map Имя файла с картой шрифтов (аналогично тому, что используется многими драйверами для преобразования документа из формата DVI в формат PostScript); можно задать более одного такого файла, используя несколько `map`-строк. Если имени `map`-файла предшествует префикс `+`, его значение будет присоединено к уже существующему набору, в противном случае — будет заменять его. Если не задано ни одного `map`-файла, производится поиск файла `psfonts.map`, задаваемого по умолчанию.

Типичный конфигурационный файл `pdftex.cfg` может выглядеть так, как это показано ниже. Он устанавливает размер бумаги A4, задает стандартное TEX'овское смещение в 1 дюйм, а также загружает два `map`-файла с определениями шрифтов.

```
output_format 1
compress_level 0
decimal_digits 2
```

```
page_width 210mm
page_height 297mm
horigin 1in
vorigin 1in
map standard.map
map +cm.map
```

2.4.1.2 Настройка шрифтов

Программа pdfTEX обычно работает со шрифтами типа Type 1 и TrueType; должны быть доступны исходные тексты всех шрифтов, используемых в документе, за исключением тех 14 базовых шрифтов, что поддерживаются программами Acrobat (это шрифты семейств Times, Helvetica, Courier, Symbol и Dingbats). В pdfTEX можно также использовать и шрифты, сформированные METAFONT'ом. Однако настоятельно рекомендуется этой возможностью не пользоваться, если в наличии имеются варианты соответствующих шрифтов в формате Type 1 или TrueType, поскольку получаемые из METAFONT-шрифтов шрифты Type 3 в существующих версиях программы Acrobat воспроизводятся очень плохо. Обычно особых проблем здесь не возникает, так как для всех шрифтов семейства Computer Modern существуют свободно доступные шрифты Type 1[1], поэтому можно работать лишь со стандартными PostScript-шрифтами.

2.4.1.3 Файлы определения шрифтов

Программа pdfTEX читает *карты шрифтов* (`map`-файлы), имена которых определены в *конфигурационном файле* (см. разд. 2.4.1.1), где заданы кодировки шрифтов и информация относительно частичной загрузки для каждого из шрифтов. Должны быть перечислены все требуемые шрифты, каждый в отдельной строке, причем отдельно от PK-шрифтов. Синтаксис каждой из этих строк аналогичен тому, что употребляется в картах шрифтов программы `dvips`[2], причем каждая из этих строк может содержать до шести полей, разделенных пробелами: *texname*, *basename*, *fontflags*, *fontfile*, *encoding* и *special*. Обязательным является только поле *texname*, которое должно идти первым. Остальные поля необязательны, но если задано поле *basename*, то оно должно быть вторым. Аналогично, если задано поле *fontflags*, то оно должно быть третьим (если присутствует поле *basename*) или вторым (если поле *basename* не задано). Поля *fontfile*, *encodingfile* и *special* можно располагать в произвольном порядке, но указанные первые три поля должны идти в перечисленном выше фиксированном порядке.

texname: имя, используемое TEX'ом, т. е. имя TFM-файла. Оно должно указываться всегда.

[1] Свободно доступные кириллические шрифты представлены в пакете `pscyr`. Пакет можно найти на FTP по одному из адресов: `ftp://mch5.chem.msu.su/pub/russian/psfonts/`, `ftp://ftp.vsu.ru/pub/tex/font-packs/pscyr/`. — *Прим. ред.*

[2] Без каких-либо затруднений можно использовать большинство из карт шрифтов программы `dvips`, причем программа pdfTEX может пользоваться ими совместно с `dvips`.

basename: имя PostScript-шрифта. Если не задано, то будет взято из шрифтового файла. Если задается имя, не совпадающее с именем шрифтового файла, pdfTEX выдает предупреждающее сообщение, поэтому данное поле лучше не использовать, если соответствующий шрифтовой ресурс доступен (это наиболее общий случай). Этот параметр в первую очередь рассчитан на использование базовых шрифтов и для обеспечения совместимости с `map`-файлами программы `dvips`.

fontflags: флаги, определяющие некоторые характеристики шрифтов. Ниже дается описание (с некоторыми изменениями), взятое из спецификации формата PDF (Bienz et al. (1996), разд. 7.9.2).

Значение ключа Flags в дескрипторе шрифта представляет собой 32-битовое целое число, которое содержит набор булевых атрибутов. Эти атрибуты имеют значение «истина» (true), если соответствующий бит в данном целом числе равен 1. Ниже дается смысл, приписываемый каждому из битов, причем самым младшим считается первый бит. Резервируемые (незанятые) биты должны иметь нулевые значения.

Положение бита	*Семантика бита*	*Пример текста*
1	Моноширинный шрифт	`Sample Text`
2	Шрифт с засечками	Sample Text
3	Символьный шрифт	
4	Рукописный шрифт	*SAMPLE TEXT*
5	Зарезервировано	
6	Шрифт использует кодировку Adobe Standard Roman Character Set	
7	Курсив	*Sample Text*
8–16	Зарезервировано	
17	Прописной	SAMPLE TEXT
18	Капитель	Sample Text
19	Задать полужирный для малых кеглей	
20–32	Зарезервировано	

Все литеры в моноширинном шрифте имеют одну и ту же ширину, тогда как литеры в пропорциональных шрифтах имеют в общем случае различную ширину. Литеры в шрифтах с засечками имеют короткие штрихи-засечки, проведенные под некоторым углом в верхней и нижней части основных штрихов литеры, тогда как в шрифтах без засечек такие элементы отсутствуют. Символьный шрифт содержит не буквы и цифры, составляющие большую часть «обычного» шрифта, а некоторые символы. Литеры рукописного шрифта имитируют написание букв и цифр от руки. Прописной шрифт используется обычно для целей шрифтового выделения, например в заголовках, названиях разделов, не содержащих строчных («малых») букв, в отличие от шрифта типа «капитель», где такие буквы содержатся, причем по рисунку они соответствуют прописным буквам, а по размеру — строчным буквам окружающего текста.

Бит 6 в рассматриваемом поле флагов показывает, что рассматриваемый шрифт использует кодировку Adobe Standard Roman Character Set или же некоторое ее подмножество, и что используются стандартные имена для таких литер.
Наконец, бит 19 определяет, надо ли прорисовывать полужирные литеры с дополнительными пикселами, даже если размер текста (кегль) совсем невелик. Обычно, если литеры малых кеглей отображаются на устройствах с очень невысокой разрешающей способностью, например на экране монитора, элементы литеры могут иметь ширину лишь в один пиксел. Поскольку это естественная граница снизу по толщине элементов на устройствах такого типа, обычные (светлые) литеры будут отображаться точно таким же образом, что и полужирные литеры, и их нельзя будет различить. Если бит 19 установить в ненулевое значение, полужирные литеры будут отображаться более толстыми штрихами.

Если шрифтовые флаги не заданы, pdfTeX будет обращаться со шрифтами как имеющими ненулевое значение бита 3, т. е. как с неким символьным шрифтом. Если корректное значение нам неизвестно, то лучше его вообще не указывать; неправильное значение шрифтовых флагов может создать помехи работе программы Acrobat.

fontfile: имя файла шрифта. Это должен быть шрифтовой файл типа Type 1 или TrueType. Данному имени шрифтового файла может предшествовать один или два специальных символа, указывающие, как надо обращаться с этим шрифтовым файлом.

- Если имени предшествует символ <, то шрифтовой файл будет *частично загружаемым*, это означает, что в состав формируемого документа будет включено (встроено) подмножество рассматриваемого шрифта, содержащее лишь те его литеры, которые реально используются в тексте данного документа. Этот способ наиболее употребителен и *настоятельно рекомендуется* применять именно его, поскольку при таком подходе обеспечивается мобильность формируемого PDF-документа и в то же время минимизируется его объем.

- Если имени шрифтового файла предшествует сдвоенный символ <<, то в формируемый файл будет включен используемый шрифт в полном объеме, т. е. все его литеры, в том числе и те, что не используются в тексте документа ни разу. Помимо того, что при таком подходе увеличивается размер PDF-файла, могут возникнуть затруднения, если используются шрифты типа TrueType, поэтому применять данный подход не рекомендуется. Он может быть оправдан лишь в случае, когда используется некий «странный» шрифт, подмножество из которого программой pdfTeX не может быть выделено корректно.

- Если имени шрифтового файла не предшествует ни одного символа, то данный файл будет читаться, но в формируемый файл он включаться не будет ни целиком, ни по частям. Из этого файла будут лишь извлечены параметры шрифта, чтобы сформировать дескриптор шрифта, который будет использован программой Acrobat для имитации тре-

буемого шрифта. Этот вариант может быть полезен в случаях, когда нежелательно встраивать шрифт даже частично в файл с текстом документа (чтобы максимально сократить его размер), но желательно использовать метрики шрифта, чтобы потребовать от программы Acrobat подобрать при отображении документа литеры, максимально близкие по характеристикам к литерам в исходном документе, в случае, когда требуемого шрифта нет на компьютере, где производится просмотр или печать PDF-документа. Чтобы использовать данный вариант, *необходимо* задать значения шрифтовых флагов, причем так, чтобы бит 6 был «включен», что означает, что будут имитироваться только шрифты Adobe Standard Roman Character Set. Единственное исключение составляет символьный шрифт Adobe Symbol, не очень полезный.

- Если имени шрифтового файла предшествует знак `!`, шрифт вообще не читается и предполагается, что он имеется на том компьютере, где будет воспроизводиться формируемый документ. Этот вариант может использоваться для формирования PDF-файлов, не содержащих никаких встроенных шрифтов. Тогда полученный PDF-файл можно будет просмотреть или отпечатать только на том компьютере, где установлены подходящие шрифты. Такой вариант не очень подходит для обмена документами, поскольку соответствующий файл нельзя считать мобильным. С другой стороны, он весьма полезен, если надо максимально ускорить обработку документа через pdfTEX, например при его тестировании. Эта черта требует, чтобы программа Acrobat имела доступ ко всем установленным шрифтам, включая и те из них, что находятся в TEX'овских каталогах.

Следует отметить, что стандартные 14 шрифтов не встраиваются в файл с документом никогда, даже если их пометить как встраиваемые в файле определения шрифтов.

encoding: имя файла, содержащего вектор кодировки, который надо использовать с данным шрифтом. Имени файла может предшествовать символ `<`, но результат будет тем же. Формат вектора кодирования идентичен используемому программой `dvips` (см. Goossens et al. (1997), разд. 11.2.4). Если никакого вектора кодирования не задано, используется по умолчанию встроенная кодировка. Его можно опустить, если есть уверенность в том, что требуемый шрифтовой ресурс имеет корректную встроенную кодировку. В общем случае последний вариант предпочтителен и *обязателен* для подмножеств шрифтов типа TrueType.

special: специальные инструкции для преобразования шрифтов как в программе `dvips`. Читаются только спецификации `SlantFont` и `ExtendFont`; остальные инструкции игнорируются.

Если программа pdfTEX не может локализовать шрифт в `map`-файле, она вначале будет пытаться найти его исходный текст как файл с расширением `pgc`, представляющий собой PGC-источник (PDF Glyph Container — Контейнер знаков PDF)[1]. Если PGC-источник отсутствует, pdfTEX попытается использовать растровые PK-шрифты тем же самым способом, что и обычные DVI-драйверы.

Строки, в которых не содержится ничего, кроме параметра *texname*, показывают, что надо использовать масштабируемый шрифт Type 3. Для шрифтов таких типов, как Type 1, TrueType и масштабируемый Type 3, все запросы к данному шрифту любого размера обеспечиваются единственным шрифтом, включенным в формируемый PDF-файл. В таком случае, если, например, в `map`-файле указан шрифт `csr10`, предполагается, что он является масштабируемым. Шрифт `csr10` будет загружаться однократно для `csr10`, `csr10 at 12pt` и т. д.

Не беда, если масштабируемый шрифт Type 3 не указан в `map`-файле, правда, исходный текст шрифта будет загружаться многократно, если требуется несколько размеров этого шрифта, т. е. формируемый PDF-файл будет несколько больше по размеру. С другой стороны, если шрифт указан в `map`-файле как масштабируемый Type 3, а его PGC-источник не является масштабируемым или же недоступен (в этом случае pdfTEX будет использовать взамен него PK-шрифт), выходной PDF-документ также будет корректным. Однако некоторые шрифты будут выглядеть уродливыми из-за того, что масштабировались растровые шрифты.

Примеры элементов файла определения шрифтов

Использовать встроенный шрифт с кодировкой, специальной для данного шрифта, т. е. не требуется ни загрузка данного шрифта в формируемый документ, ни определение внешней кодировки для него. `SlantFont` определяется тем же самым способом, что для программы `dvips`.

```
psyr        Symbol
psyro       Symbol          ".167 SlantFont"
```

Использовать встроенный шрифт с внешней кодировкой из файла (`8r.enc`). Символ `<`, предшествующий имени файла с кодировкой, можно опустить.

```
ptmri8r     Times-Italic    <8r.enc
ptmro8r     Times-Roman     <8r.enc   ".167 SlantFont"
```

Использовать частично загружаемый шрифт с внешней кодировкой:

```
putr8r      Utopia-Regular  <8r.enc <putr8a.pfb
putro8r     Utopia-Regular  <8r.enc <putr8a.pfb     ".167 SlantFont"
```

[1] Это текстовый файл, который содержит шрифт PDF Type 3, созданный Гансом Хагеном с использованием METAPOST'а и соответствующих утилит. В общем случае PGC-файлы могут содержать информацию, позволяющую поддерживать шрифты в описании PDF-страницы. В настоящее время PGC-шрифты не очень полезны, поскольку векторные шрифты Type 3 не слишком хорошо отображаются программой Acrobat. Они станут полезнее, когда улучшатся шрифты Type 3.

Использовать имя шрифта Type 1, взятое из самого загружаемого шрифта:

```
logo8      <logo8.pfb
```

Подстроить ширину литер, но не толщину их штрихов:

```
logod10          logobf10          <logobf10.pfb   ".913 ExtendFont"
```

Использовать полностью загружаемый шрифт без перекодировки его:

```
pgsr8r GillSans <<pgsr8a.pfb
```

Использовать частично загружаемый шрифт без перекодировки его:

```
pgsr8r GillSans <pgsr8a.pfb
```

Не читать шрифт совсем — этот шрифт должен быть доступен на том компьютере, где будет воспроизводиться документ:

```
pgsr8r GillSans !pgsr8a.pfb
```

Использовать полностью загружаемый шрифт с перекодировкой его:

```
pgsr8r GillSans <<pgsr8a.pfb 8r.enc
```

Использовать частично загружаемый шрифт с перекодировкой его:

```
pgsr8r GillSans <pgsr8a.pfb 8r.enc
```

Не включать шрифт в формируемый документ, но извлечь его параметры из шрифтового файла и перекодировать шрифт:[1]

```
pgsr8r GillSans 32 pgsr8a.pfb 8r.enc
```

Шрифт TrueType используется тем же самым способом, что и шрифт Type 1:

```
verdana8r Verdana  <verdana.ttf 8r.enc
```

2.4.1.4 Шрифты TrueType

Как мы уже видели, pdfTEX может работать со шрифтами TrueType, причем подключаются они непосредственным добавлением имен шрифтов в соответствующие карты шрифтов. Единственная дополнительная задача, требующая решения, состоит в создании шрифтовых метрических файлов (TFM-файлов). Существует программа, именуемая `ttf2afm` и включаемая в состав дистрибутива pdfTEX'а, которую можно применять для извлечения шрифтовых AFM-метрик из шрифтов TrueType. Использовать ее достаточно просто:

```
ttf2afm ttf-file [encoding]
```

Здесь имя шрифта TrueType записано как *ttf-file*, а необязательный параметр *encoding* задает файл с кодировкой, который точно такой же, как и в `map`-файле для pdfTEX'а и программы `dvips`. Если кодировка в явном виде не задана, все литеры в AFM-выводе отображаются на `/.notdef`. Программа `ttf2afm` записывает AFM-вывод в стандартный вывод. Отсюда можно получить TFM-файл из AFM-файла (Goossens et al. (1997, разд. 10.5)). Если требуется знать,

[1] Это будет работать только для шрифтов с кодировкой Adobe Standard Encoding. Шрифтовые флаги показывают, на что похож этот шрифт, так что программа Acrobat сможет сгенерировать нечто подобное, если соответствующий шрифтовой ресурс недоступен на целевом компьютере.

какие литеры доступны в данном шрифте, можно прогнать программу `ttf2afm` без кодировки, чтобы получить все имена литер.

Чтобы использовать новый шрифт TrueType (файл `times.ttf`), минимально необходимый набор шагов (в предположении, что упоминание о файле `test.map` включено в конфигурационный файл `pdftex.cfg`) должен быть таким (для операционной системы UNIX):

```
ttf2afm times.ttf 8r.enc >times.afm
afm2tfm times.afm -T 8r.enc
echo "times TimesNewRomanPSMT <times.ttf <8r.enc" >>test.map
```

В выдаче программы `afm2tfm` содержится упоминание о PostScript-шрифтовом имени, TimesNewRomanPSMT, однако присутствие его в `map`-файле для pdfTEX'а не является строго необходимым.

Программы `ExtendFont` и `SlantFont` работают также и для шрифтов TrueType.

2.4.2 Новые примитивы

Программа pdfTEX добавляет к TEX'у набор новых примитивов (описываемых в руководстве по pdfTEX'у в [↪PDFTEXEX]); основная их часть рассматривается в следующих разделах. Эти примитивы обеспечивают пользователю полный доступ ко всем средствам формата PDF.

2.4.2.1 Параметры настройки документа

`\pdfoutput=`*n*

Этот целочисленный параметр определяет, в каком формате будет выдаваться документ — в DVI или в PDF. Если его значение больше нуля, то формируется PDF-файл, в противном случае — DVI-файл. Этот параметр нельзя задать *после* пересылки первой страницы. Другими словами, его значение следует задать до того момента, как pdfTEX выдаст первую страницу, если в качестве выходного формата был задан PDF. Это единственный обязательный параметр, требующий определения при необходимости выдать документ в формате PDF, все остальные параметры — необязательные.

`\pdfcompresslevel=`*n*

Этот целочисленный параметр определяет уровень сжатия текста программой `zlib`. Его нулевое значение указывает на отсутствие сжатия, 1 — сжатие для наиболее быстрого исполнения, 9 — для наиболее компактного представления, $2 \div 8$ — нечто промежуточное между этими двумя состояниями. Любое значение, лежащее вне этого диапазона, будет приведено к ближайшему осмысленному значению.

`\pdfpagewidth=`*dimen*
`\pdfpageheight=`*dimen*

Эти параметры определяют размеры страницы (ее ширину и высоту) при выдаче PDF-документа. Если они не заданы, размеры страницы будут вычислены так, как это описано в разд. 2.4.1.1.

`\pdfpagesattr=`{*tokens*}

Этот список лексем определяет необязательные атрибуты для каждой страницы формируемого PDF-документа. Этими атрибутами могут быть `MediaBox` (прямоугольник, задающий естественный размер страницы), `CropBox` (прямоугольник, задающий область страницы, которая будет отображаться и печататься) и `Rotate` (параметр поворота страницы в градусах по часовой стрелке при ее отображении или печати — он может быть либо равен 0, либо кратным 90).

`\pdfpageattr=`{*tokens*}

Этот параметр аналогичен параметру `\pdfpagesattr`, но обладает более высоким приоритетом по сравнению с ним. Он может быть использован для перезаписи любых атрибутов, задаваемых с помощью параметра `\pdfpagesattr` для отдельных страниц.

2.4.2.2 Каталог и информация о документе

`\pdfinfo`{*info keys*}

Эта команда дает возможность пользователю добавить информацию в раздел документа, содержащий его описание (информационную характеристику). Если такая информация была введена, она доступна из программы Acrobat Reader через последовательность позиций меню `File`↝ `Document Info`↝ `General`. Параметр *info keys* представляет собой набор информационных пар (ключ—значение). Именам ключей должны предшествовать символы `/`, а значения должны быть заключены в скобки; все ключи являются необязательными параметрами. К числу возможных ключей относятся следующие: `/Author` (автор), `/CreationDate` (дата создания, по умолчанию — текущая дата), `/ModDate` (дата модификации), `/Creator` (программа-создатель, по умолчанию — «TeX»), `/Producer` (программа, непосредственно сформировавшая данный документ, по умолчанию — «pdfTeX»), `/Title` (заглавие), `/Subject` (тема) и `/Keywords` (ключевые слова).

Параметры `/CreationDate` и `/ModDate` выражаются в форме `D:YYYYMMDDhhmmss`, где YYYY — год, MM — месяц, DD — день, hh — час, mm — минуты, ss — секунды.

Разрешается многократное использование команды \pdfinfo; если ключ задан более одного раза, наиболее приоритетным считается его первое появление. Пример использования команды \pdfinfo выглядит следующим образом:

```
\pdfinfo{
    /Title (example.pdf)
    /Creator (TeX)
    /Producer (pdfTeX)
    /Author (Tom and Jerry)
    /CreationDate (D:19980212201000)
    /ModDate (D:19980212201000)
    /Subject (Example)
    /Keywords (cat;mouse)
}
```

\pdfcatalog{*catalog keys*} openaction {*action*}

Каталог документа (список информационных элементов, характеризующих его) аналогичен информационной секции (разделу) документа, а в число его ключей входят /URI, определяющий базовый URL документа, а также /PageMode, который определяет, как программа Acrobat отображает документ при запуске. Возможны следующие варианты:

/UseNone	Открыть документ, ни его закладку, ни дерево не показывать.
/UseOutlines	Открыть документ, показать его схему.
/UseThumbs	Открыть документ, показать его дерево.
/FullScreen	Открыть документ в полноэкранном режиме. В этом режиме нет меню, органов управления окном, других окон.

Значение по умолчанию — /UseNone.

Параметр *action* задает действие, которое предпринимается при открытии документа; оно задается тем же самым способом, что и для внутренних ссылок (см. разд.2.4.2.6), например goto page 3 {/Fit}.

2.4.2.3 Включение графики

\pdfximage width *dimen* height *dimen* depth *dimen* {*filename*}
\pdflastximage
\pdfrefximage *number*

Команда \pdfximage создает объект типа «изображение» из файла *filename*, при этом можно (по дополнительному запросу) поменять ширину, высоту, глубину изображения или любую комбинацию из этих параметров (атрибутов).

По умолчанию для глубины изображения принято нулевое значение, а для высоты и ширины — естественные размеры рисунка. Если заданы значения всех этих параметров, изображение может быть промасштабировано, чтобы подогнать его к заданным величинам. Если заданы некоторые из этих параметров (но не все), оставшиеся масштабируются пропорционально заданным так, чтобы сохранить размеры рисунка неизменными по отношению к рисунку в его исходном виде. Если ничего этого не сделано, то изображение будет выводиться согласно его естественным размерам. Каждый объект-изображение в PDF-файле имеет уникальный идентификатор; доступ к нему можно получить с помощью команды `\pdflastximage`, а отображение осуществляется с использованием команды `\pdfrefximage`. Типичный пример:

```
\pdfximage{picture.png}\pdfrefximage\pdflastximage
```

Размеры изображения можно оценить вводом команды `\pdfrefximage` в TEX'овский блок и проверкой размеров блока. Команду `\pdfrefximage` в файле можно использовать несколько раз, чтобы указать на одно и то же изображение.

Тип файла определяется расширением в его имени: `png` означает формат PNG, `pdf` — формат PDF, `tif` — формат TIFF; в остальных случаях изображение трактуется как имеющее формат JPEG.

`\pdfimageresolution=`*resolution*

Если изображение является растровым, а соответствующий файл включает информацию о разрешении, то будет использована эта информация. В противном же случае можно воспользоваться командой `\pdfimageresolution`, чтобы определить значение этого параметра. По умолчанию это 72 dpi.

2.4.2.4 Формы XObject

`\pdfxform` *number*

выводит TEX'овский бокс *number* (число) как форму XObject в PDF-файл.

`\pdflastxform`

возвращает номер объекта последней из форм XObject, записанной в PDF-файл.

`\pdfrefxform` `\`*name*

включает ссылку на форму XObject, названную как `\name`.

Эти макро обеспечивают поддержку повторного использования объектов в pdfTEX. Содержимое формы XObject соотносится с содержимым TEX'овского блока, который состоит из текста, рисунков и ссылок на другие объекты-формы типа XObject. Форма XObject может быть использована простой ссылкой на ее номер (номер объекта). Это полезно в больших документах со зна-

чительным количеством однотипных элементов, поскольку предотвращает дублирование идентичных объектов. Достаточно общий пример такого рода — стиль документа, где в колонтитуле каждой из страниц используются одни и те же графические или текстовые элементы.

2.4.2.5 Аннотации

`\pdfannot` `width` *dimen* `height` *dimen* `depth` *dimen* {*text*}

присоединяет некоторую аннотацию к заданной текущей точке в рассматриваемом тексте. Содержимое аннотации представляет собой «сырой» PDF-код, как задано в параметре *text*.

`\pdflastannot`

возвращает номер объекта, отвечающего последней аннотации, созданной командой `\pdfannot`. Эти два примитива позволяют создать аннотацию, которая не может быть создана командой `\pdfstartlink` (см. ниже).

2.4.2.6 Адресаты и ссылки

`\pdfdest` ⟨ `num` {*number*} — `name` {*refname*} ⟩ *appearance*

определяет адресатов для установления ссылок и помещения закладок. Ссылка должна идентифицироваться либо параметром *number*, либо символическим параметром *refname* и задавать программе Acrobat способ, которым должна быть определена страница. *Внешний вид* ее должен быть одного из следующих видов:

`fit`	согласовать целую страницу с окном
`fith`	выдержать ширину страницы
`fitv`	выдержать высоту страницы
`fitb`	выдержать величину ограничивающей рамки страницы
`fitbh`	выдержать ширину ограничивающей рамки страницы
`fitbv`	выдержать высоту ограничивающей рамки страницы
`xyz`	использовать текущий масштабирующий коэффициент

За параметром `xyz` может следовать необязательный параметр `zoom` *factor*, чтобы обеспечить фиксированное значение масштабирующего коэффициента. Параметр *factor* подобен TEX'овскому увеличению (magnification); т. е. 1000 — это «нормальный» вид страницы.

`\pdfstartlink` ***height*** {*dimen*} ***depth*** {*dimen*} ***attr*** {*attributes*} *action*

открывает гипертекстовую ссылку. Если необязательные размеры не заданы, они могут быть вычислены из бокса, включающего эту ссылку. Параметр

Таблица 2.13. Действия ссылок в PDF.

Действие	*Результат*
page *n*	Перейти к странице *n*.
goto num *number*	
goto name {*refname*}	Перейти к точке, определяемой как *number* или *name* командой `\pdfdest`.
goto file {*filename*}	Открыть локальный файл; это действие может быть использовано с параметром *refname* или спецификацией *number*, чтобы указать на конкретное место (точку) в файле. Например, `goto file{foo.pdf} name{intro}`, откроет файл `foo.pdf` при адресате *intro*.
thread num {*number*}	
thread name {*refname*}	Перейти к потоку новостей, идентифицируемому параметром *number* или *refname*.
user {*spec*}	Предпринять действие, заданное пользователем. Раздел 6.9 руководства по PDF содержит перечень возможных вариантов и их объяснение. Типичный пример использования здесь — задание URL, например, `/S /URI /URI (http://www.tug.org/)`

attributes (более детально разъясняется в разд. 6.6 руководства по PDF) определяет вид, который имеет рассматриваемая ссылка. Обычно она используется для того, чтобы задать цвет и толщину границы вокруг рассматриваемой ссылки. Тогда `/C [0.9 0 0] /Border [0 0 2]` задает цвет (в RGB-кодировке) как ярко-красный, а границу — толщиной в 2 пункта.

Параметр *action* может означать многое; часть из этих возможностей перечисляется в табл. 2.13.

`\pdfendlink`

завершает ссылку; весь текст между командами `\pdfstartlink` и `\pdfendlink` будет трактоваться как часть этой ссылки. Программа pdfTEX может разделить текст этой ссылки между несколькими строками или страницами, в этом случае она сформирует несколько ссылок, имеющих одно и то же содержимое.

2.4.2.7 Закладки

`\pdfoutline` *action* `count` {*n*} {*text*}

создает закладку. Первый параметр определяет действие, которое следует предпринять, причем оно будет таким же, какое допускается для команды

\pdfstartlink. Параметр *count* задает число элементов следующего уровня, подчиненных рассматриваемому элементу; он равен нулю, если данный элемент не имеет подчиненных элементов следующего уровня (в таком случае данный параметр можно не задавать). Если значение данного параметра задано отрицательным, то все подчиненные элементы будут закрытыми и абсолютное значение этого параметра определяет число подчиненных элементов. Параметр *text* определяет то содержимое, которое будет показано в окне отображения закладок документа.

2.4.2.8 Потоки новостей

```
\pdfthread num {number} name {refname}
...
\pdfendthread
```

создает потоки новостей в документе. Команда \pdfendthread должна быть в блоке той же самой глубины, что и команда \pdfthread. Все блоки на этом уровне глубины будут трактоваться как элементы данного потока. Должен быть задан идентификатор (*number* или *refname*); потоки с одним и тем же идентификатором будут объединены.

```
\pdfthreadhoffset=dimen
\pdfthreadvoffset=dimen
```

определяют поля для отображения потока.

2.4.2.9 Разное

Есть несколько команд, которые дают возможность подготовленным пользователям управлять выводом в формате PDF на еще более детальном уровне.

```
\pdfliteral{pdf text}
```

Подобно команде \special в обычном TEX'е, эта команда вставляет необработанный код PDF в формируемый документ. Она позволяет поддерживать работу с цветом, а также преобразования текста; используется в драйвере pdftex стандартного графического пакета graphics.

```
\pdfnames{data}
```

Эта команда помещает параметр *data* в словарь имен каталога PDF.

```
\pdfobj stream {text}
\pdflastobj
\pdfrefobj number
```

Эти примитивы позволяют поместить объект, определенный пользователем, в PDF-вывод. Команда \pdfobj аналогична команде \pdfliteral, однако

текст вставляется в качестве объекта PDF. Если задан необязательный ключ *stream*, содержимое объекта будет помещено в качестве потока. Команда `\pdflastobj` возвращает номер последнего объекта, созданного с помощью команды `\pdfobj`, а команда `\pdfrefobj` открывает возможность повторного использования рассматриваемого объекта.

`\pdfmovechars`

Эта команда определяет, должен ли pdfTEX пытаться переместить символы из диапазона $0 \div 31$ в более высокие области, чтобы предотвратить проблемы с некоторыми программами предварительного просмотра и принтерами.

`\pdfincludechars{`*font*`}{`*text*`}`

Эта команда заставляет pdfTEX считать, что символы, определяемые параметром *text*, должны использоваться в шрифте *font*; это означает, что соответствующие литеры будут включены в состав шрифтовых ресурсов PDF-вывода.

`\pdftexversion`

Эта команда возвращает номер версии pdfTEX'a, умноженный на 100; например, для версии `0.13b` будет выдано число 13.

`\pdftexrevision`

Эта команда выдает буквенное обозначение текущей редакции для данной версии pdfTEX'a; в частности, для `0.13b` будет выдано `b`.

2.4.3 Графика и цвет

Программа pdfTEX обеспечивает включение в документ иллюстраций в форматах PNG, JPEG и PDF. Наиболее популярная техника — включение рисунков в формате Encapsulated PostScript — заменена использованием рисунков в формате PDF. Файлы EPS можно преобразовать в формат PDF с помощью программы Ghostscript, Acrobat Distiller или какого-либо другого конвертера PostScript в PDF. Ограничивающий блок PDF-файла берется из `CropBox`, если этот параметр определен, в противном случае — из `MediaBox`. Чтобы получить правильное значение `MediaBox`, надо скорректировать EPS-файл до начала его преобразования таким образом, чтобы его стартовая точка имела координаты (0,0), а размер страницы в точности отвечал бы значению `BoundingBox`. Доступен скрипт на языке Perl ([↪EPSTOPDF]), написанный Себастианом Ратцем, позволяющий решать эту задачу.

К числу других альтернативных вариантов использования графики в pdfTEX'е относятся:

- LATEX'овский режим формирования рисунков Поскольку этот режим реализован просто с помощью шрифтовых глифов, он работает в точности так же, как и обычный.
- XY-pic Если не вызывается постпроцессор PostScript'а, пакет XY-pic использует свои собственные шрифты Type 1 и каких-либо специальных мер для работы с ним не требуется.
- tpic Команды `\special` пакета tpic (используется в некоторых макропакетах) могут быть переопределены посредством макро Ганса Хагена, что дает возможность получать вывод в «чистом» PDF.
- METAPOST формирует свой вывод в PostScript, но в существенно упрощенной форме; преобразование METAPOST-в-PDF (написано Гансом Хагеном и Тэнмоем Бхаттачариа) реализуется как набор макро, читающих вывод METAPOST'а и поддерживающих все его функциональные возможности. Пакетом `graphics` для LATEX'а поддерживается тип `mps`, предназначенный для этой цели.

Последние два макрофайла представляют собой часть макропакета ConTEXT (это файлы `supp-pdf.tex` и `supp-mis.tex`), однако они доступны также и отдельно и могут работать с LATEX'ом.

Включение необработанных PostScript-команд — метод, использованный в пакете `pstricks` (Goossens et al. (1997), гл. 4) — не разрешается[1]. Хотя PDF и является прямым потомком языка PostScript, в нем отсутствуют некоторые команды, характерные для языка программирования, и поэтому он не может иметь дело с произвольными командами PostScript'а.

Стандартные LATEX'овские пакеты `graphics` и `color` располагают опциями `pdftex`, которые позволяют применять обычные команды управления цветом, вращением текста и включением графики. Реализация включения графики организована таким образом, что сколько бы раз ни встречался в выходном документе некоторый графический объект, включен он будет в этот документ лишь один раз (даже в тех случаях, когда этот объект подвергается масштабированию и т. п.).

Некоторое число примеров вывода программы pdfTEX можно найти на Web-сервере TUG [↪PDFTEXEX].

[1] Этот метод *может* быть использован в системе VTEX фирмы MicroPress, которая включает встроенный интерпретатор языка PostScript.

Транслятор LaTeX2HTML

В этой главе подробно обсуждается транслятор LaTeX2HTML. Для интерпретации LaTeX'овского исходного кода и трансляции его в просматриваемые по Сети гипердокументы в нем наряду с другими общедоступными средствами используется разработанный Ларри Уоллом язык Perl.

После краткого исторического обзора и напоминания основных принципов создания документов для Сети, мы рассмотрим различные компоненты рассматриваемой системы — установку, пользовательскую настройку и механизмы расширения. В языке HTML практически отсутствует поддержка математики, поэтому мы подробно изучим три основных математических режима LaTeX2HTML и его расширения для оптимизации воспроизведения математики в Сети. Затем речь пойдет о работе LaTeX2HTML с исходными документами, написанными не по-английски, с особым упором на трансляции заголовков и ключевых слов, различные кодировки, специальные шрифты, препроцессоры и т. д. В заключение мы рассмотрим не менее важный вопрос о том, как с помощью определенных в пакете `html` расширений можно использовать LaTeX2HTML в качестве инструмента для создания настоящего гипертекста из LaTeX'овских исходных документов.

3.1 Введение

LaTeX, в основе которого лежит система набора текстов TeX, обладает более широкими возможностями, чем большинство программных средств обработки текстов. В его основном описании (Lamport (1994)) LaTeX представлен как «Система подготовки документов». Так и LaTeX2HTML — это нечто гораздо большее, чем LaTeX'овская опция *Save As HTML (Сохранить в формате HTML)*. Лучше всего к нему подходит определение «Система подготовки Web-документов».

LaTeX2HTML — это открытое программное средство, распространяемое по лицензии GNU. Для приведенных в этой главе примеров необходима версия 99.1 или старше. Она работает в среде операционных систем Linux, OS/2,

Windows NT, Windows 95 и DOS, а также в большинстве вариантов системы UNIX, для которой она была первоначально разработана. Версия для платформы Macintosh находится в состоянии разработки.

По умолчанию создаваемые HTML-страницы соответствуют спецификации HTML 3.2. Поэтому такие страницы можно просматривать в навигаторах, поддерживающих эту или более новую спецификацию HTML 4.0. При необходимости вместо этого можно выбрать создание страниц, которые соответствуют HTML 2.0. При таком ограничении для более сложных LaTeX'овских окружений, таких как таблицы или выравнивания в математических формулах, используется графика. И наоборот, можно выбрать создание дополнительных атрибутов `CLASS` и `ID`, соответствующих спецификации HTML 4.0. Они позволяют ассоциировать информацию стиля с конкретными типами окружения или даже с отдельными экземплярами окружений, абзацами и фрагментами текста. Более того, когда в одном документе используется несколько языков, то путем создания атрибутов `LANG` в версии HTML 4.0 обеспечивается поддержка многих языков и диалектов.

Транслятор LaTeX2HTML написан на языке Perl Ларри Уолла (Wall et al. (1996)), так что в принципе LaTeX2HTML работает в любой системе, в которой установлен язык Perl. Но для того чтобы LaTeX2HTML мог начать трансляцию, необходимы еще некоторые программные средства. Как и LaTeX, язык Perl и другое необходимое прикладное программное обеспечение являются общедоступными. Большая его часть, вероятно, была вами приобретена вместе с операционной системой при покупке машины либо была установлена системным администратором на подсоединенных к сети персональных компьютерах. Во всяком случае, он существует на компакт-дисках для почти всех компьютерных платформ. Недостающие компоненты легко получить по Сети; как это сделать, объясняется в разд. 3.2.

3.1.1 Несколько исторических замечаний

Первоначально LaTeX2HTML был разработан Никосом Дракосом. В 1995 г. Дракос представил свою программу Интернет-сообществу с помощью списка рассылки LaTeX2HTML [↪L2HLIST]. К концу 1996 г. местонахождением репозитория для использования разработчиками стал Дармштадтский университет в Германии, а впоследствии — Бейрутский университет, где его теперь поддерживает в рабочем состоянии Марек Рушаль. С тех пор многие внесли свой вклад в программу, обеспечивая оптимизацию для различных платформ, работая над мобильностью и совершенствуя процедуры инсталляции. Многие из этих разработок были непосредственной реакцией на полученные из списка рассылки запросы о поддержке отдельных LaTeX'овских команд и пакетов.

Документация, поддержка LaTeX'овских пакетов, стратегии создания изображений и расширения для самых последних спецификаций HTML, а также общая координация работ — все это, главным образом, заслуга Росса Мура[1].

[1] Полный список участников работы можно найти в Руководстве пользователя LaTeX2HTML.

3.1.2 Принципы создания Web-документов

С ранних дней своего возникновения Всемирная паутина была признана средством, благодаря которому узкоспециализированная, структурированная информация может стать легкодоступной. TEX и LATEX используются главным образом в технических областях, где сами данные гораздо важнее, чем способ их представления. Поэтому естественно настроить средство (программу) трансляции, предназначенное для представления структурированных данных, таким способом, чтобы обеспечить как удобочитаемость, так и использование структуры для легкой навигации.

Первые версии LATEX2HTML были разработаны Никосом Дракосом для того, чтобы использовать только зарождавшуюся тогда Всемирную паутину для компьютеризованного обучения и образования. Всегда требовалось представлять наиболее важную информацию. Дракос также понял, что необходимо обеспечить легкое перемещение по представленной информации согласно ее логической структуре. Он перечислил несколько принципов, которые считал необходимыми для любого серьезного программного обеспечения, предназначенного для создания Web-документов [↪DRAKOSWWW]:

1. Автоматическое создание гипертекстовых сетей на основе структуры.
2. Гибкость в определении размера узлов.
3. Автоматическое создание средств навигации.
4. Включение сложно форматируемой информации, например рисунков, таблиц, математических уравнений, диаграмм, химических формул и экзотических языков.

Основанные на этих принципах ранние версии LATEX2HTML были достаточно эффективны, даже при очень ограниченных возможностях первых версий HTML [↪HTML2SPEC]. При введенных последующими спецификациями HTML расширениях эти же принципы были сохранены и при разработке транслятора LATEX2HTML, более того, они были в полной мере использованы и расширены. Теперь все те окружения, которые обычно использовались при подготовке бумажных документов с помощью LATEX'а, автоматически конвертируются в структуры и разметку HTML, обеспечивая практически наивысшее качество результатов, которое может быть достигнуто при конкретной версии HTML.

Главной целью программы конвертации, такой как LATEX2HTML, является, конечно, принцип 1. Принцип 2 реализуется с помощью переменных конфигурации и опций командной строки, которые подробно описаны в Руководстве пользователя LATEX2HTML (Drakos and Moore (1998)). Примерами могут служить переменные `$MAX_SPLIT_DEPTH` и `$MAX_LINK_DEPTH` и опции `-link` и `-split` (см. п. 3.2.3.2 и 3.2.3.3).

Наглядной иллюстрацией принципа 3 (и, конечно, принципа 1) является пример из разд. 3.2.1, в котором показано, как LATEX2HTML автоматически транслирует LATEX'овский исходный файл в связанную гиперссылками паутину HTML-файлов, снабженную средствами навигации.

В настоящей главе содержится много примеров, иллюстрирующих, как LaTeX2HTML справляется с принципом 4, главным образом с представлением сильно структурированной математики. Многие способы, пригодные для отображения математических выражений, с успехом используются и в других ситуациях. Мы увидим, что LaTeX2HTML — это больше, чем просто средство трансляции. Мы также увидим, как информация из одного и того же LaTeX'овского исходного файла может быть разными способами представлена в виде набора HTML-страниц, с использованием пакетов, переменных конфигурации и опций командной строки. Процедура трансляции может быть специальным образом настроена с учетом таких факторов, как версия HTML, возможности конкретных навигаторов, время загрузки документа и связь документа с другими Web-страницами. Для создания документов может быть использован «условный код», обеспечивающий наилучшие результаты как для бумажной версии, набранной в LaTeX'е, так и для HTML-версии, подготовленной с использованием LaTeX2HTML. Эти возможности оправдывают определение LaTeX2HTML как системы подготовки Web-документов.

Еще одной важной областью применения принципа 4 является воспроизведение языков, в которых используются экзотические алфавиты. Для отображения языков Индийского полуострова часто используются схемы транслитерации в таком варианте, который допускает компьютерную реализацию. Существуют различные образцы программ предварительной обработки, которые конвертируют текст, закодированный с помощью схем транслитерации, в документ, который может быть сверстан в TeX'е или LaTeX'е с использованием специальных шрифтов. LaTeX2HTML поддерживает некоторые из этих программ в полностью автоматическом режиме.

Очень важное на сегодняшний день применение LaTeX2HTML, которому изначально Дракос не придавал большого значения, — создание гипертекстовых документов. Поэтому в комплект поставки LaTeX2HTML включен макропакет `html.sty`, который содержит расширения для разметки в форме новых связанных с HTML команд и окружений (подробности и короткие примеры см. в разд. 3.5).

3.2 Необходимое программное обеспечение и настройка

Последнюю версию LaTeX2HTML можно получить на нескольких сайтах, основные из которых находятся в Америке [↪L2HTUG], Германии [↪L2HSC] и в архивах CTAN [↪L2HCTAN]. Версии, находящиеся в стадии разработки, можно найти на сайте разработчиков [↪CVSREPOS]. Имена распространяемых версий имеют вид `latex2html-yy_vv.tar.gz`, где `yy` соответствует году, а `vv` — версии. Например, весной 1999 г. последней версией была `latex2html-99_1.tar.gz`.

В следующем списке перечислено другое программное обеспечение, необходимое для стандартной инсталляции LaTeX2HTML. Оно *не* входит в стан-

дартный пакет поставки (не является частью дистрибутива) и *должно* устанавливаться отдельно.

Perl. Необходима версия 5 (для некоторых платформ — 5.003 или старше). Для LaTeX2HTML необходим также модуль управления базами данных. Все, что относится к языку Perl, включая обновление до последней версии, можно найти на сайте www.perl.com [↪PERL].

LaTeX. В тех случаях, когда возможностей HTML недостаточно для адекватного воспроизведения какой-либо части информации, создается графическое изображение. Первым шагом в процессе создания этого изображения является набор соответствующего фрагмента исходного документа с помощью LaTeX'а.

`dvips` конвертирует LaTeX'овский выходной файл формата DVI в набор PostScript'овских файлов.

`Ghostscript` интерпретирует PostScript'овские файлы, воспроизводя их в виде рисунков с использованием высококачественного растрового формата. Для того чтобы LaTeX2HTML мог создавать рисунки высокого качества со «сглаженными контурами», необходима, как минимум, версия 4.02. Информацию о последней версии можно найти на странице Ghostscript [↪GSHOME].

`netpbm`. Набор графических утилит, каждая из которых предназначена для выполнения какой-либо операции по обработке графического изображения, например: обрезки, придания прозрачности фону, вращения, уменьшения насыщенности, преобразования к конкретному выходному формату. Архив можно получить на сайте X Windows [↪NETPBM] по FTP.

Прочие инструменты. Для специальных целей, таких как обработка транслитераций разновидностей индийского письма, LaTeX2HTML включает поддержку различных препроцессоров.

Благодаря тому, что транслятор LaTeX2HTML написан на скриптовом языке, он является исключительно гибким. Основной принцип его постоянного развития — использование существующих программ для решения специальных задач. Достоинством перечисленных выше программ является то, что их можно получить бесплатно. Другие программы можно использовать, например, для обработки графики.

Довольно просто писать код на Perl, создавая программы, решающие какую-либо насущную задачу лучше, быстрее или просто другим способом. При использовании совместно с LaTeX2HTML дополнительного программного обеспечения требуется решение следующих задач:

1. Распознавание тех частей исходного документа, которые должны быть обработаны с использованием нового программного обеспечения.
2. Извлечение этой информации и преобразование ее к форме, пригодной для ввода в специализированное программное обеспечение.

3. Подготовка соответствующей командной строки для запуска прикладной программы обработки необходимой информации.
4. Включение сформированного выхода в создаваемые HTML-страницы либо в виде размеченного текста, либо в виде гиперссылок на внешние файлы, такие, как графические изображения. При этом может понадобиться некоторая заключительная обработка.

Язык программирования Perl хорошо подходит для решения таких задач. Его развитые средства сопоставления с образцом идеальны для распознавания отдельных форм информации, содержащейся в большем документе.

3.2.1 Работа LaTeX2HTML с LaTeX'овским документом

Для того чтобы с помощью LaTeX2HTML обработать документ, написанный для LaTeX'а, обычно достаточно подать примерно следующую команду. Расширение `.tex` является необязательным.

```
latex2html <имя_файла>[.tex]
```

В одной команде можно задать обработку нескольких файлов, используя опции командной строки для определения места размещения получающихся HTML-страниц и описания гиперссылок между ними:

```
latex2html <опции> файл1 файл2 файл3
```

Для того чтобы убедиться, как просто работать с LaTeX2HTML, рассмотрим пример с файлом `12hexa.tex`, написанным по-французски. LaTeX'овский исходный файл показан на рис. 3.1. В строках 2–4 подключаются три пакета: `graphicx`, `francais` (пакет `babel`) и `makeidx`. Также определяется новая команда `\Lcs`, которая выводит свой аргумент на печать, ставя перед ним обратную косую (строка 5). Далее, после печати заголовка, с помощью команды `\maketitle` размещаются оглавление и списки рисунков и таблиц (строки 10–13). Основная часть текста состоит из двух разделов (строки 14–31 и 33–50). Вслед за ним размещаются библиография (строки 52–58) и алфавитный указатель, который создается программой `makeindex` из различных команд `\index` и подключается с помощью команды `\printindex` (строка 59). В первом разделе есть два расположенных рядом графика (строки 25–26), в то время как во втором разделе показаны в табличной форме несколько команд пакета `francais`.

Если обработать этот исходный файл `12hexa.tex` LaTeX'ом, затем запустить `makeindex` для обработки файла `12hexa.idx`, а в конце второй раз запустить LaTeX для разрешения перекрестных ссылок и подключить файл `12hexa.ind` с алфавитным указателем, то в результате получатся три страницы, показанные в виде PostScript'овских файлов на рис. 3.2.

Для создания HTML-файлов, соответствующих этому LaTeX'овскому исходному файлу, достаточно ввести:

```
latex2html 12hexa
```

```
\documentclass{article}
\usepackage{graphicx}
\usepackage{francais}
\usepackage{makeidx}
\newcommand{\Lcs}[1]{\texttt{\symbol{'134}#1}}
\makeindex
\title{Exemple d'un article en fran\c{c}ais}
\author{Michel Goossens}
\begin{document}
\maketitle
\tableofcontents
\listoffigures
\listoftables
\section{Une figure EPS}
\index{section}
Cette section montre comment inclure une figure PostScript\cite{bib-PS}
dans un document \LaTeX. La figure~\ref{Fpsfig}
est ins\'er\'ee dans le texte \`a l'aide de la commande
 \verb!\includegraphics{colorcir.eps}!.
\index{figure}
\index{PostScript}
\begin{figure}
 \begin{center}
  \begin{tabular}{c@{\qquad}c}
   \includegraphics[width=3cm]{colorcir} &
   \includegraphics[width=3cm]{tac2dim}
  \end{tabular}
 \end{center}
 \caption{Deux images EPS}
 \label{Fpsfig}
\end{figure}

\section{Exemple d'un tableau}

Le tableau~\ref{tab:exa} \`a la page \pageref{tab:exa}
montre l'utilisation de l'environnement \texttt{table}.

\begin{table}
 \begin{center}
  \begin{tabular}{cccccc}
   \Lcs{primo}  & \primo & \Lcs{secundo} & \secundo & \Lcs{tertio} & \tertio \\
   \Lcs{quatro} & \quatro& 1\Lcs{ier}    & 1\ier    & 1\Lcs{iere}  & 1\iere  \\
   \Lcs{fprimo)}&\fprimo)& \Lcs{No} 10   & \No 10   & \Lcs{no} 15  & \no 15  \\
   \Lcs{og} a \Lcs{fg}&\og a \fg&3\Lcs{ieme}&3\ieme & 10\Lcs{iemes}& 10\iemes
  \end{tabular}
  \end{center}
 \caption{Quelques commandes de l'option \texttt{francais} de \texttt{babel}}
 \label{tab:exa}
 \index{tableau}
\end{table}

\begin{thebibliography}{99}
\index{r\'ef\'erences}
\bibitem{bib-PS}
Adobe Inc.
\emph{PostScript, manuel de r\'ef\'erence (2\ieme \'edition)}
Inter\'Editions (France), 1992
\end{thebibliography}
\printindex
\index{index}
\end{document}
```

Рис. 3.1. Пример LaTeX'овского исходного файла, предназначенного для трансляции с помощью LaTeX2HTML.

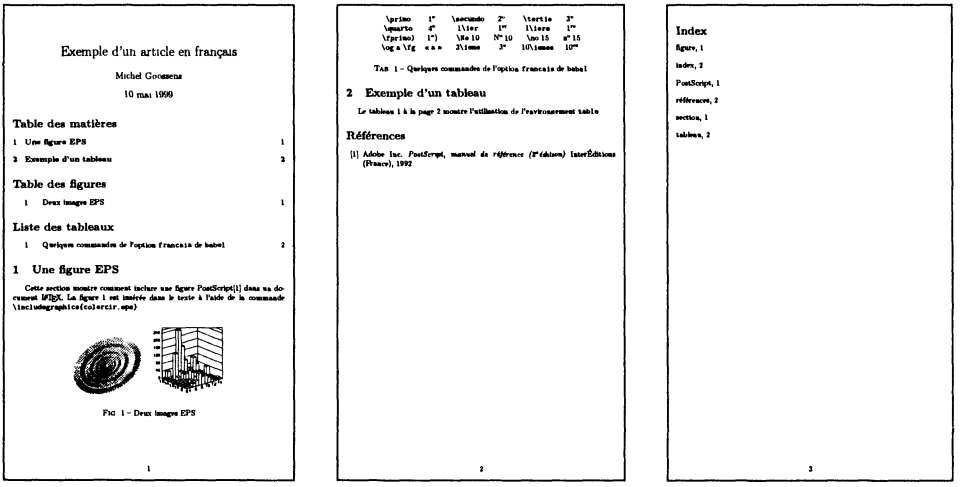


Рис. 3.2. Форматированный PostScript'овский вывод для файла, показанного на рис. 3.1.

Для того чтобы было видно, что происходит, LaTeX2HTML выводит на пользовательскую консоль следующую информацию. В данном случае программа выполняется при стандартных установках системы, а в последующих разделах будет показано, как можно управлять одной или несколькими характеристиками создаваемых выходных файлов.

```
This is LaTeX2HTML Version 99.1 release (March 30, 1999)
by Nikos Drakos, CBLU, University of Leeds.

Revised and extended by:
 Marcus Hennecke, Ross Moore, Herb Swan and others
...producing markup for HTML version 3.2

Loading /afs/.cern.ch/asis/src/TeX/archive/latex2html/versions/html3_2.pl

 *** processing declarations ***

OPENING /afs/cern.ch/project/tex/lwc/babel/l2hexa.tex

Loading /afs/.cern.ch/asis/src/TeX/archive/latex2html/styles/texdefs.perl...
Loading /afs/.cern.ch/asis/src/TeX/archive/latex2html/styles/article.perl
Loading /afs/.cern.ch/asis/src/TeX/archive/latex2html/styles/graphicx.perl
Loading /afs/.cern.ch/asis/src/TeX/archive/latex2html/styles/francais.perl
Loading /afs/.cern.ch/asis/src/TeX/archive/latex2html/styles/makeidx.perl........
Reading ...
%,++............
@@@@@@@@@@@@@@@@@@@@@@@@@
Reading /afs/cern.ch/project/tex/lwc/babel/l2hexa.aux ...
Processing macros ...++.........................
Reading /afs/cern.ch/project/tex/lwc/babel/l2hexa.lof ...
Processing macros ...++....
Reading /afs/cern.ch/project/tex/lwc/babel/l2hexa.lot ...
Processing macros ...++......
Translating ...
0/8:top of l2hexa: for l2hexa.html

 *** translating preamble ***
```

```
33  .........
34   *** preamble done ***
35  ;;.
36
37  1/8:tableofcontents:.."Table des mati\'{e}res" for node1.html
38  ;;.
39
40  2/8:listoffigures:.."Liste des figures" for node2.html
41  ;;.
42
43  3/8:listoftables:.."Liste des tableaux" for node3.html
44  ;;.
45
46  4/8:section:..."Une figure EPS" for node4.html
47  ;,,,,,....;........
48
49  5/8:section:.."Exemple d'un tableau" for node5.html
50  ;.,,,.....................................................................;....
51
52  6/8:bibliography:.."R\'{e}f\'{e}rences" for node6.html
53  ;,..........;
54
55  7/8:textohtmlindex:..."Index" for node7.html
56  ;;
57
58  8/8:sectionstar:.."\'{A} propos de ce document..." for node8.html
59  ;;.
60
61  Writing image file ...
62
63  This is TeX, Version 3.14159 (Web2C 7.2)
64  (./images.tex
65  LaTeX2e <1997/12/01> patch level 2
66  Babel <v3.6h> and hyphenation patterns for english, french, german, dumylang,
67  nohyphenation, loaded.
68
69   *** processing 2 images ***
70
71  Generating postscript images using dvips ...
72
73  images will be generated in l2h45774/
74
75  This is dvips(k) 5.78 Copyright 1998 Radical Eye Software (www.radicaleye.com)
76  ' TeX output 1998.09.07:1521' -> l2h45774/image
77  (-> l2h45774/image001) <texc.pro><special.pro><color.pro>[1<colorcir.eps>]
78  (-> l2h45774/image002) <texc.pro><special.pro><color.pro>[2<tac2dim.eps>]
79  Writing img1.gif
80  Writing img2.gif
81
82  Doing section links .....
83  Doing table of contents ..........
84  Doing the index ....
```

По умолчанию LaTeX2HTML будет создавать HTML-файлы, соответствующие спецификации HTML 3.2. После того как в строке 6 сказано самое главное, в строке 8 загружается соответствующий Perl-файл. В этом месте читается входной файл `l2hexa.tex` (строка 12) и вводятся необходимые для его анализа стилевые Perl-файлы (строки 14–18). Отметим, в частности, что для каждого пакета, используемого в исходном файле, имеется соответствующая Perl-программа (строки 16–18). Затем LaTeX2HTML читает вспомогательные LaTeX'овские файлы (строки 23, 25 и 27) и в строке 29 переходит к шагу трансляции. По умолчанию основной HTML-файл будет иметь то же самое имя, что и исходный входной файл, но с расширением `.html`. В рассматриваемом случае мы получим `l2hexa.html`. Теперь LaTeX2HTML будет создавать файлы

для каждого из подразделов исходного LaTeX'овского документа (делением на уровни управляет пользователь). Таким образом (строки 37–59), мы получим восемь HTML-файлов, от `node1.html` до `node8.html`. Из-за того, что HTML-навигаторы не могут непосредственно работать с EPS-файлами (строки 25–26 на рис. 3.1), LaTeX2HTML конвертирует эти образы в растровые изображения формата GIF (или PNG). Поэтому LaTeX2HTML копирует весь необходимый LaTeX'овский код в файл `images.tex` (строка 61), который сначала компилируется LaTeX'ом (строки 63–67), а затем с помощью программы `dvips` разрезается на одностраничные EPS-файлы (строки 75–78). Эти файлы обрабатываются утилитами `Ghostscript` и `netpbm` (как объяснено в разд. 3.2) и преобразуются в изображения формата GIF (PNG) с именами `img1.gif` и `img2.gif` (строки 79–80). И наконец, LaTeX2HTML обновляет все гиперссылки между различными компонентами документа (строки 82–84).

Результат трансляции помещается в подкаталог `l2hexa`, оглавление которого показано ниже. Там мы прежде всего видим описанные ранее HTML- и GIF-файлы. Файл `l2hexa.css` является стилем CSS, который предоставляет некоторые элементарные средства для настройки способа отображения различных элементов языка HTML (см. разд. 3.3.4). Мы также видим файл `images.tex`, который содержит LaTeX'овский код, использованный для создания изображений. (Он может оказаться очень полезным, если при создании изображений возникают проблемы. В частности, файл `images.log` будет содержать LaTeX'овские сообщения об ошибках.) В последней строке показаны вспомогательные Perl-файлы, которые связывают информацию в LaTeX'овских исходных файлах и в созданных HTML-файлах:

```
l2hexa.css   l2hexa.html
node1.html   node2.html   node3.html   node4.html
node5.html   node6.html   node7.html   node8.html
img1.gif     img2.gif
images.tex   images.aux   images.idx   images.log
images.pl    internals.pl labels.pl
```

На рис. 3.3 показано, как связаны различные файлы. В центре расположен головной файл `l2hexa.html`. Из него с помощью гиперссылок можно связаться со всеми остальными файлами:

❶ ссылки на оглавление (*Tables des matières*);

❷ ссылки на список рисунков (*Liste des figures*);

❸ ссылки на список таблиц (*Liste des tableaux*);

❹ ссылки на первый раздел (*Une figure EPS*);

❺ ссылки на второй раздел (*Example d'un tableau*);

❻ ссылки на библиографию (*Références*);

❼ ссылки на алфавитный указатель (*Index*) и

❽ ссылки на информацию о способе создания документа (*À propos de ce document...*).

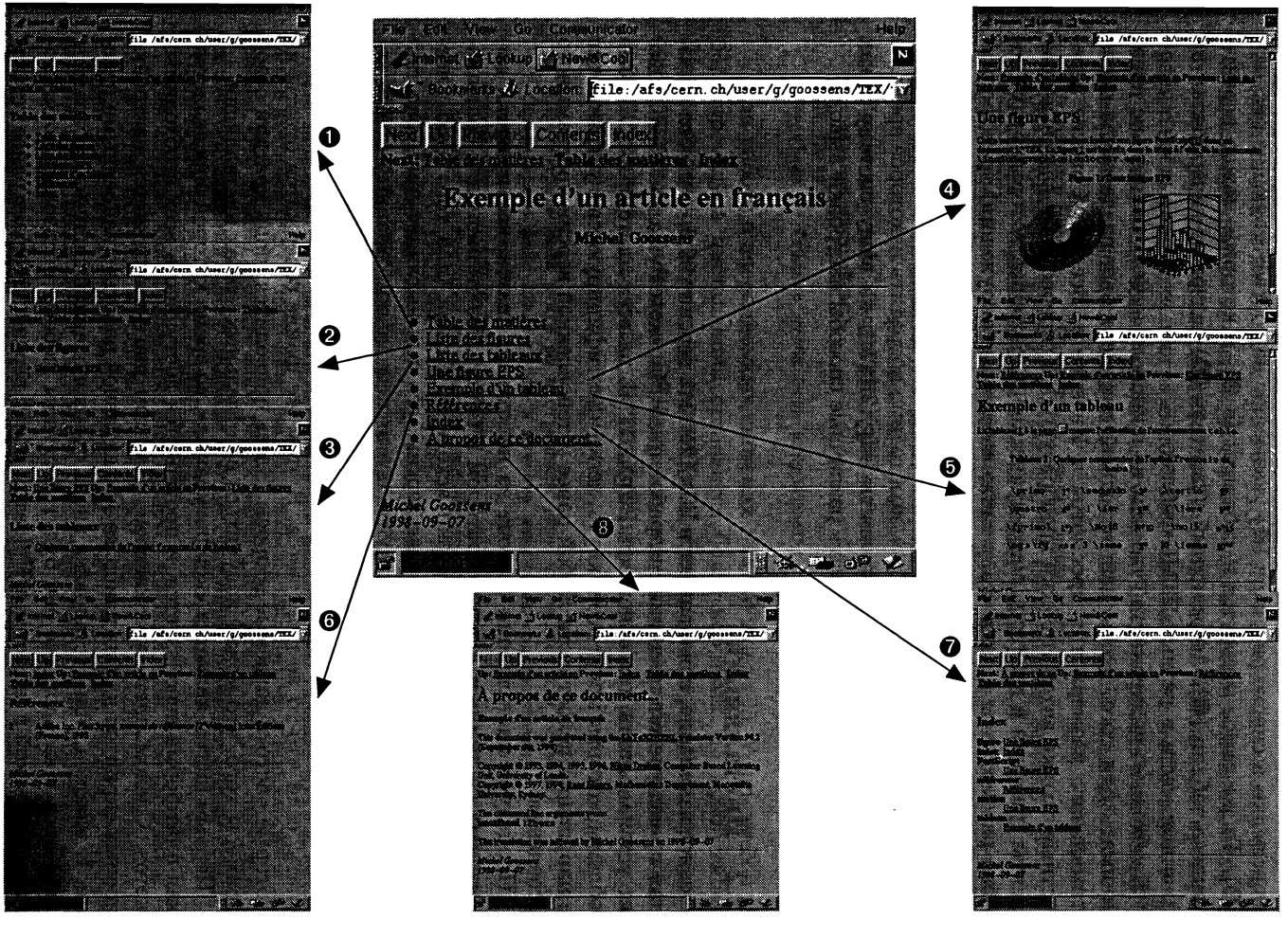


Рис. 3.3. HTML структура, созданная LaTeX2HTML из LaTeX'овского исходного файла на рис. 3.1.

Гиперссылки в списке рисунков (❷) указывают на соответствующие рисунки (в ❹). То же самое относится к таблицам (в ❺), к которым можно перейти из списка таблиц (❸). Алфавитный указатель (❼) содержит гиперссылки на различные места в разных HTML-файлах.

Необходимо подчеркнуть, что нам не пришлось ничего специально предпринимать для того, чтобы наш исходный документ был разделен на части оптимального размера для просмотра в Интернете. Все LaTeX'овские перекрестные ссылки автоматически преобразуются в гиперссылки; сложные компоненты (рисунки и графики, математические выражения, которые будут подробно рассмотрены в разд. 3.3, а также некоторые окружения) трансформируются в растровые изображения. Рассматривается даже трансляция компонентов в случае неанглийских документов (в нашем случае — французского).

Прежде чем двигаться дальше, необходимо потратить некоторое время на то, чтобы посмотреть, как установить LaTeX2HTML на вашу систему и как наилучшим способом использовать различные способы настройки вывода.

3.2.2 Инсталляция

Основное изменение, появившееся в LaTeX2HTML версии v99.2, — это более развитая процедура инсталляции, разработанная Мареком Рушалем. Она позволяет устанавливать или обновлять LaTeX2HTML на всех платформах. Любые проблемы, которые могут возникнуть, легко идентифицируются и устраняются без нарушения целостности любой ранее установленной версии программного обеспечения.

Процедура инсталляции состоит из нескольких шагов, как правило, выполняемых системным администратором в месте, недоступном для большинства пользователей. После загрузки исходного архива (для того чтобы четко указать содержащуюся в архиве версию, его имя выглядит примерно так: `latex2html-2K.1.tar.gz`) распакуйте его в специально созданный новый каталог. Четкое описание процедуры инсталляции приведено в файле с именем `INSTALL`. Там подробно объясняется каждый шаг и даются советы, как поступать, если что-то не получается. В этом же файле содержатся ответы на некоторые часто задаваемые вопросы. Здесь приводится лишь краткий обзор и примеры того, что может получиться на отдельных этапах инсталляции.

Первым шагом является «конфигурация», во время которой определяется, какая платформа используется, и проверяется доступность всех необходимых пакетов. Следом идет «компоновка». На этом шаге выполняется настройка основных скриптов для их наиболее эффективной работы с найденными пакетами. И наконец, при «установке» вновь скомпонованной системы программы копируются в каталог, где они будут доступны любому пользователю многопользовательской платформы. При этом новая версия замещает ранее установленную.

В промежутке между этими шагами можно проверить полученные результаты и исправить ошибки, которые, может быть, были допущены. Для пред-

отвращения таких же ошибок при последующих обновлениях можно записать в специальные файлы значения, которые должны использоваться вместо тех, которые определяются автоматически. Для того чтобы можно было проверить правильность установки программного обеспечения на локальном сайте, в комплект поставки включены тестовые документы. Так, при работе в системе UNIX рекомендуется выполнить следующую последовательность команд:

```
./configure --help      (посмотрите, какие опции доступны)
 (Просмотрите файл prefs.pm и отредактируйте его надлежащим образом)
./configure    (Просмотрите протокол на предмет ошибок;
                    отредактируйте файл cfgcache.pm)
make
make check     (на всякий случай!)
make test      (Проанализируйте результаты, сделайте
                    дополнительную проверку.)
 (Просмотрите файл l2hconf.pin и внесите изменения соответственно
  локальным настройкам)
make install
make clean     (только чтобы быть аккуратным!)
```

При работе в операционных системах DOS, Windows 95, Windows NT или OS/2 процедура инсталляции становится более короткой, но менее гибкой. Шаги конфигурирования и компоновки объединяются в одной программе `CONFIGURE.BAT`. (В системе OS/2 прежде всего следует заменить это имя на `CONFIGURE.CMD`, и подобным же образом переименовать остальные исполнимые файлы.)

```
 (Просмотрите файл prefs.pm и отредактируйте его надлежащим образом)
 (Просмотрите файл l2hconf.pin и внесите изменения соответственно
  локальным настройкам)
CONFIG.BAT     (CONFIG.CMD в системе OS/2)
 (Просмотрите протокол на предмет ошибок, внесите изменения в cfgcache.pm)
TEST.BAT       (TEST.CMD в системе OS/2)
 (Проанализируйте результаты, сделайте дополнительную проверку)
INSTALL.BAT    (INSTALL.CMD в системе OS/2)
 (удалите временные файлы, чтобы быть аккуратным!)
```

На заключительном шаге инсталляции необходимо указать пути к каталогам, в которых будут постоянно располагаться исполнимые и вспомогательные файлы. Это можно сделать *либо* редактируя файл `prefs.pm` (для определения их местоположения), *либо* задавая соответствующие значения опций команды `./configure` (или команды `CONFIG.BAT`).

Посмотрим, как на практике выполняется инсталляция. Предположим, что дистрибутив получен из репозитория и все файлы LaTeX2HTML скопированы в дерево каталогов; после этого перейдем к вершине этого дерева. Структура файлов и каталогов будет выглядеть примерно следующим образом:

```
BUGS                IndicTeX-HTML/      MANIFEST            XyMTeX-HTML/
cweb2html/          psfiles/            Changes             L2hos/
Makefile.in         config/             docs/               dot.latex2html-init
FAQ                 L2hos.pm            README              config.bat
changebar/          INSTALL             LICENSE             TODO
configure           hthtml/             example/
```

```
foilhtml/          icons/          l2hconf.pin     latex2html.pin
makemap            makeseg/        prefs.pm        pstoimg.pin
readme.hthtml      styles/         tests/          texexpand.pin
texinputs/         texlive.pm      versions/       wrapper/
```

В последующих разделах мы более подробно рассмотрим содержимое большинства этих каталогов. Однако сейчас продолжим инсталляцию LaTeX2HTML и выполним следующую команду, не редактируя файл `prefs.pm`:

```
./configure --prefix=/home/ross/local/share --enable-gif
```

Среди прочих будут выведены следующие строки:

```
checking for /usr/bin/tex... /usr/bin/tex
checking for /usr/bin/latex... /usr/bin/latex
checking for /usr/bin/initex... /usr/bin/initex
checking for /usr/bin/kpsewhich... /usr/bin/kpsewhich
checking for kpsewhich syntax... ok (style=2)
checking for TeX include path... /usr/lib/texmf/texmf/tex/latex/latex2html
checking for /usr/bin/dvips... /usr/bin/dvips
checking dvips version... 5.58f
checking if dvips supports the combination of -E and -i -S 1... no
checking for html4-check... no

Note: Will install...
      ... executables to   : /home/ross/share/local/bin
      ... library items to : /home/ross/share/local/lib/latex2html
```

Обратите внимание, что исполнимые и библиотечные файлы будут устанавливаться там, где определено опцией `--prefix=...`. Чтобы не запоминать это, в соответствующей строке файла запишите этот каталог:

```
$prefs{'PREFIX'} = '/home/ross/local/share';
```

Хотя пути к программам `tex` и `latex` выглядят вполне приемлемо, на самом деле они указывают на устаревшую версию с неподходящей программой `dvips`. После настройки переменной path и выполнения команды `rm cfgcache.pm` для удаления всех неправильных значений повторное выполнение `./configure` формирует пути к новым исполнимым файлам:

```
checking for tex... /usr/local/texlive4/bin/i386-linux/tex
checking for latex... /usr/local/texlive4/bin/i386-linux/latex
checking for initex... /usr/local/texlive4/bin/i386-linux/initex
checking for kpsewhich... /usr/local/texlive4/bin/i386-linux/kpsewhich
checking for kpsewhich syntax... ok (style=1)
checking for TeX include path... /home/ross/texmf/tex/latex/latex2html
checking for dvips... /usr/local/texlive4/bin/i386-linux/dvips
checking dvips version... 5.85
checking if dvips supports the combination of -E and -i -S 1... yes
checking for html4-check... /home/ross/bin/html4-check
```

Для того чтобы в будущем избежать повторения этой ошибки, скопируйте правильные пути непосредственно в файл `prefs.pm`. Обратите внимание, что упомянутые здесь компоненты «TeX include path» и «html4-check» могут быть недоступны для некоторых пользователей. Поэтому при инсталляции для общего пользования может понадобиться найти им замену и сделать соответствующую запись в файл `prefs.pm`. Для удаления старых значений перед каждым новым запуском `./configure` выполняйте `rm cfgcache.pm`.

Здесь приведена часть файла `cfgcache.pm`, являющегося результатом выполнения `./configure` на другом сайте, предназначенном в данном случае для общего пользования.

```
1  # LaTeX2HTML site specific configuration file
2  # generated by config.pl
3
4  # You may edit this file to get around deficiencies of the configuration
5  # procedure, but you have to be sure of what you are doing!
6  # If you think there are bugs in the configuration procedure, please report
7  # them. See the BUGS file on how to do it. Your help is appreciated!
8
9  package cfgcache;
10 require Exporter;
11 @ISA = qw(Exporter);
12 @EXPORT = qw(%cfg);
13
14 $cfg{'BINDIR'} = q'/usr/local/bin';
15 $cfg{'CRAYOLAFILE'} = q'/usr/local/lib/latex2html/styles/crayola.txt';
16 $cfg{'DVIPS'} = q'/usr/local/bin/dvips';
17 $cfg{'DVIPSOPT'} = q' -E';
18 $cfg{'GS'} = q'/usr/local/bin/gs';
19 $cfg{'GSALIASDEVICE'} = q'ppmraw';
20 $cfg{'GSDEVICE'} = q'pnmraw';
21 $cfg{'GSLANDSCAPE'} = q'';
22 $cfg{'GS_LIB'} = q'';
23 $cfg{'HASHBANG'} = q'1';
24 $cfg{'HTML_VALIDATOR'} = q'/usr/local/bin/html4-check';
25 $cfg{'ICONPATH'} = q'http://www-texdev.mpce.mq.edu.au/images';
26 $cfg{'ICONSTORAGE'} = q'/services/www/texdev/images';
27 $cfg{'IMAGE_TYPES'} = q'gif';
28 $cfg{'INITEX'} = q'/usr/local/bin/initex';
29 $cfg{'KPSEWHICH'} = q'/usr/local/bin/kpsewhich';
30 $cfg{'LATEX'} = q'/usr/local/bin/hugelatex';
31 $cfg{'LATEX2HTMLDIR'} = q'/usr/local/lib/latex2html';
32 $cfg{'LIBDIR'} = q'/usr/local/lib/latex2html';
   ...
54 $cfg{'TEXPATH'} = q'/usr/local/share/texmf/tex/latex/html';
55 $cfg{'TMPSPACE'} = q'/centre/tmp';
56 $cfg{'WEB2C'} = q'1';
57 $cfg{'dd'} = q'/';
58 $cfg{'distver'} = q'99.2beta5';
59 $cfg{'exec_extension'} = q'';
60 $cfg{'gif_interlace'} = q'netpbm';
61 $cfg{'gif_trans'} = q'netpbm';
62 $cfg{'have_dvipsmode'} = q'';
63 $cfg{'have_geometry'} = q'1';
64 $cfg{'have_images'} = q'1';
65 $cfg{'have_pstoimg'} = q'1';
66 $cfg{'perl_starter'} = q'';
67 $cfg{'pipes'} = q'1';
68 $cfg{'plat'} = q'unix';
69 $cfg{'scriptdir'} = q'/usr/local/bin';
70 $cfg{'scriptext'} = q'';
71 $cfg{'srcdir'} = q'/centre/l2h/latex2ht/latex2html/user';
72 $cfg{'texlive'} = q'0';
73 $cfg{'wrapper'} = q'0';
74
75 1; # must be last line
```

Нестандартные значения здесь присутствуют только в строках 25–26, 30, 54 и 55. Они обусловлены следующими входными данными в `prefs.pm`:

```
$prefs{'ICONPATH'} = 'http://www-texdev.mpce.mq.edu.au/images';
$prefs{'ICONSTORAGE'} = '/services/www/texdev/images';
```

```
$prefs{'LATEX'} = 'hugelatex';
$prefs{'TEXPATH'} = '/usr/local/share/texmf/tex/latex/html';
$prefs{'TMPSPACE'} = '/centre/tmp';
```

Значение переменной ICONPATH определяет сетевой адрес, где находятся изображения иконок, а ICONSTORAGE задает локальный адрес этого же каталога. Обратите внимание, что этот каталог *должен* находиться в иерархии, доступной для локального WWW-сервера; для того чтобы скопировать иконки в нужное место, могут потребоваться специальные привилегии доступа.

Для того чтобы успешно преобразовать объемистые LaTeX'овские первоисточники в изображения, удобно использовать hugelatex. LaTeX'овские файлы стилей помещаются в каталог TEXPATH, тогда как используемые во время выполнения временные файлы размещаются на другом диске в каталоге TMPSPACE.

На шаге «компоновки», выполняемом с помощью команды make, путем вставки записанных в cfgcache.pm путей из latex2html.pin создается Perl-скрипт latex2html. Самое важное — это определить, где следует искать программные модули, которые выполняют почти всю трансляцию LaTeX'овской разметки. В следующей строке файла latex2html.pin

```
$ENV{'LATEX2HTMLDIR'} = $LATEX2HTMLDIR = '@LATEX2HTMLDIR@';
```

строка @LATEX2HTMLDIR@ заменяется на полный путь к каталогу, заданному в cfgcache.pm с помощью $cfg{'LATEX2HTMLDIR'}. В тексте этого скрипта необходимы и другие подстановки, некоторые зависят от платформы. Аналогичным образом из texexpand.pin и pstoimg.pin путем вставки записанных в cfgcache.pm значений создаются скрипты texexpand и pstoimg.

Кроме того, для сохранения пользовательских настроек, которые могут воздействовать на все создаваемые на локальном сайте Web-документы, из l2hconf.pin создается l2hconf.pm. Предполагается, что l2hconf.pin должен редактироваться в соответствии с местными условиями для присваивания значений переменным, которые будут доступны во время выполнения. В нем подробно описывается оказываемое этими переменными воздействие, а также приводятся рекомендуемые для использования значения. Как и в других программах, многие установки вставляются из cfgcache.pm, но в l2hconf.pm автоматически созданные значения могут быть отменены или заменены другими. Во время выполнения читается только l2hconf.pm.

Для иллюстрации некоторых аспектов пользовательской настройки ниже приведен фрагмент файла l2hconf.pin, имеющего минимальные отличия от стандартного варианта, содержащегося в дистрибутиве LaTeX2HTML (файл cfgcache.pm для рассматриваемой процедуры инсталляции был приведен выше).

```
1  $HTML_VALIDATE = 1;
2  $NUMBERED_FOOTNOTES = 1;
3  $DVIPSOPT .= " -Pgs";
4  #$DVIPS_MODE = '@METAMODE@';
```

Нумерация сносок выполняется по умолчанию, а все созданные HTML-страницы автоматически проверяются на соответствие стандартному определению типа документа (DTD). (Это может быть выполнено только тогда, когда установлено

значение переменной `$HTML_VALIDATOR` — либо непосредственно, либо путем вставки `'@HTML_VALIDATOR@'` — для того чтобы указать, какая программа должна быть запущена для выполнения проверки.)

Дополнительная опция `-Pgs` в `$DVIPSOPT` определяет, что программа `dvips` должна использовать файл конфигурации с именем `config.gs`. Это, в свою очередь, приводит к тому, что из файла `psfonts.gs` будет прочитан список шрифтов, которые *не* должны встречаться в выводе. Вместо этого выполняется конфигурация `Ghostscript` для поиска `.pfb` или `.pfa` версий всех используемых в LaTeX'е шрифтов. При этой стратегии никогда не потребуется `$DVIPS_MODE`. Для выполнения соответствующей конфигурации как `dvips`, так и `Ghostscript` совершенно необходимо хорошо знать все доступные в локальной системе шрифты.

Перед запуском исполнимых файлов `make install` или `INSTALL` на не-UNIX платформах может оказаться полезным протестировать документы, найденные в подкаталоге `tests/`. Таким образом можно обнаружить допущенные при конфигурации ошибки, которые необходимо исправить. Наиболее распространенные причины, по которым LaTeX2HTML не может правильно создать HTML-страницы из LaTeX'овских исходных документов, могут быть обусловлены ошибками, допущенными при локальной конфигурации. Поэтому, затратив немного времени на проверку, можно избавиться от головной боли в дальнейшем.

Когда убедитесь, что все выглядит хорошо, запустите `make install`. Как `cfgcache.pm`, так и `l2hconf.pm` устанавливаются вместе с другими вспомогательными файлами, которые использует LaTeX2HTML. В процессе конфигурации автоматически читается любой файл `l2hconf.pm`, обнаруженный в соответствующем месте. Поэтому можно очень легко выполнить обновление до новой версии, и при этом не требуется переустанавливать параметры конфигурации. Поэтому можно безопасно выполнить `make clean` или полностью удалить используемый при инсталляции временный каталог. Однако по вашему желанию в качестве дополнительной меры предосторожности можно скопировать и/или переименовать файлы `prefs.pm` и `l2hconf.pin`.

При выполнении только что описанной процедуры инсталляции специальные пакеты Indic-TeX/HTML (для рукописных шрифтов индийских языков) и XeTeX (для химических схем) автоматически *не* устанавливаются. Поэтому если вы хотите их использовать или иметь к ним доступ, то перед выполнением `make install` (или программы `INSTALL` для системы, отличной от UNIX) скопируйте `.perl` файлы из соответствующих каталогов в подкаталог `styles/`. Для пакета Indic-TeX/HTML необходимо также установить в соответствующем месте TeX'овской системы LaTeX'овские `sty`-файлы.

В пакетах `cweb2html` и `foilhtml` также есть LaTeX'овские компоненты, которые необходимо установить, если вы хотите использовать эти пакеты или сделать их доступными на вашем сайте. Подобным образом справочную информацию об установке `hthtml` можно найти в файлах, содержащихся в каталоге `docs/hthtml`.

3.2.3 Настройка локальной инсталляции

Существуют десятки переменных конфигурации, которые так или иначе влияют на способ создания HTML-страниц с помощью LaTeX2HTML. Для сохранения настроек, которые лучше всего подходят для конкретного сайта, отдельного автора или группы авторов или для особого вида документов, LaTeX2HTML предоставляет иерархическую систему механизмов настройки.

Эти механизмы заключаются в чтении файлов, содержащих Perl-код для установки переменных и определения или переопределения подпрограмм, которые будут использоваться на последующих этапах обработки. В приведенном ниже списке эти файлы перечислены в том порядке, в котором они обрабатываются головным скриптом `latex2html`. Perl-код, содержащийся во вновь прочитанных файлах, может переопределить значение переменной, заданное в ранее загруженном файле.

3.2.3.1 Файл `l2hconf.pin`

В предыдущих версиях LaTeX2HTML для управления всеми зависящими от сайта настройками, относящимися к процедуре инсталляции или HTML-выводу, использовался файл с именем `latex2html.config`. Сейчас вместо него используется `l2hconf.pm`, который создается из `l2hconf.pin` и `prefs.pm`. Оба этих файла можно редактировать для соответствия местным требованиям.

Переустановить программное обеспечение после таких изменений очень просто. Повторно выполнять программу `./configure` необходимо только в тех случаях, когда был изменен файл `prefs.pm`. А вот перезапускать `make install`, которая будет автоматически вызывать `make`, нужно *всегда*. (На не-UNIX платформах перезапустите обе программы: `CONFIGURE` и `INSTALL`.)

3.2.3.2 Файл `.latex2html-init`

Для управления специфическими настройками пользователя применяется файл `$HOME/.latex2html-init`. Как правило, он постоянно располагается в домашнем каталоге пользователя (отсюда использование переменной `$HOME`).

> Этот файл обычно применяется для задания подписи (сигнатуры) с помощью переменной `$ADDRESS` или для использования альтернативного набора навигационных иконок посредством `$ICONSERVER`. Переменную `$TEXINPUTS` можно задать так, чтобы обеспечить возможность поиска отобранных пользователем исходных документов. Аналогично переменные `$LATEX2HTMLSTYLES` и `$LATEX2HTMLVERSIONS` могут быть настроены на поиск каталогов, содержащих нужные пользователю Perl-программы для выполнения дополнительных пакетов или расширений или для замещения ранее поставленных в комплекте LaTeX2HTML. Установка `$INFO="";` запрещает создание информационной страницы, которая в противном случае создавалась бы автоматически. Значения коэффициентов масштабирования `$FIGURE_SCALE_FACTOR`, `$MATH_SCALE_FACTOR` и `$DISP_SCALE_FACTOR`, применяемых при создании рисунков, на этом этапе могут быть изменены путем задания соответствующих величин в файле `l2hconf.pin`.

Файл `.latex2html-init` можно разместить также и в текущем каталоге. Это дает возможность определять переменные, которые будут использоваться всеми заданиями, запускаемыми из данного каталога.

Помимо переопределения остальных настроечных параметров, этот файл обычно используется для задания «степени детализации» при разбиении выходного документа на HTML-страницы по уровням секционирования. Для этого используется переменная `$MAX_LINK_DEPTH`. Переменная `$MAX_SPLIT_DEPTH` оказывает влияние на создание навигационных ссылок между страницами, в то время как `$TOC_DEPTH` определяет, на заголовки каких разделов будут сделаны гиперссылки со страницы с оглавлением. На этом этапе можно также определить требуемый уровень HTML и применяемый способ трансляции математических выражений.

3.2.3.3 Опции командной строки

Многие из упомянутых выше переменных, а также многие другие для отдельного задания могут быть определены с помощью опций командной строки (см. примеры, приведенные ниже).

Например, `-split 4` оказывает такое же действие, как и установка `$MAX_SPLIT_DEPTH=4;`, когда разбиение документа выполняется на уровне команд `\section`, а не команд `\subsection` . Аналогично, опция `-link 2` и установка `$MAX_LINK_DEPTH=2;` оказывают одинаковое воздействие на гиперссылки на заголовки более чем второго уровня вложенности, расположенные на той же самой или других HTML-страницах.

С помощью `-init_file<`*`init-file`*`>` можно загрузить дополнительные файлы конфигурации. Опции командной строки обрабатываются в порядке их появления, в том числе полностью читаются все встречающиеся файлы `<`*`init-file`*`>`. Более подробную информацию о доступных опциях, разъяснение способов их использования и сведения о соответствии между ними и переменными конфигурации можно найти в Руководстве пользователя LaTeX2HTML.

3.2.3.4 Perl-пакеты и файл `<jobname>.perl`

При обращении к какому-либо LaTeX'овскому пакету с помощью команды `\usepackage{`*`<package>`*`}`, соответствующая Perl-программа загружается из файла `<`*`package`*`>.perl` при условии, что его удается найти в рабочем каталоге или в одном из каталогов, перечисленных в переменной `$LATEX2HTMLSTYLES`. Кроме того, если для конкретного задания существует файл `<`*`jobname`*`>.perl`, то он будет загружен. Этот механизм обработки расширений объясняется ниже.

3.2.4 Механизмы расширений и LaTeX'овские пакеты

Результатом работы LaTeX2HTML можно управлять с помощью трех независимых механизмов загрузки файлов, содержащих Perl-программу. Механизмы «класс документа» и «загрузка пакета» аналогичны соответствующим способам, использующимся в LaTeX'е. В первом из них для загрузки конкретных

файлов в команду \documentclass передаются основной параметр и любые опции класса, а во втором — используется команда \usepackage с любыми опциями.

Механизм расширения имеет дело с такими элементами оттранслированного документа, которые могут отсутствовать в исходном LaTeX'овском документе, но существенны для создаваемого HTML-документа. Например, расширения управляют создаваемой версией HTML, используемой кодировкой шрифта, а также способом визуализации математических выражений — в виде графических образов или путем разбиения на маленькие фрагменты.

3.2.4.1 Классы документов и опции класса

Если в исходном тексте документа присутствует строка \documentclass {<*class*>}, то будет загружен файл <*class*>.perl, если он будет найден в текущем каталоге или в одном из каталогов, перечисленных в переменной $LATEX2HTMLSTYLES.

> Отметим, что команда \documentclass или \documentstyle не является строго обязательной для того, чтобы LaTeX2HTML воспринимал *любой* текстовый файл как первоисточник для преобразования в HTML. По возможности подстановка всех макро будет выполняться так, как если бы файл был исходным LaTeX'овским документом.

Если в команде указаны опции класса, как, например, \documentclass [<*option*>]{<*class*>}, то каждый класс может вызвать загрузку файла <*option*>.perl, если файл с таким именем может быть найден в просматриваемых каталогах. Кроме того, выполняется подпрограмма do_<*class*>_<*option*>, если она была определена в загруженных пакетах или в файлах инициализации.

> Например, в файле article.perl есть определение sub do_article_leqno, которое устанавливает флаг, указывающий на то, что номера выключных математических формул должны располагаться слева.

Команду \documentstyle документов, которые были подготовлены в LaTeX 2.09, LaTeX2HTML воспринимает как \documentclass. Таким образом, поддержка стиля и любого из названных пакетов обеспечивается с помощью объясненного ранее механизма.

> В случае стандартных классов документов в LaTeX'е обеспечивается верстка материала первоисточника на странице выходного документа. Однако при создании HTML-документов эти процедуры не нужны; единственное, что обычно выполняется в файлах article.perl, report.perl и book.perl, — это создание структуры для нумерации разделов, когда это требуется. С помощью amsart.perl и amsbook.perl определяются дополнительные команды для использования на титульной странице; поддержка математических пакетов $\mathcal{AMS}$ загружается автоматически.

3.2.4.2 Пакеты и их опции

В тех вариантах LaTeX'а, в которых содержится команда \usepackage {<*package*>}, будет загружаться файл <*package*>.perl, если он находится в текущем каталоге или в одном из каталогов, перечисленных в переменной $LATEX2HTMLSTYLES.

Таким способом поддерживается как большинство стандартных LaTeX'овских пакетов, так и многие поддерживаемые LaTeX'ом пакеты. Соответствующие .perl-файлы содержатся в подкаталоге $LATEX2HTMLDIR/styles/, имя которого является умалчиваемым значением переменной $LATEX2HTMLSTYLES. В других случаях список можно посмотреть в Руководстве пользователя LaTeX2HTML.

Для обработки опций пакета, как, например, в команде \usepackage [<*option*>]{<*package*>}, после загрузки скрипта пакета выполняется Perl-подпрограмма с именем do_<*package*>_<*option*>. Если такая подпрограмма не определена, то дважды выводится предупреждение: немедленно и по завершении задания.

Естественно, эта подпрограмма может вызвать загрузку дополнительных perl-файлов. Например, do_babel_french вызывает загрузку french.perl для использования ключевых слов и заголовков на французском языке. Аналогично, do_inputenc_latin2 загружает поддержку кодировки ISO-8859-2 с использованием файла $LATEX2HTMLDIR/versions/latin2.pl.

3.2.4.3 Расширения и версии HTML

Аргумент опции командной строки -html_version определяет дополнительные Perl-файлы, которые должны быть загружены для текущего задания. Как видно из его названия, этот переключатель определяет версию HTML, которую необходимо сгенерировать. Последующие имена файлов, добавляемые через запятую в список, указывают дополнительные файлы, которые должны быть загружены из каталогов, перечисленных в переменной $LATEX2HTMLVERSIONS.

Например, -html_version 4.0,math,unicode,frame заставляет LaTeX2HTML прочитать Perl-файлы html4_0.pl, math.pl и unicode.pl. По умолчанию эти файлы располагаются в подкаталоге $LATEX2HTMLDIR/versions/, но их местоположение можно изменить с помощью переменной $LATEX2HTMLVERSIONS. Основными доступными версиями являются: frame, html3_2, html4_0, lang, от latin1 до latin9, math, unicode и utf8.

3.3 Математические режимы в LaTeX2HTML

LaTeX2HTML предоставляет богатый набор методов отображения математических выражений в Web-страницах. В приведенных примерах используется файл sampleMath.tex, текст которого см. ниже:

```
1  \documentclass[a4paper,twoside]{article}
2  \usepackage{html}
```

```
3  %\usepackage{amsmath}
4
5  \renewcommand{\d}{\partial}\providecommand{\bm}[1]{\mathbf{#1}}
6  \providecommand{\Range}{\mathcal{R}}\providecommand{\Ker}{\mathcal{N}}
7  \providecommand{\Quat}{\vec{\mathbf{Q}}}
8
9  \newcommand{\StAndrews}{\url{http://www-groups.dcs.st-and.ac.uk/~history}}%
10 \newcommand{\Pythagorians}{\htmladdnormallink
11  {Pythagorians}{\StAndrews/Mathematicians/Pythagoras.html}}
12 \newcommand{\Fermat}{\htmladdnormallink
13  {Fermat, c.1637}{\StAndrews/HistTopics/Fermat's_last_theorem.html}}
14 \newcommand{\Wiles}{\htmladdnormallink
15  {Wiles, 1995}{http://www.pbs.org:80/wgbh/nova/proof}}
16
17 \begin{document}
18 \htmlhead[center]{section}{Math examples}
19 \begin{eqnarray}
20  \phi(\lambda) & = &  \frac{1} {2 \pi i}\int^{c+i\infty}_{c-i\infty}
21   \exp \left( u \ln u + \lambda u \right ) du \hspace{1cm}\mbox{for } c \geq 0 \\
22  \lambda       & = &  \frac{\epsilon  -\bar{\epsilon} }{\xi}
23                    - \gamma' - \beta^2 - \ln \frac{\xi} {E_{\rm max}}          \\
24  \gamma        & = &  0.577215\dots \mathrm{\hspace{5mm}(Euler's\ constant)}   \\
25  \gamma'       & = &  0.422784\dots = 1 - \gamma                               \\
26    \epsilon , \bar{\epsilon} & = & \mbox{actual/average energy loss}
27 \end{eqnarray}
28
29 Since~\ref{eqn:stress-sr} or~\ref{gdef} should hold for arbitrary $\delta\bm{c}$%
30 -vectors, it is clear that $\Ker(A) = \Range(B)$ and that when $y=B(x)$ one has...\\
31 ...the \Pythagorians{} knew infinitely many solutions in integers to $a^2+b^2=c^2$.
32 That no non-trivial integer solutions exist for $a^n+b^n=c^n$ with integers $n>2$ has long
33 been suspected (\Fermat). Only during the current decade has this been proved (\Wiles).
34
35 \begin{eqnarray}\label{eqn:stress-sr}
36  V \bm{\pi}^{sr} & = & \left<  \sum_i M_i \bm{V}_i \bm{V}_i
37   + \sum_i \sum_{j>i} \bm{R}_{ij} \bm{F}_{ij}\right> \\ \nonumber
38                  & = & \left< \sum_i M_i \bm{V}_i \bm{V}_i
39   + \sum_{i}\sum_{j>i}\sum_\alpha\sum_\beta \bm{r}_{i\alpha j\beta}\bm{f}_{i\alpha j\beta}
40   - \sum_i \sum_\alpha \bm{p}_{i\alpha} \bm{f}_{i\alpha}   \right>
41 \end{eqnarray}
42
43 \end{document} %%%  requires \usepackage{amsmath} to continue
44
45 \begin{subequations}\label{bgdefs}
46 \begin{align} B_{ij}^\alpha     & =
47         \left(B_{ij}^\alpha\right)_0 + \left(B_{ij}^\alpha\right)_a \label{bdef}  \\
48   \left(B_{ij}^\alpha\right)_0  & = \frac{1}{2}\left(\frac{\d N_i^\alpha}{\d X_j}
49         + \frac{\d N_j^\alpha} {\d X_i} \right)                    \label{b0def} \\
50   \left(B_{ij}^\alpha\right)_a  & = H_{ij}^{\alpha \beta} a^\beta   \label{budef} \\
51   H_{ij}^{\alpha \beta}         & =
52     \frac{1}{2}\left( \frac{\d N_k^\alpha}{\d X_i} \frac{\d N_k^\beta}{\d X_j}
53     + \frac{\d N_k^\beta}{\d X_i} \frac{\d N_k^\alpha}{\d X_j} \right)  \label{gdef}
54 \end{align}
55 \end{subequations}
56 \end{document}
```

Второй используемый файл `sampleAMS.tex` идентичен первому за исключением того, что в нем закомментирована команда `\end{document}` (строка 43) и подключен пакет `amsmath` (убран знак комментария `%` в строке 3).

3.3.1 Обзор математических режимов LaTeX2HTML

В основном, в LaTeX2HTML используется три способа отображения математических выражений: режим «новичок», режим «профессионал» и режим «экс-

перт». Некоторые из них обладают дополнительными возможностями. Рассмотрим все режимы по очереди.

3.3.1.1 Режим «новичок»

Режим «новичок», который обычно устанавливается по умолчанию, называют также «простой математикой». Выражения, в которых используются только обычные алфавитные, цифровые и арифметические символы, представляются с помощью текстового шрифта навигатора, который в подходящих случаях печатается курсивом. Верхние и нижние индексы используются в версиях HTML 3.2 и более старших. Любые другие символы или макро представляются графическими образами, которые создаются либо из всего внутритекстового выражения, либо из логической части выровненного окружения. Исключениями являются макрокоманды стилей, такие, как `\mathrm`, `\mathbf`, `\mathtt`, `\boldmath` и `\bm`. Для обеспечения совместимости с более старыми документами допускается использование макрокоманд текстового режима, таких, как `\textrm`, `\textit`, `\texttt` и `\rm`, `\bf`, `\it`, `\tt`. Этот режим можно применять только в тех случаях, когда в документе в основном нематематического содержания встречаются лишь очень простые математические выражения. На рис. 3.4 показано, как математические выражения изображаются в режиме «новичок», который устанавливается с помощью команды:

```
latex2html sampleMath.tex
```

На самом деле в файле `sampleMath.tex` слишком много математики, чтобы использовать этот режим. Обратите внимание, что на рис. 3.4 некоторые встроенные выражения воспроизведены с помощью текстового шрифта навигатора, тогда как для остальных понадобилось создать рисунки. Для выравнивания рисунков необходимо центрировать их относительно базовой линии каждый раз, когда встречаются выступающие вниз детали, например свисающие элементы литер или нижние части скобок. Только некоторые навигаторы могут выполнить такое выравнивание. В любом случае межстрочный интервал становится непостоянным либо слишком большим; с недостатками выполняется и центрирование в выключных формулах. Обратите внимание на различия в размерах и расположении знаков равенства «=». Справиться с такими проблемами можно, используя HTML 4.0 и стиль CSS, как впоследствии будет видно на рис. 3.10.

3.3.1.2 Профессиональный режим

Профессиональный режим хорош для документов со множеством математических формул, как выключных, так и внутритекстовых, а также при частом использовании математических символов. Этот режим устанавливается по умолчанию, когда загружается какой-либо из $\mathcal{AMS}$-пакетов: `amsmath`, `amsthm`, `amsopn` или `amstex`, а также для документов, имеющих класс `amsart` или `amsbook`. Для поддержки единого стиля у внутритекстовых и выключных

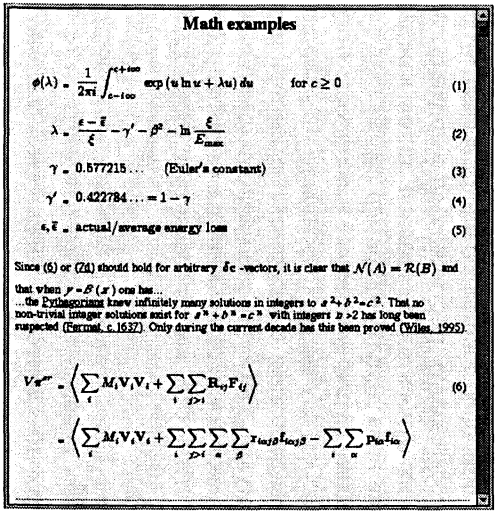


Рис. 3.4. Воспроизведение математики в LaTeX2HTML в режиме «новичок» (стандартные установки).

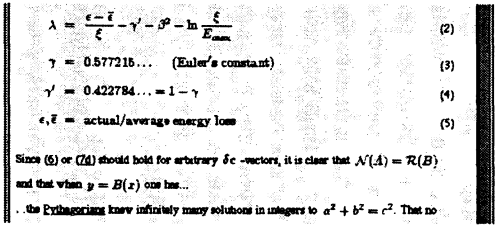


Рис. 3.6. Воспроизведение математики на HTML-странице с использованием профессионального режима.

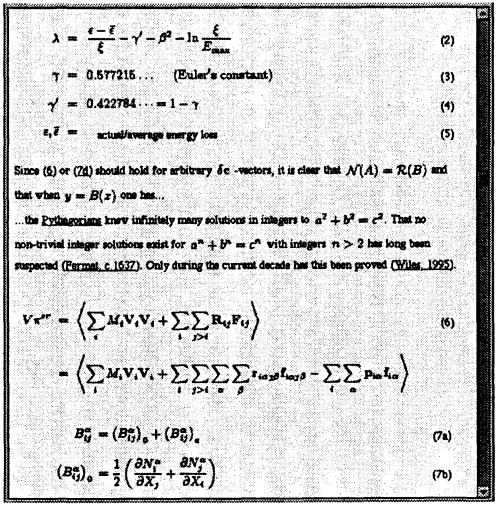


Рис. 3.5. Воспроизведение математики в профессиональном режиме, устанавливаемом по умолчанию с пакетами $\mathcal{AMS}$.

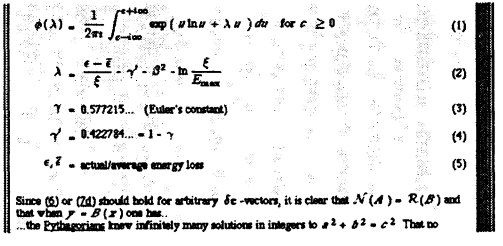


Рис. 3.7. Воспроизведение математики на HTML-странице с использованием режима «эксперт».

формул рисунки создаются из всех внутритекстовых математических выражений, а также из каждого уравнения, формулы или выравниваемого элемента выключной формулы. Там, где это возможно, для аргументов макрокоманд \mbox и \text используется текстовый шрифт навигатора. Рисунок 3.5 иллюстрирует профессиональный режим, который вызывается командой

```
latex2html sampleAMS.tex
```

Окончательный результат, показанный на рис. 3.5, выглядит более сбалансированным по сравнению с тем, который приведен на рис. 3.4. Хотя расстояние между строками, содержащими внутритекстовые математические формулы, фиксировано, оно слегка великовато, из-за того что в каждой строке присутствует рисунок с выступающими вниз элементами («с глубиной»). Даже простой знак «=» в уравнениях представляет собой рисунок, который встречается семь раз. В последнем элементе выражения (5) для \mbox используется текстовый шрифт навигатора. Из-за того что взаимное выравнивание элементов в строке <TABLE> выполняется относительно середины[1] текста, а не его базовой линии, не удается правильно выровнять рисунки в пределах одной строки. В уравнении (2) слова помещены внутрь рисунков, потому что в противном случае нельзя было бы использовать команду \hspace.

При использовании опции -no_math единство стиля, характерного для профессионального режима, достигается без загрузки $\mathcal{AMS}$ пакетов. Это означает, что анализ «непростой математики» выполняется путем использования графических образов. То же самое действие оказывает и установка переменной $NO_SIMPLE_MATH=1;. Для примера, который представлен на рис. 3.6, была использована команда

```
latex2html -no_math -white -no_transparent sampleMath.tex
```

Рисунки показаны на непрозрачном белом фоне (опции -no_transparent и -white).

Заключение

Профессиональный режим предлагает единый стиль для всех математических выражений, в частности при использовании элементов <TABLE> языка HTML для выравниваний и нумерации уравнений. Его недостатком является то, что для любой математической формулы требуется создавать рисунок, и в результате количество файлов с растровыми рисунками может оказаться бóльшим. Вероятно, это не проблема, когда время загрузки рисунков не имеет решающего значения.

[1] Используется атрибут VALIGN="MIDDLE", так как использовать "TOP" и "BOTTOM", очевидно, неправильно.

3.3.1.3 Режим «эксперт» (специальный режим)

В этом режиме структура математических выражений разбивается на меньшие части (фрагменты). Везде, где это возможно, используется текстовый шрифт навигатора, а рисунки создаются из таких фрагментов, как дроби и символы. Для загрузки получающегося при этом множества маленьких рисунков часто требуется меньше времени, чем для загрузки меньшего числа, но больших изображений. Действительно, маленькие рисунки часто можно повторно использовать во многих HTML-страницах, благодаря чему можно существенно уменьшить общий размер Web-документа. В случае очень больших документов общий объем дискового пространства, необходимый для документа в целом, становится существенно меньше, чем при использовании профессионального режима.

Примеры использования специального режима приведены на рис. 3.7–3.10. В случае, показанном на рис. 3.7, была использована команда

```
latex2html -no_math -html_version 3.2,math -white \
           -no_transparent sampleMath.tex
```

3.3.2 Расширение math для сложных математических выражений

Для специального математического режима и детального анализа математических выражений следует загрузить модуль расширения Perl-команд `math`. Примером может служить рис. 3.8, где были использованы следующие опции командной строки:

```
latex2html -no_math -html_version 3.2,math sampleAMS.tex
```

и где также должна быть определена версия HTML. Везде, где это возможно, используется текстовый шрифт навигатора, а рисунки применяются только для специальных символов и конструкций, в которых требуется вертикальное выравнивание, например для дробей, производных, интегралов, знаков суммирования и больших скобок. Подробности показаны на рис. 3.7, где благодаря непрозрачному фону ясно видно, какие именно математические элементы воспроизводятся в виде рисунков, а также, как достигается выравнивание (за счет выравнивания глубины и высоты).

Видимый результат обычно бывает достаточно хорошим, хотя в нем и нет полного соответствия с более крупными рисунками, полученными в профессиональном режиме. Однако теперь есть возможность управлять размером и гарнитурой шрифта, используемого в математических выражениях. Масштаб рисунков не изменяется, но их исходным размером также можно управлять. На размер рисунков оказывают влияние следующие переменные (их можно применять в любом режиме):

`$MATH_SCALE_FACTOR` масштабный коэффициент для создания рисунков из математических формул и фрагментов текстовой строки.

Рекомендуемое значение: 1,4 (соответствует размеру шрифта навигатора 14 пунктов).

$MATH_DISPLAY_FACTOR дополнительное увеличение, применяемое к рисункам, созданным из выключных формул; является множителем при **$MATH_SCALE_FACTOR**.

Рекомендуемое значение: 1,2 (соответствует результирующему размеру шрифта навигатора 17 пунктов).

$FIGURE_SCALE_FACTOR коэффициент увеличения, применяемый при создании изображений из рисунков и целых окружений. Рекомендуемое значение: 1,6.

К используемым в выключных математических формулах шрифтовым символам применяется элемент **<BIG>**. Благодаря **$MATH_DISPLAY_FACTOR** обеспечивается соответствующее масштабирование рисунков. При этом все же остается некоторое рассогласование между смежными элементами, содержащими текст и рисунки, но оно менее бросается в глаза. Вызывает раздражение то, что верхние индексы не удается разместить строго над нижними. Еще сильнее раздражает появляющийся в некоторых навигаторах дополнительный пробел после букв, написанных курсивом, особенно перед нижним индексом.

Главное преимущество специального режима заключается в уменьшении объема дискового пространства, требуемого для хранения более мелких изображений. Достигается это за счет существенного уменьшения участков, отображаемых с использованием растровых рисунков. Это также приводит к уменьшению объема передаваемых данных, благодаря чему сокращается время доступа. Однако это частично компенсируется необходимостью большего числа отдельных соединений для того, чтобы получить большее число файлов для каждой HTML-страницы.

В случае всего лишь одной страницы, достаточно насыщенной математикой, выгода не очень велика, но когда страниц много, то одни и те же изображения символов еще и еще раз повторно используются. Несколько первых страниц могут загружаться медленно; последующие обычно загружаются гораздо быстрее, поскольку большинство требуемых изображений уже локально доступно в кэше изображений навигатора. На самом деле повторное использование рисунков может быть настолько эффективным, что при специальном режиме рисунков может стать *меньше*. В реальной ситуации при использовании профессионального режима статья по математической логике была преобразована в 36 HTML-страниц общим объемом 1,3 Mb с 392 рисунками. В специальном режиме для этого понадобилось только 800 Kb и всего 223 рисунка.

Специальный режим обладает и другими достоинствами, которые гарантируют, что информация, содержащаяся в подлиннике исходного LaTeX'овского документа, передается получателю без искажений. Это справедливо даже в тех случаях, когда возникают проблемы с получением всех изображений для представления требуемого рисунка в окне навигатора.

Полный код рисунков, воспроизводящих маленькие фрагменты исходного LaTeX'овского документа, содержится в атрибуте **ALT** элемента **<IMG>**. На-

пример, показанные на рис. 3.8 суммы (знаки суммирования) размечаются на языке HTML следующим образом:

```
1  <TD ALIGN="LEFT" NOWRAP><IMG
2   WIDTH="233" HEIGHT="63" ALIGN="MIDDLE" BORDER="0"
3   SRC="img19.gif"
4   ALT="$\displaystyle \left&lt; \sum_i M_i \mathbf{V}_i \mathbf{V}_i
5  + \sum_i \sum_{j&gt;i} \mathbf{R}_{ij} \mathbf{F}_{ij}\right&gt;$"></TD>
```

Этот пример демонстрирует также использование объектов `<` (строка 4) и `>` (строка 5) вместо знаков < и >, которые в языке HTML имеют свои особые значения. Большинство рисунков мельче и короче, как в следующем примере:

```
1  - <IMG
2   WIDTH="21" HEIGHT="35" ALIGN="MIDDLE" BORDER="0"
3   SRC="img10.gif"
4   ALT="$\displaystyle \beta^{2}_{}$">
```

Обратите внимание на строку 4, где в `\beta^{2}_{}` автоматически вставлены фигурные скобки и пустой нижний индекс. Это гарантирует хороший LaTeX'овский стиль и одинаковое положение верхнего индекса в сходных рисунках. Таким образом, полную информацию часто можно получить даже с помощью навигатора, воспроизводящего только текст, либо в режиме работы навигатора, в котором рисунки загружаются лишь по требованию.

В соответствии с этой концепцией, когда математическое окружение делится на шрифтовые символы и маленькие рисунки, *целый исходный документ* вставляется в HTML-файл в виде комментария. Следующий пример показывает, как из исходного HTML-файла извлекается фрагмент документа, показанный на рис. 3.8. Может быть выполнен поиск встречающихся в этих комментариях отдельных частей математических выражений. Найденные фрагменты легко извлечь для повторного использования в других LaTeX'овских документах. Нумерация объектов в комментариях необязательна.

```
1  <DIV ALIGN="CENTER">
2  <!-- MATH: \begin{eqnarray}
3  \phi(\lambda) & = &  \frac{1} {2 \pi i}\int^{c+i\infty}_{c-i\infty}
4   \exp \left( u\ln u +\lambda u \right) du \hspace{1cm}\mbox{for } c\geq 0\\
5   \lambda        & = &  \frac{\epsilon  -\bar{\epsilon} }{\xi}
6    - \gamma' - \beta^2 - \ln \frac{\xi} {E_{\rm max}}          \\
7   \gamma  & = &  0.577215\dots \mathrm{\hspace{5mm}(Euler's\ constant)}   \\
8   \gamma' & = &  0.422784\dots = 1 - \gamma \\
9   \epsilon , \bar{\epsilon} & = & \mbox{actual/average energy loss}
10 \end{eqnarray} -->
11
12 <TABLE ALIGN="CENTER" CELLPADDING="0" WIDTH="100%">
13 <TR VALIGN="MIDDLE"><TD NOWRAP ALIGN="RIGHT"><IMG
14  WIDTH="35" HEIGHT="31" ALIGN="MIDDLE" BORDER="0"
15  SRC="img1.gif"
16  ALT="$\displaystyle \phi(\lambda)$"></TD>
17 <TD ALIGN="CENTER" NOWRAP><IMG
```

При возможном увеличении времени загрузки (скачивания) все остальные преимущества профессионального режима можно получить одновременно с рисунками, используя следующий набор опций командной строки:

```
latex2html -no_math_parsing -html_version 3.2,math filename
```

Заключение

Использование специального режима вместе с расширением `math` обеспечивает согласованность всех математических выражений. При этом исходный LaTeX'овский документ включается в HTML-файлы в виде комментария, благодаря чему становится возможным поиск текста внутри математических выражений. Такая стратегия приводит к увеличению размеров рисунков, а следовательно, и к увеличению времени загрузки. Однако это решение является идеальным, если нужно совместить максимальный объем информации с высоким качеством изображений.

3.3.3 Шрифты Unicode и именованные объекты в специальном режиме

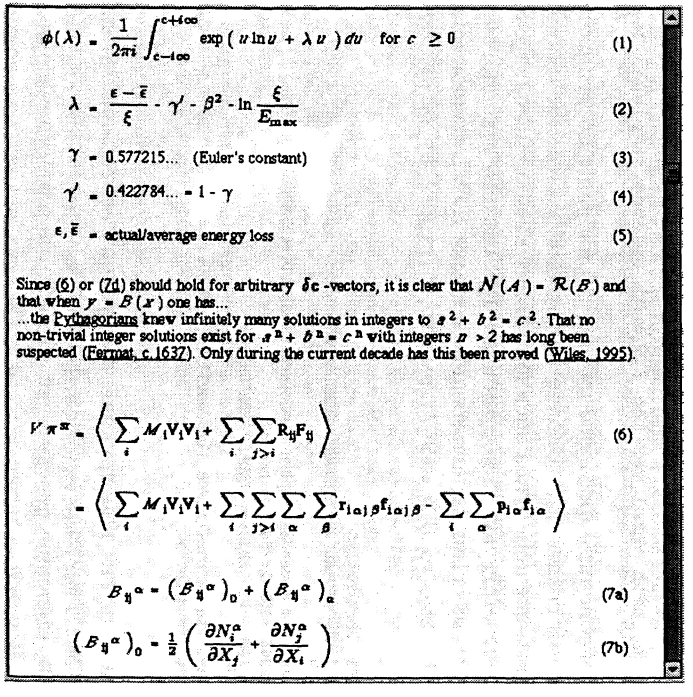
Количество рисунков можно еще больше сократить, если использовать в навигаторе специальные шрифты для греческих букв и некоторых математических символов. Коды для таких символов определяются стандартом Unicode (или ISO-10646-1) (см. приложение С.2). Это возможно в HTML-страницах, удовлетворяющих требованиям спецификаций 3.2 или 4.0. В настоящее время навигаторы поддерживают в лучшем случае только некоторые из этих символов. На рис. 3.9 показан результат, который получается, когда LaTeX2HTML загружает расширение `unicode` с помощью команды

```
latex2html -no_math -html_version 3.2,math,unicode sampleAMS.tex
```

Почти все выключные формулы на рис. 3.9 набраны доступными навигатору шрифтовыми символами. Вполне приемлемы маленькие буквы α и β в индексах. Видоизмененная буква эпсилон входит в набор используемого здесь шрифта; для соответствующей коррекции LaTeX'овского исходного кода можно взять `\bar\varepsilon`. Использование в языке HTML ссылок на символы, например `&#946;` вместо β и `&#947;` вместо γ (строка 5) затрудняет чтение `html`-файлов.

```
1  <TD ALIGN="LEFT" NOWRAP><IMG
2   WIDTH="40" HEIGHT="47" ALIGN="MIDDLE" BORDER="0"
3   SRC="img5.gif"
4   ALT="$\displaystyle {\frac{\epsilon -\bar{\epsilon} }{\xi}}$">
5  - &#947;' - &#946;<SUP>2</SUP> - ln<IMG
6   WIDTH="44" HEIGHT="51" ALIGN="MIDDLE" BORDER="0"
7   SRC="img6.gif"
8   ALT="$\displaystyle {\frac{\xi}{E_{\rm max}}}$"></TD>
9  <TD WIDTH=10 ALIGN="RIGHT">
10 (2)</TD></TR>
```

Для дробей и других конструкций, требующих вертикального выравнивания, по-прежнему требуются рисунки. Наиболее часто встречаются, пожалуй, знаки операций, имеющие переменный размер, например интегралы и знаки суммирования. Верхние и нижние индексы в этих операторах вставляются в рисунки, поскольку другими способами нельзя достичь их правильного расположения.



$$\phi(\lambda) = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} \exp\left(u\ln u + \lambda u\right) du \quad \text{for } c \geq 0 \tag{1}$$

$$\lambda = \frac{\epsilon - \bar{\epsilon}}{\xi} - \gamma' - \beta^2 - \ln\frac{\xi}{E_{max}} \tag{2}$$

$$\gamma = 0.577215\ldots \text{ (Euler's constant)} \tag{3}$$

$$\gamma' = 0.422784\ldots = 1 - \gamma \tag{4}$$

$$\epsilon, \bar{\epsilon} = \text{actual/average energy loss} \tag{5}$$

Since (6) or (7d) should hold for arbitrary $\delta\mathbf{c}$ -vectors, it is clear that $\mathcal{N}(A) = \mathcal{R}(B)$ and that when $y = B(x)$ one has...
...the Pythagorians knew infinitely many solutions in integers to $a^2 + b^2 = c^2$. That no non-trivial integer solutions exist for $a^n + b^n = c^n$ with integers $n > 2$ has long been suspected (Fermat, c.1637). Only during the current decade has this been proved (Wiles, 1995).

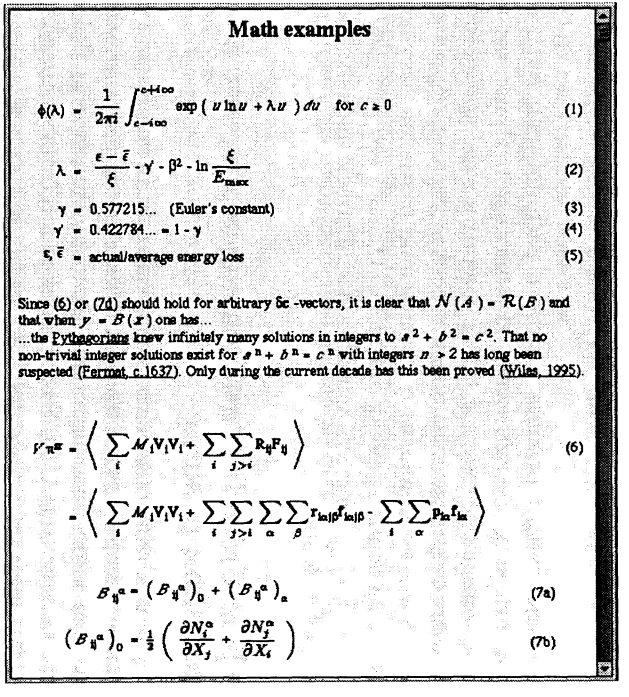
$$V^{\pi\pi} = \left\langle \sum_i M_i V_i V_i + \sum_i \sum_{j>i} R_{ij} F_{ij} \right\rangle \tag{6}$$

$$= \left\langle \sum_i M_i V_i V_i + \sum_i \sum_{j>i} \sum_\alpha \sum_\beta r_{i\alpha j\beta} f_{i\alpha j\beta} - \sum_i \sum_\alpha p_{i\alpha} f_{i\alpha} \right\rangle$$

$$B_{ij}{}^{\alpha} = \left(B_{ij}{}^{\alpha}\right)_0 + \left(B_{ij}{}^{\alpha}\right)_a \tag{7a}$$

$$\left(B_{ij}{}^{\alpha}\right)_0 = \tfrac{1}{2}\left(\frac{\partial N_i^{\alpha}}{\partial X_j} + \frac{\partial N_j^{\alpha}}{\partial X_i}\right) \tag{7b}$$

Рис. 3.8. Воспроизведение математики в специальном режиме со специальным расширением `math`.



Math examples

$$\phi(\lambda) = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} \exp\left(u\ln u + \lambda u\right) du \quad \text{for } c \geq 0 \tag{1}$$

$$\lambda = \frac{\epsilon - \bar{\epsilon}}{\xi} - \gamma' - \beta^2 - \ln\frac{\xi}{E_{max}} \tag{2}$$

$$\gamma = 0.577215\ldots \text{ (Euler's constant)} \tag{3}$$

$$\gamma' = 0.422784\ldots = 1 - \gamma \tag{4}$$

$$\epsilon, \bar{\epsilon} = \text{actual/average energy loss} \tag{5}$$

Since (6) or (7d) should hold for arbitrary $\delta\mathbf{c}$ -vectors, it is clear that $\mathcal{N}(A) = \mathcal{R}(B)$ and that when $y = B(x)$ one has...
...the Pythagorians knew infinitely many solutions in integers to $a^2 + b^2 = c^2$. That no non-trivial integer solutions exist for $a^n + b^n = c^n$ with integers $n > 2$ has long been suspected (Fermat, c.1637). Only during the current decade has this been proved (Wiles, 1995).

$$V^{\pi\pi} = \left\langle \sum_i M_i V_i V_i + \sum_i \sum_{j>i} R_{ij} F_{ij} \right\rangle \tag{6}$$

$$= \left\langle \sum_i M_i V_i V_i + \sum_i \sum_{j>i} \sum_\alpha \sum_\beta r_{i\alpha j\beta} f_{i\alpha j\beta} - \sum_i \sum_\alpha p_{i\alpha} f_{i\alpha} \right\rangle$$

$$B_{ij}{}^{\alpha} = \left(B_{ij}{}^{\alpha}\right)_0 + \left(B_{ij}{}^{\alpha}\right)_a \tag{7a}$$

$$\left(B_{ij}{}^{\alpha}\right)_0 = \tfrac{1}{2}\left(\frac{\partial N_i^{\alpha}}{\partial X_j} + \frac{\partial N_j^{\alpha}}{\partial X_i}\right) \tag{7b}$$

Рис. 3.9. Воспроизведение математики с использованием специального режима и символов шрифта Unicode.

При использовании опции командной строки `-entities` числовые ссылки на символы замещаются именованными ссылками на объекты, например `&beta;` и `&gamma;`. Хотя такие ссылки и входят во множество объектов, доступных в версии HTML 4.0, только самые новейшие навигаторы действительно их поддерживают.

Заключение

Специальный режим, использующий кодировку Unicode, требует существенно меньшего количества рисунков, чем описанные ранее способы, потому что в набор шрифта входят знаки для многих математических символов. Однако современные навигаторы лишь в ограниченной степени поддерживают или совсем не поддерживают стандарт Unicode, а также не могут использовать шрифты, содержащие греческие буквы и математические символы. Этот подход несомненно приобретет большее значение в обозримом будущем и позволит очень сильно сократить время загрузки. По только что упомянутым причинам его практическое влияние до поры до времени будет оставаться весьма ограниченным.

3.3.4 HTML 4.0 и стили

При генерации кода для спецификации HTML 4.0 имеется больше возможностей для исправления некоторых из упомянутых ранее визуальных дефектов. Доступны все три математических режима вместе со своими вариантами. Вот несколько примеров их использования:

```
latex2html -html_version 4.0 sampleMath.tex
latex2html -no_math -html_version 4.0 sampleMath.tex
latex2html -no_math -html_version 4.0,math sampleMath.tex
latex2html -no_math -html_version 4.0,math,unicode sampleMath.tex
latex2html -no_math_parsing -html_version 4.0,math sampleMath.tex
```

Некоторые простые результаты применения стилей показаны на рис. 3.10, который был получен с помощью команды:

```
latex2html -no_math_parsing -html_version 4.0,math \
                    -style sampleAMS.css sampleAMS.tex
```

Действие стилей проявляется в следующем (номера строк соответствуют приведенному ниже стилю CSS):

- цветной заголовок в рамке другого цвета (строка 1);
- полужирный цветной шрифт нумерации формул (строка 2);
- окрашенный фон окружения `subequations` (строка 3) и
- постоянная высота строк в абзацах, содержащих внутритекстовые изображения математических выражений (строка 4).

LaTeX2HTML вместе с HTML-страницами при обнаружении файла `sampleAMS.tex` автоматически создает «стандартный» стиль CSS `sampleAMS.css`. Затем этот стиль можно редактировать, чтобы включить в него декларации, которые обеспечивают выполнение желаемых действий, например:

```
1 H2 { border : 2pt solid brown ; color : maroon ; padding : 5pt }
2 TD.eqno {color : green ; fontweight : 700 } /* equation-number cells */
3 TABLE.subequations { background-color : #E0E0E0 ; }
4 P  { line-height : 18pt ; }
```

Естественно, эти эффекты будут видны только в том случае, когда HTML-страницы просматриваются с помощью навигатора, который в состоянии интерпретировать язык стилей. Рисунок 3.10 был получен с помощью навигатора, который понимает язык CSS.

В настоящее время в качестве языка таблиц стилей поддерживается только CSS. В будущем, по мере того как другие языки таблиц стилей будут становиться общедоступными, к LaTeX2HTML будут добавлены средства, позволяющие использовать их возможности. Более подробная информация о языке CSS представлена в разд. 7.4.

В HTML-страницах, создаваемых с помощью LaTeX2HTML, можно использовать несколько способов определения элементов стилей CSS.

1. По завершении работы LaTeX2HTML можно, используя любой текстовый редактор, вручную отредактировать автоматически созданный стиль. Для каждого окружения в исходном LaTeX'овском документе уже будет создана пустая «заглушка», которая затем может быть дополнена желаемыми свойствами и значениями.
2. Можно сослаться на заранее подготовленный стиль при помощи опции командной строки `-style` *style-file*. Можно также задать строковое значение переменной `$STYLESHEET`, равное *style-file*, указав здесь либо полный URL местоположения файла со стилем, либо относительный URL от того места, где будут постоянно находиться HTML-страницы.
3. Можно включить стилевую информацию непосредственно в сам исходный LaTeX'овский документ. Это можно сделать для окружений, использующих необязательный аргумент команды `\begin` (для этого следует загрузить пакет `html`).

Вряд ли это хорошая идея — сильно перемешивать информацию о стиле с исходным кодом, хотя иногда полезно задействовать микропрограммное управление элементами какого-либо фрагмента текста или формулы. Поэтому рассмотрим третий вариант более подробно и выясним, как можно управлять стилем посредством необязательного аргумента команды `\begin` какого-либо окружения. Далее описываются три способа использования этой возможности:

`\begin[`*style-info*`]`

В этом первом способе в открывающую метку `<DIV>`, `<SPAN>` или `<TABLE>`, созданную для содержимого окружения, включается атрибут `ID="`*unid*`"`. Уни-

кальный идентификатор ***unid*** связывает соответствующий элемент с определением *style-info* в стиле.

`\begin[`*class*`]`

В этом втором способе с созданным элементом языка HTML ассоциируется атрибут `CLASS="`*class*`"`. В результате будет унаследована информация о стиле, связанная с *class*. В таблицу стилей записывается «заглушка», позволяющая впоследствии добавить необходимую информацию.

`\begin[`*class*`|`*style-info*`]`

Третий метод является комбинацией двух предыдущих, причем для разделения различных типов используется «|».

Содержимое *style-info* может в точности представлять собой код стиля. В другом варианте это может быть набор пар ***key=value*** (ключ=значение), в которых в качестве разделителя используется «=» или «:». Пары отделяются друг от друга знаком «;», тогда как «,» используется для разделения нескольких значений одного ключа. В именах атрибута *class* используются алфавитно-цифровые символы. Если разделитель | отсутствует, то присутствие =, : или ; обозначает, что аргумент содержит *style-info* и не содержит имен *class*.

Следующий LaTeX'овский код позволяет формировать входную информацию для стиля, действие которого показано на рис. 3.10 (описание команды `\htmlsetstyle` см. в разд. 3.5.2). Приведенные ниже строки 1–3 эквивалентны строкам 1, 2 и 4 стиля CSS, показанного на с. 140, в то время как строка 3 стиля заменена спецификацией, расположенной непосредственно в начале окружений `subequations` (строка 5).

```
1 \htmlsetstyle[H2]{}{border=2pt,solid,brown;color=maroon;padding:5pt}
2 \htmlsetstyle[TD]{eqno}{color : green ; fontweight : 700 }
3 \htmlsetstyle[P]{}{ line-height : 18pt }
4
5 \begin[background-color=\#E0E0E0]{subequations}
6   ...
7 \end{subequations}
```

Заключение

HTML 4 и CSS позволяют использовать любое количество стилей для создания стилевых эффектов. Хотя многие ранее созданные навигаторы имеют ограниченную поддержку стилей либо вовсе ее не имеют, эта ситуация быстро улучшается. Поэтому через некоторое время сочетание HTML 4 и CSS станет идеальным способом, обеспечивающим полное управление воспроизведением информации.

3.3.5 Большие рисунки и HTML 2.0

Другой подход к воспроизведению математических выражений в Web-страницах заключается в том, чтобы использовать более крупные рисунки, т. е. изображения целых формул (окружений). Наглядный вид действительно мог бы иметь решающее значение, несмотря на объем данных и неподходящую скорость их передачи. Такая ситуация может возникнуть, например, при просмотре компакт-дисков CD-ROM или использовании отдельной машины либо локальной сети. При необходимости обеспечить совместимость с навигаторами, способными интерпретировать разметку только версии HTML 2.0, такой подход оказывается единственно жизнеспособным, поскольку элементы `<TABLE>`[1] не поддерживаются.

Крупные рисунки действительно могут быть очень привлекательными, как показано на рис. 3.11, который был получен с помощью команды

```
latex2html -no_math -html_version 2.0 sampleAMS.tex
```

Однако из-за того, что номера уравнений становятся частью изображений, повторное использование рисунков становится невозможным. Гиперссылки, связанные с отдельными уравнениями или «подуравнениями», должны адресоваться ко всему рисунку как единому целому. При использовании HTML 2.0 возможность выравнивания ограничивается целыми абзацами и рисунками.

Большие рисунки можно обрабатывать также и с более поздними версиями HTML посредством команды `\htmlimage` (строка 9) и окружения `makeimage` (строки 2–6), которые определены в пакете `html`. При использовании `makeimage` необходимо в явном виде задать выравнивание (окружение `flushright` в строках 1–7) для того, чтобы номера формул ровнялись по правому краю.

```
1  \begin{flushright}
2    \begin{makeimage}
3      \begin{eqnarray}
4        ...
5      \end{eqnarray}
6    \end{makeimage}
7  \end{flushright}
```

```
8  \begin{flushright}
9  \begin{eqnarray}
10   \htmlimage{}
11     ...
12 \end{eqnarray}
13 \end{flushright}
```

Команда `\htmlimage` должна иметь аргумент, который может быть пустым. Пример показан на рис. 3.12. В нем использован документ из файла `sampleMathImages.tex`, который является модификацией указанного ранее `sampleAMS.tex`.

Полный список возможных аргументов команды `\htmlimage` приведен в разд. 3.5. Один из них способствует снижению трудностей представления больших рисунков за счет того, что непосредственно в HTML-страницу вставляется только меньшая по размеру «пиктограмма»:

```
\htmlimage{thumbnail=.4}
```

[1] В настоящее время таких навигаторов, вероятно, уже нет.— *Прим. ред.*

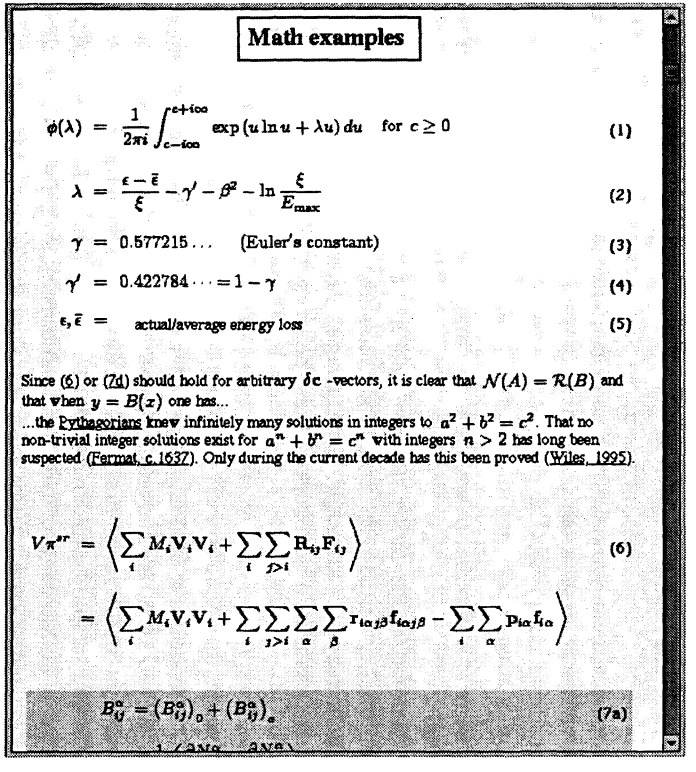


Рис. 3.10. Воспроизведение математики в специальном режиме, с большими рисунками, HTML 4.0 и эффектами таблицы стилей.

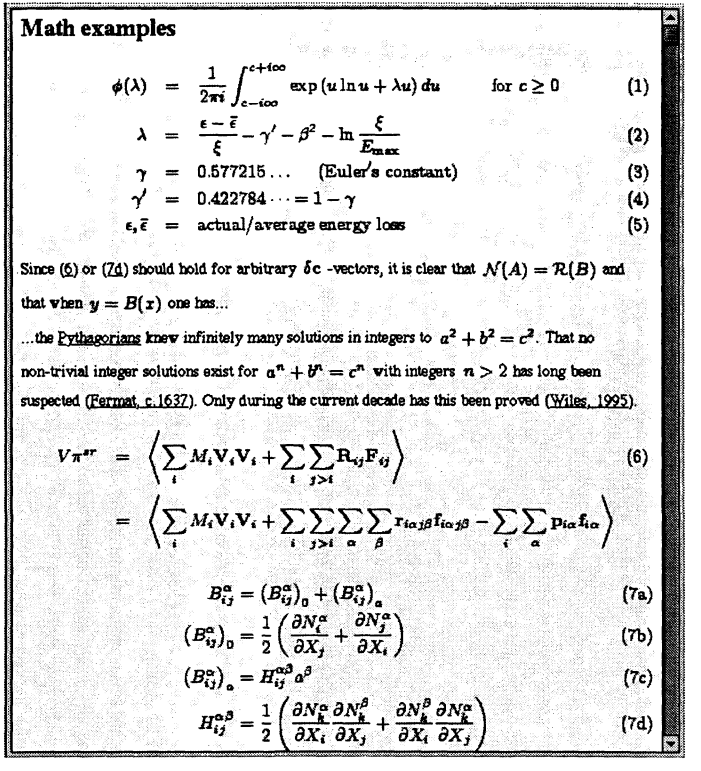


Рис. 3.11. Воспроизведение математики с помощью рисунков целых окружений (HTML 2.0).

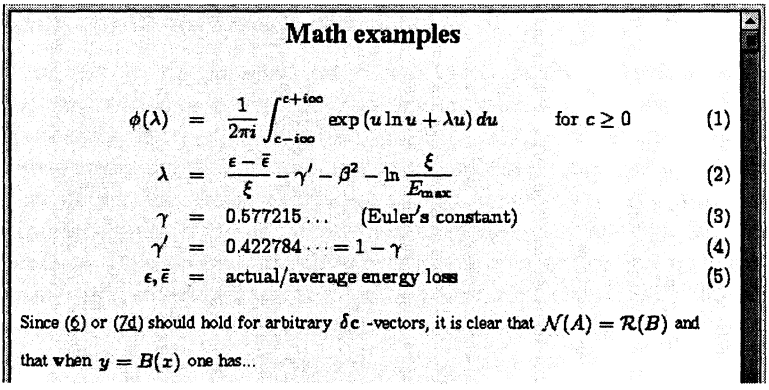


Рис. 3.12. Воспроизведение математики рисунками целых окружений.

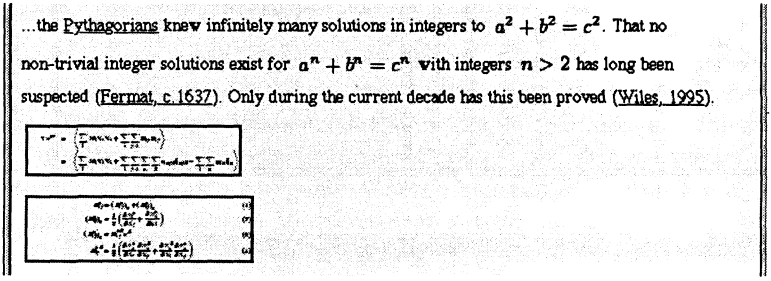


Рис. 3.13. Воспроизведение математики с помощью «иконок», реализующих гиперссылки на полноразмерные рисунки.

Она действует как кнопка, реализующая гиперссылку к полномасштабной версии рисунка.

На рис. 3.13 показан результат, полученный из `sampleMathThumb.tex`, который идентичен файлу `sampleAMS.tex` за исключением того, что для выключных формул используется команда `\htmlimage`. Этот способ был достаточно эффективно использован в некоторых электронных журналах.

Заключение

Все проблемы, связанные с большими рисунками, являются общим недостатком подходов HTML 2.0 к обеспечению правильного и с высоким качеством TeX'овского набора представления математики. Время передачи документа может существенно возрасти, оказывается занятым большой объем дискового пространства, а также отсутствует возможность повторного использования рисунков. Тем не менее этот режим по-прежнему является полезным вариантом при использовании более старых навигаторов, а также тогда, когда время передачи не имеет слишком большого значения.

3.3.6 Использование MathML в будущем

Реализуемое в специальном режиме разбиение математических выражений на отдельные элементы представляет собой определенный шаг вперед к созданию полноценной разметки языка MathML. Уже выполнена определенная работа над LaTeX2HTML с целью создания мастера WebEQ, генерирующего Java-апплеты и файлы с разметкой MathML. Для этих целей в HTML 3.2 создаются элементы `<APPLET>`, а в HTML 4.0 — элементы `<OBJECT>`. Однако в настоящее время эта процедура работает только с некоторым подмножеством LaTeX'овской математики, поэтому здесь она больше не описывается. В будущем эта работа будет развиваться, с тем чтобы охватить все стороны LaTeX'овской математики, на которые распространяется MathML.

3.4 Поддержка различных языков

Благодаря использованию специальных шрифтов и макро-пакетов, TeX позволяет выполнять набор текста на любых языках. Бóльшая часть этого программного обеспечения может быть использована и с LaTeX2HTML. Для многих языков соответствующие интерфейсы уже существуют и включены в дистрибутив LaTeX2HTML. Не представляет большого труда получить интерфейс для других языков путем адаптации какого-либо из уже существующих.

LaTeX2HTML может несколькими способами обеспечить поддержку создания HTML-страниц, написанных не по-английски.

Заголовки и ключевые слова. Для многих языков достаточно перевести ключевые слова и заголовки, которые LaTeX2HTML автоматически размещает на HTML-страницах. Это, например, заголовки наподобие Глава, Аннотация (Реферат), Список литературы, Предметный указатель, а также команды перемещения, такие, как Следующий, Наверх и Предыдущий. Нужно также перевести заголовок «Об этом документе...» для информационной страницы вместе со всей реально содержащейся там информацией.

Кодирование алфавита. В некоторых языках используются символы, которые не входят в набор шрифтовых кодов Latin 1 (ISO-8859-1). Набор Latin 1 включает в себя все символы многих западноевропейских и скандинавских языков, в то время как Latin 2 (ISO-8859-2) — большинства восточноевропейских языков, основанных на латинском алфавите. Наборы кодов Latin 3 (ISO-8859-3), Latin 4 (ISO-8859-4), Latin 5 (ISO-8859-9) и Latin 6 (ISO-8859-10) распространяются на некоторые языки Средиземноморья, Среднего Востока и Северной Европы. LaTeX2HTML предоставляет средства для корректной обработки документов, написанных с использованием этих кодировок. Доступна также поддержка некоторого подмножества кодировки Unicode и имен объектов для букв со знаками ударения и других символов.

Рисунки (использование специальных шрифтов). LaTeX2HTML может распознавать макрокоманды явного переключения шрифтов. Поэтому существует возможность создать рисунок из соответствующего фрагмента текста при условии, что он явным образом заключен в фигурные скобки. Такой подход вполне эффективен в отношении отдельных слов и коротких предложений, но в случае кусков текста большего размера отдельные рисунки создаются из каждого абзаца.

Предварительная обработка и рисунки. В случае более сложных языков, наподобие тех, которые используются в Индии и странах Юго-Восточной Азии, использование различных схем транслитерации и транскрипции позволяет получить кодированное представление с использованием символов, доступных на стандартной «Латинской» клавиатуре. Уже существует программное обеспечение для предварительной обработки с целью преобразования этих кодировок в TeX'овские команды доступа к шрифтам, воспроизводящим традиционные формы. LaTeX2HTML идеально приспособлен для написания сценариев таких препроцессоров, предназначенных для выполнения необходимых преобразований (перекодировок), предшествующих созданию рисунков из традиционного текста. Соответствующие интерфейсы к нескольким препроцессорам для индийских языков включены в состав LaTeX2HTML под именем Indic-TeX/HTML; сами программы предварительной обработки должны устанавливаться отдельно.

В последующих подразделах более подробно обсуждается поддержка, осуществляемая LaTeX2HTML для каждой из этих стратегий.

3.4.1 Заголовки и ключевые слова

Комплект поставки LaTeX2HTML включает в себя интерфейсы к Perl-пакетам для большинства языков и диалектов, на которые распространяется система Babel Йоханнеса Брамса. К ним относятся `english.perl`, `german.perl`, `usorbian.perl` и т. д. Они соответствуют файлам определения языка (`ldf`), входящим в дистрибутив системы Babel. Однако для полной языковой поддержки LaTeX2HTML требуется больше ключевых слов, чем представлено в `ldf`-файлах.

Как правило, подобный файл «языкового интерфейса» определяет Perl-подпрограмму для установки заголовков и ключевых слов, например `german_titles`. Кроме того, в строке идентификации языка, например `german`, задаются две переменные: `$default_language` и `$TITLES_LANGUAGE`. Более того, для использования в управляемом LaTeX'ом пакете Babel могут быть определены специальные макро или символьные последовательности, которые отличаются от стандартного англо-американского использования LaTeX'а. Например, для набора букв со знаком умляута существуют последовательности наподобие `"a`, `"e`, ..., `"u`, а другие последовательности позволяют иметь дело с кавычками и выполнять переносы слов. Соответствующий `perl`-файл содержит требуемый

код, с помощью которого эти символьные последовательности могут быть правильно перенесены на HTML-страницу. Если в языке используются символы, отсутствующие в наборе символов Latin 1, то для поддержки требуемой кодировки будет загружен соответствующий файл (см. разд. 3.4.2).

Для некоторых языков, в частности, голландского, финского, французского, немецкого, испанского, шведского и турецкого, должны быть переведены все строки, необходимые для LaTeX2HTML, в то время как для других языков и диалектов, таких как африкаанс, бахаса, бразильский, бретонский, каталонский, хорватский, чешский, датский и эсперанто, заголовки и ключевые слова берутся из `ldf` файлов пакета Babel. Для восполнения некоторых пробелов требуются дополнения, а также очень желательны исправления и улучшения.

Файл «языкового интерфейса» загружается различными способами, так что используемые по умолчанию английские значения заменяются переведенными словами и фразами.

- Скопируйте содержимое в файл `l2hconf.pin` и замените определенные там английские строки. Повторно выполните процедуру инсталляции. Таким образом устанавливается новый язык, который будет приниматься по умолчанию для всех документов, обрабатываемых этой новой инсталляцией LaTeX2HTML.
- Создайте копию файла `l2hconf.pm` и замените принимаемые по умолчанию значения так, как показано выше. Теперь установите переменную окружения `L2HCONFIG` так, чтобы загружался этот альтернативный файл конфигурации. Это позволяет использовать конфигурации с различными языками без переустановки LaTeX2HTML. Необходимо только задать соответствующее значение переменной окружения `L2HCONFIG`.
- Вставьте в файл `.latex2html-init` Perl-команду, такую, как

  ```
  &do_require_package("$LATEX2HTMLSTYLES/german.perl");
  ```

- Поместите LaTeX'оподобный код непосредственно в документ, чтобы использовать один из рассмотренных в разд. 3.2.4 механизмов загрузки пакетов. Любая из перечисленных ниже LaTeX'овских команд будет работать:

  `\usepackage{german}` `\usepackage[german]{babel}`
  `\documentstyle[german]{`*style*`}` `\documentclass[german]{`*class*`}`

  Здесь в последнем примере имя языка должно быть разрешенной опцией для класса документа `<`*class*`>`. В действительности это необходимо только для того, чтобы LaTeX правильно обрабатывал документ, либо тогда, когда с помощью LaTeX'а должны быть созданы рисунки. LaTeX2HTML загружает соответствующий `perl` файл и пытается выполнить трансляцию в HTML независимо от того, является ли документ действительно LaTeX'овским или нет.

Многоязычные документы можно создавать, используя пакет `babel`. Как показано в разд. 3.4.3, стили воспроизведения языковых сегментов могут быть разными.

3.4.2 Кодировка алфавита

При восьмибитовом кодировании символов языка, даже в случае европейских языков, использующих латинский алфавит, в верхней части кодовой таблицы (значения кодов 128–255) используются различные начертания символов. Буквы с такими кодами, присутствующие в исходном LaTeX'овском документе, автоматически изображаются в виде объектов, соответствующих входной кодировке[1], если они являются частью обычного текста. Для кодирования объектов могут использоваться различные методы, поэтому, для того чтобы навигатор изображал правильные символы, он должен знать, какая кодовая таблица используется. Кодировка определяется как атрибут элемента `<META>` внутри раздела `HEAD` HTML-документа[2], например:

```
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html"
      CHARSET="iso-8859-1">
...
</HEAD>
```

Если не определено другое, LaTeX2HTML использует кодировку Latin 1 (ISO-8859-1), так как в HTML 3.2 предполагается использование именно этого набора кодов. Поддерживаются и другие наборы кодов от Latin 2 до Latin 9 (см. табл. С.4), а также Unicode (UTF-8). При использовании стандарта Unicode буквы со знаками ударения и другие символы (см. [↪HTMLENTS]) задаются в виде числовых ссылок (например, `&#233;`), *если не* определена опция командной строки `-entities`, либо как UTF-8 байтовые строки, если `utf8.pl` загружен как часть опции `-html_version`.

Каждой из поддерживаемых кодировок соответствует расположенный в подкаталоге `$LATEX2HTMLVERSIONS` файл с именем наподобие `latin1.pl`,..., `latin6.pl`, `unicode.pl`. Загрузить эти файлы и поменять кодировку можно различными способами. Если загружено несколько файлов, то используемую в HTML-страницах кодировку определяет последний из них, за исключением тех случаев, когда в их числе загружен файл `unicode.pl`. В этом случае используется кодировка UTF-8.

Наборы кодов могут быть загружены различными способами:

- Загрузите поддержку конкретной кодировки, например:

```
latex2html -html_version 3.2,math,latin2 мой_документ.tex
latex2html -html_version 4.0,latin2,unicode,latin2 мой_документ.tex

latex2html -html_version 4.0,latin5,unicode,utf8 мой_документ.tex
```

  Если определено более одной кодировки, то загружаются все файлы. Первый из них принимается за входную кодировку, тогда как остальные определяют другие возможные кодировки. Если задана вторая латинская ко-

[1] С соответствующей модификацией при использовании в аргументах изменяющих регистр макрокоманд, таких, как `\MakeUppercase`, `\MakeLowercase`, `\uppercase` и `\lowercase`.

[2] Для языков, где распространены несколько кодировок (включая русский), использование элемента `<META>`, задающего кодировку, противопоказано. Причина этого — в существовании перекодирующих серверов, которые изменяют кодировку, после чего такой элемент `<META>` будет только вводить в заблуждение пользовательские навигаторы.— *Прим. ред.*

дировка, то она принимается за основную кодировку документа; будут использоваться исходные 8-битовые символы. Если также используется опция `-entities`, то остальные графические примитивы представляются как числовые или именованные объекты.

- Загрузите во вводной части документа пакет `inputenc`, определив при этом требуемую кодировку, например `\usepackage[latin2]{inputenc}`.
- Загрузите описанным в разд. 3.4.1 способом файл «языкового интерфейса», который неявным образом загружает поддержку конкретной кодировки.

3.4.3 Использование пакета `babel` для многоязычных документов

Как уже упоминалось, можно получить специальную поддержку отдельных языков, загружая пакет `babel` с опциями, соответствующими требуемым языкам. Внутри документа выбор языка для заданной части текста осуществляется командой `\selectlanguage`. Описанные ниже элементы `LANG` предполагают использование HTML 4.0, в противном случае такие атрибуты в HTML-страницу не помещаются.

Команда `\selectlanguage{`*language*`}`, расположенная в преамбуле документа, определяет язык для заголовков, ключевых слов и навигационных панелей. Если имеется несколько команд `\selectlanguage`, то реализуется последняя из них. Практически, в качестве значения атрибута `LANG` элемента `<BODY>` используется идентификатор языка ISO 639 (ISO:639, 1988) (см. табл. C.1 на с. 527). Более того, соответственно этому языку выбирается и кодировка самого документа. Например, результатом выполнения команды `\selectlanguage{turkish}` является `<BODY LANG="tr">`, и, кроме того, для документа берется набор кодов Latin 5 (ISO-8859-9). Переход к другому диалекту языка может привести к изменению формата представления дат на титульной странице; например, команды `\selectlanguage{american}` и `\selectlanguage{english}` отличаются друг от друга в этом отношении.

Использование команд `\selectlanguage` {*language*} в теле документа не оказывает влияния на кодировку. Вместо этого, когда язык (*language*) отличается от того, который был установлен для документа в целом, все абзацы в указанной части документа наследуют соответствующий атрибут `LANG`. Например, абзацы, следующие за командой `\selectlanguage{austrian}`, будут начинаться с `<P LANG="de-AT">` до тех пор, пока не встретится другая команда `\selectlanguage` или не будет закрыта текущая TEX'овская группа. Аналогично атрибут языка наследуют и дескрипторы `<TABLE>`.

```
\htmllanguagestyle{german}
```

Команда `\htmllanguagestyle`, которая определена в пакете `html`, связывает имя класса с последующими абзацами, например `<P LANG="de" CLASS="de">`. Таким образом, определенный стиль можно связать с заданным языком, в данном случае — с немецким (`de`).

3.4.4 Рисунки, использующие специальные шрифты

В LaTeX2HTML обеспечивается специальная поддержка как LaTeX'овской команды `\newfont`, так и TeX'овской команды `\font`.[1] Они применяются для определения макрокоманд, предназначенных для использования шрифтов специального назначения в небольших фрагментах текста. Например, макрокоманда выбора шрифта может быть определена следующим образом, возможно, внутри отдельного стилевого файла или файла класса документа:

```
\font\wncyr=wncyr10      % Cyrillic Roman font
\font\wncyi=wncyi10      % Cyrillic Italiced font
\newfont{\SHa}{sinha10}  % Haralambous' Sinhala Font A
\newfont{\SHb}{sinhb10}  % Haralambous' Sinhala Font B
\newfont{\SHc}{sinhc10}  % Haralambous' Sinhala Font C
```

Подобные макро обычно используются как описатели в пределах групп, заключенных в фигурные скобки, например[2]:

```
({\wncyi Russko-Singal\char126ski\char26\ Slovar\char126\/}),
{\wncyr Rus\-ski ...
```

При обработке с помощью LaTeX2HTML для каждой группы создается отдельный рисунок. Это дает возможность использовать в одном HTML-документе несколько языков, не задумываясь о кодировке алфавита или способности навигатора воспроизводить редкие шрифты. Однако при использовании рисунков возникают проблемы, аналогичные тем, с которыми мы сталкивались в связи с внутритекстовыми математическими выражениями. Показанные выше строки TeX'овского исходного кода наряду с кириллицей демонстрируют код, созданный по транслитерационной схеме для сингальского языка. Такие методы рассматриваются в следующем разделе.

3.4.5 Преобразование транслитераций с использованием препроцессоров

При подготовке компьютерных документов, например TeX'овских или LaTeX'овских, на языках, алфавиты которых основаны не на латинице, возникают отдельные проблемы. Традиционное решение дает «схема транслитерации»[3], с помощью которой совокупность латинских букв может представлять один символ или слог языка, который нужно воспроизвести. Для подготовки документов используются цепочки литер, формируемые в соответствии со

[1] Команды `\newfont` и `\font` нельзя рассматривать как наиболее предпочтительный способ определения шрифтовых команд в LaTeX 2_ε, в котором используется Новая схема выбора шрифта (NFSS—New Font Selection Scheme). Тем не менее, они остаются работоспособными, часто встречаются в более старых документах, и их удобно использовать для отдельных шрифтов особых размеров.

[2] Речь идет о кириллице в Вашингтонской кодировке, близкой к транслитерации — *Прим. ред.*

[3] Здесь нет необходимости различать «транслитерацию» и «транскрипцию». Подразумевается, что единый термин «схема транслитерации» обозначает оба этих взаимосвязанных понятия.

```
1  \bibitem{belko} {\wncyr Bel\char126koviq, A.A.} #Srusiyaanu"si.mhala
2  "sabdako.saya#N ({\wncyi Russko-Singal\char126ski\char26\
3  Slovar\char126\/}), {\wncyr Rus\-ski\char26\ \char23zyk}, 1983.
4
5  \bibitem{belko} {\wncyr Bel\char126koviq, A.A.}
6  {\SHb\char29a\char8}{\SHb\-\char69i}{\SHb\-\char21a\char0}{\SHa\-\char213u}
7  {\SHb\-\char53i}{\SHa\char11}{\SHb\-\char77a}{\SHb\-\char37a}{\SHb\char53a}
8  {\SHa\-\char237}{\SHb\-\char163a}{\SHa\-\char5\char77a\char7}{\SHb\-\char61a}
9  {\SHb\-\char21a} ({\wncyi Russko-Singal\char126ski\char26\Slovar\char126\/}),
10 {\wncyr Rus\-ski\char26\ \char23zyk}, 1983.
```

Bibliography

1 Белькович, А.А. රුසියානු-සල සබ්දකෝෂය (*Русско-Сингальский Словарь*), Русский Язык, 1983.

2 Clough, Rev. B., සිංහල ඉංග්‍රීසි අකාරාදිය (*Sinhalese-English Dictionary*), Wesleyan Mission Press, Kollupitiya, 1892, facsimile edition by Asian Educational Services, New Delhi, 1982.

Рис. 3.14. Пример предварительной обработки и транслитерации в LaTeX2HTML.

схемой транслитерации. Затем с помощью некоторых других программ выполняется предварительная обработка, чтобы получить TeX'овский код, открывающий доступ к соответствующей литере или к совокупности литер из специально разработанного шрифта.

Пример на рис. 3.14 взят из документа, подготовленного таким способом из рукописи на сингальском языке, распространенном в Южной Индии и Шри-Ланке. Во-первых, здесь показано то, что автор действительно набирал на клавиатуре, причем написанные по-сингальски части ограничены знаками #S и #N (строки 1–3). Во-вторых, показан TeX'овский код после предварительной обработки (строки 5–10) и в заключение — окончательный результат, который можно увидеть в Сети.

В этом примере был использован препроцессор Indica, который поставляется вместе с Sinhala-TeX [↪SINTEX], разработанным Яннисом Хараламбусом. `Indica` работает с двенадцатью различными индийскими языками и диалектами, используя для этого четыре разные схемы транслитерации или транскрипции. Он позволяет создавать выходные документы, которые могут быть обработаны TeX'ом или LaTeX'ом с помощью набора из трех специальных сингальских шрифтов, а также может выполнять непосредственное преобразование в числовые коды литер по стандарту Unicode.

LaTeX2HTML может не только обрабатывать TeX'овские исходные документы, полученные в результате предварительной обработки, такие как на рис. 3.14, но также непосредственно переводить оригинальные «компьютерные рукописи» на язык HTML, *минуя* стадию первоначальной предварительной обработки. Естественно, что для этого необходимо использовать механизм генерации рисунков, но в рассматриваемом случае соответствующим образом размеченные сегменты (например, ограниченные разделителями #S и #N) по-

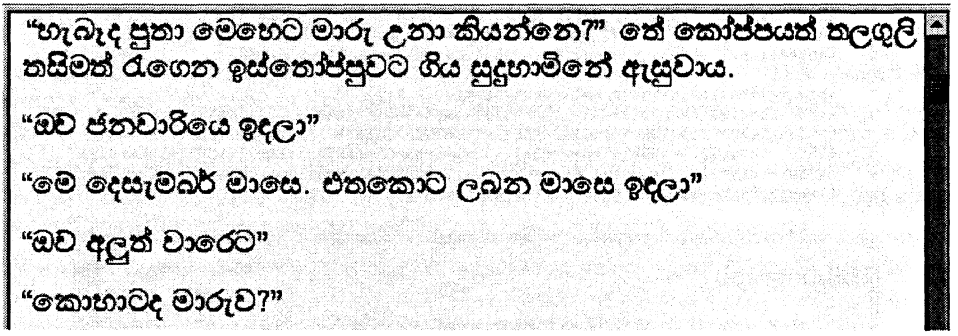


Рис. 3.15. Пример сингалезского (или сингальского) письма, полученного с использованием препроцессора `Indica` для интерпретации транслитерированного исходного документа.

мещаются в один файл с именем `images.pre`. Именно из этого файла препроцессор `Indica` создает обычный файл `images.tex`, который впоследствии обрабатывается LaTeX'ом, `dvips`, `Ghostscript` и т. д. с целью создания рисунков для HTML-страниц. Фрагмент перевода демонстрационного файла с использованием Sinhala-TeX [↪SINTEX] представлен на рис. 3.15.

При трансляции не подвергавшегося предварительной обработке исходного документа из каждого транслитерированного абзаца создается отдельный рисунок, чтобы сохранить размеры рисунков на приемлемом уровне. Как правило, такая стратегия оказывается лучше, чем подход, при котором рисунки создаются из каждой буквы или слога. В последнем случае число рисунков на отдельной странице может достигать нескольких сотен, даже несмотря на возможность многократного использования рисунков на разных страницах. Ширина таких рисунков определяется значением переменной `$PAPERSIZE`. В примере, показанном на рис. 3.15, было использовано значение `$PAPERSIZE = 'b5';`.

Следует отметить, что как и в случае, когда для воспроизведения математических выражений использовались большие рисунки, оригинал транслитерированного исходного текста вставляется в HTML-документ в виде комментария, предшествующего графическому изображению каждого абзаца. Если он не слишком велик, то он также содержится в виде атрибута `ALT` рисунка, возможно, вместе с громоздкими символами, заменяемыми TeX'овскими эквивалентами. Ниже приведен HTML-код, полученный для второго и третьего абзацев на рис. 3.15. Особо отметим способ задания транслитерируемого исходного текста, сначала в виде комментария (строки 1–3 и 9–11), а затем с помощью атрибута `ALT` (строки 7 и 15).

```
1  <!-- INDICA S
2   ''ov janava~riye i"ndhala~''
3   -->
4  <P><IMG
5   WIDTH="186" HEIGHT="22" ALIGN="BOTTOM" BORDER="0"
6   SRC="img2.gif"
7   ALT="\lq\lq ov janava~riye i''ndhala~''"></P>
8
```

```
<!-- INDICA S
 ``me~ dhesa^mbar ma~se. ethakota labana ma~se i"ndhala~''
 -->
<P><IMG
 WIDTH="430" HEIGHT="22" ALIGN="BOTTOM" BORDER="0"
 SRC="img3.gif"
 ALT="\lq\lq me~ dhesa^mbar ma~se. ethakota labana ma~se i''ndhala~''"></P>
```

Независимо от применяемого метода трансляции Sinhala-TEX [↪SINTEX] и `Indica` должны быть доступны в локальной системе, чтобы их можно было использовать. Интерфейсные файлы в подкаталоге `$LATEX2HTMLDIR/Indic-HTML/` поставляются вместе с дистрибутивом LATEX2HTML. В сочетании с соответствующими реализациями на языке Perl они представляют собой LATEX'овские пакеты для различных индийских языков. Для обработки исходных документов необходимо загрузить подходящий пакет с опциями, соответствующими препроцессору и схеме транслитерации. Например, для документа, содержащего сингальский язык в транскрипции *samanala*, можно использовать любую из следующих команд:

```
\usepackage[indica,samanala]{sinhlese}
\usepackage[sinhala,samanala]{indica}
```

Инсталляция Indic-TEX/HTML состоит в обеспечении доступа LATEX'а и LATEX2HTML к файлам языковых интерфейсов (имена файлов могут содержать не более восьми символов в основной части, поэтому некоторые имена выглядят несколько странно). Эти файлы следует скопировать или переместить в TEX'овскую иерархию, либо связать их с ней. После этого способом, рассмотренным в разд. 3.2.3 и 3.2.4, в переменных конфигурации следует указать подкаталог `$LATEX2HTMLDIR/styles/` либо местоположение этих файлов.

Аналогично и файл `indica.perl` должен быть доступен для LATEX2HTML. В начальной части этого файла получают свои значения некоторые Perl-переменные. Возможно, что потребуется их корректировка применительно к локальной инсталляции, а также новые значения могут быть присвоены в файлах `l2hconf.pin` или `.latex2html-init`. Наиболее важна переменная `$INDICA`, которая должна содержать полный путь к препроцессору, если задания самой команды `indica` оказывается недостаточно во время выполнения.

3.4.5.1 Поддерживаемые языки и препроцессоры

Подобным же образом под общим названием Indic-TEX/HTML обеспечивается поддержка других препроцессоров для индийских языков; см. TUGIndia Journal [↪TUGINDIA]. После появления этой статьи была добавлена поддержка препроцессора `itrans` [↪ITRANS] Авинаша Чопде, так что количество языков и схем транслитерации, которые могут быть обработаны LATEX2HTML, увеличилось. В настоящее время Indic-TEX/HTML поддерживает следующие препроцессоры и шрифты:

`Indica` поддерживает бенгали, гуджарати, гурмукхи, хинди, каннада, малаялам, ория, санскрит, сингальский, тамильский, телугу и тибетский языки, используя Velthuis, CSX (расширенный ISO-646), *samanala* [↪SAMANALA]

и стандартизованную LaTeX'овскую транслитерацию. Яннисом Хараламбусом) были разработаны специальные шрифты `sinha`, `sinhb` и `sinhc`. Имеется документация по Sinhala-TeX [↪SINTEX] в виде, оттранслированном LaTeX2HTML [↪SINDOC].

`itrans` поддерживает бенгали, деванагари, гуджарати, хинди, каннада, маратхи, пенджаби (гурмукхи), тамильский, телугу и латинизированный санскрит, используя множество шрифтов, например `ItxBeng` и `BWTI Bengali`, `devnac` и `Xdvng`, `ItxGuj`, `kan`, `pun`, `tel`, `tamil`, `wntml` и `CSUtopia`.

`devnag`, разработанная в Вашингтонском университете, транслирует деванагари в транслитерации Velthuis в хинди, маратхи, непальский и санскрит. В нем используется шрифт `dvng`.

`tamilize` и `tmilize` переводят на тамильский рукописный шрифт с помощью шрифта `wntml`, созданного в Вашингтонском университете.

`patc` и `mm` препроцессоры, используемые совместно с системой *Malayalam*-TeX [↪MALAYALAM] Джероена Хеллингмана. Они поддерживают как традиционное, так и реформированное малаяламское письмо, используя шрифты `mm` и латинизированную разновидность, в которой применяются знаки ударения и диакритические знаки. Они также поддерживают различные транскрипции тамильского языка, используя шрифт `wntml`, и деванараги, используя `devnag` и шрифт `dvng`.

Интерфейсные файлы для каждого из этих препроцессоров поставляются вместе с LaTeX2HTML в составе Indic-TeX/HTML. Сами препроцессоры и любые используемые ими шрифты *должны* отдельно устанавливаться для использования на локальной системе.

Файл `itrans.perl`, предназначенный для препроцессора `itrans` [↪ITRANS], в настоящее время работает со всеми поддерживаемыми им языками. В будущем должна появиться возможность использовать `itrans` как опцию для отдельных языковых пакетов, таких как `Indica`, и обеспечивать выполнение других настроек на конкретный язык. Приведенные ниже команды являются в настоящее время эквивалентными способами загрузки `itrans` [↪ITRANS] для трансляции документа на санскрите с помощью расширенной транслитерации классического санскрита (CSX— Classical Sanskrit Extended) (подобным же образом выглядят соответствующие команды для других языков и схем транслитерации).

```
\usepackage[csx]{itrans}
\usepackage[csx,sanskrit]{itrans}
```

В начале файла `itrans.perl` устанавливаются значения некоторых Perl-переменных. Может возникнуть необходимость их настройки в соответствии с локальной инсталляцией — например, путем присваивания новых значений в файлах `l2hconf.pin` или `.latex2html-init`. Переменная `$ITRANSPATH` гарантирует, что препроцессор сможет найти все необходимые ему ресурсы. Другая

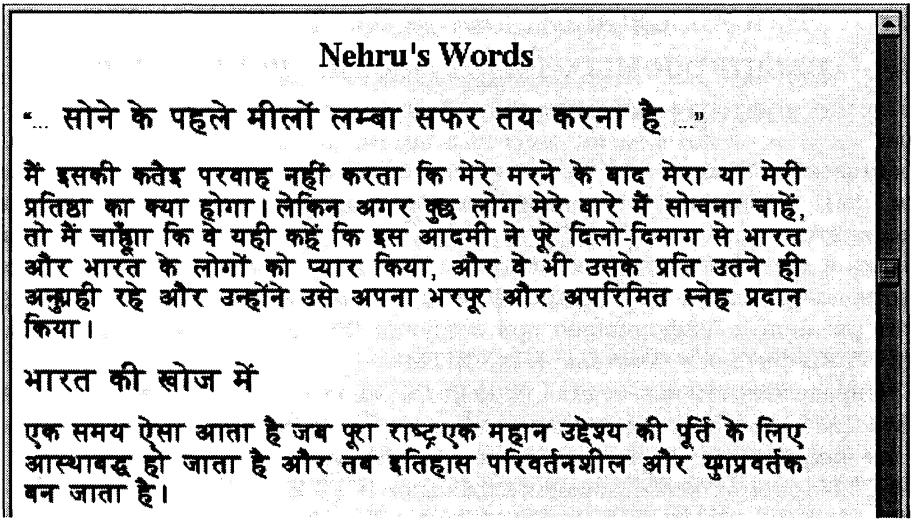


Рис. 3.16. Пример письма деванагари для языка хинди, набранный шрифтом **devnag** и полученный с помощью препроцессора **itrans** [↪ITRANS].

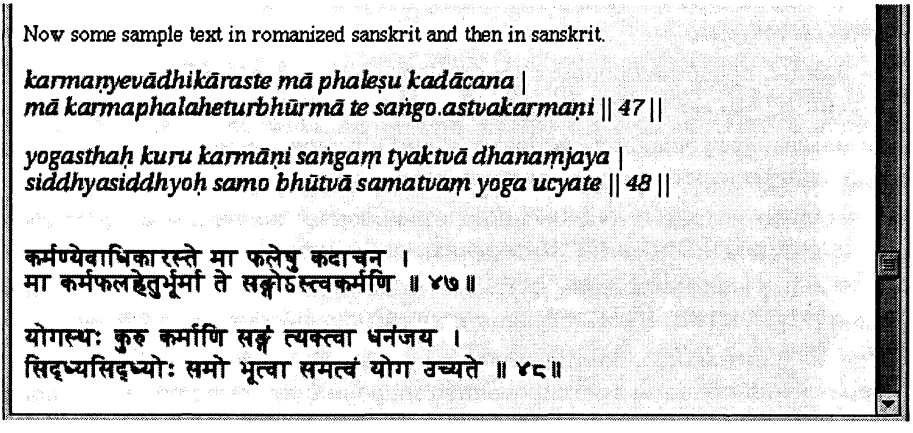


Рис. 3.17. Пример традиционной и латинизированной форм письма на санскрите, полученный с помощью препроцессора **itrans** [↪ITRANS].

переменная **$ITRANS** определяет команду запуска **itrans** [↪ITRANS] в локальной инсталляции; она должна содержать полный путь к препроцессору, если самой команды **itrans** оказывается недостаточно во время выполнения.

На сайте Авинаша Чопде можно получить документацию по **itrans** [↪ITRANS] в переводе LaTeX2HTML [↪ITRANS]. В ней описываются используемые для каждого языка схемы транслитерации. Фрагменты этих страниц показаны на рис. 3.16 и 3.17.

3.5 Использование пакета html для расширяемых LaTeX'овских документов с гипертекстовыми командами

В комплект поставки LaTeX2HTML входит LaTeX'овский пакетный файл с именем `html`, в котором определяются макроимена, команды и окружения. Благодаря этому создаваемые HTML-страницы могут содержать гипертекстовые ссылки и другие специальные эффекты. Некоторые команды при наборе компьютерной рукописи в LaTeX'е не работают, тогда как другие команды используют различную информацию при создании DVI-файла с помощью LaTeX'а или HTML-файлов с помощью LaTeX2HTML.

Определенные в пакете `html` команды можно разделить на несколько классов в соответствии с их назначением и функциональными возможностями. Те из них, которые имеют необязательный аргумент, *не* работают с версией LaTeX 2.09; для их использования необходим LaTeX 2_ε.

- Команды, создающие гиперссылки на внешние Web-страницы или рисунки, т. е. гиперссылки на информацию, которая не является целиком частью данного документа и не появляется при обработке LaTeX'ом того же самого первоисточника.
- Команды, представляющие ту же информацию, что и LaTeX, но которые в версии HTML могут производить дополнительные зрительные эффекты.
- Команды, разрешающие использование альтернативного текста в активных гиперссылках на цитаты и перекрестных ссылках.
- Несколько условных окружений, позволяющих использовать различную информацию в DVI- и HTML-версиях документа.
- Команды, настраивающие визуальную компоновку создаваемых с помощью LaTeX2HTML HTML-страниц и не имеющие аналогов в DVI-версии.
- Специальные макро, помогающие при разработке и отладке, например для управления объемом трассировочной информации, выводимой на экран в процессе обработки документа.
- Макро, разработанные для облегчения использования специальной функции Сегментации документа. Это свойство позволяет LaTeX2HTML обрабатывать большой исходный документ меньшими частями, сохраняя тем не менее в неприкосновенности информацию о перекрестных ссылках между отдельными частями так, будто бы структура HTML-документа была создана из единого первоисточника.

В последующих разделах обсуждаются команды, принадлежащие каждому из этих классов, и приводятся некоторые примеры наиболее эффективных способов их использования.

3.5.1 Гиперссылки на внешние документы

Команды служат для создания гиперссылок на ресурсы Сети \htmladdnormallink и \htmladdnormallinkfoot. Существует также специальная команда \htmladdimg для внутритекстовых гиперссылок на изображения, использующие элемент <IMG>. Она воспроизводит рисунок на HTML-странице как если бы он был частью данного документа.

\htmladdnormallink[*name*]{*link-text*}{*URL*}
\htmladdnormallinkfoot[*name*]{*link-text*}{*URL*}

На HTML-странице навигатор показывает *текст ссылки* как активную гиперссылку, обычно выделяя его другим цветом или подчеркиванием. Щелкнув по *тексту ссылки*, можно перейти к URL назначения. В LaTeX'овской версии *текст ссылки* выглядит обычным образом и ничто не указывает на то, что в этом отдельном фрагменте текста есть что-либо особенное. Необязательный параметр *name* позволяет отметить конкретное место на HTML-странице с помощью имени, несущего смысловую нагрузку.

Вообще говоря, это плохой стиль — вставлять длинные адреса URL непосредственно в основной текст печатного документа; это просто-напросто плохо выглядит, не говоря уже о практических проблемах, связанных с разрывами строк и переносами. При необходимости представить такую информацию в явном виде удобно воспользоваться командой \htmladdnormallinkfoot. На HTML-странице она действует точно так же, как и команда \htmladdnormallink, но при наборе с помощью LaTeX'а вставляет URL в виде сноски.

Команды \htmladdnormallink и \htmladdnormallinkfoot могут использоваться непосредственно в теле документа-первоисточника, хотя делать это не рекомендуется. Вместо этого их лучше всего использовать в расширениях макро, определенных в преамбуле документа, либо во входных файлах, содержащих много подобных определений. Обычно URL указывает на место, расположенное за пределами той области, к которой автор документа имеет непосредственный доступ. Если этот адрес меняется или по какой-либо причине перестает быть действующим, гиперссылку необходимо изменить. Если собрать вместе все присутствующие в документе URL, то это облегчит работу с ними, в том числе и подобное обновление.

Например, в математических примерах, приведенных в разд. 3.3 , эти команды определяются в строках 9–15 и используются в строках 31–33. Как это выглядит в навигаторе, можно увидеть на рис. 3.4 и на других рисунках того же раздела.

\htmlurl{*URL*}

Этот специальный макро обладает следующим свойством. Если макро \url уже определен в другом пакете, например таком, как url, то \htmlurl обращается непосредственно к нему. Аналогично, если другой пакет еще не загружен,

но впоследствии будет загружен, то \htmlurl получит те же самые выполняемые функции и расширение, что и \url. При отсутствии других пакетов \htmlurl печатает URL малым шрифтом пишущей машинки, не поддерживая специальные символы и переносы, при этом \url определяется точно так же, как и \htmlurl.

Чаще всего \htmlurl используется в расширениях других макро, особенно в тех случаях, когда один URL должен использоваться в нескольких окружениях. Например, благодаря командам

```
\newcommand{\rossURL}{\htmlurl{http://www.maths.mq.edu.au/~ross/}}
\newcommand{\thisauthor}{\htmladdnormallink{present author}{\rossURL}}
\newcommand{\RossMoore}{\htmladdnormallinkfoot{Ross Moore}{\rossURL}}
```

ожидаемые гиперссылки генерируются каждый раз, когда в документе используются \rossURL, \thisauthor или \RossMoore. Более того, в LaTeX'овской версии используются соответствующие стили, перенос слов и воспроизведение специальных символов.

\htmladdimg[*attributes*]{*URL*}

Эта команда создает элемент <IMG>, источником данных для которого служит заданный URL. Информация о ширине, высоте и способах выравнивания может быть задана парами «ключ=значение» (***key=value***) в необязательных *атрибутах*. Следовательно, при воспроизведении рисунков их размер может отличаться от исходного. Можно использовать выравнивание для того, чтобы текст «обтекал» рисунок, например, команда \htmladdimage[align=left] помещает рисунок в левой части окна навигатора, в то время как абзацы текста располагаются справа.

Следующий пример взят из библиографии Руководства пользователя LaTeX2HTML. В нем демонстрируется результат, полученный в навигаторе.

```
\newcommand{\AWcsengURL}{http://www.awl.com/cseng}
\newcommand{\AddWes}[1]{\htmladdnormallink{Addison--Wesley}{#1}}
\newcommand{\AWtheLGC}{\AWcsengURL/titles/0-201-85469-4}
\newcommand{\LGCcover}{\AWtheLGC/coversm.gif}

\htmladdimg[ALIGN=RIGHT]{\LGCcover}
\bibitem{goossens:latexGraphics}
Michel Goossens, Sebastian Rahtz and Frank Mittelbach,
\newblock \emph{The }\LaTeX\emph{ Graphics Companion}.
\newblock ISBN 0-201-85469-4, Softcover 608 pages,
 \AddWes{\AWtheLGC/}, 1997.

\bibitem{drakos:bask}
\NikosDrakos,
\newblock Text to Hypertext conversion with \latextohtml.
...
```

3 Michel Goossens, Sebastian Rahtz and Frank Mittelbach,
The LaTeX *Graphics Companion* .
ISBN 0-201-85469-4, Softcover 608 pages, Addison-Wesley, 1997.

4 Nikos Drakos,
Text to Hypertext conversion with LaTeX2HTML.
Baskerville , December 1993, Vol.3, No.2, pp 12-15.
May 1994, CERN, Geneva, Switzerland.
http://cbl.leeds.ac.uk/nikos/doc/www94/www94.html

Обратите внимание на то, как оба пункта располагаются слева от рисунка. Использованные здесь макро основаны на принципе, согласно которому всю информацию, которая может быть использована повторно, например дату издания, следует определять в одном месте, желательно в преамбуле документа.

`\htmlrule[`*attributes*`]`
`\htmlrule*[`*attributes*`]`

Команда `\htmlrule` устанавливает разрыв строки и горизонтальную линейку на HTML-странице; в варианте со звездочкой (*) разрыв строки игнорируется. При выводе страницы на печать не следует слишком часто использовать команды `\hrule` в качестве разделителя, так как обычно бывает достаточно нескольких дополнительных пробелов. Однако навигаторы часто оставляют слишком много пустого места, так что для зрительного восприятия более привлекательным будет дробление этого пустого пространства с помощью линейки. Такие свойства, как ширина, толщина и тень, задаются с помощью списка *атрибутов*, составленного из пар «ключ=значение» (*key=value*). Допускается указывать только значение, если по нему ключ может быть однозначно определен, как в следующем примере:

```
<BR CLEAR="LEFT">
<HR WIDTH="50%" SIZE="5" NOSHADE>      \htmlrule[50\% 5 noshade left]
```

3.5.2 Дополнительные возможности HTML

`\latextohtml`

Создает логотип LaTeX2HTML, как в примере из предыдущего раздела. Для родственных логотипов, таких как TeX, LaTeX, $\mathcal{AMS}$ и XY, существуют встроенные в LaTeX2HTML специальные команды. Доступ ко многим из них возможен из пакета `texnames`.

`\begin{makeimage}`

Это окружение вызывает создание рисунка из своего содержимого и воспроизведение его с шириной, определенной в переменной `$PAPERSIZE`. Поскольку

по умолчанию трансляция нестандартного окружения заключается в создании рисунка, данное окружение в основном используется для того, чтобы подавить стандартный способ трансляции. Например, таблицы иногда выглядят лучше, если их полностью набрать в LaTeX'е, а в разбитом на абзацы тексте может оказаться желательным специальное форматирование. При использовании пакета `seminar` часто лучше оперировать с окружениями `slide` как с рисунками. В содержимом окружения выполняется поиск команд `\label`, благодаря чему перекрестные ссылки могут действовать как активные гиперссылки.

Обычно окружение `figure` целиком преобразуется в один рисунок. Однако если в нем содержится одно или несколько подокружений `makeimage`, то содержимое окружения `figure` обыкновенно конвертируется в HTML как отдельная секция `<DIV>`. Рисунки создаются только из содержимого каждого подокружения `makeimage`. В частности, можно использовать пустое окружение `makeimage` для того, чтобы предотвратить преобразование окружения `figure` в рисунок.

`\htmlimage{`*graphic effects*`}`

Эта команда вызывает создание рисунка из окружения, которое в обычных условиях не должно обрабатываться таким способом. Она может также использоваться для достижения специальных графических эффектов при создании рисунка.

Команда `\htmlimage` действует на окружения `figure` и `table`, `displaymath`, `equation`, `eqnarray`, `math`, внутритекстовые математические формулы и выровненные окружения пакетов `amsmath` и `amstex` точно так же, как на окружение `slide` пакета `seminar` и любые окружения других пакетов, не имеющие специальной поддержки в LaTeX2HTML. Для того чтобы инициировать создание рисунка, команда `\htmlimage` со своим (возможно, пустым) аргументом может находиться в любом месте внутри окружения. LaTeX игнорирует как саму эту команду, так и ее аргумент.

Перечисленные ниже аргументы *графических эффектов* позволяют определить специальные эффекты, которые должны быть использованы при создании конкретного рисунка.

`height=`*pixels* задает значение атрибута `HEIGHT` дескриптора `<IMG>` равным *pixels*.

`width=`*pixels* задает значение атрибута `WIDTH` дескриптора `<IMG>` равным *pixels*.

`align=`*option* устанавливает атрибут `ALIGN` дескриптора `<IMG>`. Допустимыми значениями являются `TOP`, `BOTTOM`, `MIDDLE`, `LEFT`, `RIGHT`, `CENTER`.

`external` создает гиперссылку на внешний файл с изображением, вместо того чтобы вставлять рисунок внутрь текста.

`scale=`*scale-factor* масштабирует рисунок в соответствии с заданным значением масштабного коэффициента *scale-factor*.

`thumbnail=`*scale-factor* объединяет две предыдущие опции `external` и `scale` и создает вариант рисунка в виде «иконки» с размером, определяемым масштабным коэффициентом *scale-factor*. Эта иконка располагается внутри текста и используется как активная кнопочная гиперссылка, указывающая на полномасштабную версию рисунка.

`map=`*URL* создает карту на серверной стороне;
подробности см. в Руководстве пользователя LaTeX2HTML.

`usemap=`*URL* создает карту на клиентской стороне;
подробности см. в Руководстве пользователя LaTeX2HTML.

`flip=`*option* вращает или переворачивает рисунок в соответствии с допустимыми опциями (*option*):
`leftright`, `topbottom`, `rotate90` (или `r90`), `rotate270` (или `r270`).

`transparent` делает фон рисунка прозрачным, когда по умолчанию он должен оставаться непрозрачным.

`notransparent` или `no_transparent` сохраняет фон рисунка непрозрачным, когда по умолчанию он должен быть прозрачным.

`antialias` предписывает выполнение сглаживания символов шрифта независимо от того, какой режим установлен по умолчанию для данного рисунка.

`noantialias` или `no_antialias` подавляют использование сглаженных шрифтовых символов независимо от режима, установленного по умолчанию для данного рисунка.

`\begin[`*style-sheet information*`]`

Использование необязательного аргумента в LaTeX'овской команде `\begin` проиллюстрировано в примерах разд. 3.3.4. Это имеет смысл только при генерации кода для версии HTML 4.0, в которой обеспечивается поддержка языка CSS. В LaTeX'е необязательный аргумент игнорируется.

Информация стиля (*style-sheet information*) записывается в автоматически создаваемый файл стиля `css` для последующего использования в атрибутах `CLASS` и `ID` . Она может включать в себя имена дополнительных классов (`CLASS`), к которым будет принадлежать последующее окружение, а также тот предварительно заданный `CLASS`, к которому оно принадлежит по умолчанию. С помощью атрибута `ID` можно задать информацию, которая должна применяться только к последующему окружению.

`\htmlsetstyle[`*tags*`]{`*class*`}{`*style-info*`}`
`\htmladdtostyle[`*tags*`]{`*class*`}{`*style-info*`}`

Эти команды в LaTeX'е игнорируются. В LaTeX2HTML они предназначены для того, чтобы записать информацию о стиле (*style-info*) в таблицу стилей CSS. Впоследствии она будет использована для элементов *tags* и класса *class*. Параметр *tags* может быть иерархическим списком элементов, напри-

мер TABLE.TR.TD, или единственным элементом, и даже может быть пустым. Класс также может быть пустым. В качестве информации о стиле (*style-info*) может быть задана любая строка. В настоящее время единственная проверка правильности информации заключается в преобразовании «=» в «:» и замене каждой запятой «,» пробелом.

`\htmllanguagestyle[`*class*`]{`*language*`}`

Назначение этой команды состоит в том, чтобы включить класс *class* в число атрибутов элементов, у которых язык (*language*) является значением атрибута LANG. Если класс *class* не задан, то для параметра *language* используется умалчиваемое значение, равное коду языка по ISO 639. Для использования специфической информации о стиле следует использовать команду \htmlsetstyle. Например, следующий фрагмент кода обеспечивает запись входных данных в стиль CSS, чтобы с помощью пакета babel в документе, в котором немецкий язык не является основным, выделить красным цветом написанные по-немецки абзацы.

```
\htmllanguagestyle{german}
\htmlsetstyle[P]{de}{color:red}
```

`\htmlborder{`*attributes*`}`

Эта команда, как и команда \htmlimage, действует на то окружение, в котором она содержится, и при этом требует, чтобы содержимое окружения было взято в рамку. Она действует на окружения minipage, tabbing, table, и makeimage, а также на окружения, о которых шла речь раньше при обсуждении команды \htmlimage. Кроме того, она действует на окружения типа theorem.

Эффект рамки достигается за счет того, что содержимое помещается внутрь одной ячейки <TD> конструкции <TABLE> языка HTML. Аргумент *attributes* может включать в себя любой атрибут, допустимый для элемента <TABLE> языка HTML, как, например, толщина рамки (значение, определяющее толщину, и передаваемое атрибуту BORDER), выравнивание, ширина, CELLSPACING и CELLPADDING.

`\begin{htmllist}[`*marker*`]`
`\htmlitemmark{`*marker*`}`

Это окружение заменяет собой окружение description, формирующее список описаний. В качестве маркеров элементов списка в нем используются цветные шарики. В LaTeX'е это выглядит как обычный список обозначений. Это окружение определено в пакете htmllist, который обычно загружается одной командой вместе с пакетом html. Для установки цветов используется аргумент *marker*, который может принимать значения RedBall, OrangeBall, YellowBall,

GreenBall, BlueBall, PurpleBall, PinkBall или WhiteBall. Они соответствуют имеющимся изображениям навигационных иконок.

```
\usepackage{html,htmllist}

\begin{htmllist}[WhiteBall]
\item[Item 1:]This will have a white ball.
\item[Item 2:]
 This will also have a white ball.
\htmlitemmark{RedBall}%
\item[Item 3:]This will have a red ball.
\end{htmllist}
```

- **Item 1:**
  This will have a white ball.
- **Item 2:**
  This will also have a white ball.
- **Item 3:**
  This will have a red ball.

\strikeout{*text*}

Эта команда вызывает применение к тексту *text* элементов <STRIKE>, в результате чего ненужная информация вычеркивается одной чертой. Сам *text* просто заимствуется из набранного в LaTeX'е варианта.

```
\item can deal sensibly with \strikeout{at least the \emph{Common}
\LaTeX{} \emph{Commands} summarised at the back of\\}%
virtually all of the concepts and commands described in
the \LaTeX{} \htmlcite{blue book}{lamp:latex}, where there is
a meaningful interpretation appropriate to an \texttt{HTML} document.
```

- can deal sensibly with ~~at least the *Common* LaTeX *Commands* summarised at the back of~~ virtually all of the concepts and commands described in the LaTeX blue book, where there is a meaningful interpretation appropriate to an HTML document. Also many other LaTeX

3.5.3 Альтернативный текст для гиперссылок

В HTML-документе нумерация страниц не имеет значения, поскольку все перемещения выполняются посредством активных гиперссылок. Так как гиперссылки используются и для перекрестных ссылок, нумерация разделов, рисунков и ссылок на литературу также становится менее важной. В дополнение к LaTeX'овским командам \ref и \cite, команды, описываемые в данном разделе, предоставляют альтернативные способы работы с перекрестными ссылками и ссылками на литературу.

\hyperref[*keyword*] ... *arguments* ...

Эту команду рекомендуется использовать вместо LaTeX'овских команд \ref и \pageref. Необязательный параметр *keyword* определяет, какая из форм должна использоваться при обработке документа LaTeX'ом; подразумеваемым значением является \ref.

\hyperref[ref]{*link-text*}{*pre-latex*}{*post- latex*}{*label*}
\hyperref{*link-text*}{*pre-latex*}{*post-latex*}{*label*}

\hyperref[page]{*link-text*}{*pre-latex*}{*post- latex*}{*label*}
\hyperref[pageref]{*link-text*}{*pre-latex*}{*post- latex*}{*label*}

\hyperref[no]{*link-text*}{*latex-text*}{*label*}
\hyperref[noref]{*link-text*}{*latex-text*}{*label*}

В этих парах оба варианта приводят к одинаковым результатам. Текст ссылки (*link-text*) используется только в HTML-страницах как текст, связанный с гиперссылкой, указывающей на местонахождение команды \label{*label*} в том же самом или другом документе, созданном с помощью LaTeX2HTML. Компоненты *pre-latex* и *post-latex* используются только в LaTeX'е. Они располагаются с любой стороны связанного с меткой *label* текста (или номера) ссылки. В двух последних командах LaTeX не использует аргумент *label*, хотя в HTML-версии гиперссылка по-прежнему создается. В следующем примере ясно видны различия в использовании текста разных аргументов LaTeX'ом (слева) и транслятором LaTeX2HTML (внизу).

...HTML pages for which there is no direct LaTeX counterpart. Most of these commands are discussed in detail in Section 4.8.

```
...\texttt{HTML} pages for which there
is no direct \LaTeX{} counterpart.
Most of these commands are discussed
in detail in \hyperref{a later section}
{Section~}{}{misceffects}.
```

The following commands implement effects on the HTML pages for which there is no direct LaTeX counterpart. Most of these commands are discussed in detail in a later section.

При обработке LaTeX'ом используются макро, соответствующие каждой опции, а именно: \hyperrefref, \hyperrefpageref, \hyperrefpage, \hyperrefno и \hyperrefnoref. Эти команды обрабатывают аргументы, следующие за первым необязательным. Команда \hyperrefdef всегда действует стандартным способом без каких-либо вариантов. Благодаря этому информация, которая представляется с помощью какой-либо специальной опции, может быть использована другим способом, причем для этого достаточно переопределить одну команду, а не всю структуру \hyperref.

\hyperref[hyper]{*URL*}{*category*}{*name*}{*text*}
\hyperref[hyper][*label*]{*text*}

\hyperref[html]{*link-text*}{*pre-latex*}{*post-latex*}{*label*}

Необходимость введения этих дополнительных вариантов обусловлена тем, что в разработанном Себастианом Ратцем пакете hyperref (см. разд. 2.3) также используется команда с именем \hyperref. В первом варианте отсутствует символическая метка *label*, а используется он аналогично команде \htmladdnormallink, хотя в нем и допускаются структуры в адресате гипер-

ссылки. Во втором варианте наряду с обязательным аргументом, задающим текст гиперссылки, используется необязательный аргумент, позволяющий задать символическую метку. Если команду `\hyperref` в любом из этих двух вариантов каждый раз использовать с опцией `[hyper]`, то LaTeX2HTML будет правильно обрабатывать документы, загружающие пакет `hyperref`. С другой стороны, если команда начинается с `\hyperref[html]`, то это означает, что в данном случае необходимо использовать версию команды `\hyperref` из пакета `html`. В то время как LaTeX обрабатывает обязательные аргументы, для обработки этих опций используются команды `\hyperrefhtml` и `\hyperrefhyper`.

`\htmlref{`*link-text*`}{`*label*`}`
`\htmlref[ext]{`*link-text*`}[`*prefix*`]{`*label*`}`

Это более простой способ задания гиперссылки в HTML-версии. Он эквивалентен команде `\hyperref[noref]`, причем *latex-text* совпадает с *link-text*. Например,

With \htmlref it's easy to make links.

```
With \verb|\htmlref|
\htmlref{it's easy to make links}
{fig:example}.
```

With \htmlref it's easy to make links.

Опция `ext` означает, что место, на которое делается ссылка, находится во внешнем документе. Как использовать этот механизм, а также необязательный аргумент *prefix* объясняется ниже в описаниях команд `\externalref` и `\externallabels`. В случае альтернативной обработки LaTeX'ом макро `\htmlrefdef` и `\htmlrefext` могут быть переопределены так, чтобы обрабатывать все аргументы, кроме первого необязательного.

`\hypercite[`*keyword*`]` ... *arguments* ...

Эту команду рекомендуется использовать вместо LaTeX'овских команд `\cite` и `\nocite`. Необязательный ключ *keyword* определяет ту форму, которая должна применяться при обработке документа LaTeX'ом; по умолчанию используется `\cite`.

`\hypercite[int]{`*link-text*`}{`*latex-text*`}{`*opt-latex*`}{`*label*`}`
`\hypercite[cite]{`*link-text*`}{`*latex-text*`}{`*opt- latex*`}{`*label*`}`
`\hypercite{`*link-text*`}{`*latex-text*`}{`*opt-latex*`}{`*label*`}`

`\hypercite[nocite]{`*link-text*`}{`*latex-text*`}[`*prefix*`]{`*label*`}`
`\hypercite[no]{`*link-text*`}{`*latex-text*`}[`*prefix*`]{`*label*`}`
`\hypercite[ext]{`*link-text*`}{`*latex-text*`}[`*prefix*`]{`*label*`}`

При наборе в LaTeX'е все команды первой группы действуют одинаковым способом — сначала данные помещаются в аргумент *latex-text*, после чего выпол-

няется команда \cite[*opt-latex*]{*label*}. Вторая группа команд помещает команду \nocite{*label*} вместе со следующими за ней данными в *latex-text* для того, чтобы создать элемент библиографического списка, не ставя на печатной странице какого-либо явного знака ссылки.

Аргументу *link-text* на HTML-странице соответствует гиперссылка, указывающая на место, адрес которого определяется символической меткой *label*. Оно может находиться на странице со списком литературы, присоединенной к создаваемому Web-документу, либо на каком-либо внешнем сайте. Использование символических меток, доступных на внешних сайтах, и применение префиксов (*prefix*) объясняется в разделе, посвященном команде \externallabels.

При обработке LaTeX'ом для каждой опции существует соответствующий ей макро, а именно: \hyperciteint, \hypercitecite, \hypercitenocite, \hypercitenо и \hyperciteext. Эти команды обрабатывают аргументы, следующие за первым необязательным. Команда \hypercitedef всегда действует стандартным способом без каких-либо вариантов. Благодаря этому информация, которая представляется с помощью какой-либо специальной опции, может быть использована другим способом, причем для этого достаточно переопределить одну команду, а не всю структуру \hypercite.

\htmlcite{*link-text*}{*label*}
\htmlcite[ext]{*link-text*}[*prefix*]{*label*}

Этот способ позволяет более просто определить на HTML-странице цитату, на которую указывает гиперссылка. Он эквивалентен команде \hypercite[nocite], причем *latex-text* совпадает с *link-text*; однако при наличии опции ext LaTeX пропускает элементы библиографии. Об использовании префикса *prefix* говорится ниже в описании команды \externallabels. В случае альтернативной обработки LaTeX'ом макрокоманды \htmlcitedef и \htmlciteext могут быть переопределены так, чтобы обрабатывать все аргументы кроме первого необязательного.

Допускаются также ссылки на объекты, идентифицируемые с помощью символических меток. Такие метки могут находиться в других Web-документах, созданных с помощью LaTeX2HTML, причем эти документы могут даже находиться на удаленных сайтах. Для того чтобы создать гиперссылку, указывающую на определенные места на удаленных HTML-страницах, необходимо задать следующую команду:

\externallabels[*prefix*]{*URL*}{*local copy of* labels.pl *file*}

Параметр *URL* устанавливает то удаленное местоположение, где должны находиться нужные HTML-страницы. Эти страницы становятся объектом гиперссылок в локальном документе. Прежде всего необходимо скопировать файл, который обычно называется labels.pl и который содержит информацию о символических ссылках(*label*), использовавшихся при создании удаленных страниц. (См. с. 118, где при обсуждении функционирования LaTeX2HTML упоминается

о существовании этого файла.) Во избежание совпадения имен у меток, используемых в различных документах, целесообразно задать необязательный аргумент *prefix*, который будет использоваться в качестве префикса при каждой ссылке на метку *label* с данного сайта. Созданная копия файла `labels.pl` может быть расположена в любом удобном месте локальной системы. Полный путь и имя этого файла должны быть заданы в качестве последнего аргумента команды `\externallabels`.

`\externalref[`*prefix*`]{`*label*`}`

Эта команда является аналогом команды `\ref`, но для метки, определенной в удаленном документе. При наборе в LaTeX'е метка *label* будет проигнорирована. В HTML-версии создается гиперссылка, в качестве видимого маркера которой используется маленькая иконка. Для текстовых гиперссылок вместо данной команды следует использовать `\htmlref` или `\hyperref[noref]`. Об использовании префикса *prefix* см. приведенное выше описание команды `\externallabels`.

`\externalcite[`*prefix*`]{`*label*`}`

Это эквивалент команды `\nocite` для случая, когда библиографические данные находятся в удаленном документе. При наборе в LaTeX'е метка *label* будет проигнорирована. В HTML-версии создается гиперссылка, в качестве видимого маркера которой используется маленькая иконка. Для текстовых гиперссылок вместо данной команды следует взять `\htmlcite` или `\hypercite[ext]`. О назначении префикса (*prefix*) см. приведенное выше описание команды `\externallabels`. Пример использования этих команд обсуждается в разд. 3.5.6.

3.5.4 Условные окружения

Часто бывает так, что некоторые данные относятся только к версии документа, набираемой в LaTeX'е или, наоборот, только к HTML-версии, создаваемой с помощью LaTeX2HTML. При этом, скорее всего по техническим причинам, может оказаться необходимым, чтобы некоторый макро имел разные расширения для этих двух версий. В подобных ситуациях используются специальные окружения.

`\begin{htmlonly}`

Содержимое окружения `htmlonly` используется только в тех случаях, когда документ обрабатывается транслятором LaTeX2HTML и совершенно игнорируется LaTeX'ом.

`\begin{latexonly}`

Содержимое окружения `latexonly` используется только в тех случаях, когда документ обрабатывается LaTeX'ом и совершенно игнорируется транслятором LaTeX2HTML. Обратите внимание на то, что содержимое этого окружения попадает внутрь TeX'овской группы `\begingroup... \endgroup`. При помощи этого окружения нельзя задавать значения длин или счетчиков, а также определять расширения макро, предназначенных для использования только в LaTeX'е. В случае возникновения такой необходимости следует применять описанную ниже конструкцию.

```
%begin{latexonly}
...
...
%end{latexonly}
```

Эти специальные комментарии имеют значение только для LaTeX2HTML. Он отбрасывает их вместе с той частью исходного документа, которую они окружают. С другой стороны, поскольку LaTeX рассматривает «`%`» просто как знак комментария, он полностью игнорирует эти строки. Поскольку при этом содержимое остается нетронутым, то любые определения `\newcommand` или `\newenvironment`, а также установки значений длин или счетчиков будут оказывать свое обычное действие.

`\html{`*text for HTML only*`}`
`\latex{`*text for LaTeX only*`}`
`\latexhtml{`*latex-text*`}{`*html-text*`}`

Это краткие формы рассмотренных ранее окружений, предназначенные для малых фрагментов текста. Сама команда и ее аргументы прочитываются и, возможно, обрабатываются до сбрасывания. При вводе больших порций исходного документа лучше использовать окружения, чтобы наверняка избежать нежелательных побочных эффектов. Однако из-за того, как TeX читает и размечает введенный первоисточник, могут возникать ситуации, в которых `\html` работает, а `htmlonly` — нет. Такое происходит, например, при применении специальных символов, вроде «`&`».

`\begin{imagesonly}`

Содержимое окружения `imagesonly` записывается в файл `images.tex` для совместного использования с LaTeX'овским кодом, который должен быть набран при создании изображений для HTML-документа. Например, таким способом можно для всех рисунков задать цвета фона и переднего плана. Эта функция иногда бывает необходима для того, чтобы задействовать требуемое макрорасширение.

`\begin{comment}`

Содержимое окружения comment всегда игнорируется как LaTeX'ом, так и LaTeX2HTML.

`\begin{rawhtml}`

Окружение rawhtml позволяет непосредственно вставить исходный HTML-код в то место документа (относительно текста и абзацев), где он располагается в оригинале. При этом правильность вставляемого кода и его пригодность для используемой версии HTML не проверяются. Решение этих проблем полностью возлагается на автора документа. В описании следующей команды \HTMLcode рассказывается о более надежном способе конструирования HTML-кода.

`\HTMLcode[`*attributes*`]{`*tag*`}`
`\HTMLcode[`*attributes*`]{`*tag*`}{`*content*`}`

Эта команда создает элемент языка HTML в соответствии с заданным параметром *tag* и с атрибутами, определяемыми списком атрибутов (*attributes*), при условии, что этот элемент допустим в выбранной версии HTML. В тех случаях, когда создаваемой метке должна соответствовать закрывающая метка, содержимое задается аргументом *content*. Неправильные атрибуты метки *tag* удаляются из списка, но если сама метка *tag* задана неправильно, то команда вообще не выполняется.

Атрибуты (*attributes*) могут быть заданы в виде списка пар ключ=значение (*key=value*), разделенных пробелами, запятыми или символами разделения строк. Допустимо также задавать только сами желаемые значения, если при этом можно определить, какой из атрибутов должен получить то или иное значение. Например,

```
<HR WIDTH="50%" SIZE="3"
    NOSHADE ALIGN="center">          \HTMLcode[50\% 3 noshade center]{HR}
```

В каждом из аргументов команды \HTMLcode (*content*, *tag* и *attributes*) присутствует макрорасширение. Благодаря этому при использовании повторяющихся структур ввод с клавиатуры сводится к минимуму. Рисунок 3.18 демонстрирует, как это можно использовать на практике.

В отличие от LaTeX'а LaTeX2HTML не игнорирует присутствующие в LaTeX'овском исходном документе символы конца строки. Наоборот, они сохраняются в создаваемых HTML-файлах. Как правило, это не оказывает влияния на создаваемое навигаторами изображение. Однако это существенно влияет на расположение информации в html-файлах, поскольку они предназначены для чтения человеком. Это объясняет, почему в предыдущем примере так много макроопределений.

Перечень элементов, которые могут быть использованы с командой \HTMLcode, может быть расширен. С этой целью для каждого нового элемента на языке Perl определяется «регулярное выражение», которое будет подби-

```
\newcommand{\myalign}{center}\newcommand{\mylist}{UL}
\newcommand{\myitem}[2]{\HTMLcode[disc]{LI}{\simpletest{#1}{#2}}}
\newcommand{\simpletest}[2]{%
 \HTMLcode{#1}{ a simple test of ``#2'',} using \HTMLcode{CODE}{<#1>} .}
\newcommand{\tableopts}{10,border=5}
\newcommand{\tablelist}[4][left]{\HTMLcode[#1]{DIV}{%
 \HTMLcode[\tableopts]{TABLE}{\HTMLcode[bottom]{CAPTION}{#3}%
  \HTMLcode{TR}{\HTMLcode{TD}{\HTMLcode{#2}{#4}}}%
 }}\HTMLcode[all]{BR}}

\tablelist[\myalign]{\mylist}{%
 \textbf{A listing of the different text styles available in HTML 3.2}}{%
  \myitem{B}{bold-face}\myitem{I}{italics}\myitem{TT}{teletype-text}
  \myitem{U}{underlining}
  \HTMLcode[circle]{LI}{\simpletest{STRIKE}{strikeout}}
  \myitem{EM}{emphasis style}\myitem{STRONG}{strong style}
  \myitem{CODE}{code style}\myitem{CITE}{citation style}
  \myitem{DFN}{definition style}
  \HTMLcode[square]{LI}{\simpletest{SAMP}{sample style}}
  \HTMLcode[square]{LI}{\simpletest{KBD}{keyboard style}}
  \myitem{VAR}{variable style}}
```

- **a simple test of ``bold-face'',** using <B> .
- *a simple test of ``italics'',* using <I> .
- a simple test of ``teletype-text'', using <TT> .
- a simple test of ``underlining'', using <U> .
- ~~a simple test of ``strikeout'',~~ using <STRIKE> .
- *a simple test of ``emphasis style'',* using <EM> .
- **a simple test of ``strong style'',** using <STRONG> .
- a simple test of ``code style'', using <CODE> .
- *a simple test of ``citation style'',* using <CITE> .
- a simple test of ``definition style'', using <DFN> .
- a simple test of ``sample style'', using <SAMP> .
- a simple test of ``keyboard style'', using <KBD> .
- *a simple test of ``variable style'',* using <VAR> .

A listing of the different text styles available in HTML 3.2

Рис. 3.18. Использование команд `\HTMLcode` в LaTeX2HTML.

рать имена допустимых для него атрибутов. При этом для каждого атрибута также должно существовать регулярное выражение, с помощью которого для него будут выбираться возможные значения. Если читателю интересно узнать, как это достигается, он может изучить файлы `versions/html3_2.pl` и `versions/html4_0.pl`, которые входят в комплект поставки LaTeX2HTML.

3.5.5 Навигация и компоновка HTML-страниц

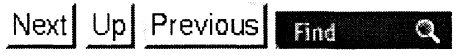
Описываемые в данном разделе команды полностью игнорируются LaTeX'ом.

`\htmladdtonavigation{`*code*`}`

Эта команда предоставляет простой способ расширения навигационной панели, которую LaTeX2HTML создает автоматически. В качестве видимого указателя

гиперссылки в аргументе *code* обычно используются другие команды, с помощью которых можно отображать текст или рисунок. Благодаря тому, что при каждом использовании команды \htmladdtonavigation к навигационной панели добавляются новые инструменты, достигается возможность весьма многосторонней настройки. Такая настройка действует только в пределах одного документа. При необходимости распространить настройку на все документы, которые должны обрабатываться с помощью LaTeX2HTML, можно изменить код навигационной панели, содержащийся в файле latex2html.config.

```
\htmladdtonavigation{\htmladdnormallink
 {\htmladdimg[bottom]{http://bonk.ethz.ch/icon-files/find.gif}}
 {http://bonk.ethz.ch/asearch.html}}
```



Next: Geologic Setting **Up**: Nature and cause of **Previous**: Nature and cause of

В этом примере, предоставленном Юлием Уортманном, обеспечен удобный доступ к механизму поиска, позволяющий облегчить поиск ключевых слов в Web-страницах, содержащихся в документе.

\tableofchildlinks[*option*]
\tableofchildlinks*[*option*]

Существует возможность создания больших HTML-страниц, содержащих подразделы, пункты и еще более мелкие структурные элементы. В зависимости от уровня используемых ссылок, такие страницы обычно начинаются с таблицы активных гиперссылок на содержащиеся там подразделы. Приведенные выше команды позволяют передвинуть такое «мини-оглавление» в то место, где находится команда \tableofchildlinks, либо вообще его пропустить. Аргумент *option* может принимать следующие допустимые значения:

`off` пропустить мини-оглавление на данной HTML-странице.
`none` пропустить мини-оглавление на данной и всех последующих страницах.
`on` восстановить мини-оглавление для данной HTML-страницы.
`all` восстановить мини-оглавление для данной и всех последующих страниц.

В варианте команды со звездочкой * пропускается элемент разрыва (<BR>), который в противном случае был бы вставлен перед мини-оглавлением.

\htmlinfo[*option*]
\htmlinfo*[*option*]

Эта команда позволяет изменять место расположения технической информации «Об этом документе». По умолчанию она размещается на отдельной стра-

```
\htmlrule
\htmlhead[center]{subsubsection}{About this document.}
\htmladding[ALIGN=right]{../logos/aaslogo.gif}
\begin{flushleft}
This Web site has been constructed by \RossMoore, using the material
supplied by the authors and the conference organisers.

Most of this material will become available in book form, published by
\OzAcadSci, during 1997.
\end{flushleft}

\htmlrule[all,width=350]\htmlinfo*
```

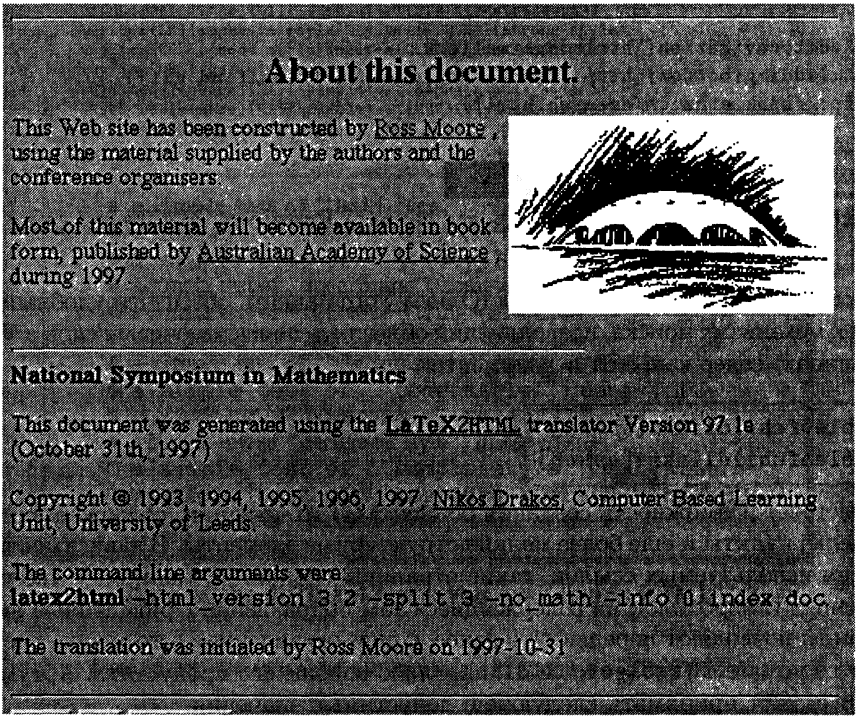


Рис. 3.19. Использование команды `\htmlinfo` в LaTeX2HTML.

нице в конце документа. Альтернативное положение определяется командой `\htmlinfo`. В варианте со звездочкой * пропускается заголовок, благодаря чему автоматически создаваемую информацию можно вставить как часть документа, а дополнительные данные присоединить позже. При использовании опции `off` или `none` информация пропускается полностью.

С помощью этих опций техническую информацию можно представить в более привлекательном и выразительном виде. Пример использования этой команды для оформления титульной страницы можно увидеть на рис. 3.19 (и в [↪MATHSYMP]).

\bodytext{*attributes*}
\htmlbody[*attributes*]

Каждая из этих команд воздействует на содержимое элемента <BODY>, который необходим на каждой HTML-странице. В самом общем случае они используются для определения цветов переднего плана и фона. Этим же способом могут быть заданы и цвета для гиперссылок. Команда \bodytext используется для переопределения параметров, задаваемых в навигаторе по умолчанию; все подлежащие изменению атрибуты должны быть одновременно определены в аргументе *attributes*. Это определение будет действительно как для текущей HTML-страницы, так и для всех последующих страниц. В команде \htmlbody можно присвоить заданное значение *value* отдельному атрибуту, сохранив все остальные предыдущие установки. Эти команды можно применять несколько раз на одной и той же странице, причем каждая новая команда будет дополнять (\htmlbody) или подавлять (\bodytext) действие предыдущих команд.

Цвета текста и фона могут быть заданы также командами \color и \pagecolor, при условии что предварительно был загружен LaTeX'овский пакет color.

\htmlbase{*URL*}

Эта команда позволяет указать в элементе <BASE> каталог, который отличается от того, где находится текущий документ.

3.5.6 Пример связывания нескольких внешних документов

В этом разделе показано, как извлечь пользу из описанных в разд. 3.5 гипертекстовых расширений при работе с составными документами.

За отправную точку принимается показанный на рис. 3.1 исходный LaTeX'овский документ, который в демонстрационных целях делится на четыре поддокумента. Как показано на рис. 3.20, к этим поддокументам относятся «основной» файл (ex20.tex) и три вторичных файла (ex21.tex, ex22.tex и ex2bib.tex). Все эти файлы сначала «пропускаются» через LaTeX, а затем *в определенном порядке* — через LaTeX2HTML. В самом деле, поскольку для обращения к элементам внешних документов используются перекрестные ссылки (с помощью команд \externalref и \externallabels, представленных в разд. 3.5.3), необходимо сначала обработать вторичные файлы ex21.tex, ex22.tex и ex2bib.tex, и только после этого браться за основной файл ex20.tex.

По умолчанию LaTeX2HTML записывает создаваемые им файлы в подкаталог, имя которого совпадает с именем исходного файла. Поэтому по окончании

Основной файл (ex20.tex)

```
\documentclass{article}
\usepackage{html}
%begin{latexonly}
\usepackage[T1]{fontenc}
%end{latexonly}
\usepackage[dvips]{graphicx}
\usepackage{francais}
\begin{document}
\begin{center}\Large
  Exemple d'un document composé\\
\end{center}

\htmladdnormallink{Les Images}{../ex21/ex21.html}

\externallabels{../ex21}{../ex21/labels.pl}
Référence à une figure externe~\externalref{Fpsfig}.

\htmladdnormallink{Les tableaux}{../ex22/ex22.html}

\externallabels{../ex22}{../ex22/labels.pl}
Référence à un tableau externe~\externalref{tab-exa}.

\htmladdnormallink{La bibliographie}%
                  {../ex2bib/ex2bib.html}
\end{document}
```

Файл, содержащий рисунки (ex21.tex)

```
\documentclass{article}
\usepackage{html}
%begin{latexonly}
\usepackage[T1]{fontenc}
%end{latexonly}
\usepackage[dvips]{graphicx}
\usepackage{francais}
\makeindex
\begin{document}
\section{Une figure EPS}
\label{sec-figure}
Cette section montre comment inclure une figure
\externallabels{../ex2bib}{../ex2bib/labels.pl}%
PostScript\externalref{bibPS} dans un document \LaTeX.
La \hyperref{figure}{figure }{}{Fpsfig}
est insérée dans le texte à l'aide de la commande
\verb!\includegraphics{colorcir}!.
\begin{figure}
\htmlimage{thumbnail=0.4}
\centering
  \begin{tabular}{c@{\qquad}c}
    \includegraphics[width=3cm]{colorcir} &
    \includegraphics[width=3cm]{tac2dim}
  \end{tabular}
  \caption{Deux images EPS}\label{Fpsfig}
\end{figure}
\end{document}
```

Рис. 3.20. Связывание внешних файлов (LaTeX'овские файлы).

Файл, содержащий таблицу (ex22.tex)

```
\documentclass{article}
\usepackage{html}
%begin{latexonly}
\usepackage[T1]{fontenc}
%end{latexonly}
\usepackage[dvips]{graphicx}
\usepackage{francais}
\newcommand{\Lcs}[1]{\texttt{\symbol{'134}#1}}
\begin{document}
\section{Exemple d'un tableau}
\label{sec-tableau}
Le \hyperref{tableau}{tableau }{}{tab-exa} montre
l'utilisation de l'environnement \texttt{table}.
\begin{table}
\centering
 \begin{tabular}{cccccc}
  \Lcs{primo}  & \primo & \Lcs{secundo} & \secundo &
  \Lcs{tertio} & \tertio                           \\
  \Lcs{quatro} & \quatro& 1\Lcs{ier}    & 1\ier    &
  1\Lcs{iere}  & 1\iere                            \\
  \Lcs{fprimo)}&\fprimo)& \Lcs{No} 10   & \No 10   &
  \Lcs{no} 15  & \no 15                            \\
  \Lcs{og} a \Lcs{fg}&\og a \fg&3\Lcs{ieme}&3\ieme &
  10\Lcs{iemes}& 10\iemes
 \end{tabular}
\caption{Quelques commandes de l'option
  \texttt{french} de \texttt{babel}}\label{tab-exa}
\end{table}
\end{document}
```

Файл с библиографией (ex2bib.tex)

```
\documentclass{article}
\usepackage{html}
%begin{latexonly}
\usepackage[T1]{fontenc}
%end{latexonly}
\usepackage[dvips]{graphicx}
\usepackage{francais}
\makeindex
\begin{document}
\begin{thebibliography}{99}
\bibitem{bib-PS}\label{bibPS}
Adobe Inc.
\emph{PostScript, manuel de référence (2ième édition)}
Interčditions (France), 1992
\end{thebibliography}
\end{document}
```

Рис. 3.20. *Продолжение.*

работы с файлами возникают четыре показанных ниже подкаталога (имена каталогов подчеркнуты):

```
ex20:   ex20.css    ex20.html   labels.pl

ex21:   ex21.css    ex21.html   labels.pl
        img1.gif    Timg1.gif
        images.idx  images.tex  images.log  images.aux  images.pl
ex22:   ex22.css    ex22.html   labels.pl   internals.pl

ex2bib: ex2bib.css  ex2bib.html labels.pl   internals.pl
```

Файлы `labels.pl` из разных каталогов содержат информацию, которая связывает символические ключи команд `\label` в исходных LaTeX'овских файлах с физическими файлами. Например, в строках 21–22 файла `ex20.tex` делается ссылка на ключ «`tab-exa`», который определен в строке 27 файла `ex22.tex`. В действительности файл `labels.pl` в каталоге `ex22` содержит следующую информацию:

```
1  # LaTeX2HTML  99.1 (March 30, 1999)
2  # Associate labels original text with physical files.
3  $key = q/tab-exa/;
4  $external_labels{$key} = "$URL/" . q|ex22.html|;
5  $noresave{$key} = "$nosave";
6
7  $key = q/sec-tableau/;
8  $external_labels{$key} = "$URL/" . q|ex22.html|;
9  $noresave{$key} = "$nosave";
10 1;
```

Здесь, действительно, есть и сам ключ (строка 3), и определение файла, в котором он находится (строка 4).

Другие файлы в этих каталогах — это HTML- и CSS-файлы. Они содержатся там вместе с графическими изображениями формата GIF, созданными из тех данных, которые LaTeX2HTML не может качественно преобразовать в формат HTML. В данном примере только однажды — с рисунком в файле `ex21.tex` — возникает ситуация, в которой рисунок целиком создается из таблицы с помощью двух EPS-файлов (строки 21–24 в файле `ex21.tex` на рис. 3.20). Кроме того, в строке 19 указывается, что в HTML-файл надо вставлять не полномасштабный рисунок, а уменьшенный до 40% ярлычок. Поэтому в каталоге `ex21` можно увидеть два графических GIF-файла: полный рисунок `img1.gif` и соответствующий ярлычок `Timg1.gif`.

Вдобавок, во всех исходных LaTeX'овских файлах на рис. 3.20 строки 3–5 указывают на то, что при трансляции с помощью LaTeX2HTML команда `\usepackage[T1]{fontenc}` *игнорируется*. Причина в том, что кодировка шрифта имеет значение только в LaTeX'е, а все символы набора Latin 1, в том числе и встречающиеся в некоторых строках исходного документа французские диакритические знаки, LaTeX2HTML обрабатывает правильно без дополнительного вмешательства. На самом деле и более привычное воспроизведение букв со знаками ударения в LaTeX'овском первоисточнике, основанное на применении макрокоманд (использованное на рис. 3.1), и непосредственное использование кодировки символов Latin 1 после трансляции в HTML приводят к одинаковым результатам.

Для управления процессом трансляции этих документов с помощью LaTeX2HTML используется также файл настройки `myinit.pl`, в котором содержатся установки Perl констант.

```
1  # File myinit.pl
2  $ADDRESS = "<em>Michel Goossens<BR>" .
```

```
3 \            "Division IT<BR>"          .
4              "T&eacute;l. 767.5028<BR>"
5              "$address_data[1]</em>";
6  $MAX_SPLIT_DEPTH = 0; # do not split document
7  $NO_NAVIGATION = 1;   # no navigation panel
8  1;       # Mandatory last line
```

В строках 2–5 определяется отображаемая в конце каждой HTML-страницы адресная информация, в строке 6 объявляется, что весь документ должен быть сохранен как единое целое, а строка 7 удаляет навигационные панели.

При использовании этого настроечного файла для того, чтобы запустить LaTeX2HTML на обработку показанных на рис. 3.20 файлов, надо ввести команду:

```
latex2html -init_file myinit.pl -info "External links test"filenames
```

Указатель `-init_file` загружает файл настройки `myinit.pl`, а флажок `-info` изменяет информацию «Об этом документе... (*À propos de ce document...*)». И, наконец, параметр *filenames* (имена файлов) означает список из четырех LaTeX'овских файлов — `ex2bib.tex`, `ex21.tex`, `ex22.tex` и `ex20.tex`, — которые обрабатываются в надлежащем порядке. Результат выполнения этой команды показан на рис. 3.21.

3.5.7 Дополнительные функции

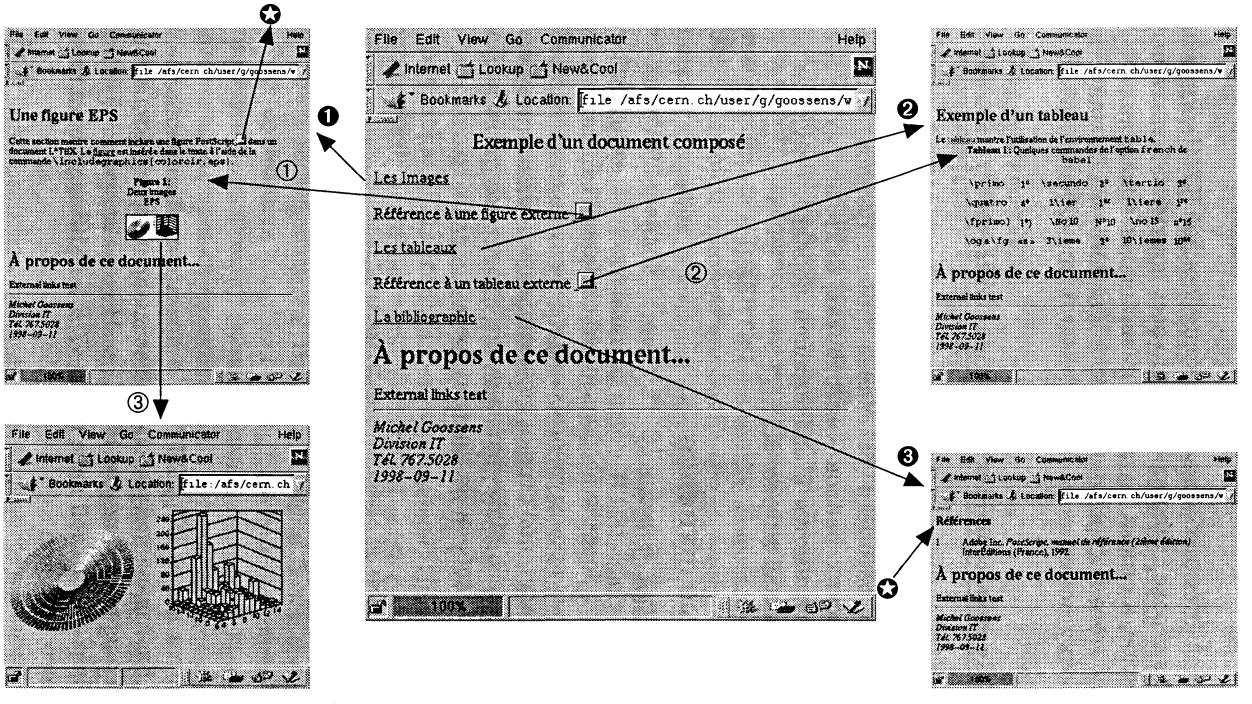
3.5.7.1 Использование Make-файла

В системе UNIX часто бывает полезно воспользоваться `Makefile` для упрощения команд запуска отдельных процедур, особенно в тех случаях, когда требуется большое число опций командной строки или используются длинные имена файлов. С помощью следующего Make-файла очень просто получить результат, показанный на рис. 3.21, набрав на клавиатуре `make ex2`.

```
1  # Makefile for Example 2
2  LTX = latex
3  L2H2 = latex2html -init_file myinit.pl -info "External_links_test"
4
5  ex2 :   ex21.aux ex22.aux
6          $(L2H2) ex2bib ex21 ex22 ex20
7  ex21.aux :
8          $(LTX) ex21.tex
9  ex22.aux :
10         $(LTX) ex22.tex
```

В строке 6 содержится вызов LaTeX2HTML; в предыдущей строке определяется «зависимость», благодаря чему обеспечивается предварительная обработка сегментированных первоисточников LaTeX'ом. Номера рисунков и таблиц содержатся в `.aux` файлах. В каждой паре строк (7–8 и 9–10) осуществляется вызов LaTeX'а для одного сегмента.

Приведенная программа — это простейший пример использования Make-файла, но даже его достаточно, чтобы показать, насколько можно сократить



В соответствии с предписаниями файла настройки `myinit.pl` навигационные панели отсутствуют и создано новое поле адреса. Выполнена также настройка информации *À propos de ce document...*. Стрелки, указывающие на цифры ❶, ❷ и ❸, соответствуют гиперссылкам, которые с помощью команды `\htmladdnormallink` в LaTeX'овском первоисточнике (строки 13, 18 и 23 файла `ex20.tex`) указывают на HTML-документ. Стрелки ① и ② обозначают перекрестные ссылки, созданные с помощью команд `\externalref`. В них используются символические имена, определенные как аргумент команд `\label` в документах, на которых делаются ссылки (строки 16 и 21 в файле `ex20.tex` ссылаются на метки, определенные, соответственно, в строке 25 файла `ex21.tex` и строке 27 файла `ex22.tex`). Стрелка ③ соответствует гиперссылке, соединяющей ярлычок в тексте с полномасштабным рисунком, доступным в виде отдельного внешнего GIF-файла. И, наконец, отправная (строка 14 в `ex2bib.tex`) и конечная (строка 11 в `ex2bib.tex`) точки библиографической ссылки обозначены знаком ✪.

Рис. 3.21. Воспроизведение навигатором структуры HTML-файла, полученного из составного документа и его поддокументов (рис. 3.20).

объем ввода с клавиатуры. Make-файл для другого варианта того же примера показан на рис. 3.22 в виде «сегментированного» документа. В следующем подразделе речь пойдет о специально разработанных для этого способа макро из пакета html. Сам пример будет подробно рассмотрен в разд. 3.5.7.3.

3.5.7.2 Использование «сегментированных» документов

Описанные в данном разделе команды предназначены для использования главным образом в рамках стратегии «Сегментирования документов», хотя некоторые из них могут оказаться полезными и в других ситуациях. Сегментация включает в себя деление исходных LaTeX'овских документов на составные части. Все сегменты содержатся в отдельных файлах и обрабатываются транслятором LaTeX2HTML поодиночке.

Ранние версии LaTeX2HTML могли требовать очень большого объема памяти, особенно на этапе создания изображений. Позднейшие усовершенствования снизили эту потребность, но сегментация остается плодотворной стратегией работы с Web-документами, требующими регулярного обновления. Сегменты могут обновляться поодиночке или группами, и при этом нет необходимости повторной обработки всего документа. Более подробно с вопросами использования этой стратегии можно познакомиться в Руководстве пользователя LaTeX2HTML.

\segment[*align*]{*file*}{*sectioning*}{*title*}

Предположим, что параметр *sectioning* имеет значение section; тогда LaTeX будет обрабатывать эту команду как команду \section{*title*}, за которой следует команда \input *file*. Прежде всего в файл с именем *file*.ptr записывается информация о счетчиках в виде LaTeX'овских команд для установки значений счетчиков. При обработке сегмента с помощью LaTeX2HTML этот файл является частью задания. Параметр *sectioning* может принимать значения в виде любой допустимой формы команд секционирования: part, chapter, section, subsection, paragraph и т. д.

Сегмент *file* не должен содержать своего названия, если только оно не спрятано внутри окружения \begin{htmlonly}...\end{htmlonly}, так как название уже задано аргументом *title*. При использовании LaTeX2HTML этот заголовок *title* с помощью команды \htmlhead (см. ниже) сохраняется в соответствующем .ptr файле. В случае необходимости сохранение заголовка можно отменить командой \htmlnohead.

\htmlhead[*align*]{*sectioning*}{*title*}
\htmlnohead

Команда \htmlhead обычно читается из .ptr файла (см. описание команды \segment выше). В этой форме она становится самой первой информацией, которая должна появиться на HTML-странице вслед за средствами навигации. Однако может оказаться желательным вместо этого сначала поместить

```
# Makefile for Example 3
L2H = $(L2HNEW)/latex2html -link 4
L2HTOP = $(L2HNEW)/latex2html -split 0
LTX = latex
EX = ex3
HTML = .html
TX = .tex
TEXES = $(EX)*$(TX)
TOP = ex30
BIB = $(EX)bib
TEXTOP = $(TOP)$(TX)
INT = internals.pl
BIBREF = '\#BIBLIO'
# Directories containing the segments
EX30 = ../$(TOP)/$(TOP)
EX31 = ../$(EX)1/$(EX)1
EX32 = ../$(EX)2/$(EX)2
EX3b = ../$(TOP)/$(TOP)$(HTML)
# Titles for navigation to the segments
EX30t = Exemple_d\'un_document_segmentщ
EX31t = Une_figure_EPS
EX32t = Exemple_d\'un_tableau
EX3bt = Rщfщrences

COMMON = -info 0 -split 1 -link 4 -no_auto_link -biblio $(EX3b) -external_file $(TOP) \
        -up_url $(EX30)$(HTML) -up_title $(EX30t) -index $(EX3b) -index_in_navigation
update: $(TOP).ind
        make $(TOP)/$(INT) $(EX)2/$(INT) $(EX)1/$(INT); make $(TOP)/$(INT)
fresh:
        rm $(EX)*/$(INT) $(TOP).aux; make $(TOP).ind;
        make $(TOP)$(HTML) $(EX)2$(HTML) $(EX)1$(HTML); make $(TOP)$(HTML)
$(TOP)$(HTML) :
        $(L2HTOP) -down_url $(EX31)$(HTML) -down_title $(EX31t) -biblio $(BIBREF) $(TOP)
$(TOP)/$(INT) : $(TEXTOP) $(BIB)$(TX) $(EX)1/$(INT) $(EX)2/$(INT)
        make $(TOP)$(HTML)
$(EX)1$(HTML) :
        $(L2H) $(COMMON) -t $(EX31t) -prev_url $(EX30)$(HTML) -prev_title $(EX30t) \
        -down_url $(EX32)$(HTML) -down_title $(EX32t) $(EX)1
$(EX)1/$(INT) : $(EX)1$(TX)
        make $(EX)1$(HTML)
$(EX)2$(HTML) :
        $(L2H) $(COMMON) -t $(EX32t) -prev_url $(EX31)$(HTML) -prev_title $(EX31t) \
        -down_url $(EX3b)$(BIBREF) -down_title $(EX3bt) $(EX)2
$(EX)2/$(INT) : $(EX)2$(TX)
        make $(EX)2$(HTML)

# handle  LaTeX/dvips/makeindex  etc.
$(TOP).aux: $(TEXES)
        make dvi
$(TOP).dvi:
        make dvi
dvi:
        $(LTX) $(TEXTOP); $(LTX) $(TEXTOP); make $(TOP).ind
$(TOP).ind: $(TOP).aux
        makeindex $(TOP).idx; $(LTX) $(TEXTOP); touch $(TOP).ind
ps:
        make $(TOP).ps
$(TOP).ps: $(TOP).dvi
        dvips $(TOP).dvi -o
clean:
        rm *.ps *.dvi *.log *.ilg
```

Рис. 3.22. Пример Make-файла для документов, показанных на рис. 3.23.

на страницу другую информацию, например какой-нибудь рисунок. В этом случае следует использовать команду \htmlnohead, а следом за ней в подходящем месте исходного документа — команду \htmlhead. В действительности можно использовать разные варианты заголовков *title*, благодаря чему одна и та же HTML-страница может иметь разные названия в зависимости от того, рассматривается ли она как самостоятельный документ или как составная часть большего документа.

С помощью необязательного аргумента *align* расположенное в верхней части HTML-страницы название *title* можно выровнять по центру (center), по правому (right) или по левому (left) краю. По умолчанию выравнивание выполняется по левому краю. Конечно же, в LaTeX'е обе команды \htmlnohead и \htmlhead игнорируются вместе со своими аргументами.

\startdocument

По отношению к сегменту документа эта команда действует как искусственный маркер, обозначающий место, где начинается тело данного сегмента. Данные, расположенные до этого места, обрабатываются так, как если бы они находились в преамбуле всего LaTeX'овского документа, и не будут выведены на HTML-страницу. Типичным примером таких данных могут служить определения \newcommand, содержащиеся внутри \begin{htmlonly}...\end{htmlonly}.

\internal[*type*]{*prefix*}

Эта команда заставляет LaTeX2HTML загрузить Perl-файл с информацией, созданной в отдельном сегменте данного (или другого) документа. Например,

```
\internal[contents]{../segA/C}
```

Задание такой последовательности приводит к чтению файла ../segA/Ccontents.pl, причем путь к нему указывается от каталога, в котором выполняется обработка документа. Этот файл должен содержать информацию из оглавления, созданного с помощью команды \tableofcontents, при условии, что она должна быть воспроизведена в обрабатываемом сегменте. Допустимыми значениями параметра *type* являются section, contents, figure, table, index, images и internal.

\segmentcolor[*model*]{*color*}
\segmentpagecolor[*model*]{*color*}

Эти команды можно использовать только в ptr-файлах. Они передают информацию о цвете, определенную LaTeX'овскими командами \color и \pagecolor, от одного сегмента к следующему либо от родительского документа к его сегментам.

\endsegment[*sectioning*]

Эта команда иногда нужна для того, чтобы обеспечить правильную нумерацию разделов в документе, который загружает другие сегменты. В LaTeX'е она

не выполняет никаких действий, но в LaTeX2HTML устраняет проблемы с навигационными панелями и оглавлением, которые без нее могут оказаться беспорядочными. Для того чтобы не возникало путаницы между гиперссылками на последующие страницы и текущим или предыдущими сегментами, ей должна предшествовать команда `\segment`. Значение параметра *sectioning* должно соответствовать предшествующему сегменту.

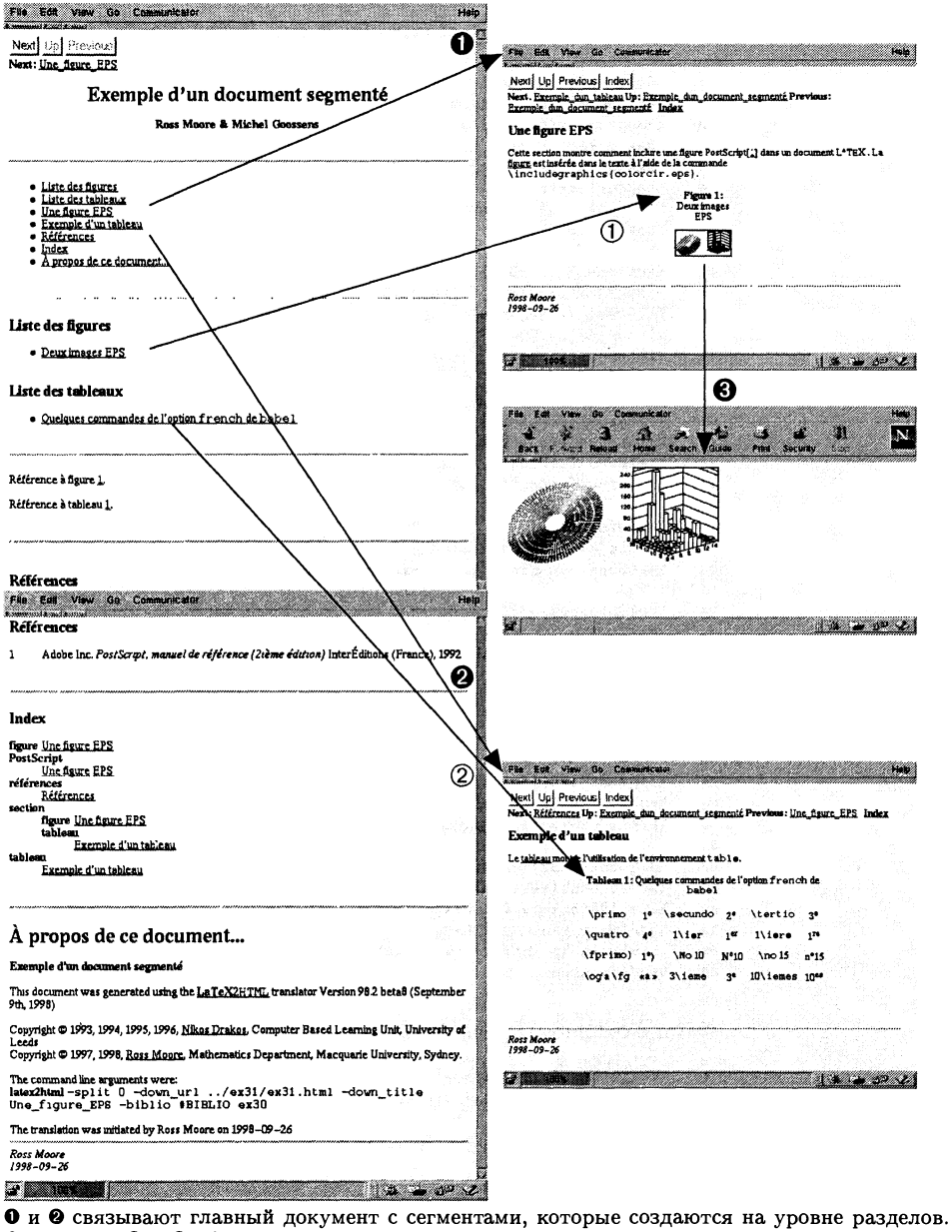
3.5.7.3 Пример использования Make-файла для сегментации

На рис. 3.23 показаны HTML-страницы, которые можно получить, если применить стратегию сегментации для первоисточников LaTeX'овских документов. Из сравнения рис. 3.23 с рис. 3.2 становится ясно, что полезность отдельных страниц можно увеличить, если изменить порядок воспроизведения информации. На рис. 3.23 длина основной HTML-страницы превышает высоту экрана, поэтому она была разбита на два изображения. Они показаны в левой части рисунка совмещенными по вертикали. Естественно, что при использовании навигатора доступ ко всей информации обеспечивается с помощью вертикальной полосы прокрутки.

Распечатки исходных текстов сегментов приведены на рис. 3.24. Они аналогичны приведенным на рис. 3.20, но при создании HTML-страниц отдельные LaTeX'овские фрагменты обрабатываются независимо. Как правило, они оказываются несколько короче из-за того, что отпадает необходимость в обработке всего LaTeX'овского документа с помощью LaTeX2HTML. Посмотрим распечатку файла `ex30.tex`. В нем для каждого сегмента вместо команд `\input` или `\include` используется команда `\segment`. Поэтому при обработке LaTeX'ом каждый из входных файлов `ex31.tex` и `ex32.tex` начинает новый раздел. Получаемый в результате `dvi`-файл должен выглядеть точно так же, как на рис. 3.2, за исключением нескольких терминов указателя. Интенсивное использование команд `\htmlrule` помогает разделять большие куски взаимосвязанной информации на HTML-странице.

Чтобы указать LaTeX2HTML место начала основной части документа, в каждом из файлов `ex31.tex` и `ex32.tex` вместо команды `\begin{document}` используется `\startdocument`. Перед ней размещается вся необходимая вводная информация, причем, во избежание повторения при обработке документа LaTeX'ом, большая ее часть — внутри условных окружений `htmlonly`. В файле `ex31.tex` присутствуют команды `\documentclass` и `\usepackage`, которые будут нужны при создании рисунка. В файле `ex32.tex` команда `\documentclass` не нужна, но пакеты `makeidx`, `francais` и `html` по-прежнему необходимы. Для правильной нумерации разделов в обоих сегментах необходимо с помощью команды `\input` указать соответствующий `ptr`-файл.

Обратите внимание на то, что использовавшиеся ранее (на рис. 3.20) команды `\externallabels` и `\externalref` теперь не нужны. Вместо этого перемещение по документу осуществляется с помощью обычных LaTeX'овских команд `\ref`. Это становится возможным благодаря присутствию в `ex30.tex` команд `\internal`, которые читают из сегментов информацию, задаваемую символическими метками. Аналогично, путем вставки соответствующей ко-



❶ и ❷ связывают главный документ с сегментами, которые создаются на уровне разделов. Символами ① и ② обозначены перекрестные ссылки из списков рисунков и таблиц на сами рисунки и таблицы. ❸ связывает указатель рисунка с настоящим рисунком.

Рис. 3.23. Сегментированная HTML структура, созданная с помощью LaTeX2HTML из исходного LaTeX'овского документа, показанного на рис. 3.24.

Основной файл (ex30.tex)

```
\documentclass{article}
\usepackage[dvips]{graphicx}
\usepackage{francais}
\usepackage{html}
\usepackage{makeidx}
%begin{latexonly}
\usepackage[T1]{fontenc}
%end{latexonly}

\internal{../ex31/}
\internal[sections]{../ex31/}
%\internal[contents]{../ex31/}
\internal[figure]{../ex31/}
\internal[index]{../ex31/}

\internal{../ex32/}
\internal[sections]{../ex32/}
%\internal[contents]{../ex32/}
\internal[table]{../ex32/}
\internal[index]{../ex32/}

\makeindex
\title{Exemple d'un document segmenté}
\author{Ross Moore \& Michel Goossens}

\begin{document}
\maketitle\htmlrule
\tableofchildlinks\htmlrule
%begin{latexonly}
\tableofcontents
%end{latexonly}
\listoffigures\listoftables\htmlrule

\segment{ex31}{section}{Une figure EPS}
\endsegment[section]
\html{Référence à figure~\ref{Fpsfig}.}

\segment{ex32}{section}{Exemple d'un tableau}
\endsegment[section]
\html{Référence à tableau~\ref{tab-exa}.}
\htmlrule \input{ex3bib}
\htmlrule \printindex \htmlrule
\end{document}
```

Файл с библиографией (ex3bib.tex)

```
\begin{thebibliography}{99}
\label{BIBLIO}\index{références}
\bibitem{bib-PS}
Adobe Inc. \emph{PostScript, manuel de référence
 (2ième édition)} Interéditions (France), 1992
\end{thebibliography}
```

Рис. 3.24. Пример сегментации (LaTeX'овские файлы).

Файл, содержащий рисунки (ex31.tex)

```
\begin{htmlonly}
\documentclass{article}
\usepackage{makeidx,html}
\usepackage[T1]{fontenc}
\usepackage[dvips]{graphicx}
\usepackage{francais}
\input ex31.ptr
\end{htmlonly}
\startdocument
\index{section!figure}

Cette section montre comment inclure une figure
PostScript\cite{bib-PS} dans un document \LaTeX.
La \hyperref{figure}{figure }{}{Fpsfig}
est insщгще dans le texte p l'aide de la commande
\verb!\includegraphics{colorcir}!.
\index{figure}
\index{PostScript}
\begin{figure}[h]
\htmlimage{thumbnail=0.4}
\centering
  \begin{tabular}{c@{\qquad}c}
    \includegraphics[width=3cm]{colorcir} &
    \includegraphics[width=3cm]{tac2dim}
  \end{tabular}
  \caption{Deux images EPS}\label{Fpsfig}
\end{figure}
```

Файл, содержащий таблицу (ex32.tex)

```
\begin{htmlonly}
\usepackage{makeidx,html}
\usepackage{francais}
\input ex32.ptr
\end{htmlonly}
\newcommand{\Lcs}[1]{\texttt{\symbol{'134}#1}}
\startdocument
\index{section!tableau}

Le \hyperref{tableau}{tableau }{}{tab-exa}
montre l'utilisation de l'environnement \texttt{table}.
\begin{table}[h]
\centering
 \begin{tabular}{cccccc}
  \Lcs{primo}  & \primo & \Lcs{secundo} & \secundo
  & \Lcs{tertio} & \tertio \\
  \Lcs{quatro} & \quatro& 1\Lcs{ier}    & 1\ier
  & 1\Lcs{iere}  & 1\iere  \\
  \Lcs{fprimo)}&\fprimo)& \Lcs{No} 10   & \No 10
  & \Lcs{no} 15  & \no 15  \\
  \Lcs{og} a \Lcs{fg}&\og a \fg&3\Lcs{ieme}&3\ieme
  & 10\Lcs{iemes}& 10\iemes
 \end{tabular}
\caption{Quelques commandes de l'option \texttt{french}
         de \texttt{babel}}\label{tab-exa}
\index{tableau}
\end{table}
```

Рис. 3.24. *Продолжение.*

манды \internal, можно разрешить перекрестные ссылки между различными сегментами. Информация для создания списков рисунков и таблиц берется из команд \internal[figure] и \internal[table], тогда как данные о разделах каждого сегмента получаются с помощью команд \internal[section]. Аналогичную или, может быть, незначительно отличающуюся информацию для оглавления можно получить, используя команду \internal[contents]. В данном примере это делать необязательно, потому что вся необходимая информация содержится в автоматически создаваемом «мини»-оглавлении. Это также объясняет, почему команда \tableofcontents используется только LaTeX'ом, при том что расположение этого мини-оглавления определяется командой \tableofchildlinks.

Сбор информации для указателя осуществляется командами \internal[index]. При загрузке пакета makeidx становятся доступными дополнительные возможности, такие, как вывод иерархического списка указателей. Если пакет makeidx не загружен, то может быть использован упрощенный вариант указателя. Лучше всего загружать пакет makeidx во всех сегментах; в противном случае в некоторых элементах указателя вместо названий разделов будут приведены имена html-файлов. При использовании команд \cite в ex31.tex адресат гиперссылки определяется в зависимости от того, в каком html-файле должен появиться список литературы. Получение такой информации при обработке соответствующего сегмента обеспечивается опцией командной строки -biblio. Кнопки, расположенные в верхней части каждой HTML-страницы, позволяют переходить от сегмента к сегменту, точно так же как и в случае отдельного несегментированного документа, независимо от наличия в LaTeX'овском первоисточнике каких-либо перекрестных ссылок или команд секционирования. Соответствующие гиперссылки определяются с помощью опций командной строки. В результате командная строка становится весьма сложной, и это приводит к необходимости использования Make-файла.

Рассмотрим структуру Make-файла, листинг которого приведен на рис. 3.22. Обычно в нем определяются переменные для записи строк, которые либо должны будут часто использоваться, либо с большой степенью вероятности будут изменяться в процессе работы над документом. (Это похоже на использование макро в LaTeX'е.) Такие определения содержатся в строках 2–26, а их значения можно использовать с помощью разыменовывающей записи $(..).

В качестве примера в строках 25–26 создается переменная COMMON, которая должна содержать опции командной строки, используемые при обработке сегментов. Эти опции командной строки, ряд которых создан путем разыменовывания других переменных, будут использоваться при обработке обоих сегментов. Сами командные строки в окончательном виде формируются в строках 36–38 и 41–43. В этих группах строк с помощью опций командной строки определяется название документа. Определяются также гиперссылки для навигационных кнопок «следующий/предыдущий», а также текущий файл, подлежащий обработке. Сформированная командная строка приведена ниже (строки 1–6) вместе с начальной частью протокольных сообщений для одного из рассматриваемых сегментов (строки 8–27). Поскольку сама командная строка очень длинна, то она была преобразована к виду, более удобному для чтения.

```
1  latex2html -link 4 -info 0 -split 1 -link 4 -no_auto_link -biblio ../ex30/ex30.html \
2  -external_file ex30 -up_url ../ex30/ex30.html -up_title Exemple_d'un_document_segmentщ \
3  -index ../ex30/ex30.html -index_in_navigation \
4  -t Une_figure_EPS -prev_url ../ex30/ex30.html \
5  -prev_title Exemple_d'un_document_segmentщ  -down_url ../ex32/ex32.html \
6  -down_title Exemple_d'un_tableau ex31
7
8  This is LaTeX2HTML Version 99.1 release (March 30, 1999)
9  Drakos, CBLU, University of Leeds.
10
11 Revised and extended by:
12  Marcus Hennecke, Ross Moore, Herb Swan and others
13 ...producing markup for HTML version 3.2
14
15 Loading /usr/local/share/latex2html/versions/html3.2.pl
16
17  *** processing declarations ***
18
19  OPENING /services/www/texdev/LWC/MICHEL/ex31.tex
20
21 Cannot create directory ex31/: File exists, reusing it.
22 Reusing directory ex31/:
23
24 Loading /usr/local/share/latex2html/styles/texdefs.perl...
25 Loading /usr/local/share/latex2html/styles/article.perl
26 Loading /usr/local/share/latex2html/styles/makeidx.perl
27 Loading /usr/local/share/latex2html/styles/html.perl
```

Этот Make-файл упрощает сопровождение рассмотренного в примере документа. Достаточно лишь ввести в командной строке `make ex31.html`, чтобы выполнить команды, определенные в строках 35–37. Если вместо этого ввести `make ex31/internals.pl`, то команда будет выполняться, только если в LaTeX'овском исходном файле `ex31.tex` произошли какие-то изменения. Эта «зависимость» проверяется в строке 38, а обращение к файлу `make ex31.html` из строки 39 выполняется только в случае необходимости[1]; в противном случае нет нужды реконструировать `ex31/ex31.html`. В строках 43–44 выполняется подобное условное обновление файла `ex32/ex32.html`. Обновление основной страницы `ex30/ex30.html` оказывается несколько более сложным, поскольку зависит от изменений, сделанных в более чем одном LaTeX'овском исходном файле, а также от наличия обновленных данных в файлах `internals.pl`, соответствующих другим сегментам. Эта зависимость устанавливается в строке 33. В отличие от LaTeX'овского aux-файла, который переписывается после каждого обращения к LaTeX'у для обработки исходного файла, файл `internals.pl` переписывается *только* тогда, когда была изменена информация, которую он должен содержать. Следовательно, можно внести небольшие изменения в сегмент, не изменяя дату (модификации) его файла `internals.pl`. Такое изменение даты привело бы к необходимости обновления головного сегмента.

Возможность условного обновления только тех HTML-страниц, которые в этом нуждаются, зафиксирована в строках 26–27, благодаря чему в результате выполнения команды `make update` анализируются все указанные зависимости. Эта процедура включает в себя запуск LaTeX'а, для того чтобы обеспечить обновление версии `dvi` файла и корректную информацию в aux-файле.

[1] Такая необходимость может возникнуть, только если файл, стоящий справа от знака «:», имеет более позднюю дату/время модификации, чем файл, стоящий слева.

Если что-то не удается или в результате сделанных изменений не происходит соответствующего обновления, то следует, используя инструкцию, содержащуюся в строках 28–30, выполнить полную повторную обработку, для чего нужно ввести с клавиатуры `make refresh`. Обычно такая необходимость возникает после переопределения макрорасширения в файле стиля.

Конечно же, зависимость от такого файла стиля можно дополнительно указать в соответствующем месте Make-файла. В строках 47–58 определяются правила запуска LaTeX'а с помощью `make dvi` и `dvips` с помощью `make ps`. Первая команда несколько раз запускает LaTeX, затем `makeindex` и в конце — снова LaTeX. Ненужные протокольные файлы удаляются командой `make clean` (строки 59–60).

Следует знать, что отдельные незначительные детали программы `make` могут быть разными для различных платформ UNIX и даже для различных командных процессоров на одной и той же платформе. Обратите внимание на то, что специальные символы типа # и ' необходимо вводить в составе экранирующих последовательностей, используя \# и \'. На конкретной платформе и командном процессоре, которыми пользовался автор, были недопустимы пробелы в строках, которые появляются в командной строке; этим объясняется использование символа _ в названиях сегментов.

3.5.7.4 Трассировка и отладка

Команды данного раздела предназначены главным образом для помощи разработчикам LaTeX2HTML и расширения возможностей программы за счет специализированных пользовательских модулей.

```
\htmltracing{level}
\htmltracenv{level}
```

Обе эти команды устанавливают уровень трассировки равным заданному уровню *level*, который может быть целым числом от 0 (трассировка не выполняется) до 9; принимаемое по умолчанию значение уровня равно 1. Разница между командами `\htmltracing` и `\htmltracenv` зависит от времени обработки каждой команды. Команда `\htmltracenv` обрабатывается в одной последовательности с окружениями, тогда как `\htmltracing` выполняется как команда с постоянным статическим расширением. Поэтому обработку можно отложить до появления окружений такого же уровня TeX'овской группировки. Чем выше значение уровня *level*, тем больше генерируется сообщений, относящихся к большему числу технических аспектов процесса трансляции. О том, какая информация воспроизводится на каждом уровне, можно узнать в Руководстве пользователя LaTeX2HTML.

```
\htmlset{variable}{value}
\htmlset[hash]{key}{value}
\htmlsetenv{variable}{value}
\htmlsetenv[hash]{variable or key}{value}
```

Команды `\htmlset` и `\htmlsetenv` разрешают установку или изменение Perl переменных непосредственно в процессе обработки с помощью LaTeX2HTML. Раз-

личие между этими двумя формами определяется тем, когда данная команда фактически обрабатывается. Команда `\htmlsetenv` обрабатывается совместно с окружениями, тогда как `\htmlset` выполняется как команда со статическим расширением. Поэтому обработка может быть приостановлена и выполнена только после появления окружения того же уровня TEX'овской группировки. При использовании необязательного аргумента параметр *key* получает значение *value* для хэш, определенного как *hash*.

3.6 Заключение

В этой главе показано, как при помощи LATEX2HTML преобразовать исходные LATEX'овские документы в сшитую гиперссылками паутину HTML-файлов, представляя в виде рисунков те фрагменты исходных документов, например математические формулы, которые нельзя непосредственно воспроизвести посредством эквивалентных конструкций языка HTML. Для того чтобы получить отличные результаты, часто бывает достаточно использовать в процессе трансляции задаваемые по умолчанию настройки, обходясь без вмешательства пользователя. Однако пользователь имеет возможность непосредственно влиять на процесс трансляции, задавая опции командной строки или редактируя один или несколько файлов, специально предназначенных для облегчения пользовательской настройки.

Были подробно изучены способы точной настройки воспроизведения математических выражений и рассмотрены имеющиеся средства, позволяющие использовать разные языки. В приведенных примерах рассматривались средства для поддержки языков Индийского полуострова, однако описанные способы являются более общими.

В заключительной части мы научились извлекать пользу из богатого набора расширений пакета `html`. Это не только открывает доступ ко всем возможностям HTML, но и позволяет создавать настоящую сеть из гиперсвязанных внешних документов, связанных гиперссылками.

В конечном счете, LATEX2HTML представляет собой очень гибкий и совершенный инструмент. Даже новичок может воспользоваться этим автоматическим транслятором как своего рода *черным ящиком*, для того чтобы простым и понятным способом сделать LATEX'овские исходные документы доступными по Сети. Однако в руках посвященного LATEX2HTML превращается в законченное приложение, позволяющее создавать, редактировать и сопровождать его гипертекстовые документы. Таким образом, LATEX2HTML — это удобный мост между миром LATEX'а и Интернетом.

Глава 4

Трансляция из LaTeX'а в HTML с помощью TeX4ht

TeX4ht — это система, которая предоставляет способ создания настраиваемых гипертекстовых версий LaTeX'овских документов; она является расширением TeX'а вообще и LaTeX'а — в частности. В этой главе мы сосредоточимся на устанавливаемой по умолчанию конфигурации, которая обеспечивает создание HTML-документов.

Система состоит из двух частей: набора стилевых файлов, которые дополняют существующие LaTeX'овские файлы возможностями HTML, и постпроцессора для выделения HTML-файлов из результатов работы TeX'а. Основную задачу превращения исходных документов в DVI-код TeX4ht оставляет самому TeX'у. Следовательно, TeX4ht имеет доступ ко всем возможностям TeX'а в области работы со шрифтами, макро, переменными, кодами категорий и другим важным свойствам, которые весьма часто остаются спрятанными от пользователя, но существенны для стилевых файлов. В большинстве приложений оказываются необходимыми только несколько основных функций этой системы. При создании HTML-документов из LaTeX'овских источников пользователь вряд ли заметит вмешательство системы.

Несмотря на то, что TeX4ht — это гибкая авторская система с богатыми возможностями, для большинства применений его функциональность остается незаметной. Как правило, компиляция и просмотр документов, использующих TeX4ht, выполняются так же просто, как и без него. Первые три раздела этой главы посвящены именно таким приложениям — в них объясняется, как с рассматриваемой системой могут работать обычные пользователи, которым не требуется специальная конфигурация. Перечисляются наиболее часто используемые опции пакета, приводится законченный пример, содержащий как исходные данные, так и результат, а также коротко описывается ход процесса.

В четвертом разделе описываются новые команды пакета, предназначенные для авторов, которые хотят добавить явные ссылки, вручную получить HTML-вывод или добавить информацию к стилям CSS. В пятом разделе объясняется, как можно существенным образом изменить конфигурацию си-

стемы для получения различных результатов из одной и той же LATEX'овской разметки.

В заключительной части этой главы более подробно рассматриваются выполнение различных этапов обработки с помощью TEX4ht и способы их изменения. Этот раздел будет особенно полезен системным администраторам, устанавливающим TEX4ht для коллективного пользования.

Получить TEX4ht можно либо в TEX'овских архивах, либо на его Web-странице [↪TEX4HT], на которой имеются также многочисленные примеры. Работа над TEX4ht все еще продолжается. Некоторые из описанных здесь функций могут со временем измениться, но ожидается, что эти изменения будут незначительными, поскольку большинство функций уже стабилизировалось.

4.1 Использование TEX4ht

Для преобразования исходного LATEX'овского файла в HTML многим пользователям будет достаточно просто загрузить входящий в состав TEX4ht пакет `tex4ht.sty` (рис. 4.1).

Процесс трансляции запускается командой вида `ht latex` *filename* или аналогичной системно-зависимой командой. Результат трансляции сохраняется в наборе файлов, корневой файл которого имеет имя *filename*`.html`.

4.1.1 Опции пакета

С помощью опций пакета можно изменять результат трансляции исходных LATEX'овских файлов в HTML (см., например, рис. 4.2).

Пользователь имеет доступ к ряду опций пакета. Некоторые из них представляют общий интерес; большинство же не существенны. Некоторые опции играют роль механизмов быстрого вызова, предоставляющих удобные способы задействования полезных функций без необходимости их подробного изучения.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
        HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<!--try.html from try.tex
  (TeX4ht, 1998-05-27 00:01:00)-->
<TITLE>try.html</TITLE>
<LINK REL="stylesheet"
    TYPE="text/css" HREF="try.css">
</HEAD><BODY>
hello world
</BODY> </HTML>
```

```
\documentclass{article}
\usepackage{tex4ht}
\begin{document}
hello world
\end{document}
```

Рис. 4.1. Простой исходный файл и полученный HTML-код.

A Perfect Day

A Perfect Day
L. Reed
June 26, 1998

1 Introduction
References

1 Introduction

[next] [tail] [up]

1 Introduction

See [1].

[next] [front] [up]

References

[prev] [prev-tail] [tail] [up]

References

[1] Nothing yet.

[prev] [prev-tail] [front] [up]

```
\documentclass{article}
  \usepackage
     [html,2,sections+]{tex4ht}
  \title{A Perfect Day}
  \author{L. Reed}
\begin{document}
  \maketitle
  \section{Introduction}
     See \cite{x}.
  \begin{thebibliography}{}
     \bibitem{x} Nothing yet.
  \end{thebibliography}
\end{document}
```

Рис. 4.2. Исходный документ и вывод его HTML-файлов.

Например, опции 1, 2, 3 и 4 — это просто быстрый вызов одной команды `\tableofcontents` и нескольких команд `\CutAt` и `\TocAt*`. Пользователи, которым нужен результат более изысканный, чем тот, который можно получить с помощью указанных опций, должны приложить некоторые усилия к изучению этих команд (см. разд. 4.5.2 на с. 207 и разд. 4.5.3 на с. 210) и использовать их явным образом.

Первая опция пакета должна быть либо именем файла конфигурации (см. разд. 4.5.1 на с. 205), либо опцией `html`. В противном случае первая опция игнорируется. Последующие опции могут перечисляться в любом порядке.

В соответствии со своим именем опция `html` требует, чтобы результат создавался в виде HTML-файла. Без этой опции получился бы стандартный DVI-файл. Ниже перечислены некоторые другие опции:

1, 2, 3 или 4 Эти опции задают членение документов на иерархии гипертекстовых страниц. Эти иерархии отражают логическую структуру содержимого, определенную командами секционирования `\part`, `\chapter`, `\section` и т. д. Значения этих опций определяют желаемую глубину иерархии гипертекстовых страниц.

sections+ Элементы оглавлений обеспечивают гипертекстовые ссылки с разделами, на которые они ссылаются. Эта опция требует создания дополнительных обратных гипертекстовых ссылок от названий разделов к оглавлению.

next В тех случаях, когда документ делится на гипертекстовые страницы по разделам, чтобы определить пути от страницы к странице устанавливаются навигационные ссылки. При стандартной установке некоторые из путей формируются на основании соотношений «предыдущий-последующий», которые отражают древовидную логическую структуру разделов, обеспечивая соединения между ближайшими вершинами дерева. Эта опция требует установления альтернативных навигационных ссылок «предыдущий-последующий», отражающих линейную последовательность всех разделов документа.

pic-array, pic-displaylines, pic-eqnarray, pic-tabbing, pic-tabular Эти опции вызывают графическое представление указанных окружений вместо их воспроизведения с применением HTML-таблиц.

_13, ^13, no_, no^ TEX4ht модифицирует исходные определения специальных символов _ и ^, чтобы сделать возможным включение меток языка HTML для верхних и нижних индексов в математических формулах. Первые две опции вызывают альтернативные определения (с кодом категории этих символов 13 вместо 12). Вторая пара опций отключает этот режим.

refcaption Ссылки на окружения рисунков и таблиц, вставленные с помощью команды **\ref**, становятся гипертекстовыми ссылками, указывающими на точки входа этих окружений. Данная опция создает гипертекстовые ссылки, которые указывают на *заголовки* в этих окружениях.

3.2 Эта опция связывает с опцией **html** вызов версии HTML 3.2 вместо устанавливаемой по умолчанию ссылки с версией HTML 4.0 Transitional. Более поздняя версия по своей природе значительно ближе к LATEX'у, чем версия 3.2. В ней использование стилей не зависит от содержания и логических структур, и мы рекомендуем пользоваться этой версией при любой возможности.

fonts+ При обычной установке предполагается, что вместо стандартного шрифта документа в навигаторах должны использоваться их собственные стандартные шрифты. Эта опция предлагает конкретный шрифт.

no_*style* Эта опция отключает загрузку функций HTML, специально предназначенных для указанного *стилевого файла*. Например, опция **no_amsart** предписывает транслятору TEX4ht не загружать имеющуюся у него программу для пакета **amsart**, а опция **no_array** оказывает такое же действие в отношении программы, которая в TeX4ht предназначена специально для пакета **array**. В подобных случаях от пользователя может понадобиться создание собственных дополнений к соответствующим пакетам.

info Эта опция требует, чтобы в протокольный файл была записана информация о функциональных возможностях конкретного пакета.

`htm` Эта опция является разновидностью опции `html`, в которой в качестве имен файлов используются собственные имена, состоящие не более чем из восьми символов, а имена HTML-файлов имеют расширение `htm`.

4.1.2 Графическое представление отдельных элементов

Основная задача — это описать все транслируемые объекты в терминах гипертекстовых элементов, которые отражают как структуру, так и семантику содержимого. При невозможности достичь этого необходимо использовать альтернативные представления.

4.1.2.1 Математические выражения и специальные символы

Для воспроизведения содержимого математических окружений — как внутритекстовых \(*formula*\), так и выключных \[*formula*\] — требуется графическое представление.

С другой стороны, plain-TeX'овские математические окружения — внутритекстовые $*formula*$ и выключные $$*formula*$$ — смешивают текстовый вывод простых частей формулы с графическими изображениями остальных элементов.

Кроме математических выражений, некоторые буквы со знаками ударения, другие специальные символы и LaTeX'овские графические окружения при стандартной установке автоматически транслируются в рисунки. Для трансляции других объектов в растровые изображения необходим явный запрос.

4.1.2.2 Создание рисунков и ссылки на них

Воспроизводимые навигаторами растровые изображения могут быть описаны в автономных файлах, ссылки на которые содержатся в HTML-коде. Команда

`\Picture[`*alt*`]{`*filename attributes*`}`

создает такую ссылку на существующий файл, а также определяет атрибуты и альтернативное текстовое представление для этих рисунков. Компонента [*alt*] является необязательной; при ее отсутствии используется предусмотренное в системе стандартное текстовое представление. Если атрибуты заданы, то они должны быть отделены от имени файла пробелом.

Например, команда

```
\Picture[TUG logo]{http://www.tug.org/logo.gif ID="tuglogo"}
```

ссылается на GIF-файл, содержащий логотип группы пользователей TeX'а TUG (TeX Users Group). Эта команда создает следующий HTML-код:

```
<IMG SRC="http://www.tug.org/logo.gif" ALT="TUG logo" ID="tuglogo">
```

А вот вариант, в котором используются две команды:

\Picture+[*alt*]{*filename attributes*}
\EndPicture

Они указывают, что растровый рисунок должен быть *создан* из того, что располагается между ними. Однако здесь отсутствующий параметр [*alt*] рассматривается как запрос на создание альтернативного, определяемого содержимым текстового представления рисунка.

```
                                    \Picture+{}%
                                      \begin{tabular}{lcr}
                                      1 & 2 & 3\\
<IMG SRC="try0x.gif"                  xxx & xxx & xxx\\
ALT="1      2     3                   1 & 2 & 3
xxx   xxx  xxx                        \end{tabular}%
1      2     3">                    \EndPicture
```

Вместо знака (+) можно использовать звездочку (*). В этом случае содержимое печатается в вертикальной рамке. Имя файла, как и расширение файла, в команде необязательны. Если одно из них опущено, то используется создаваемый системой элемент.

Имя файла последнего созданного растрового рисунка записывается в

\PictureFile

Команда

\NextPictureFile{*filename*}

определяет имя файла для *следующего* растрового рисунка, если это имя файла не определено командой \Picture. Команда \NextPictureFile может оказаться удобной для доступа к скрытым командам \Picture.

```
                                    \NextPictureFile{mypic.gif}
<IMG SRC="mypic.gif" ALT="ab"> and  \[\alpha^\beta\] and
<IMG SRC="mypic.gif" ALT="[Picture]">  \Picture{\PictureFile} use
use the same bitmap file.           the same bitmap file.
```

При стандартной установке предполагается, что TEX4ht присваивает файлам с растровыми рисунками имена с расширением gif. С помощью опций пакета jpg и png можно установить альтернативные расширения jpg и png соответственно. С другой стороны, команду

\ConfigurePictureFormat{*extension*}

можно использовать для динамического изменения этой настройки.

4.2 Полный пример

На рис. 4.3 показано, как HTML-файл отображается навигатором, который способен реализовать возможности HTML 4.0. Соответствующее изображение выхода в формате DVI показано на рис. 4.4, а исходный LaTeX'овский файл — на рис. 4.5.

HTML-файл содержит растровые рисунки — для символов ϕ, λ и ∞ и для массива в выключном математическом окружении \[...\]. Качество рисунков зависит от используемого монитора и плотности растровых изображений. Растровые изображения на рис. 4.4 имеют плотность 110 точек на дюйм, а на рис. 4.6 представлен фрагмент с плотностью 144 точки на дюйм.

Simulation of Energy Loss Straggling

Maria Physicist

August 6, 1998

1 Landau theory

The Landau probability distribution may be written in terms of the universal Landau function $\phi(\lambda)$ as [1]. The oscillator strengths f_i and the atomic level energies E_i should satisfy the constraints

$$f_1 + f_2 = 1 \quad (1)$$

$$f_1 \ln E_1 + f_2 \ln E_2 = \ln I \quad (2)$$

The following values have been chosen:

$$f_2 = \begin{cases} 0 & \text{if } Z \leq 2 \\ 2/Z & \text{if } Z > 2 \end{cases} \Rightarrow f_1 = 1 - f_2$$

$$E_2 = 10Z^2\text{eV} \Rightarrow E_1 = \left(\frac{I}{E_2^{f_2}}\right)^{\frac{1}{f_1}}$$

$$r = 0.4$$

The following values are obtained with $c = 4$:

n_3	$n_{B,max}$	n_3	$n_{B,max}$
16	16	2000	29.63
100	27.59	∞	32.00

References

[1] L.Landau. On the Energy Loss of Fast Particles by Ionisation. Originally published in *J. Phys.*, 8:201, 1944. Reprinted in D.ter Haar, Editor, *L.D.Landau, Collected papers*, page 417. Pergamon Press, Oxford, 1965.

Рис. 4.3. Изображение HTML-файла, созданного с помощью TeX4ht.

Simulation of Energy Loss Straggling

Maria Physicist

August 6, 1998

1 Landau theory

The Landau probability distribution may be written in terms of the universal Landau function $\phi(\lambda)$ as [1]. The oscillator strengths f_i and the atomic level energies E_i should satisfy the constraints

$$f_1 + f_2 = 1 \quad (1)$$

$$f_1 \ln E_1 + f_2 \ln E_2 = \ln I \quad (2)$$

The following values have been chosen:

$$f_2 = \begin{cases} 0 & \text{if } Z \leq 2 \\ 2/Z & \text{if } Z > 2 \end{cases} \quad \Rightarrow \quad f_1 = 1 - f_2$$

$$E_2 = 10Z^2\text{eV} \quad \Rightarrow \quad E_1 = \left(\frac{I}{E_2^{f_2}}\right)^{\frac{1}{f_1}}$$

$$r = 0.4$$

The following values are obtained with $c = 4$:

n_3	$n_{B,max}$	n_3	$n_{B,max}$
16	16	2000	29.63
100	27.59	∞	32.00

References

[1] L.Landau. On the Energy Loss of Fast Particles by Ionisation. Originally published in *J. Phys.*, 8:201, 1944. Reprinted in D.ter Haar, Editor, *L.D.Landau, Collected papers*, page 417. Pergamon Press, Oxford, 1965.

Рис. 4.4. Стандартный LaTeX'овский вывод для файла с рис. 4.3.

```
\documentclass{article}
\usepackage{tex4ht}
  \title{Simulation of  Energy Loss  Straggling}
  \author{Maria Physicist}
\begin{document}
\maketitle

\section{Landau theory}
\label{sec:phys332-1}
The Landau probability distribution may be written in terms of the
universal Landau function $\phi(\lambda)$ as \cite{bib-LAND}. The
oscillator strengths $f_i$ and the atomic level energies $E_i$ should
satisfy the constraints
\begin{eqnarray}
f_1 + f_2 & = & 1  \label{eq:fisum}\\
f_1 \ln E_1 + f_2 \ln E_2 & = & \ln I \label{eq:flnsum}
\end{eqnarray}

The following values have been chosen:
\[ \begin{array}{lcl}
f_2 = \left\{ \begin{array}{ll}
              0   & \mathrm{if}\, Z \leq 2 \\
              2/Z & \mathrm{if}\, Z > 2 \\
              \end{array} \right.    &\Rightarrow & f_1 = 1 - f_2 \\
E_2 = 10 Z^2 \mathrm{eV} &\Rightarrow &E_1 = \left(\frac{I}{E_{2}^{f_2}}
                                                \right)^{\frac{1}{f_1}} \\
r  = 0.4 & & \\
\end{array} \]
The following values are obtained with $c=4$:

\begin{tabular}{llcrr}
$n_3$  & $n_{B,max}$ & & $n_3$  & $n_{B,max}$\\ \hline
16   & 16       & & 2000  & 29.63\\
100  & 27.59    & & $\infty$ & 32.00
\end{tabular}

\begin{thebibliography}{10}
\bibitem{bib-LAND}
L.Landau.
On the Energy Loss of Fast Particles by Ionisation.
Originally published in \emph{J. Phys.}, 8:201, 1944.
Reprinted in D.ter Haar, Editor, \emph{L.D.Landau, Collected
papers}, page 417.  Pergamon Press, Oxford, 1965.
\end{thebibliography}
\end{document}
```

Рис. 4.5. Исходный LaTeX'овский файл для документа, показанного на рис. 4.3.

$$f_2 = \begin{cases} 0 & \text{if } Z \leq 2 \\ 2/Z & \text{if } Z > 2 \end{cases} \quad \Rightarrow \quad f_1 = 1 - f_2$$

$$E_2 = 10Z^2\text{eV} \quad \Rightarrow \quad E_1 = \left(\frac{I}{E_2^{f_2}}\right)^{\frac{1}{f_1}}$$

$$r = 0.4$$

Рис. 4.6. Растровое изображение с плотностью 144 точки на дюйм.

Simulation of Energy Loss Straggling

Maria Physicist

August 7, 1998

1 Landau theory

The Landau probability distribution may be written in terms of the universal Landau function $\phi(\lambda)$ as [1]. The oscillator strengths f_i and the atomic level energies E_i should satisfy the constraints

$$f_1 + f_2 = 1 \quad (1)$$

$$f_1 \ln E_1 + f_2 \ln E_2 = \ln I \quad (2)$$

The following values have been chosen:

$$f_2 = \begin{cases} 0 & \text{if } Z \leq 2 \\ 2/Z & \text{if } Z > 2 \end{cases} \quad \Rightarrow \quad f_1 = 1 - f_2$$

$$E_2 = 10Z^2\text{eV} \quad \Rightarrow \quad E_1 = \left(\frac{I}{E_2^{f_2}}\right)^{\frac{1}{f_1}}$$

$$r = 0.4$$

The following values are obtained with $c = 4$:

n_3	$n_{B,max}$	n_3	$n_{B,max}$
16	16	2000	29.63
100	27.59	∞	32.00

References

[1] L.Landau. On the Energy Loss of Fast Particles by Ionisation. Originally published in *J. Phys.*, 8:201, 1944. Reprinted in D.ter Haar, Editor, *L.D.Landau, Collected papers*, page 417. Pergamon Press, Oxford, 1965.

Рис. 4.7. Вариант рис. 4.3.

Для сравнения на рис. 4.7 показан вариант рис. 4.3, который был получен с использованием следующего CSS-кода (см. разд. 4.3.4) при компиляции исходного файла:

```
\Css{.maketitle{ border:solid 5px; width: 100\% }}
\Css{.sectionHead, .likesectionHead {
   text-align:right;
   font-family: cursive;
   border-bottom:solid 2px; }}
\Css{.thebibliography { font-size : 70\%; }}
\Css{body { text-align: justify; }}
```

4.3 Создание элементов гипертекста вручную

Относительно малое количество низкоуровневых команд составляет тот фундамент, на котором (обычно) можно создать требуемый гипертекстовый результат. Эти команды обеспечивают непосредственный доступ к элементам HTML, средства для создания гипертекстовых страниц, необходимую поддержку для установления гиперссылок и метод для определения конкретной формы представления содержимого.

4.3.1 Исходный гипертекстовый код

Грубый доступ ко многим полезным функциональным возможностям можно получить путем вставки маленьких фрагментов HTML-кода. В подобных ситуациях удобно использовать команду

\HCode{*content*}

поскольку она задерживает обработку своего содержимого вплоть до макрорасширения. В таком ограниченном режиме символы выводятся необработанными, а не заменяются на соответствующие символы текущего шрифта (работа TeX4ht со шрифтами описывается в разд. 4.6.7).

```
Make it
<STRONG>&lt;STRONG&gt;</STRONG>.
```

```
Make it
\HCode{<STRONG>}\texttt
{<STRONG>}\HCode{</STRONG>}.
```

Команда

\Hnewline

может быть использована для принудительного разрыва строк внутри параметров команд \HCode.

С другой стороны, альтернативная команда

\HChar{*html-character-code*}

при каждом обращении к ней вводит один необработанный символ. Например, при стандартной установке специальный LaTeX'овский символ тильды ~ преобразуется в \HChar{160}.

Следует отметить, что в отличие от команды \HCode параметр LaTeX'овской команды \verb необязательно преобразуется к точно такому же формату в HTML-файле. Пользователю достаточно показать нечто похожее на содержимое первоисточника. То же самое относится и к тексту, заключенному внутри окружения verbatim.

4.3.2 Гипертекстовые страницы

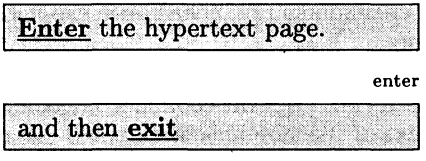
Автоматическое членение документов на гипертекстовые страницы основывается на их логической структуре и выбранных опциях. Для принудительного создания гипертекстовых страниц можно использовать следующую пару команд:

```
\HPage{entry-anchor}
\EndHPage{}
```

Страница может содержать навигационную кнопку с командой вида

```
\ExitHPage{exit-anchor}
```

для определения обратного пути к тому месту, откуда была вызвана данная страница.




```
\HPage{Enter}
and then \ExitHPage{exit}
\EndHPage{}
the hypertext page.
```

Если параметр \ExitHPage пуст, то навигационные кнопки задействуют анкеры, определенные командами \HPage.

4.3.3 Гипертекстовые ссылки

В языке HTML для обозначения гипертекстовых ссылок используется элемент <A HREF="*target-file*#*target-loc*" NAME="*cur-loc*" *parameters*>*anchor*</A>. В каждом таком элементе содержится информация о местонахождении файла-адресата и о желаемой точке входа в этот файл. Кроме того, каждый из этих элементов присваивает имя текущему местоположению и позволяет указать другие параметры.

Следующая команда выполняет аналогичные функции, автоматически определяя при этом локальные файлы-адресаты, если они не заданы явным образом.

```
\Link[target-file parameters]{target-loc}{cur-loc}anchor\EndLink
```

В частном случае, когда параметр *target-file* пуст, TeX4ht предполагает, что файл-адресат принадлежит текущему документу и сам выполняет поиск это-

го файла. Поиск файла, содержащего адрес *target-loc*, выполняется командой `\Link`.

Если оба аргумента *target-file* и *parameters* отсутствуют, то компонента [*target-file parameters*] необязательна. Если аргумент *parameters* присутствует, то перед ним обязательно должен стоять пробел.

```
An external link to
\Link[http://www.tug.org
ID="ORG"]{}{}TUG\EndLink.
```

An external link to TUG.
An internal link to here.

```
An internal \Link
{to}{}link\EndLink{} to
\Link{}{to}here\EndLink.
```

В параметрах команды `\Link` специальные символы `~`, `_` и `%` следует вводить с помощью команд `\string~`, `\string_` и `\%` соответственно.

4.3.4 Каскадные стили

HTML — это язык для идентификации структур в гипертекстовых документах; он похож на способ применения LaTeX'овских инструкций `\section` для идентификации логических структур в исходных документах. В обоих случаях вопрос о том, в каком виде эти структуры представляются читателю, не затрагивается. Его решение возлагается на спецификации, введенные где-либо с помощью языков специального назначения.

Каскадные стили (Cascading Style Sheets — CSS) — это язык для определения способов представления объектов языка HTML. TeX4ht собирает подобные запросы в один файл с именем *jobname*`.css` и выдает еще несколько своих собственных запросов.

Описание языка CSS выходит за рамки данной главы. Тем не менее понятно, что авторы могут легко изучить и использовать этот язык, если они знают основы языка HTML и немного знакомы с терминологией настольных издательских систем.

Команда

`\Css{`*CSS code*`}`

предназначена для задания способа представления элементов документа.

Если исходный код задан в виде

```
\section{Header} Text.
\par
More text
```

то TeX4ht создаст примерно такой HTML-выход

```
<H2>1 Header</H2> <P CLASS="noindent"> Text.
<P CLASS="indent"> More text
```

и код на языке CSS наподобие

```
P.noindent { text-indent: 0em } P.indent { text-indent: 1.5em }
```

С помощью команды вида `\Css{ H2 { color: green } }` можно потребовать, чтобы для заголовка использовался зеленый цвет.

Можно запросить конкретный файл CSS с помощью окружения

```
\CssFile[list-of-files]
content
\EndCssFile
```

CSS-код можно импортировать из других файлов (*list-of-files*), а также явным образом задать внутри данного окружения.

Команда `\EndPreamble` вызывает это окружение для создания файла CSS при условии, что до этого пользователь не запрашивал такой файл.

В отдельной строке файла CSS должен содержаться комментарий `/* css.sty */`, чтобы `t4ht` — постпроцессор транслятора `tex4ht` (см. разд. 4.6.4) — мог идентифицировать его как место для вставки содержимого инструкций `\Css`. При отсутствии такой строки содержимое инструкций будет проигнорировано. Имя этого файла и его исходное содержимое могут быть изменены с помощью команды `\Configure{CssFile}{`*filename*`}{`*content*`}`.

Встроенный (внутритекстовый) CSS-код можно создать с помощью окружения

```
\Css content\EndCss
```

Рассмотрим следующие исходный файл и файл конфигурации:

```
% try.tex
\documentclass{article}
  \usepackage[try]{tex4ht}
\begin{document}
  \Css
      H2 { color : red; }
  \EndCss
  \Css{ H2 { color : green; }}
\end{document}
```

```
% try.cfg
\Preamble{html}
\begin{document}

\CssFile
   H2 { color : blue; }
   /* css.sty */
\EndCssFile
\EndPreamble
```

Из них будут созданы файлы HTML и CSS:

```
<!--- try.html -->
<STYLE TYPE="text/css">
  H2 { color : red; }
</STYLE>
```

```
/* try.css */
H2 { color : blue; }
H2 { color : green; }
```

4.4 Как работает TEX4ht

Процесс трансляции исходного документа в гипертекст (см. разд. 4.6.1 на с. 220) состоит из трех этапов: компиляции исходного текста TEX'овской программой в DVI-код, обработки DVI-кода программой `tex4ht` и выполнения заключительных процедур, необходимых для завершения трансляции.

4.4.1 От LaTeX'а к DVI

Для разрешения перекрестных ссылок в LaTeX'е требуется дважды выполнить компиляцию исходного файла с помощью TeX'а, а чтобы вместо них при использовании TeX4ht получить все гипертекстовые ссылки, может понадобиться и третья компиляция. Иногда, в случае табличных окружений с большим количеством объединяемых ячеек, может понадобиться выполнить компиляцию большее число раз, чтобы система смогла определить, как должны выглядеть эти ячейки.

Когда LaTeX загружает пакет TeX4ht, он загружает файл `tex4ht.sty` и анализирует там всего лишь несколько строк. Затем он формулирует запрос на повторную загрузку этого файла впоследствии, при которой будет просмотрена оставшаяся часть файла. Второй раз загрузка выполняется при обнаружении кода `\begin{document}`, и на этот раз учитываются также требования, обусловленные опциями пакета.

Поскольку TeX4ht «выходит на сцену» только после команды `\begin{document}`, некоторые сделанные ранее пользовательские определения могут оказаться для него недоступными, если только они не были повторно определены в файле конфигурации. Например, могут возникнуть трудности при генерации элементов HTML для показателя степени (верхнего индекса) в макро `\newcommand{\x}{a^{b}}`, который был определен до начала окружения `document`.

4.4.2 От DVI к HTML

DVI — это язык описания страницы, содержащий инструкции для определения того, что именно должно появляться в каждом месте, предназначенном для печати документа (носителя информации). И наоборот, HTML — это структурно-ориентированный язык, в котором очень мало внимания уделяется вопросам размещения информации.

Следовательно, преобразование из DVI в HTML — это во многих отношениях обратный процесс, при котором нужно восстановить информацию, возможно, потерянную при преобразовании из LaTeX'а в DVI. Этот обратный процесс может оказаться совершенно неудачным, если DVI-код был получен из исходного документа, в котором нарушено обычное расположение элементов; примером этого может быть `\hspace{-0.6em}`.

В процессе трансляции из DVI в HTML обращения к шрифтам обрабатываются с помощью виртуальных гипертекстовых шрифтов (см. разд. 4.6.7 на с. 226). Если они отсутствуют или непригодны, то пользователь может без особых усилий сформировать новые шрифты.

4.4.3 Кое-что еще

На последнем этапе процесса трансляции основное внимание уделяется созданию растровых изображений из DVI-кода. В этом вопросе TeX4ht полагается

на средства, доступные на данной платформе и позволяющие создавать изображения несколькими путями (способами).

Создание рисунков является узким местом процесса трансляции, для выполнения которого может потребоваться много времени. Для ускорения этого процесса существует ряд приемов: например, если уже доступны растровые изображения, созданные при предыдущих выполнениях процедуры компиляции, то с помощью системно-зависимых утилит их можно использовать повторно.

4.5 Расширенная настройка TEX4ht

Большинству пользователей LATEX'а может понадобиться лишь очень немного вводной информации (если вообще понадобится), кроме той, о которой уже говорилось в этой главе. Но TEX4ht — это большая система, интересная во многих отношениях. Первое представление об этом мире дают последующие разделы, в которых затрагиваются отдельные аспекты настройки и работы системы.

Большинству читателей, вероятно, будет достаточно бегло просмотреть представленный ниже материал, чтобы получить общее впечатление о затрагиваемых темах. Подробности почти не влияют на изложение материала этих разделов, и поэтому их можно пропустить, пока они не понадобятся для выполнения особых требований.

4.5.1 Файлы конфигурации

TEX4ht вводит свои собственные промежуточные интерфейсы, занимающие промежуточное положение между интерфейсами LATEX'а для исходных файлов и интерфейсами HTML для выходных файлов. Это делается путем вставки «зацепок» в LATEX'овские файлы стилей. Они представляют собой команды, которые пользователь может переопределить для получения желаемого результата. Эти промежуточные интерфейсы, с одной стороны, не связаны с LATEX'овскими «заботами о стиле», а с другой — предлагают конструкции, аналогичные имеющимся в HTML. В результате пользователи могут очень легко получать различные результаты, просто определяя те или иные маркеры ссылки для создания соответствующего HTML-кода.

Обычно имеет смысл не только размещать определения все вместе в отдельных файлах настройки, но и делать то же самое с командами конфигурации TEX4ht. Однако дополнительный аргумент в пользу файлов конфигурации обусловлен необходимостью того, что получаемый результат должен удовлетворять структурным требованиям к HTML-файлам.

Реально HTML-файл состоит из заголовка и тела (основной части), причем предполагается, что эти части содержат разные типы информации. Файлы конфигурации определяют эти части и предоставляют содержимое для заголовка.

4.5.1.1 Неявные и явные файлы

Если первая опция пакета не является именем файла конфигурации, то используется стандартная конфигурация (по умолчанию). Она выглядит примерно так:

```
\Preamble{options}
\begin{document}
\EndPreamble
```

Однако если первая опция *является* именем файла конфигурации, то этот файл конфигурации должен иметь следующую структуру:

```
ранние определения
\Preamble{опции}
Преамбула
\begin{document}
вставки в заголовки HTML-файла
\EndPreamble
```

На рис. 4.8 показан пример исходного файла и файла конфигурации вместе с создаваемыми ими HTML- и CSS-файлами.

```
\documentclass{article}
  \usepackage[try,html]{tex4ht}
\begin{document}
  \begin{itemize}
  \item First
  \item Second
  \end{itemize}
\end{document}
```

(a)

```
\Preamble{}
   \Css{ UL{border : solid 1px;} }
\begin{document}
   \HCode{<META NAME="description"
               CONTENT="example">}
\EndPreamble
```

(b)

```
<HTML><HEAD>
 <META NAME="description"
    CONTENT="example">
 <LINK REL="stylesheet"
       TYPE="text/css"
       HREF="try.css">
</HEAD><BODY>
 <UL><LI>First
     <LI>Second
</UL></BODY></HTML>
```

(c)

```
UL{border : solid 1px;}
```

(d)

Рис. 4.8. Входные файлы: (a) `try.tex` и (b) `try.cfg`. Выходные файлы: (c) `try.html` и (d) `try.css`.

Как только встретится команда \usepackage, файл tex4ht.sty частично загружается для поиска нескольких определений. Затем выполняется чтение файла конфигурации, пока не встретится команда \Preamble. Остальная часть стилевого файла читается и выполняется до тех пор, пока в исходном документе не появится команда \begin{document}.

За исключением опции пакета, обозначающей имя файла конфигурации, порядок расположения других опций между командами \usepackage и \Preamble не имеет значения. Более того, в отличие от ситуации с первой опцией команды \usepackage на тип первой опции команды \Preamble не накладывается никаких ограничений.

4.5.1.2 Внутритекстовая конфигурация

Файл конфигурации можно также вставить непосредственно внутрь исходного документа, вместо того чтобы косвенно подключать его с помощью команды \usepackage. Такой подход дал бы возможность более свободно размещать инструкцию \begin{document} в файле конфигурации. Однако для того, чтобы сохранить поддерживаемый LATEX'ом авторский стиль, пользователям настоятельно рекомендуется не использовать этот подход.

Вложение может быть выполнено путем замены инструкции \usepackage[*options*]{tex4ht} на команду \input{tex4ht.sty} и подстановки содержимого файла конфигурации в команду \begin{document} в исходном документе. В таком случае опции команды \usepackage нужно переместить в список опций команды \Preamble.

Следующий исходный документ — это тот же документ, который показан на рис. 4.8(a), но со встроенным в него файлом конфигурации, показанным на рис. 4.8(b).

```
\documentclass{article}
   \input{tex4ht.sty}
   \Preamble{html}
      \Css{UL { border : solid 1px; }}
\begin{document}
      \HCode{<META NAME="description"  CONTENT="example">}
   \EndPreamble
   \begin{itemize} \item First  \item Second  \end{itemize}
\end{document}
```

4.5.2 Оглавления

LATEX'овская команда \tableofcontents в TEX4ht обогащается новыми возможностями. Для того чтобы обеспечить простое управление типом включаемых в нее элементов, способом их представления и их возможным расположением, некоторые из этих функций косвенно активизируются с помощью опций пакета 1, 2, 3 и 4 (см. разд. 4.1.1 на с. 191).

Нужно помнить, что поскольку из одного LaTeX'овского файла часто создается много HTML-файлов, каждый результирующий документ (выход) может иметь свое собственное оглавление. Поэтому обычный термин «оглавление» используется здесь во множественном числе: «оглавления».

4.5.2.1 Выбор записей

Типы элементов, которые должны быть включены в оглавления, определяются LaTeX'ом стандартным способом. В качестве альтернативного варианта в TeX4ht добавляется модифицированная команда `\tableofcontents[`*units*`]`, в которой тип входов должен быть явно определен. Параметр *units* представляет собой список имен команд секционирования (без обратных косых черт), в котором в качестве разделителя используется запятая. Варианты команд со звездочкой (заканчивающиеся знаком `*`) заменяются именами с префиксами `like`, а приложения вызываются словом `appendix`. Таким образом команда

```
\tableofcontents[chapter,appendix,section,likesection]
```

предписывает создать оглавление, элементы которого указывают на логические единицы, созданные командами `\chapter`, `\section` и `\section*`.

4.5.2.2 Локальные оглавления

Команда

`\TocAt{`*units*`}`

предписывает создавать оглавление в начале каждой логической единицы определенного типа. Параметр *units* — это список с запятой в качестве разделителя, аналогичный тому, который использовался в новом варианте команды `\tableofcontents`; первое имя в этом списке определяет раздел, который должен иметь локальное оглавление. Остальные имена определяют, какие типы элементов должны включаться в это оглавление. Имена, перед которыми стоит косая черта, определяют *конечную точку* оглавлений. Таким образом, команда

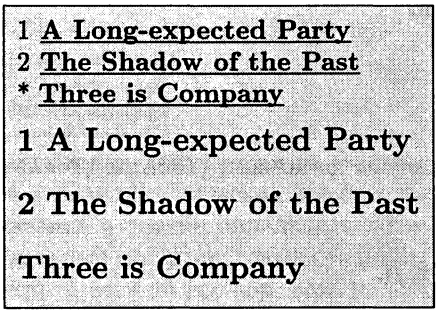
```
\TocAt[chapter,section,/likesection]
```

предписывает создавать оглавление в начале каждой главы. В него должны включаться элементы, соответствующие последующим командам `\section`, но при достижении команды `\section*` или следующей команды `\chapter` список завершается.

Оглавления, созданные командой `\TocAt{`*units*`}`, размещаются сразу же после заголовков структурных единиц. При использовании варианта команды вида

`\TocAt*{`*units*`}`

создаются аналогичные оглавления, которые располагаются не сразу после заголовка, а после вводной части (преамбулы) раздела.




```
\ConfigureToc{section}
   {\null}{~}{}{ }
\ConfigureToc{likesection}
   {}{*~}{}{ }
\tableofcontents[section,likesection]
\section{A Long-expected Party}
\section{The Shadow of the Past}
\section*{Three is Company}
```

Рис. 4.9. Конфигурирование оглавлений.

4.5.2.3 Конфигурирование элементов оглавлений

Каждый элемент оглавления образуется из трех полей: маркера, заголовка и номера страницы. Обычно маркер — это номер раздела (или он просто пуст), а номер страницы в данном контексте практически не имеет значения.

Приведенную ниже команду можно использовать для того, чтобы определить, как должны создаваться элементы оглавления для логических частей типа *unit*. На имена этих частей распространяются те же самые соглашения, которые использовались для параметра расширенной команды `\tableofcontents`.

`\ConfigureToc{`*unit*`}{`*mark*`}{`*title*`}{`*page*`}{`*end*`}`

Каждый элемент оглавления будет формироваться из параметра *mark*, за которым следует поле маркера, параметра *title* со следующим за ним заголовком и параметра *page*, после которого стоит номер страницы. И на последнем месте стоит параметр *end*. Если какой-либо из параметров пуст, то соответствующее поле в оглавлении *пропускается*. Результат показан на рис. 4.9.

4.5.2.4 Конфигурирование оглавлений

Оглавления содержат «зацепки» перед своими точками входа и после точек выхода, после своих последних элементов, а также в начале каждого абзаца, как с отступом, так и без него. Конфигурация «зацепки» выполняется следующей командой:

`\Configure{tableofcontents}{`*before*`}{`*end*`}{`*after*`}{`*n-par*`}{`*i-par*`}`

«Зацепка», расположенная за последним элементом, обрабатывается в окружениях оглавления. «Зацепки», расположенные после *точки выхода* из оглавления, обрабатываются в окружении, содержащем эти оглавления.

Дополнительную конфигурацию локальных оглавлений можно выполнить аналогичным образом с помощью команды

```
\Configure{TocAt}{before}{after}
\Configure{TocAt*}{before}{after}
```

Новые конфигурации должны быть определены до команд `\TocAt` и `\TocAt*`.

4.5.3 Части, главы, разделы и т. д.

Команды секционирования определяют базовые структуры документов и очень часто управляют членением гипертекстовых документов на файлы. В этом подразделе показывается, как можно выполнить настройку подобных объектов.

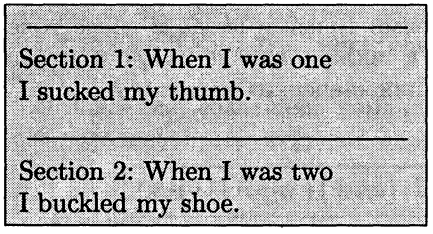
4.5.3.1 Конфигурирование граничных точек и заголовков

Команды секционирования создают логические элементы, которые характеризуются как своими начальными и конечными точками, так и своими заголовками. Следующая команда наполняет эти элементы содержимым, которое вставляется в начало и конец элементов, а также перед заголовками и, соответственно, после них.

```
\Configure{unit}{top}{bottom}{before}{after}
```

Если оба параметра *top* и *bottom* пусты, то соответствующая часть команды конфигурирования игнорируется и продолжают действовать старые значения. То же самое относится и к параметрам *before* и *after*. Результат показан на рис. 4.10.

В отношении названий логических элементов действуют такие же соглашения, как и для параметров расширенной команды `\tableofcontents`.




```
\def\EGpagesize{.8\hsize}%
\Configure{section}
  {\HCode{<HR>}} {}
  {Section \thesection: } {}

\section{When I was one}
  I sucked my thumb.
\section{When I was two}
  I buckled my shoe.
```

Рис. 4.10. Конфигурирование заголовков раздела.

4.5.3.2 Разбиение на файлы

Опции пакета 1, 2, 3 и 4 (см. разд. 4.1.1 на с. 191) неявным образом активизируют команду

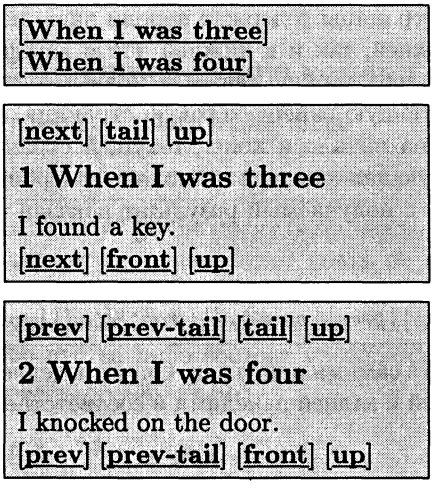
`\CutAt{`*units*`}`

для деления документов на файлы. Параметр *units* представляет собой список названий логических элементов с запятой в качестве разделителя. Первое название в списке определяет логические единицы, для которых требуются отдельные гипертекстовые страницы. Созданные гипертекстовые страницы не закрываются до тех пор, пока не начнется логический элемент, название которого присутствует в остальной части списка. Таким образом, команда

```
\CutAt{chapter,likechapter,appendix,part,likepart}
```

предписывает создание гипертекстовых страниц для логических единиц, определенных командой `\chapter`. Более того, в этой команде говорится, что главы, помеченные звездочкой, приложения, обычные части и части, помеченные звездочкой, являются логическими единицами, которые не надо включать в главы.

Команда `\CutAt{`*units*`}` обычно используется совместно с оглавлением, элементы которого содержат ссылки на гипертекстовые страницы, создание которых предписывается данной командой. Возможной альтернативой этому является использование модифицированной команды `\CutAt{+`*units*`}`, которая помимо самих гипертекстовых страниц создает также и ссылки на эти страницы. Результат показан на рис. 4.11.




```
\CutAt{+section}

\section{When I was three}
I found a key.

\section{When I was four}
I knocked on the door.
```

Рис. 4.11. Конфигурирование разделов для создания нескольких файлов.

Ссылки на эти страницы заключаются между ограничителями, которые формируются командой

\Configure{+CutAt}{*unit*}{*ldel*}{*rdel*}

Таким образом, команда

```
\Configure{+CutAt}{section}{*~}{}
```

указывает, что для ссылок на страницы, которые создаются командой \section, в качестве левого ограничителя должен использоваться знак *~, а правый ограничитель отсутствует.

4.5.3.3 Установка граничных точек

Команды \CutAt{*units*} и \CutAt{*+units*} создают конечные точки разделов, которые размещаются в отдельных файлах. Конечные точки других разделов должны быть определены следующей командой:

\Configure{*endunit*}{*units*}

Параметр *endunit* означает название логического элемента, перед которым стоит префикс end, а параметр *units* — это список названий элементов, разделенных запятой.

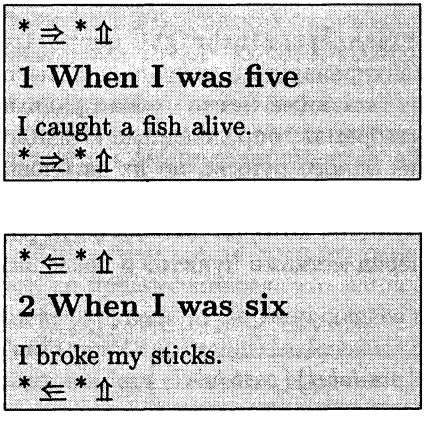
4.5.3.4 Настройка навигационных кнопок

Созданные из разделов гипертекстовые страницы содержат панели навигационных кнопок, расположенные как в верхней, так и в нижней части каждой страницы. Каждая из этих кнопок вставлена между левым и правым ограничителями. Кнопки указывают на следующую гипертекстовую страницу, на начало и конец предыдущей страницы, на начало и конец текущей страницы, а также на исходную страницу. Следующая команда позволяет настроить как сами кнопки, так и их ограничители, а получаемый результат показан на рис. 4.12

\Configure{crosslinks}{*ldel*}{*rdel*}{*next*}{*prev*}{*prev-tail*}{*front*}{*tail*}{*up*}

Представленная ниже команда имеет дело с самими панелями. Она определяет, что должно быть помещено перед передней и задней панелями и соответственно после:

\Configure{crosslinks+}{*before-front*}{*after- front*}{*before-tail*}{*after-tail*}

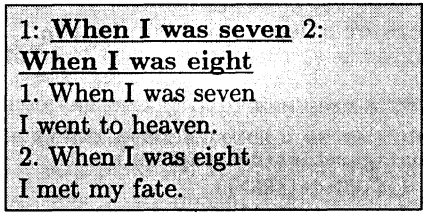



```
\Configure{crosslinks}{*~}{ }
   {$\Rightarrow$}{$\Leftarrow$}
   {}{}{}{$\Uparrow$}

\section{When I was five}
I caught a fish alive.

\section{When I was six}
I broke my sticks.
```

Рис. 4.12. Конфигурирование навигационных кнопок.




```
\NewSection{\head}{\arabic{head}}
\Configure{head}
   {\addtocounter{head}{1}}{}
   {\par\arabic{head}. }{}
\ConfigureToc{head}{ } {: }{}{}
\newcounter{head}
\tableofcontents[head]
\head{When I was seven}
  I went to heaven.
\head{When I was eight}
  I met my fate.
```

Рис. 4.13. Определение новых команд секционирования.

4.5.4 Определение команд секционирования

С помощью инструкций вида

`\NewSection{`*cmdname*`}{`*marker*`}`

для пакета TEX4ht могут быть введены новые команды секционирования. Благодаря этому пакет будет знать об их существовании и о выполняемых ими стандартных функциях. Параметр *marker* определяет маркеры, которые будут передаваться в оглавление вместе с заголовками. Как правило, такие маркеры либо пусты, либо состоят из порядкового номера текущего раздела. Пример показан на рис. 4.13.

4.5.5 Перечни

Окружения `list` и `trivlist` являются основными структурами LATEX'а, на базе которых определяется довольно много окружений. Некоторые из них сами

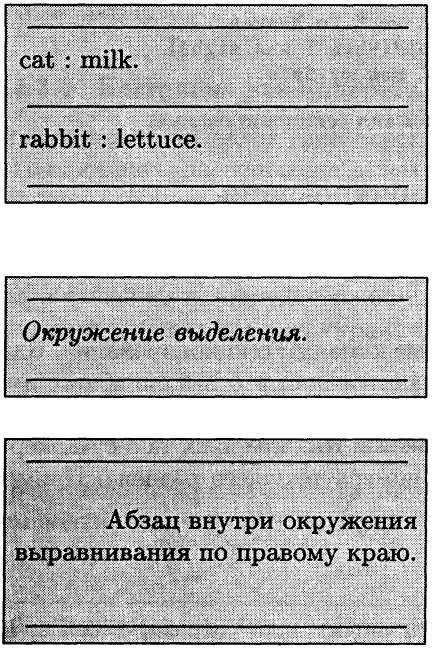
являются вариантами окружений перечня, например окружения `description`, `itemize`, `enumerate` и `thebibliography`.

Другие окружения, например `center`, `flushleft`, `flushright`, `quotation`, `quote`, `verbatim` и `verse`, в сущности ориентированы на визуальный вывод данных и используют перечни из одного пункта и без метки только ради их характеристик верстки. Окружения типа «теорема», определенные командой `\newtheorem`, также являются перечнями из одного пункта, но их заголовки вводятся как метки команд `\item`.

Следующая команда определяет информацию, которая должна быть размещена перед перечнями, после перечней, перед метками пунктов и после них. Результат показан на рис. 4.14.

`\ConfigureList{`*name*`}{`*pre-list*`}{`*post-list*`}{`*pre-label*`}{`*post-label*`}`

Если описывающие перечень окружения определяются на основе других аналогичных окружений, то добавление команды `\ConfigureList` оказывает влияние только на перечни самого верхнего уровня. Поскольку в LaTeX'е окружение `description` определяется на основе окружения `list`, а в TeX4ht оба эти окружения конфигурируются с помощью команды `\ConfigureList`, то заданная конфигурация окружения `list` не действует в перечнях `description`.




```
\ConfigureList{description}
   {}{\HCode{<HR>}}
   {\HCode{<HR>}} {: }
\begin{description}
\item[cat] milk.
\item[rabbit] lettuce.
\end{description}

\ConfigureEnv{em}
   {\HCode{<HR>}}
   {\HCode{<HR>}} {}{}
\begin{em}
Окружение выделения.
\end{em}

\ConfigureEnv {flushright}
   {\HCode{<DIV ALIGN="RIGHT">}}
   {\HCode{</DIV>}}
   {\HCode{<HR>}} {\HCode{<HR>}}
\begin{flushright}
Абзац внутри окружения
выравнивания по правому краю.
\end{flushright}
```

Рис. 4.14. Конфигурирование перечней и окружений.

4.5.6 Окружения

Структуры LATEX'овских окружений можно настроить с помощью команд, вызываемых в точках входа и выхода:

```
\ConfigureEnv{name}{before-env}{after-env}{before-list}{after- list}
```

Параметры *before-environment* и *after-environment* определяют данные, которые должны быть размещены до и после указанного окружения. Если оба эти параметра пусты, то они игнорируются. Пример показан на рис. 4.14.

Аналогично, если хотя бы один из параметров *before-list* или *after-list* не пуст, то предполагается, что данное окружение должно быть реализовано на основе окружения, формирующего перечень. После этого вызывается команда `\ConfigureList{`*name*`}{`*before-list*`}{`*after-list*`}{}{}` для конфигурирования базовых команд создания перечня.

4.5.7 Таблицы

TEX4ht прилагает большие усилия к удовлетворительному воспроизведению таблиц, но он не дает исчерпывающего решения. Иногда он терпит неудачу, и в этих случаях могут потребоваться специальные конструкции или рисунки.

4.5.7.1 Окружения `array` и `tabular`

Различие между окружениями `array` и `tabular` состоит только в том, что первое из них используется в математическом режиме, а второе обрабатывается в обычном режиме. Настройка этих окружений выполняется следующей командой (перед таблицами и после таблицы, перед каждой строкой и после каждой строки, перед каждым элементом и после каждого элемента):

```
\Configure{table}{pre-tbl}{post-tbl}
  {pre-row}{post-row}{pre-entry}{post-entry}
```

Для облегчения конфигурирования таблиц можно использовать макро `\HRow`, `\HCol` и `\ALIGN`. Первые два макро генерируют номера строк и столбцов, в которых появляются команды; третий макро формирует кодированное представление информации о выравнивании в данной таблице (см. ниже):

```
\Configure{tabular} {}{}
  {\HRow: }{\HCode{<BR>}}
  {}{(\HCol)}
\begin{tabular}{ccc}
A&B&C\\ D&E&F
\end{tabular}
```

1: A(1) B(2) C(3)
2: D(1) E(2) F(3)

Для столбцов, выравниваемых по центру, макро `\ALIGN` выдает триплет, сформированный из цифры 0, номера столбца и знака минус -. Аналогичные

триплеты создаются для столбцов, выравниваемых по левому или по правому краю, а также для столбцов, содержащих абзацы текста. Единственное отличие состоит в том, что вместо знака - используются символы <, > и p соответственно.

Для стабилизации таблиц, содержащих ячейки типа \multicolumn, необходимо несколько раз выполнить LaTeX'овскую компиляцию. Это связано с тем, что информация о размерах колонок, полученная при предыдущих выполнениях компиляции, медленно передается в TeX4ht. При конфигурировании ячеек таблиц для определения количества охватываемых этими ячейками колонок можно использовать макро \MULTISPAN.

Рассмотрим следующий исходный код:

```
\Configure{tabular} {\HCode{<TABLE>}} {\HCode{</TABLE>}}
  {\HCode{<TR>}} {\HCode{</TR>}}
  {\HCode{<TD \ifnum \MULTISPAN>1 COLSPAN="\MULTISPAN"\fi>}}
  {\HCode{</TD>}}
\begin{tabular}{lr}   \multicolumn{2}{c}{merge}\\
first & second        \end{tabular}
```

Из него получается следующий результат:

```
<TABLE><TR>   <TD COLSPAN="2">merge</TD>      </TR>
       <TR>   <TD>first</TD> <TD>second</TD> </TR></TABLE>
```

Опции пакета pic-array и pic-tabular (разд. 4.1.1 на с. 191) предписывают создание графических вариантов всех таблиц array и tabular соответственно.

Команда \\ выступает в роли разделителя строк. Чтобы избежать появления нежелательных пустых строк в конце таблицы, не следует вставлять \\ после последней строки. С другой стороны, с помощью символа ~ пустые ячейки можно наполнить невидимым содержимым. Благодаря этому навигаторы смогут по-разному обрабатывать пустые и непустые ячейки.

4.5.7.2 Окружение eqnarray и ему подобные

Конфигурацию различных вариантов окружения eqnarray можно изменить с помощью команд \Configure, которые аналогичны командам, использующимся для окружений array и tabular. Альтернативным вариантом может быть создание рисунка с помощью опции pic-eqnarray.

4.5.7.3 Окружение tabbing

Следующая команда определяет, какая информация должна быть помещена перед и после строк, а также перед и после ячеек окружения tabbing. Кроме того, в этой команде предусматривается возможность задать в виде десятичного числа коэффициент увеличения для ширины граф.

`\Configure{tabbing}[mag]{pre-row}{post-row}{pre- entry}{post-entry}`

В тех случаях, когда изменять увеличение не требуется, компонента [*mag*] необязательна. Если все параметры *pre-row*, *post-row*, *pre-entry* и *post-entry* пусты, все предлагаемые ими вставки игнорируются. С помощью команды `\TABBING` можно задать ширину всех ячеек, на ширину которых не наложено ограничений, и заданное значение ширины равно нулю. Этим свойством обладают замыкающие ячейки строк.

Реконфигурировать таблицы так, чтобы не ухудшить их свойства, — это задача, для решения которой могут понадобиться более глубокие знания об основах программирования в TEX'е, чем те, которыми обладает большинство пользователей. Однако объем требуемого TEX'овского кода, как правило, бывает достаточно мал.

```
\newcount\c
\def\Width#1//{\gdef\TABBING{#1}%
   \ifnum\c>0 \HCode{ WIDTH="\the\c"}\fi}
\ConfigureEnv{tabbing}{}{}{\Configure{HtmlPar}{}{}{}{}}{}
\Configure{tabbing}
   {\HCode{<TABLE><TR>}}   {\HCode{</TR></TABLE>}}
   {\HCode{<TD}\afterassignment\Width\c\TABBING//\HCode{>}}
   {\HCode{</TD>}}
\begin{tabbing}
LaTeX: \=tabbing\\
TeX: \>settabs
\end{tabbing}
```

Приведенный выше фрагмент исходного LATEX'овского документа преобразуется в следующий HTML-код:

```
<TABLE><TR><TD WIDTH="71">LaTeX:</TD>
           <TD>tabbing</TD>          </TR></TABLE>
<TABLE><TR><TD WIDTH="71">TeX:</TD>
           <TD>settabs</TD>          </TR></TABLE>
```

С помощью опции пакета `pic-tabbing` можно запросить воспроизведение таблицы в виде рисунка. Разновидность `pic-tabbing'` применяется только в тех случаях, когда в окружении `tabbing` используется директива `\'`. TEX4ht не в полной мере поддерживает эту директиву.

4.5.8 Незначительные подробности

Большинство из описанных до сих пор возможностей связано с конкретными конструкциями LATEX'а и очень редко используется в каком-либо другом месте. Следующие свойства, более универсальные по своей природе, являются более низкоуровневыми.

4.5.8.1 Имена файлов

На процесс создания HTML-файлов влияют опции пакета 1, 2, 3 и 4, а также команды \CutAt и \HPage. В таких случаях система автоматически генерирует имена файлов, если только пользователи не предлагают свои собственные варианты имен.

Чтобы узнать имя текущего файла, можно использовать команду \FileName. С другой стороны, команду

\NextFile{*filename*}

можно применить для того, чтобы предложить имя для следующего HTML-файла.

4.5.8.2 Условный код

Команда

\ifHtml *true-part*\else *false-part*\fi

позволяет выбрать содержимое в зависимости от того, используется или нет опция пакета html.

4.5.8.3 Окружения для скриптов

Команда \HCode позволяет пользователю записывать в HTML-файл небольшие фрагменты исходного кода. Команда

\ScriptEnv{*name*}{*prefix*}{*suffix*}

предоставляет средства определения окружений для записи более крупных фрагментов исходного кода.

```
<STYLE TYPE="text/css">
<!--
  UL { border : solid 1px; }
  H1 { color: green }
-->
</STYLE>
```

```
\ScriptEnv{css}
  {\HCode{<STYLE TYPE="text/css">
  \Hnewline<!--}\Hnewline}
  {\HCode{-->\Hnewline</STYLE>}}
\begin{css}
  UL { border : solid 1px; }
  H1 { color: green; }
\end{css}
```

4.5.8.4 Содержимое для концов абзаца

Следующая команда дает возможность определить, какие данные должны быть вставлены в начале некоторого абзаца, а также, что должно быть сохранено в этом месте в команде \EndP. Отдельные параметры предусмотрены для случаев, когда первая строка абзаца начинается с отступа или без него.

\Configure{HtmlPar} {*noindent-P*} {*indent-P*} {*noindent-save*} {*indent-save*}

Задача команды \EndP обычно состоит в том, чтобы передать код из начала абзаца в его конец.

```
\Configure{HtmlPar}
   {\HCode{<P><H2>}}
   {\EndP \HCode{<P>}}
   {\HCode{</H2>}}  {}
\noindent Head \par Body
```

<P><H2>Head </H2><P> Body

Существуют дополнительные команды, позволяющие выполнить более тонкую местную настройку информации, добавляемой в начале абзацев. Команда \IgnorePar игнорирует вставку содержимого в начало следующего абзаца, а команда \ShowPar задает содержимое в начале следующего абзаца. Подобным же образом команда \NoIndent определяет, что первая строка следующего абзаца должна набираться без отступа, а команда \Indent определяет, что первая строка следующего абзаца должна иметь отступ.

4.5.8.5 Создание новых «зацепок» для TEX4ht

Ядро TEX4ht запрограммировано для работы в стандартных ситуациях, возникающих при работе базовых TEX'овских алгоритмов, и, как правило, оно хорошо справляется с этой работой. Однако его основные функции обычно почти не связаны со *структурными* свойствами команд, определенных в частных и общих стилевых файлах. Для того чтобы охватить такие свойства, эти определения необходимо расширить так, чтобы они включали в себя *зацепки*. Чтобы зацепки приносили больше пользы, они должны быть реконфигурируемыми. Для этой цели предназначена команда

\NewConfigure{*name*}[*digit*]{*assignments*}

Эта команда вводит зацепку, конфигурацию которой можно изменять с помощью команды \Configure. Параметр *digit* определяет количество необходимых для команды \Configure реконфигурируемых полей. Доступ к этим полям можно получить, задавая в команде \NewConfigure имена параметров #1, #2 и т. д. Пример показан на рис. 4.15.

```
\newcommand{\try}
          {\Pre Just trying.\Post}
\NewConfigure{try}[2]{%
   \newcommand\Pre{#1}%
   \newcommand\Post{#2}}
\Configure{try}{\HCode{<H2>}}
                {\HCode{</H2>}}
\try
\Configure{try}{}{}
\try
```

Just trying.

Just trying.

Рис. 4.15. Добавление новых маркеров ссылки.

TeX4ht создает зацепки с заданными начальными значениями для команд, содержащихся в LaTeX'овских стилевых файлах, plain-TeX'е, стилевых файлах пакетов $\mathcal{AMS}$-LaTeX и $\mathcal{AMS}$-TeX, а также в других часто используемых пакетах.

4.6 Как работает TeX4ht

Понимание того, как работает система, может помочь при ее инсталляции и способствовать улучшению и расширению ее использования. Кроме того, это может объяснить возможности системы и присущие ей ограничения. Хотя большинство этих вопросов обычно имеет значение только для небольшого числа наиболее квалифицированных пользователей, многие читатели, может быть, захотят бегло просмотреть этот раздел только лишь для того, чтобы получить общее впечатление о рассматриваемых там темах.

4.6.1 Процесс трансляции

Командная строка

```
ht latex filename
```

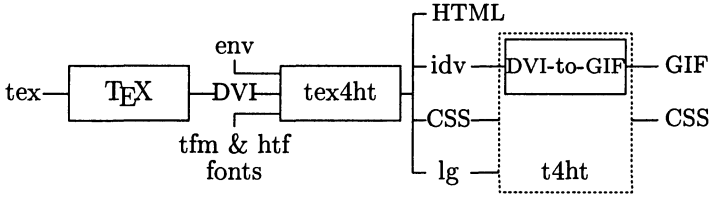
вызывает трансляцию исходного файла *filename* в HTML. Скрипт `ht` вызывает различные утилиты, используемые в процессе трансляции (рис. 4.16); он состоит из пяти шагов, первые три из которых заключаются в выполнении LaTeX'а:

```
latex  filename
latex  filename
latex  filename
tex4ht filename
t4ht   filename
```

4.6.2 Работа LaTeX'а

Троекратная компиляция исходного LaTeX'овского документа TeX'ом необходима для того, чтобы обеспечить как правильные ссылки в гипертекстовых ссылках, так и правильное расположение ячеек в таблицах, содержащих команду `\multicolumn`. Для исходных документов, в которых команда `\multicolumn` объединяет большее количество ячеек, компиляцию, возможно, придется выполнить большее число раз.

Создаваемый при компиляции протокольный файл будет содержать информацию, подобную той, которая показана на рис. 4.17 (без номеров строк). В третьей строке приведенного примера содержится запрос на обработку выходного файла `try.dvi` программой `tex4ht` для создания файла `try.html`, а также, возможно, и других файлов. В пятой строке фигурирует стилевой файл



Входные файлы	
`tex4ht.sty`	Стилевой файл для LATEX'а
`*.4ht`	Дополнительные стилевые файлы для `tex4ht.sty`
`*.cfg`	Конфигурации для `tex4ht.sty`
`*.htf`	Виртуальные гипертекстовые шрифты
Выходные файлы	
`*.html`	HTML-файлы
`*.css`	Стиль для HTML-кода
`*.gif`	растровые рисунки
Временные файлы	
jobname`.lg`	Скрипт из `tex4ht` для `t4ht`
jobname`.dvi`	TEX'овский вывод для `tex4ht`
jobname`.idv`	DVI-код для растровых рисунков
jobname`.xref`	Сохраняет перекрестные ссылки между TEX'овскими компиляциями
jobname`.toc`	Сохраняет оглавление между TEX'овскими компиляциями
jobname`.otc`	Сохраняет оглавление в процесс TEX'овской компиляции
`tex4ht.tmp`	Вспомогательный файл, используемый при компиляции

Рис. 4.16. Последовательность выполняемых действий и файлы TEX4ht.

```
1  (tex4ht.sty) (url.sty)
2  (try.cfg (tex4ht.sty
3  --- needs --- tex4ht try ---
4  (tex4ht.tmp) (try.xref)
5  --- file try.css ---
6  (tex4hta.sty)) (try.aux))  [1] [2]
7  (try.otc[3]) [4]
8  l. 19 Writing try.idv[1] (try0x.gif)
```

Рис. 4.17. Оперативная информация в протокольном файле LATEX'а.

с именем `try.css`, предназначенный для HTML-выхода. В восьмой строке упоминается файл `try0x.gif`, содержащий растровое изображение, которое должно быть создано программой `t4ht` из первого рисунка в файле `try.idv` (см. разд. 4.6.4 на с. 223).

Первая и вторая строки показывают, что пакетный файл `tex4ht.sty` читается два раза. В первый раз читается команда `\Preamble` и около пяти зацепок в стандартной конфигурации. Повторное чтение инициируется командой

\Preamble, и при этом читаются те части стилевого файла, которые отбираются с помощью опций. Первоначальная установка зацепок допускает настройку заголовков, прежде чем они будут записаны при повторной загрузке этого файла. К ним относятся следующие зацепки: HTML, HEAD, BODY, TITLE, TITLE+ и HtmlPar (их использование обсуждается в разд. B.2.1.4).

В процессе компиляции TeX4ht сохраняет исходные данные для оглавлений в файле toc; они должны быть использованы при следующем проходе по исходному документу. Данные, полученные при предыдущем проходе, перемещаются в файл otc, благодаря чему к ним обеспечивается доступ при текущем повторе компиляции. Примером использования такого файла является показанный в седьмой строке демонстрационного протокольного файла файл try.otc.

4.6.3 Работа программы tex4ht

В результате обработки исходного файла LaTeX'ом создается стандартный DVI-файл, содержащий специальные инструкции для пакета tex4ht. Программа заключительной обработки (постпроцессор) tex4ht[1] использует эти инструкции, чтобы определить, как должен обрабатываться DVI-код. Они передают в tex4ht информацию о том, где должны начинаться и заканчиваться выходные файлы, какие имена должны быть им присвоены, какие отличительные признаки HTML должны быть присвоены символам различных шрифтов, где должны располагаться коды растровых изображений и т. д.

Для того чтобы выполнить трансляцию в HTML, утилита tex4ht должна иметь информацию о том, где находятся TeX'овские метрики (атрибуты) шрифта (tfm файлы), где хранятся частные гипертекстовые шрифты для tex4ht (о шрифтах htf см. разд. 4.6.7 на с. 226), и другую информацию о среде (об условиях), в которой выполняется эта утилита. Часть этой информации может быть включена в исполнимый код tex4ht в процессе компиляции. Остальная ее часть определяется в управляющем (env) файле (см. разд. 4.6.8).

Вызов tex4ht вообще без параметров приводит к появлению примерно следующего (диагностического) сообщения

```
tex4ht in-file[.dvi]
   [-ttfm-font-dir]
   [-ihtf-font-dir]
   [-eenv-dir]
   [-dout-dir]
   [-gbitmap-file-ext]   [-blg-divide-script]   [-slg-gif-script]
```

[1] Утилита tex4ht написана на языке C и использует системные вызовы только очень немногих простых стандартных функций. Ее гипертекстовые шрифты представляют собой системно-независимые файлы, содержащие обычный текст, и подобно шрифтам, используемым в TeX'е и в создаваемых им выходных DVI-файлах, доступны во всех системах. Это означает, что утилиту tex4ht можно свободно переносить с платформы на платформу, и что результат не зависит от того, на какой платформе выполняется эта программа.

```
file try.html
(/n/candy/tex/texmf/fonts/tfm/public/cm/cmr10.tfm)
(/n/soda/tex4ht.dir/cmr.htf)
(/n/candy/tex/texmf/fonts/tfm/public/cm/cmbx10.tfm)
Loading 'cmr.htf' for 'cmbx10.htf'
(/n/soda/tex4ht.dir/cmr.htf)
(/n/candy/tex/texmf/fonts/tfm/jknappen/ec/ecsl1000.tfm)
--- warning --- Couldn't find font 'ecsl1000.htf' (char codes: 0--255)
[1 file try.css ]
Execute script 'try.lg'
```

Рис. 4.18. Сообщения о работе `tex4ht`.

В большинстве случаев пользователь будет просто задавать имя dvi-файла. Например, в результате выполнения команды `tex4ht try.dvi` создается головной (основной) файл с именем `try.html`.

Опция командной строки `-g`*`bitmap-file-ext`* определяет расширение имени файла для файлов, содержащих растровые изображения рисованных символов. Опция `-d`*`out-dir`* определяет каталог для выходных файлов (например, `tex4ht try -d/tmp/` на платформе UNIX). Следующие три опции: `-e`*`env-dir`*, `-i`*`htf-font-dir`* и `-t`*`tfm-font-dir`* определяют каталоги, в которых должен выполняться поиск управляющих файлов (см. разд. 4.6.8 на с. 229), виртуальных гипертекстовых шрифтов и TFM-файлов.

На рис. 4.18 показан пример сообщений, генерируемых в процессе работы `tex4ht`. Вторая строка показывает, что была найдена TEX'овская метрика `cmr10`, а в третьей строке содержится сообщение о загрузке соответствующего виртуального гипертекстового шрифта. В пятой строке сообщается о том, что вместо файла `cmbx10.htf` был загружен файл `cmr.htf`. В языке HTML символы полужирного и обычного шрифтов изображаются одинаково. Однако благодаря информации о шрифте, содержащейся в CSS-файле, эти два шрифта будут воспроизводиться по-разному.

Десятая строка содержит запрос на выполнение некоторого скрипта (см. разд. 4.6.6).

4.6.4 Взгляд на `t4ht`

После того как `tex4ht` завершает обработку TEX'овского DVI-файла, остается сокращенный DVI-файл, имеющий расширение `idv`. Он состоит из всех фрагментов DVI-кода, которые должны быть преобразованы в рисунки, причем каждая страница содержит точно один рисунок. Первая половина этого файла предназначена для рисунков, требующихся в исходном LATEX'овском файле, а вторая — для рисунков, которые нужны для шрифтов.

Информация о том, как должны называться эти рисунки, записывается в `lg`-файл; часть этой информации содержится также в TEX'овском протокольном файле. Например, в протоколе на рис. 4.17 восьмая строка содержит

```
1 Entering try.lg
2 dvips -mode ibmvga -D 110 -f try.idv -pp 1 > tmp.ps
3 convert -crop 0x0 -density 110x110 -transparent #FFFFFF tmp.ps try0x.gif
4 cmsy10-2a.gif already in
```

Рис. 4.19. Сообщения о работе `t4ht`.

утверждение `try.idv[1] (try0x.gif)`. В нем говорится, что рисунок с первой страницы файла `try.idv` будет храниться в файле с именем `try0x.gif`.

Соответствующий `lg`-файл может также содержать информацию о стиле для `css`-файла данного документа, запросы пользователя на обращения к системным функциям и другие виды входных данных. Поэтому `lg`-файл можно рассматривать как скрипт для записи действий, которые должны быть предприняты, после того как утилита `tex4ht` завершит свою работу.

Скрипт `t4ht` должен выполнить все, что записано в `lg`-файле. Это может быть выполнено с помощью какой-либо системно-зависимой программы для интерпретации скрипта. Если же сам скрипт записан на языке подготовки сценариев, который распознается на данной платформе, то он может быть выполнен непосредственно.

4.6.5 От DVI к GIF

TeX4ht не содержит средств для преобразования DVI-кода в форму растрового изображения. В этом он полагается на внешние средства, доступные при выполнении задания.

Во второй и третьей строках на рис. 4.19 показано двухступенчатое преобразование рисунка в формате DVI в GIF-файл, содержащий растровое изображение. На первом шаге для преобразования рисунка в промежуточный PostScript'овский файл используется драйвер `dvips` [↪DVIPS]. На втором шаге для завершения работы вызывается программа `convert` (являющаяся частью пакета ImageMagick [↪IMAGEMAGICK]).

В четвертой строке примера сообщается, что файл `cmsy10-2a.gif` уже существует и поэтому он заново не создается.

Размеры рисунков зависят от размера исходных документов и, в случае рисованных символов, от размеров используемых шрифтов. Однако размеры и качество рисунков могут также зависеть от выбранных значений установочных параметров для используемых внешних утилит и, конечно, от экрана монитора, используемого для просмотра HTML-страниц.

4.6.6 Знакомство с `lg`-файлом

По существу, `lg`-файл — это всего лишь составленный программой `tex4ht` список пожеланий. Некоторые из элементов этого списка создаются в самой программе. Остальные же являются запросами, сформулированными в исходном

документе; при этом **tex4ht** играет роль промежуточного звена, передающего запросы в этот файл.

В исходном документе запросы могут быть сделаны с помощью команды вида

\Needs{*request*}

Они попадут в lg-файл вложенными в «конверт». Конфигурацию самого конверта можно выполнить с помощью команды \Configure{Needs}{*content*}, в которой для ссылки на параметр команды \Needs следует использовать метку #1.

Рассмотрим следующий LATEX'овский код:

```
\Needs{chmod 644 *.html}
\Configure{Needs}{#1}
\Needs{Say hello}
```

При использовании конфигурации

```
\Configure{Needs}{l. \the\inputlineno\space--- needs --- #1 ---}
```

этот исходный код добавляет к lg-файлу следующие две строки:

```
l. 12 --- needs --- "chmod 644 *.html" ---
Say hello
```

Стандартная программа **t4ht**, поставляемая вместе с TEX4ht, будет интерпретировать формат первой строки

```
l. integer --- needs --- "content" ---
```

как системный вызов UNIX'овской команды **chmod 644 *.html**. С другой стороны, вторая строка игнорируется, потому что распознавание шаблона этой строки в данной утилите не запрограммировано.

В начале lg-файла располагаются все вставки, сделанные в исходном документе, как правило, содержащимися в стилевых файлах командами \Needs. Затем в нем перечисляются вставки, созданные утилитой **tex4ht**. Эти два типа вставок в lg-файле разделяются разграничительной строкой, которую вставляет туда **tex4ht**.

Стандартные шаблоны для вставок, создаваемых утилитой **tex4ht**, определяются во время компиляции утилиты применительно к используемой платформе. Эти шаблоны могут быть переопределены в файле оперативного управления.

Обычно lg-файлы состоят из запросов на создание растровых изображений, явных вставок в CSS-файл и информации о шрифте, которая неявно обращается к вставкам в CSS-файл.

4.6.7 Файлы управления шрифтом

В стандартном LaTeX'овском выводе текст часто содержит множество символов, принадлежащих разным шрифтам. Эти символы помещаются в исходный файл с помощью знака или макрокоманды, указывающих на символ в текущем (действующем) шрифте. Следовательно, если используется шрифт Computer Modern Roman размером 10 pt, то знак `а` во вводимых данных означает «установить символ номер 98 из шрифта cmr10», а макро `\Gamma` означает «установить символ номер 1». Поскольку перед TeX4ht не стоит задача воспроизведения символов, то он вместо того, чтобы выбирать из файла со шрифтом изображения символов (как обычно делает драйвер DVI), для каждого символа задает его значение. Это значение описывается в файлах виртуальных гипертекстовых шрифтов TeX4ht и используется Web-навигатором для воспроизведения на экране реального символа.

4.6.7.1 Использование файлов

Для заданного LaTeX'овского шрифта `tex4ht` выбирает виртуальный гипертекстовый шрифт, у которого основная часть имени файла представляет собой часть имени LaTeX'овского шрифта. Если доступны несколько таких файлов, то выбирается файл с самым длинным именем. Таким образом, если необходим виртуальный гипертекстовый шрифт для LaTeX'овского шрифта с именем `cmr10`, то `tex4ht` будет последовательно искать файлы `cmr10.htf`, `cmr1.htf`, `cmr.htf`, `cm.htf` и `c.htf`.

Каждый файл виртуального гипертекстового шрифта начинается и заканчивается одинаковыми опознавательными строками, которые определяют имя шрифта, а также числовые коды первого и последнего символов (рис. 4.20).

Каждому символу в этом файле соответствует строка, состоящая из трех полей: текстовой строки, номера класса и (возможно, пустого) комментария. Первое и второе поля должны быть ограничены каким-либо одинарным символом; можно использовать любые ограничители, но в пределах одной строки все ограничители должны быть одинаковыми.

```
cmr 0 127
'G'     '1'  Gamma    0
=D=     =1=  Delta    1
'Q'     '1'  Theta    2
'/\\'   '1'  Lambda   3
'E'     '1'  Xi       4
'TT'    '1'  Pi       5
.......................
'ffi'   ''           14
'ffl'   ''           15
.......................
'a'     ''           97
'b'     ''           98
.......................
'&#34;' ''          125
'~'     ''          126
'\168\' ''          127
cmr 0 127
```

Рис. 4.20. Фрагменты файла виртуального гипертекстового шрифта `cmr.htf`.

Класс — это число от 0 до 255, причем пустое поле класса трактуется как 0. Если номер класса четный, то содержимое первого поля строки трактуется как символ (шрифта). Если же номер класса нечетный, то это означает, что для воспроизведения символа должен использоваться растровый рисунок, при этом в первом поле содержится альтернативное представление символа для текстовых навигаторов.

Для введенных данных «a Γ» TEX4ht затребует из файла `cmr10-0.gif` растровое изображение для `\Gamma` и в результате создаст в HTML-файле следующую строку: `a <IMG SRC="cmr10-0.gif" ALT="G">`. Часть имени файла `cmr10` указывает на имя LATEX'овского шрифта, а `0` указывает на порядковый номер символа в шрифте.

Первое поле строки может непосредственно ссылаться на символы шрифта. Для этого соответствующий числовой код символа надо поместить между двумя обратными косыми чертами. С другой стороны, сам символ обратной косой черты \ должен представляться парой таких символов: `\\`.

Символы <, > и & могут быть представлены, например, строками `<`, `>` и `&` соответственно.

4.6.7.2 Конфигурирование шрифтов

Информация о символах, извлеченная из таблиц виртуальных гипертекстовых шрифтов, записывается в HTML-файлы в формате, который определяется приведенной ниже командой. Она формирует состоящий из семи компонент шаблон для символов заданного класса. В качестве разделителей должны использоваться символы, не встречающиеся в компонентах шаблона.

```
\Configure{htf} {class} {delimiter}
  {parameter-1} {parameter-2} {parameter-3}
  {parameter-4} {parameter-5} {parameter-6}
  {parameter-7}
```

Например, из LATEX'овского кода `\textsc{a}` в версии HTML 3.2 создается `<SMALL>A</SMALL>`, а в HTML 4.0 Transitional — `<SPAN CLASS="small-caps">A </SPAN>`. В любом режиме для символа 'a' шрифт `htf` обеспечивает содержимое `A` класса 4. В первом режиме при стандартной конфигурации, установленной для символов класса 4, разметка задается командой

```
\Configure{htf}{4}{+}{<SMALL>}{}{}{}{}{}{</SMALL>}
```

Второй режим отличается только своей стандартной конфигурацией

```
\Configure{htf}{4}{+}{<SPAN CLASS="}{}{}{}{}{small-caps">}{</SPAN>}
```

Если класс символа является четным числом, то первый параметр печатается буквально. Второй параметр должен соответствовать соглашениям языка C и, если он не пуст, используется для вывода имени шрифта. Третий и четвертый параметры используются аналогичным образом для записи размера

шрифта и кратности его увеличения соответственно. Остальные параметры записываются буквально, причем либо пятый, либо шестой параметр должен быть пустым. Непосредственно перед последним параметром вставляется строка, добавленная из htf-файла.

В случае, когда класс символов нечетный, параметры используются аналогичным образом для вывода имени шрифта, дублирующей строки шрифтом htf, второй копии имени шрифта, размера шрифта, коэффициента увеличения шрифта, если он отличается от 100%, и числового кода символа. Конфигурация для класса 0 используется также для создания дополнительной разметки символов других классов.

Команды \NoFonts и \EndNoFonts приостанавливают и возобновляют, соответственно, действие команды \Configure{htf}.

4.6.7.3 Добавление стиля

Добавлением htf шрифтов в CSS-файл можно управлять с помощью команд

\Configure{htf-css}{*class*}{*content*}
\Configure{htf-css}{*fontname*}{*attributes*}

Команда

```
\Configure{htf-css}{4}{.small-caps {font-variant: small-caps; }}
```

создает строку .small-caps{font-variant: small-caps; } для символов класса 4. С другой стороны, результатом выполнения команды

```
\Configure{htf-css}{cmmi}{font-style: italic;}
```

являются вставки типа .cmmi-7{ font-size:70%; font-style: italic;} и .cmmi-10{ font-style: italic;}.

4.6.7.4 Советы, как пользоваться шрифтами

Пользователи могут модернизировать существующие виртуальные гипертекстовые шрифты, чтобы получить другой вариант вывода; можно также создавать новые шрифты, если нужные шрифты отсутствуют. Если tex4ht не сможет найти подходящих htf-шрифтов, то он будет выдавать предупреждения, например такие

```
---warning --- Couldn't find font `fontname.htf'
 (char codes: first--last)
```

до тех пор, пока не будут получены новые шрифты.

При использовании опции пакета ShowFont исходный код наподобие следующего:

```
\font\x=fontname \ShowFont\x
```

создает рисунок, показывающий стандартный результат для различных символов заданного шрифта.

4.6.8 Управляющий файл

Задача файла управления заключается в том, чтобы дать утилите `tex4ht` возможность самостоятельно, причем без перекомпиляции, приспособиться к платформе, на которой она работает, и к запросам пользователей. Этот файл называется либо `tex4ht.env`, либо `.tex4ht`, и в конкретной конфигурации системы он может иметь несколько копий. Например (в приоритетном порядке), одна копия может постоянно находиться в каталоге, на который указывает опция командной строки `-e` (см. разд. 4.6.3 на с. 222), второй файл может располагаться в рабочем каталоге, а третий — в каталоге, местоположение которого зашито в программе.

Сам файл составляется из записей, идентифицируемых по первому символу в каждой строке. Некоторые из возможных вариантов перечислены ниже:

t Определяет каталог, в котором должен выполняться поиск TEX'овских (`tfm`) файлов с метрикой шрифта.

i Определяет каталог, в котором должен выполняться поиск виртуальных гипертекстовых шрифтов (`htf`) TEX4ht.

g Определяет расширение, присваиваемое файлам с растровыми изображениями. В настоящее время такие растровые файлы используются только для рисованных символов в виртуальных гипертекстовых шрифтах.

Рассмотрим следующий управляющий файл (номера строк не являются частью файла):

```
1  t/n/candy/tex/texmf/fonts/tfm/!
2  i/n/soda/tex4ht.dir/
3  i/n/soda/tex4ht.dir/ht-fonts/!
4   acomment
5  g.jpg
```

В первой строке указывается каталог для поиска файлов с метрикой шрифта, которые отсутствуют в текущем каталоге; восклицательный знак `!` означает, что этот поиск должен распространяться на вложенные подкаталоги всех уровней.

Во второй и третьей строках определяются каталоги для размещения `htf`-шрифтов, тогда как рекурсивный поиск в подкаталогах разрешается в пределах каталога, указанного в третьей строке.

Четвертая строка игнорируется, потому что она начинается с символа пробела, а этот символ не входит в число допустимых опций для элементов управляющего файла.

В пятой строке указывается, что файлы с растровыми изображениями рисованных символов вместо стандартного расширения `gif` должны иметь расширение `jpg`.

4.7 Заключение

В этой главе показано, как посредством TeX4ht преобразовать LaTeX'овские документы в HTML-файлы, пользуясь при этом широким набором функциональных возможностей для формирования результатов. Сила этой системы состоит в том, что для чтения файла она использует сам LaTeX и по сравнению с другими трансляторами допускает применение гораздо большего числа LaTeX'овских конструкций (таких, как сложные макро).

Поскольку большая часть работы TeX4ht выполняется зацепками в LaTeX'овском стилевом файле, довольно просто внести в него изменения, чтобы создать другую разметку. В приложении B.2 на с. 461 мы посмотрим, как заставить систему создать разметку XML, а в разд. 8.2.3.2 на с. 438 будут даны конкретные примеры транслятора, преобразующего LaTeX в XML, в том числе в MathML.

Глава 5

Непосредственное отображение LaTeX'а в Сети

В этой главе мы обсуждаем ряд приложений, которые воспринимают в качестве входных данных LaTeX'овский текст и воспроизводят его непосредственно в среде некоторого Web-навигатора. Как правило, такие приложения представляют собой либо динамически подключаемый дополнительный программный модуль (plug-in), либо апплет языка Java (Java-апплет). Программные средства такого рода не основываются на «реальном» TeX'овском исходном коде; они не используют также METAFONT'овские шрифты и DVI-файлы. В настоящее время ни одно из таких приложений не воспроизводит полностью все возможности LaTeX'а, однако имеет смысл поискать на Web-сайтах разработчиков новую информацию и обновленные версии.

Наиболее часто используемым динамически подключаемым модулем, расширяющим возможности Web-навигаторов в части отображения LaTeX'овских текстов, является гипермедийный навигатор `techexplorer` фирмы IBM. Он предназначен для программ Netscape Navigator и Microsoft Internet Explorer и имеет варианты для нескольких программно-аппаратных платформ. Данный модуль позволяет воспроизводить значительную часть нематематической LaTeX'овской разметки и пригоден как для отображения содержащихся в HTML-документах математических формул, так и для воспроизведения документов в целом. Большая часть этой главы посвящена тому, как можно дополнить LaTeX'овские документы для их наилучшего интерактивного воспроизведения в `techexplorer`'е.

Для воспроизведения в навигаторе математических формул часто применяется Java-апплет WebEQ. Будучи написанным на языке Java, он более или менее автоматически совместим с несколькими платформами. Апплет WebEQ обладает богатыми функциональными возможностями в части воспроизведения математических выражений, но в отношении работы с текстом его возможности существенно ограничены. Вероятно, что чаще всего вы будете использовать этот апплет в HTML-страницах.

При использовании любой из этих программ следует помнить, что обычный монитор компьютера имеет существенно меньшую разрешающую способность,

чем принтер. Поэтому относительные размеры элементов, например таких, как основные выражения и индексы, могут отличаться от тех, которые воспроизводятся на печатной странице. Кроме того, более низкое разрешение может приводить к тому, что толщина линейки будет изменяться в зависимости от ее положения на экране.

Есть еще несколько проблем практического характера, возникающих при использовании рассматриваемых программ для отображения встроенных в HTML-документы математических выражений. Эти проблемы мы проанализируем в заключительной части настоящей главы, и там же обсудим возможности улучшения сложившейся ситуации с помощью перспективных навигаторов и программ визуализации. Мы также опишем возможные способы взаимодействия этих приложений с другими программами, обеспечивающие получение действительно интерактивных научных документов.

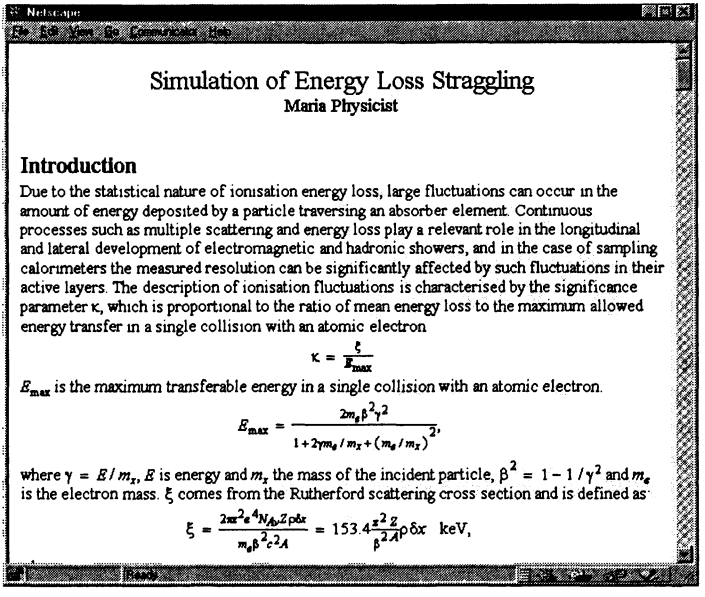
5.1 Навигатор techexplorer фирмы IBM

Разработанная фирмой IBM программа `techexplorer`[1] представляет собой динамически подключаемый модуль, предназначенный для навигаторов Netscape Navigator и Microsoft Internet Explorer. Модуль `techexplorer` фирмы IBM непосредственно отображает некоторое подмножество TeX'а и LaTeX'а и может использоваться как для изображения встроенных в HTML-страницу математических выражений, так и для воспроизведения целых документов в окне навигатора. На рис. 5.1 показано изображение двух разделов нашего тестового документа, непосредственно полученное с использованием `techexplorer`'а (см. разд. A.1 на с. 448). В следующих разделах мы рассмотрим некоторые способы получения более живых и красочных документов.

Программа `techexplorer` претендует на большее, нежели просто отображение разметки научного текста. Ее *вводное издание* (которое можно получить на сайте [↪TXPL]), не только воспроизводит документы, но и обладает некоторыми дополнительными функциональными возможностями, например: гипертекстовыми ссылками, просмотром изображений в форматах GIF и JPEG, пользовательскими меню и иерархической навигацией по документу. *Профессиональное издание* (которое нужно приобретать у фирмы IBM) в дополнение к функциональным возможностям свободно распространяемого *вводного издания* обеспечивает поддержку вывода на печать, поиска, встроенного видео, интерфейсов для программирования на Java/JavaScript и имеет дополнительные встроенные структуры, обеспечивающие взаимодействие `techexplorer`'а с другими приложениями. В настоящее время `techexplorer` совершенствуется путем добавления поддержки языка математической разметки Mathematical Markup Language (см. разд. 8.1 на с. 422).

Программа `techexplorer` свободно оперирует с фреймами навигатора и поэтому может использоваться для создания сложных сайтов, объединяющих

[1] `techexplorer` Hypermedia Browser — это товарный знак корпорации IBM.



Simulation of Energy Loss Straggling

Maria Physicist

Introduction

Due to the statistical nature of ionisation energy loss, large fluctuations can occur in the amount of energy deposited by a particle traversing an absorber element. Continuous processes such as multiple scattering and energy loss play a relevant role in the longitudinal and lateral development of electromagnetic and hadronic showers, and in the case of sampling calorimeters the measured resolution can be significantly affected by such fluctuations in their active layers. The description of ionisation fluctuations is characterised by the significance parameter κ, which is proportional to the ratio of mean energy loss to the maximum allowed energy transfer in a single collision with an atomic electron

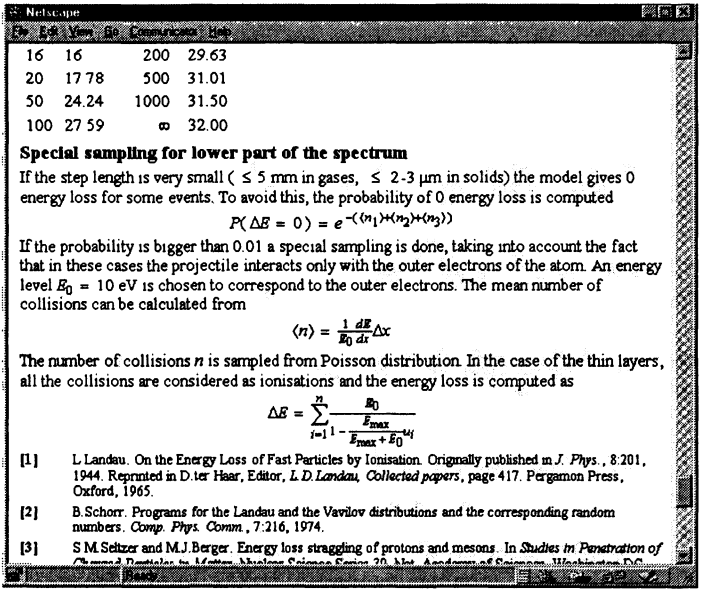
$$\kappa = \frac{\xi}{E_{max}}$$

E_{max} is the maximum transferable energy in a single collision with an atomic electron.

$$E_{max} = \frac{2m_e\beta^2\gamma^2}{1+2\gamma m_e/m_x+(m_e/m_x)^2},$$

where $\gamma = E/m_x$, E is energy and m_x the mass of the incident particle, $\beta^2 = 1 - 1/\gamma^2$ and m_e is the electron mass. ξ comes from the Rutherford scattering cross section and is defined as

$$\xi = \frac{2\pi r_e^2 c^4 N_{Av} Z\rho\delta x}{m_e\beta^2 c^2 A} = 153.4\frac{z^2 Z}{\beta^2 A}\rho\delta x \quad \text{keV},$$



16	16	200	29.63
20	17 78	500	31.01
50	24.24	1000	31.50
100	27 59	∞	32.00

Special sampling for lower part of the spectrum

If the step length is very small (≤ 5 mm in gases, ≤ 2-3 μm in solids) the model gives 0 energy loss for some events. To avoid this, the probability of 0 energy loss is computed

$$P(\Delta E = 0) = e^{-(\langle n_1\rangle+\langle n_2\rangle+\langle n_3\rangle)}$$

If the probability is bigger than 0.01 a special sampling is done, taking into account the fact that in these cases the projectile interacts only with the outer electrons of the atom. An energy level E_0 = 10 eV is chosen to correspond to the outer electrons. The mean number of collisions can be calculated from

$$\langle n\rangle = \frac{1}{E_0}\frac{dE}{dx}\Delta x$$

The number of collisions n is sampled from Poisson distribution. In the case of the thin layers, all the collisions are considered as ionisations and the energy loss is computed as

$$\Delta E = \sum_{i=1}^{n} \frac{E_0}{1-\frac{E_{max}}{E_{max}+E_0}u_i}$$

[1] L. Landau. On the Energy Loss of Fast Particles by Ionisation. Originally published in *J. Phys.*, 8:201, 1944. Reprinted in D.ter Haar, Editor, *L.D.Landau, Collected papers*, page 417. Pergamon Press, Oxford, 1965.

[2] B.Schorr. Programs for the Landau and the Vavilov distributions and the corresponding random numbers. *Comp. Phys. Comm.*, 7:216, 1974.

[3] S.M.Seltzer and M.J.Berger. Energy loss straggling of protons and mesons. In *Studies in Penetration of*

Рис. 5.1. Два примера отображения текста и математических выражений с помощью techexplorer'а.

HTML-страницы, окна с документами techexplorer'а и Java-апплеты, которые динамически обновляют документы techexplorer'а и реагируют на действия пользователя в пределах их окон.

Поскольку techexplorer является динамически подключаемым программным модулем, необходимо, чтобы его версия соответствовала используемой операционной системе. В наличии имеются версии для операционных систем Microsoft Windows 9X и Windows NT, а так же для различных вариантов системы UNIX, включая IBM AIX, SUN Solaris, SGI IRIX и Linux. Новую информацию и обновленные версии для других программно-аппаратных платформ можно найти на Web-сайте [↪TXPL]. Далее речь будет идти о techexplorer'е версии 2.0.

Полный комплект документации по techexplorer'у поставляется вместе с программным продуктом, а также доступен на Web-сайте. Мы не будем цитировать здесь приведенные в документации справочные данные, поскольку techexplorer продолжает совершенствоваться и, в частности, с каждой новой версией расширяется поддерживаемое подмножество TeX'а и LaTeX'а.

В разделе интерактивной документации, посвященном вопросам совместимости, описываются воспроизводимые techexplorer'ом функциональные возможности TeX'а и LaTeX'а. Характерные особенности techexplorer'а, которые можно использовать для создания интерактивных научных и технических документов, описываются в разделе, посвященном созданию документов. Для того чтобы облегчить создание электронных документов, отображаемых на экране с таким же качеством, как и при печати с использованием LaTeX'а, был разработан стилевой файл, который обеспечивает выполнение большинства новых команд, определенных в techexplorer'е. Этот стилевой файл и описание предоставляемых им возможностей можно найти в интерактивной документации.

В последующих разделах мы описываем основные принципы построения techexplorer'а и обсуждаем возможности того, как можно использовать расширения techexplorer'а для TeX'а и LaTeX'а, чтобы создаваемые интерактивные документы были существенно лучше, чем простое электронное воспроизведение печатных документов.

5.1.1 Основы форматирования

Программа techexplorer *эмулирует* TeX и LaTeX, но не использует саму программу TeX. Некоторые основные свойства TeX'а, например коды категорий, не поддерживаются, а основной упор делается на обеспечение стандартных LaTeX'овских макро и окружений. Стилевые файлы не поддерживаются, но входные файлы и простые plain-TeX'овские макро \def могут использоваться.

Несмотря на то что techexplorer поддерживает многие TeX'овские и LaTeX'овские команды, иногда он воспринимает команды, но затем никак на них не реагирует. Подобным же образом он может поддерживать лишь часть

функций, выполняемых тем или иным LaTeX'овским окружением. Не воспринимаемые techexplorer'ом команды и символы отображаются в тексте красным цветом![1] Имеющаяся на Web-сайте документация по techexplorer'у содержит постоянно обновляемый список поддерживаемых символов, команд и окружений.

Версия 2.0 Профессионального издания techexplorer'а поддерживает следующие команды:

```
$ $$ -- --- \! \$ \& \> \, \/ \: \; \{ \} \[ \] \( \) \^ \_ \\ \| ^ _
\acute \addtocounter \Alph \alph \arabic \arccos \arcsin \arctan \arg \atop
\author \begingroup \bf \bgroup \bibitem \big \bigl \bigm \bigr \Bigl \Big
\Bigm \Bigr \bigg \biggl \biggm \biggr \Bigg \Biggl \Biggm \Biggr \bigskip
\Bmatrix \bmatrix \bmod \bold \boxed \break \caption \cases \cdots \centering
\centerline \cfrac \chapter \choose \cite \colon \color \colorbox \cos \cosh
\cot \coth \csc \csch \date \ddot \ddots \ddotsb \ddotsc \ddotsi \ddotsm \def
\deg \det \dfrac \dim \displaylines \displaystyle \dot \egroup \em \emph
\endgroup \enskip \enspace \ensuredisplaymath \ensuremath \eqno \erf
\errmessage \exp \fbox \fcolorbox \fnsymbol \footnotesize \frac \framebox \gcd
\grave \H \hat \hbox \hfil \hfill \hfilll \hline \hom \hphantom \hrule \hsize
\hskip \hspace \hspace* \hss \Huge \huge \idotsint \iff \ifmmode \iint \iiint
\iiiint \impliedby \implies \includegraphics \index \inf \it \joinrel \kern
\ker \label \LARGE \Large \large \LaTeX \lbrace \lbrack \lcm \ldots \left
\leftline \leqno \lg \lim \liminf \limsup \llap \ln \log \lower \lowercase
\lVert \lvert \makebox \maketitle \mathbb \mathbf \mathbin \mathcal \mathchoice
\mathclose \mathit \mathop \mathop \mathopen \mathord \mathrel \mathsf
\mathstrut \mathtt \matrix \max \mbox \medbreak \medspace \medskip \min \mit
\negmedspace \negthinspace \newcommand \newcounter \newenvironment \newline
\normalsize \not \notin \null \operatorname \over \overbrace \overbracket
\overline \overrightarrow \overset \pagecolor \par \paragraph \parbox \part
\phantom \pmatrix \pmod \pod \Pr \prime \providecommand \qed \qedsymbol \qquad
\quad \quote \raggedleft \raise \raisebox \rbrace \rbrack \refstepcounter
\relax \renewcommand \renewenvironment \right \rightline \rlap \rm \root \Roman
\roman \rule \rVert \rvert \sb \sc \scriptscriptsize \scriptscriptstyle
\scriptsize \scriptstyle \shadowbox \sec \sech \section \setstyle \sf \sin
\sinh \sl \small \smallskip \smallmatrix \smash \sp \space \sqrt \stackrel
\stepcounter \strut \subparagraph \subsection \subsubsection \sup \tan \tanh
\TeX \text \textbf \textcolor \textit \textrm \textsf \textsl \textstyle
\texttt \tfrac \thanks \thebibliography \thechapter \theenumi \theenumii
\theenumiii \theenumiv \theequation \thefigure \thefootnote \thempfootnote
\thepage \theparagraph \thepart \thesection \thesubparagraph \thesubsection
\thesubsubsection \thetable \thickspace \thinspace \tilde \tiny \title \today
\tt \underbar \underbrace \underbracket \underline \underset \uppercase \value
\vbox \vdots \verb \Vmatrix \vmatrix \vphantom \vrule \vskip \vss \vtop
\widehat \widetilde \zag \zig
```

[1] В отличие от HTML-навигаторов, которые игнорируют непонятную им разметку, techexplorer отображает ее для того, чтобы облегчить авторам отладку их документов. Таким образом, techexplorer всегда полностью изображает содержимое математического выражения, вместо того чтобы тайком пропустить некоторую его часть.

Следующие команды допускаются, но игнорируются:

```
\@ \- \addcontentsline \addtocontents \addtolength \allowbreak \and
\bibliographystyle \boldmath \break \brokenpenalty \bye \cleardoublepage
\clearpage \cline \clubpenalty \DeclareMathOperator \definecolor
\displaywidowpenalty \documentclass \documentstyle \eject \end \floatingpenalty
\font \fontencoding \fontfamily \fontseries \fontshape \fontsize \footnotemark
\frenchspacing \fussy \goodbreak \hline \hyphenation \indent \interlinepenalty
\let \limits \linebreak \long \looseness \markboth \markright \multicolumn
\newblock \newif \newlength \newpage \newtheorem \noalign \nobreak \nocite
\nocorr \nofrenchspacing \noindent \nolimits \nolinebreak \nomargins \nonumber
\nopagebreak \nopagenumbers \null \pagebreak \pagenumbering \pagestyle
\postdisplaypenalty \predisplaypenalty \protect \putat \raggedbottom \relax
\rgb \selectfont \setlength \settodepth \settoheight \settowidth \singlespace
\sloppy \special \thispagestyle \typeout \unboldmath \usefont \usepackage
\widowpenalty
```

Следующие окружения поддерживаются, по крайней мере, частично:

```
align align* abstract array Bmatrix bmatrix center description displaymath document
enumerate eqnarray eqnarray* equation figure flushleft flushright gather gather*
Huge huge itemize LARGE Large large math matrix minipage normalsize pmatrix
quotation quote slide small smallmatrix tabbing table tabular tiny titlepage
verbatim Vmatrix vmatrix
```

В дальнейшем при обсуждении поддерживаемой `techexplorer`'ом разметки мы для простоты будем употреблять термин «LaTeX'овская разметка» вместо более длинного, но зато более точного выражения «подмножество разметки TeX'а и LaTeX'а».

LaTeX'овские команды `\newcommand` и `\newenvironment` можно использовать, но необязательные аргументы в них пока не допускаются. Отметим, что макроопределения являются глобальными по отношению к тому документу, в котором они определены, если только не используются команды `\gdef` или `\globalnewcommand`. В этом последнем случае макроопределения доступны для всех документов, находящихся в данное время в памяти. Еще раз подчеркнем, что `techexplorer` не допускает локальных макроопределений, действующих в тех группах, где они определены. Макроопределения всегда глобальны: либо в определяющем их документе, либо во всех активных документах. Ситуация, когда одновременно открыто несколько документов, может возникнуть, если вы используете `techexplorer` для изображения нескольких математических выражений в HTML-документе, либо когда используются фреймы навигатора и с помощью `techexplorer`'а просматриваются документы более чем в одном фрейме.

При работе на платформах Microsoft Windows 95 и Windows NT `techexplorer` использует шрифты типа TrueType. Профессиональное издание `techexplorer`'а предоставляет набор символьных шрифтов, полученных от Blue Sky и Y&Y PostScript'овских версий шрифтов Computer Modern, LaTeX'а и AMS Symbol. При работе на платформах UNIX программа `techexplorer` также

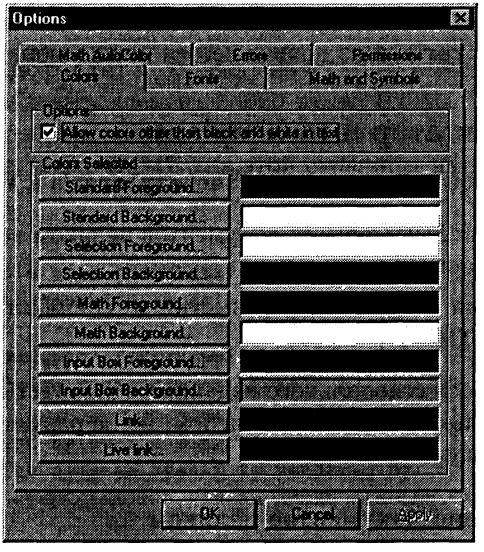


Рис. 5.2. Настройка цветов, используемых при отображении ваших документов.

использует PostScript'овские версии символьных шрифтов, но не использует METAFONT'овские шрифты.

Программа `techexplorer` не обрабатывает DVI-файлы, а непосредственно читает и воспроизводит сам исходный текст. Благодаря этому `techexplorer` может отображать LaTeX'овскую разметку, которая динамически создается Java-апплетом или другим приложением.

Внешний вид страницы зависит прежде всего от размеров окна `techexplorer`'а, а определенные в документе параметры стиля играют второстепенную роль. Например, абзацы обычно располагаются по всей ширине окна. При изменении размеров окна `techexplorer` постарается соответствующим образом переформатировать ваш документ.[1]

Так как вся работа по формированию документа выполняется внутри вашего навигатора, `techexplorer` предоставляет широкие возможности для настройки параметров (среды) отображения. Например, вы можете установить набор стандартных используемых шрифтов, а также цвета символов и фона при выводе текста. По умолчанию для ссылок используется синий цвет, но вы можете поменять его на любой другой по своему желанию. На рис. 5.2 показана стандартная вкладка свойств, используемая для установки выбранных цветов. Для того чтобы получить доступ к опциям этой вкладки, надо щелк-

[1] Отметим, что в некоторых случаях Web-навигатор вообще не сообщает `techexplorer`'у об изменении размеров окна, и поэтому документ не переформатируется. Чаще всего это происходит при использовании фреймов.

нуть правой клавишей мыши на свободном месте окна, в котором отображается документ `techexplorer`'а, и из появившегося меню выбрать пункт **Options...**.

5.1.2 Ваш навигатор и techexplorer

Возможны два способа использования `techexplorer`'а в Web-навигаторе:

- для отображения LaTeX'овских документов целиком во всей рабочей области окна навигатора и
- для отображения LaTeX'овской разметки в одном или нескольких окнах внутри одной HTML-страницы.

В версии 2.0 Профессионального издания `techexplorer`'а поддерживается вывод на печать только документов в целом, но не разметки, отображаемой в окнах HTML-страницы.

5.1.2.1 Отображение документов целиком

Если в вашем документе содержатся достаточно сложные внутритекстовые математические выражения (в противоположность выключным, т. е. расположенным между абзацами), то документ, созданный посредством `techexplorer`'а, будет выглядеть лучше, чем HTML-страница, в которой математические выражения воспроизводятся посредством рисунков, динамически подключаемых модулей или в окнах Java-апплета. Это объясняется тем, что `techexplorer` может более точно располагать и соразмерять математические выражения с окружающим текстом. Более того, стиль форматирования и фон страницы будут одинаковыми как для текста, так и для математики.

Поскольку в предварительных настройках большинства Web-навигаторов и серверов указывается, что файлы с расширением `.tex` имеют также предусмотренный стандартом MIME тип `application/x-tex`, то `techexplorer` будет автоматически вызываться для LaTeX'овских документов, если только их унифицированные указатели ресурса (URL) будут оканчиваться на `.tex`. Вы можете открывать свои LaTeX'овские документы с помощью гипертекстовых ссылок, содержащихся в HTML-документах или в других созданных с помощью `techexplorer`'а документах, а также используя стандартные способы, применяемые в навигаторах для определения унифицированных указателей ресурса (URL), которые требуется открыть.[1]

Документы `techexplorer`'а можно использовать во фреймах навигатора точно так же, как и HTML-документы. Однако если размеры окна навигатора будут изменены, то документы `techexplorer`'а, скорее всего, придется перезагрузить, чтобы скомпоновать их в соответствии с установленной шириной экрана (это недостаток навигатора, а не `techexplorer`'а).

[1] В то время, когда писался этот раздел, Netscape Navigator позволял более надежно открывать локальные файлы `techexplorer`'а из содержащихся в навигаторе списков файлов, чем Microsoft Internet Explorer. При использовании Internet Explorer'а может потребоваться явный ввод полного имени локального файла.

5.1.2.2 Отображение математических выражений на HTML-страницах

Для того чтобы techexplorer отображал на HTML-странице LaTeX'овскую разметку, можно использовать элемент EMBED языка HTML. На рис. 5.3 показана коммутативная диаграмма, вставленная внутрь некоторого HTML-текста. Исходный документ на языке HTML для этой страницы имеет вид:

```
<HTML>
<BODY>
This is an example of a commutative diagram
from an algebraic geometry article.
<CENTER>
<EMBED SRC="excomm.tex" TYPE="application/x-tex"
HEIGHT=110 WIDTH=400 NAME="comm-diagram">
</CENTER>
The text above and below the diagram are part of
the HTML page. The diagram itself is rendered by
<STRONG>techexplorer</STRONG> via an EMBED element.
</BODY>
</HTML>
```

Для techexplorer'а имеют значение шесть атрибутов элемента EMBED:

SRC — URL (унифицированный указатель ресурса) документа, содержащего вашу LaTeX'овскую разметку.

TYPE — тип данных (в соответствии со стандартом MIME), содержащихся в вашем документе.

HEIGHT — высота (в пикселах) прямоугольника, в котором должна отображаться разметка.

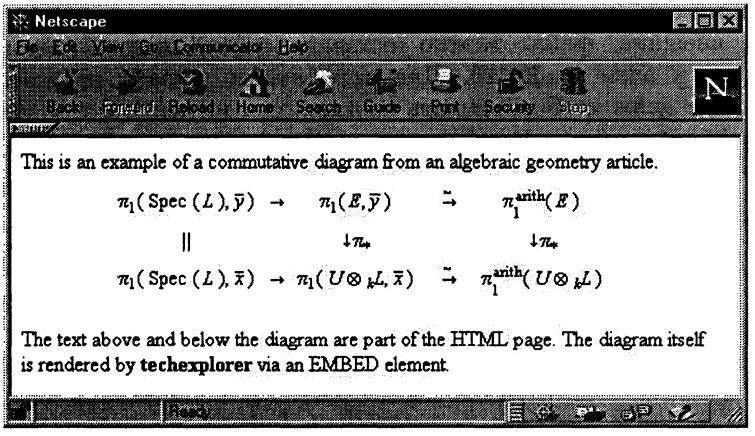


Рис. 5.3. Отображение techexplorer'ом коммутативной диаграммы, встроенной в HTML-страницу.

`WIDTH` — ширина (в пикселах) прямоугольника, в котором должна отображаться разметка.

`ALIGN` Используйте атрибут `ALIGN=MIDDLE`, если вы хотите обеспечить возможность вертикального перемещения выражения, чтобы оно более правильно располагалось относительно базовой линии окружающего текста. Для простых выражений вроде $x - 1$ этот элемент не требуется, но более сложные конструкции наподобие дробей ($\frac{2}{3}$) или матриц ($\begin{bmatrix} 3 & 2 \\ 1 & 4 \end{bmatrix}$) будут выглядеть лучше, если этот атрибут будет задан.

`NAME` — уникальное имя, которое идентифицирует отдельное вложенное окно `techexplorer`'а. Его нужно задавать, если вы используете «сценарный интерфейс» `techexplorer`'а для работы с разметкой `techexplorer`'а и событиями (действиями пользователя), которые происходят внутри этого окна. Это имя можно опустить, если вы заинтересованы только в отображении выражений.

Поскольку у рассматриваемого в этом примере вложенного LaTeX'овского документа файл имеет расширение `.tex`, то атрибут `TYPE`, вероятно, является излишним. Тем не менее мы рекомендуем его использовать.

Если выбранные значения высоты или ширины окажутся слишком маленькими, то окно `techexplorer`'а будет содержать вертикальную или горизонтальную полосу прокрутки соответственно. Поскольку с помощью окна настройки шрифтов пользователи могут сами выбирать размер шрифта, невозможно заранее быть уверенным в том, что выбранный вами размер окна будет достаточен для всех. Кроме того, `techexplorer` окружает отображаемую разметку узкими полями, поэтому необходимый размер окна может оказаться несколько больше ожидаемого. Чтобы выражение заполнило всю ширину HTML-страницы, можно попытаться применить `WIDTH="100%"`.

При использовании навигаторов, которые совместимы с Netscape Navigator по динамически подключаемым модулям, вместо того, чтобы указывать на внешний файл, можно вставлять LaTeX'овские данные непосредственно в элемент `EMBED`. Для того чтобы указать навигатору, что `techexplorer` должен обрабатывать данные, одновременно с указанием типа `TYPE="application/x-techexplorer"` используйте атрибут `TEXDATA`, задавая разметку в качестве его значения. Например,

```
<EMBED TYPE="application/x-techexplorer"
TEXDATA="\[\pmatrix{2&3&4&5\cr 6&7&8&9\cr -1&-2&-3&-4}\]"
WIDTH=200 HEIGHT=90 ALIGN=MIDDLE>
```

изображает матрицу размера 3×4

$$\begin{pmatrix} 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 \\ -1 & -2 & -3 & -4 \end{pmatrix}$$

в месте расположения элемента `EMBED`.

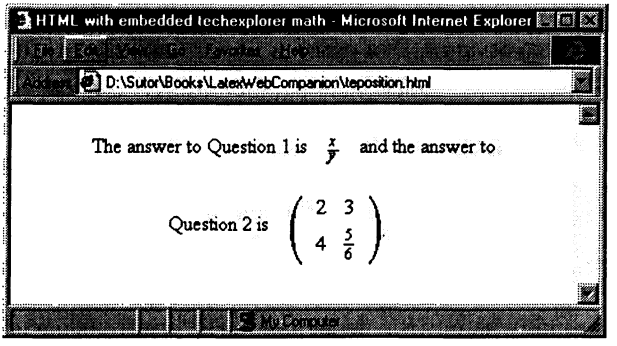


Рис. 5.4. Отображение `techexplorer`'ом двух выражений, встроенных в HTML-страницу.

Ниже приведен другой пример (изображенный на рис. 5.4 с помощью Microsoft Internet Explorer), в котором представлены две математические формулы. Первая из них — внутритекстовая, а другая — выключная.

```
<HTML>
<HEAD>
<TITLE>HTML with embedded techexplorer math</TITLE>
</HEAD>
<BODY>
<P ALIGN=CENTER> The answer to Question 1 is
<EMBED TYPE="application/x-techexplorer"
       TEXDATA="\pagecolor{white}$\frac{x}{y}$"
       ALIGN=MIDDLE
       WIDTH=33 HEIGHT=45>
and the answer to<BR>Question 2 is
<EMBED TYPE="application/x-techexplorer"
       TEXDATA="\pagecolor{white}$\pmatrix{2&3\cr 4&\frac{5}{6}}$."
       ALIGN=MIDDLE
       WIDTH=114 HEIGHT=78>
</P>
</BODY>
</HTML>
```

5.1.3 Добавление гипертекстовых ссылок

Одно из важнейших преимуществ электронных документов перед бумажными заключается в возможности использования гипертекстовых ссылок. Быстрое распространение Web-сайтов и навигаторов, использующих язык HTML, превратило возможность работы с гипертекстом в основное требование к любому приложению, интерактивно отображающему документы.

При работе в среде операционной системы Microsoft Windows 95 максимальная длина документа, с которым может работать `techexplorer`, соответствует примерно 50 печатным страницам. По этой причине более длинные до-

кументы надо будет разбивать на меньшие по размеру и устанавливать между ними гипертекстовые ссылки. Эти ссылки могут располагаться либо непосредственно в тексте, либо во всплывающих меню (см. разд. 5.1.6 на с. 250).

Универсальной командой `techexplorer`'а для создания гипертекстовых ссылок является команда `\docLink`.

`\docLink[`*frameName*`]{`*url*`}[`*label*`]{`*expression*`}`

Двумя обязательными аргументами являются *url* и *expression*. Содержимое аргумента *expression* — это то, что вы увидите на экране, а *url* — это адрес того документа, к которому навигатор перейдет, когда читатель щелкнет мышкой на заданном аргументом *expression* (выражении). Чтение адреса *url* выполняется в специальном режиме, поэтому в нем могут содержаться символы наподобие обратной косой черты.

Обратите внимание, что содержимое аргумента *expression* не обязано быть простым текстом: это может быть все, что `techexplorer` в состоянии отобразить, например текст, рисунки и математические выражения. Как уже упоминалось, по умолчанию *expression* будет воспроизводиться синим цветом. Если вы хотите, чтобы отдельная гиперссылка воспроизводилась заданным цветом, используйте внутри аргумента *expression* команду `\color`. Например,

```
Эта \docLink{Hyperlink.tex}{\color{black}гиперссылка}
может быть неотличима от окружающего текста.
```

Если цвет окружающего текста будет черным, то слово «гиперссылка» в приведенном здесь предложении не будет выглядеть как-то по-особому. Однако курсор мыши по-прежнему будет менять свою форму на кисть руки, а в строке состояния будет отображаться адресат ссылки, т. е. `Hyperlink.tex`.

В отличие от применяемого в навигаторах стандартного форматирования, в `techexplorer`'е гиперссылки не подчеркиваются; подчеркивать ссылки внутри математических выражений не имеет смысла. Действительно, в математических выражениях подчеркивание часто имеет семантическое значение.

Адрес *url* передается в Web-навигатор, даже если соответствующий документ представляет собой другой LaTeX-овский файл, который должен быть обработан `techexplorer`'ом. Для такого LaTeX'овского файла используйте необязательный аргумент *label*, чтобы задать то место в документе, до которого `techexplorer` должен пролистать («прокрутить») документ при его отображении. При невозможности найти метку *label* в окне будет отображаться начало документа.

Примеры

- Для адресов Web-сайтов используйте стандартный синтаксис URL.

```
Много полезной информации о \LaTeX{}'е содержится на Web-сайте
\docLink{http://www.tug.org}{Группы пользователей \TeX{}'а}
```

- Вы можете делать ссылки на локальные имена файлов, однако следует помнить, что они могут различаться в зависимости от ОС, а также, что они совсем не обязательно будут одинаковыми на компьютерах всех пользователей.

```
В \docLink{c:\classes\m101\probset1.tex}{постановке первой
задачи} содержатся инструкции по представлению вашей домашней
работы.
```

- Для ссылок на файлы, расположенные в том же каталоге, что и текущий файл, можно применять относительную адресацию. Здесь мы переходим к метке *lecture2*, находящейся в исходном LaTeX'овском файле `reading.tex`. Он находится в том же самом каталоге, что и просматриваемый в это время документ.

```
Для выполнения задания по чтению ко \docLink{reading.tex}
[lecture2]{второй лекции} потребуется около 30 часов.
```

- В действительности ссылки необязательно должны указывать на сами файлы. В следующем примере содержится ссылка `mailto`, поэтому когда пользователь щелкнет клавишей мыши на этой ссылке, будет запущена почтовая программа читателя.

```
При желании вы можете послать сообщение
\docLink{mailto:techexpl@watson.ibm.com}{разработчикам}
techexplorer'а.
```

LaTeX'овские документы, отображаемые навигатором с помощью `techexplorer`'а, регистрируются в общем журнале навигатора, и, в частности, доступ к ним можно получить с помощью навигационных кнопок «назад» и «вперед». Тогда как навигатор сохраняет информацию о том, до какого места пролистывались отображаемые им HTML-документы, `techexplorer` в настоящее время этого не делает. Частично эта проблема состоит в том, что при переходе от одного LaTeX'овского документа к другому `techexplorer` обычно полностью выгружается (из памяти), а затем снова загружается. Поэтому он не может сохранить в памяти текущее состояние просмотра. Этот недостаток может быть исправлен в последующих версиях.

Если задан параметр *frameName* и при этом текущий документ `techexplorer`'а включен в набор фреймов HTML, то новый документ отображается во фрейме с именем *frameName*. Если параметр *frameName* пропущен, то новый документ отображается в текущем фрейме.

В следующем примере показаны два фрейма, в каждом из которых содержится окно `techexplorer`'а. В левом фрейме с именем «toc» содержится оглавление, а в правом, имя которого «body», — разделы. Рисунок 5.5 — это

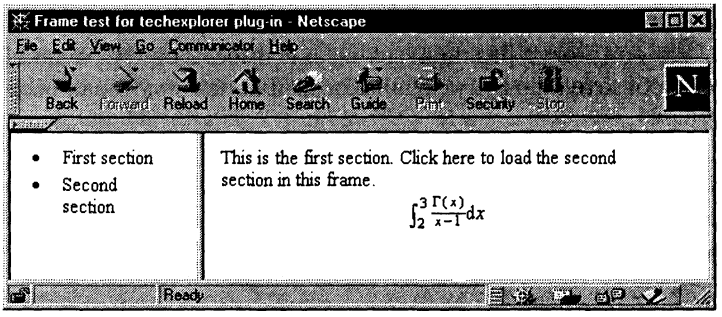


Рис. 5.5. Два фрейма, содержащих окна `techexplorer`'а.

копия экрана с оглавлением и первым разделом. Определение набора фреймов HTML приведено ниже:

```
<HTML>
  <HEAD>
    <TITLE>
      Frame test for techexplorer plug-in
    </TITLE>
  </HEAD>
  <FRAMESET COLS="150,*">
    <FRAME MARGINWIDTH="4" SRC="toc.tex" NAME="toc">
    <FRAME MARGINWIDTH="4" SRC="first.tex" NAME="body">
  </FRAMESET>
</HTML>
```

Команды \docLink, содержащиеся в LaTeX'овском файле с оглавлением, указывают, что файл с соответствующим разделом должен отображаться в расположенном справа фрейме «body».

```
% toc.tex
\begin{itemize}
\item \docLink[body]{first.tex}{First section}
\item \docLink[body]{second.tex}{Second section}
\end{itemize}
```

В первом разделе содержится ссылка на второй. В данном случае имя фрейма необязательно, потому что мы хотим открывать все разделы в одном и том же фрейме.

```
% first.tex
This is the first section. Click here to load the
\docLink{second.tex}{second section} in this frame.

\[
\int_2^3 \, \frac{\Gamma(x)}{x - 1} \mathrm{d}x
\]
```

Точно так же мы не включаем имя фрейма в следующую ссылку на первый раздел.

```
% second.tex
This is the second section. Click here to load the
\docLink{first.tex}{first section} in this frame.

$$
\bmatrix{
1 & 0 & 0 \cr
0 & 1 & 0 \cr
0 & 0 & 1
}
$$
```

`\labelLink{`*label*`}{`*text*`}`

Если место, на которое указывает гипертекстовая ссылка, находится где-либо в том же самом документе, то вместо команды `\docLink` используйте `\labelLink`.

Команда `\docLink` тоже будет работать, но она излишне мощна для этой ситуации. Вам понадобится определять URL документа, а кроме того, если вы решите переименовать свои файлы, могут возникнуть проблемы с сопровождением документа.

В следующем примере создается локальная гиперссылка на последний раздел текущего документа:

```
В \labelLink{c5:final-section}{заключительном разделе} мы анализируем
эти проблемы и обсуждаем вопрос о том, как перспективные навигаторы и
программы визуализации смогут улучшить сложившуюся ситуацию.
```

Как в команде `\docLink`, так и в `\labelLink` аргумент *text* может содержать другие ссылки. Правило разрешения ссылок состоит в том, что приоритет имеет самая глубокая (внутренняя) ссылка. Рассмотрим следующий фрагмент документа:

```
На все \labelLink{label:red-phones}{красные
\labelLink{label:phones}{телефоны}}
дается гарантия возврата денег в течение 30 дней.
```

Когда указатель мыши находится на слове «телефоны», при нажатии на клавишу мыши будет выполняться ссылка `label:phones`. А когда указатель будет на слове «красные», сработает ссылка `label:red-phones`.

5.1.4 Всплывающие окна и сноски

Когда документ воспроизводится на экране с помощью **techexplorer**'а, он не разбивается на отдельные страницы. И бывает очень неудобно пролистывать длинный документ до самого конца, чтобы посмотреть сноски (примеча-

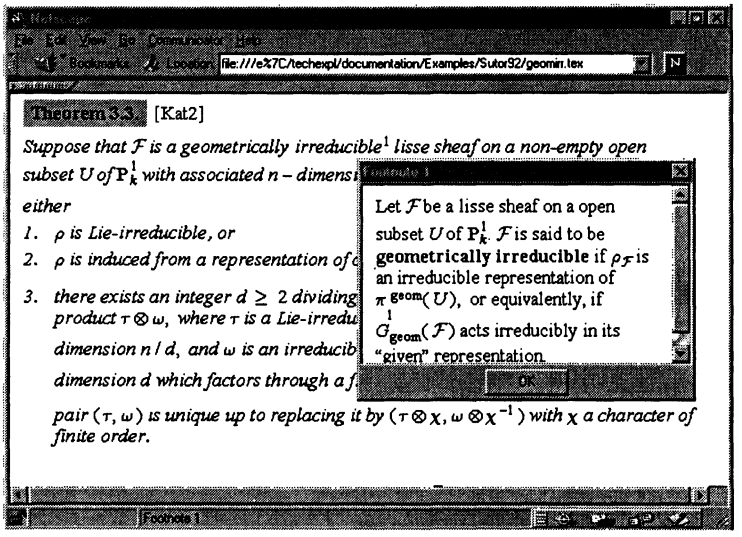


Рис. 5.6. Всплывающая сноска в `techexplorer`'е.

ния). Более того, от сноски, вставленной в содержащуюся на HTML-странице математическую формулу, нельзя перейти в «конец документа». Поэтому `techexplorer` поддерживает команду `\footnote`, создавая на экране новое окно при щелчке (клавишей мыши) на номере сноски в тексте. Сноска в документе `techexplorer`'а показана на рис. 5.6.

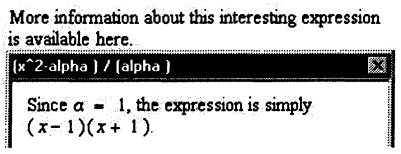
`\popupLink{`*windowText*`}{`*caption*`}{`*text*`}`

Сноска является примером *ссылки на временное всплывающее окно.* Такие ссылки создаются с помощью команды `\popupLink`.

Содержимое аргумента *text* отображается в самом документе. При щелчке в этом месте документа возникает временное всплывающее окно, содержимое которого определяется параметром *windowText.* Заголовок этого окна, задаваемый параметром *caption*, отображается в строке заголовка. Когда курсор мыши движется по *text*, в строке состояния навигатора отображается *caption.*

Обратите внимание, что *caption* (при необходимости) будет упрощен так, чтобы его можно было воспроизвести одним шрифтом, применяемым для строки заголовка. В частности, содержащиеся в нем математические выражения не будут воспроизводиться в их естественном виде. В следующем примере

```
More information about this interesting expression is available
\popupLink{Since $\alpha = 1$, the expression is simply
$(x-1)(x+1)$.}{$\frac{x^2-\alpha}{\alpha}$}{here}.
```



в строке заголовка и в строке состояния будет воспроизводиться текст `(x^2-alpha)/(alpha)`.

Текст во всплывающем окне полностью форматируется и может содержать математические выражения. Ссылки внутри окна пока не функционируют, хотя они и изображаются цветом, используемым для ссылок.Если вы вставляете в расположенный во временном окне текст рисунок, звукозапись или видеоизображение, то убедитесь, что вы используете абсолютный адрес URL местоположения оригинала.

5.1.5 Использование рисунков, звукозаписи и видеоизображений

Мультимедийные возможности `techexplorer`'а реализуются представлением рисунков, звукозаписей и видеоизображений. Наиболее полная поддержка обеспечивается в версиях для Microsoft Windows, тогда как редакции для UNIX поддерживают только рисунки. Поскольку мультимедийные файлы могут иметь большой объем, то в тех случаях, когда они получаются по сети, перед выводом изображения или воспроизведением аудио- или видео-файла может возникнуть пауза.

`\includegraphics[`*lowerLeft*`][`*lowerRight*`]{`*url*`}`

Для подключения рисунка используйте команду `\includegraphics`. Она позволяет использовать важнейшие средства LaTeX'овского пакета `graphics`.

Необязательные аргументы *lowerLeft* и *lowerRight* хотя и допускаются, но игнорируются. Если рисунок, на который указывает *url*, не существует или еще не получен из сети, на его месте возникает заставка «отсутствующий рисунок» («missing image»).

Вы можете использовать изображения типов GIF и JPEG на платформе любой операционной системы, поддерживаемой `techexplorer`'ом. Для рисунков типа GIF файл в адресе *url* должен иметь расширение `.gif`. Для файлов формата JPEG файлы с рисунками могут иметь расширения `.jpg`, `.jpeg` или `.jpe`. Версия для Microsoft Windows также поддерживает:

- растровые файлы Windows с расширениями имени файла `.bmp` и `.dib`;
- рисунки формата PCX с расширением имени файла `.pcx`;
- рисунки формата Targa с расширением имени файла `.tga` и
- рисунки формата TIFF с расширениями имени файла `.tif` или `.tiff`.

Для достижения максимальной мобильности документов мы рекомендуем использовать изображения форматов GIF и JPEG. Анимированные изображе-

ния и слайды пока еще не поддерживаются ни на одной платформе; не поддерживаются также инкапсулированные PostScript'овские файлы. При печати документа `techexplorer`'а каждый рисунок масштабируется таким образом, что его размер остается неизменным по отношению к тексту.

`\backgroundimage{`*url*`}`

Используйте команду `\backgroundimage` для заполнения фона окна `techexplorer`'а каким-либо рисунком. Для этого можно применить любой рисунок, допустимый в команде `\includegraphics`. При выводе документа `techexplorer`'а на печать фоновый рисунок игнорируется. Если вы включите в свой документ несколько команд `\backgroundimage`, то использоваться будет последняя из них. Если рисунок, на который указывает адрес *url*, не существует или не может быть получен по сети, то фоновый рисунок вообще не изображается.

`\backgroundsound{`*url*`}`
`\includeaudio{`*url*`}`

Если ваш компьютер оснащен соответствующей звуковой картой, то с помощью команд `\backgroundsound` и `\includeaudio` можно воспроизводить звуковые файлы. Звуковые файлы проигрываются только один раз; возможно, будущие версии будут допускать циклическое воспроизведение. Поддерживаются только файлы формата WAV с расширением `.wav` и к тому же только на платформах Microsoft Windows. Эта функция поддерживается только в Профессиональном издании `techexplorer`'а.

По умолчанию звуковой файл проигрывается сразу же после того, как он может быть загружен. На рис. 5.7 показана стандартная вкладка свойств для задания разрешенных действий `techexplorer`'а. Если вы не хотите, чтобы звукозапись проигрывалась в фоновом режиме при загрузке страниц, снимите флажок в окошке с меткой «Enable audio to autoplay when the document is opened».

`\audioLink{`*url*`}{`*text*`}`

Чтобы звукозапись воспроизводилась, когда читатель щелкнет клавишей мыши в пределах некоторой области экрана, используйте *аудиоссылку*. По адресу *url* будет найден звуковой файл, а содержимое параметра *text* будет отображено на экране. Как и при использовании команды `\backgroundsound`, поддерживаются только файлы формата WAV с расширением `.wav` и только в операционной системе Microsoft Windows. На других платформах при разрешении этой ссылки никакие звуки не воспроизводятся.

`\videoLink{`*url*`}{`*text*`}`

Используйте *видеоссылку*, для того чтобы воспроизвести видеозапись по щелчку клавишей мыши. Соответствующий видеофайл извлекается по адресу *url*,

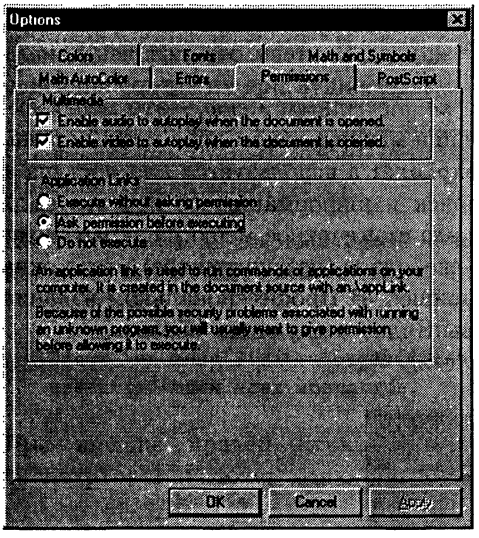


Рис. 5.7. Задание допустимых действий `techexplorer`'а.

Рис. 5.8. Встроенное видео в Профессиональном издании `techexplorer`'а.

а на экране отображается содержимое параметра *text*. Поддерживаются только файлы формата AVI с расширением `.avi` и только на платформе Microsoft Windows. После запуска видеозапись воспроизводится до конца и временное окно закрывается.

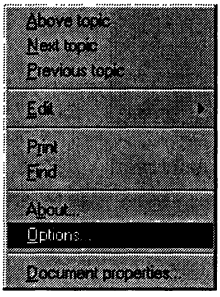
Обладатели Профессионального издания `techexplorer`'а могут обратиться к более интересной видеоопции, позволяющей использовать внутритекстовую (встроенную) видеозапись.

$$\boxed{\texttt{\textbackslash includevideo}\{autoPlay\}\{alternateText\}\{url\}}$$

Из указанного адреса *url* извлекается видеофайл. Если его невозможно найти, или он еще не получен, то на экране отображается альтернативный текст, заданный параметром *alternateText* (альтернативный текст всегда воспроизводится при выводе документа на печать). Если параметр *autoPlay* имеет значение «`t`», то воспроизведение видеозаписи начнется сразу же после ее загрузки. Вы можете самостоятельно отменить режим автоматического воспроизведения видеозаписи, сняв флажок в окошке с меткой «Enable video to autoplay when the document is opened» на вкладке свойств для задания опций доступа (см. рис. 5.7). Для приостановки, остановки или возобновления воспроизведения видеозаписи щелкните правой клавишей мыши на изображении и выберите нужный пункт из меню. На рис. 5.8 показан пример встроенного видео вместе с меню для управления воспроизведением.

5.1.6 Определение и использование всплывающих меню

Всплывающее меню, называемое также *контекстным меню*, — это меню, активизируемое при нажатии правой клавиши мыши в каком-либо месте экрана. Стандартное всплывающее меню `techexplorer`'а используется для доступа к опциям настройки, включая печать, поиск и информацию об используемой версии `techexplorer`'а. Чтобы вывести это стандартное меню на экран, нужно поместить указатель мыши в любое свободное (незаполненное) место окна `techexplorer`'а и щелкнуть правой клавишей. В Профессиональном издании `techexplorer`'а это стандартное меню выглядит следующим образом:



Вы не можете вносить изменения в это меню, но в зависимости от того, какие определения были сделаны в вашем документе для облегчения перемещения по его иерархической структуре, отдельные пункты первого раздела («topic») могут быть либо активными, либо пассивными (см. разд. 5.1.8 на с. 256).

Вы можете определять новые меню, которые будут активизироваться при каждом щелчке правой клавишей мыши в заданной области текста. Что может быть элементами таких меню? Ссылки, линейки и другие меню.

Ссылка в меню определяет текст пункта меню и действие, которое совершается при выборе этого пункта. В определениях меню чаще всего используются гипертекстовые ссылки и ссылки, обеспечивающие воспроизведение аудио- или видеозаписи.

Вот пример описания меню из трех простых пунктов, предназначенного для воспроизведения песен:

```
\newmenu{SinatraSongs}{
\audioLink{http://www.sinatrafan.com/songs/young.wav}
   {You make me feel so young}
\audioLink{http://www.sinatrafan.com/songs/pennies.wav}
   {Pennies from heaven}
\audioLink{http://www.sinatrafan.com/songs/anything.wav}
   {Anything goes}
}
```

Как мы видим, команда \newmenu используется для того, чтобы задать описание именованного меню. В данном случае именем является *SinatraSongs*, а описание содержит три аудиоссылки \audioLink.

\newmenu{*menuName*}{*menuDefinition*}
\usemenu{*menuName*}{*text*}

Команда \usemenu связывает меню с некоторым текстом. Одно и то же описание меню может быть использовано для любого количества команд \usemenu.

Например,

```
Песни \usemenu{SinatraSongs}{Фрэнка Синатры}
нравятся людям всех возрастов.
```

При щелчке правой клавишей мыши на словах «Фрэнка Синатры» techexplorer покажет меню

Если щелкнуть на любом из пунктов меню, то будет исполнена соответствующая песня![1]

Для задания разделительной линии между элементами используйте линейку в описании меню. Поскольку линейка в действительности не рисуется, а просто используется как указатель положения разделительной линии, можно использовать любую из команд: \hrule, \vrule или \rule.

Вот описание, которое содержит как песни, так и фильмы Фрэнка Синатры.

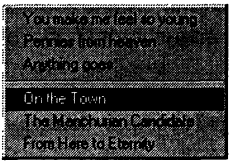
```
\newmenu{SinatraSongsAndMovies}{
\audioLink{http://www.sinatrafan.com/songs/young.wav}
   {You make me feel so young}
\audioLink{http://www.sinatrafan.com/songs/pennies.wav}
   {Pennies from heaven}
\audioLink{http://www.sinatrafan.com/songs/anything.wav}
   {Anything goes}
\hrule % provides a separator line
\videoLink{http://www.sinatrafan.com/movies/town.avi}
   {On the Town}
\videoLink{http://www.sinatrafan.com/movies/manchurian.avi}
   {The Manchurian Candidate}
\videoLink{http://www.sinatrafan.com/movies/eternity.avi}
   {From Here to Eternity}
}
```

[1] Если, конечно, существует такой Web-сайт, а на нем есть записи этих песен!

Это меню используется точно так же, как и предыдущее

```
Песни и фильмы \usemenu{SinatraSongsAndMovies}{Фрэнка Синатры}
очень популярны.
```

Теперь при щелчке правой клавишей мыши на словах «Фрэнк Синатра» появится меню



При щелчке на любом из пунктов меню будет воспроизведена соответствующая песня или (фрагмент) фильм с нашего гипотетического Web-сайта.

Если поместить команду \usemenu внутри описания \newmenu, то будет создано субменю. Переработаем последнее меню так, чтобы песни и фильмы перечислялись раздельно в своих собственных субменю. Первоначальное описание меню *SinatraSongs* оставим неизменным

```
\newmenu{SinatraSongs}{
\audioLink{http://www.sinatrafan.com/songs/young.wav}
   {You make me feel so young}
\audioLink{http://www.sinatrafan.com/songs/pennies.wav}
   {Pennies from heaven}
\audioLink{http://www.sinatrafan.com/songs/anything.wav}
   {Anything goes}
}
```

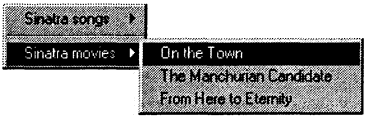
и добавим аналогичное описание для меню *SinatraMovies*:

```
\newmenu{SinatraMovies}{
\videoLink{http://www.sinatrafan.com/movies/town.avi}
   {On the Town}
\videoLink{http://www.sinatrafan.com/movies/manchurian.avi}
   {The Manchurian Candidate}
\videoLink{http://www.sinatrafan.com/movies/eternity.avi}
   {From Here to Eternity}
}
```

В заключение определим меню *SinatraSongsAndMovies2*, содержащее два субменю, и добавим разделитель:

```
\newmenu{SinatraSongsAndMovies2}{
\usemenu{SinatraSongs}{Sinatra songs}
\hrule % дает разделительную линейку
\usemenu{SinatraMovies}{Sinatra movies}
}
```

Окончательное, полностью открытое меню может выглядеть как одно из следующих:



В описании меню можно использовать почти все ссылки, кроме `\altLink`. Если вы определяете ссылку командой, в число аргументов которой входит адрес *url*, но оставляете этот аргумент пустым (например, `\labelLink{}{Section 5}`), то соответствующий пункт меню будет пассивным и будет изображен серым цветом.

5.1.7 Использование цвета в ваших документах

`techexplorer` поддерживает следующие команды из LaTeX'овского пакета `color`: `\color`, `\textcolor`, `\colorbox`, `\fcolorbox` и `\pagecolor`. Могут быть использованы основные цвета из пакета `color` и еще несколько других цветов, подходящих для воспроизведения на экране. В приведенном ниже списке поддерживаемых цветов полужирным шрифтом набраны названия новых цветов, поддержка которых добавлена в `techexplorer`'е: **aqua** (цвет морской волны), black (черный), blue (синий), cyan (голубой), **darkgray** (темно-серый), **fuchsia** (цвет фуксии), **gray** (серый), green (зеленый), **lightgray** (светло-серый), **lime** (лайм), magenta (пурпурный), **maroon** (красно-коричневый), **navy** (темно-синий), **olive** (желто-зеленый), **purple** (фиолетовый), red (красный), **silver** (серебристый), **teal** (цвет чирка), white (белый) и yellow (желтый).

Для того чтобы изменить цвет линеек, поместите команду `\color` перед определением линейки.

```
\bgroup
\color{red}%
\hrule height 3pt % эта линейка будет красной
\egroup
\hrule height 3pt % а эта, вероятно, будет черной
```

Мы вставляем команды изменения цвета и задания линейки внутрь пары ограничителей `\bgroup`/`\egroup`, для того чтобы сохранить стандартную установку цвета.

`\backgroundcolor{`*color*`}`

Команду `\backgroundcolor` можно использовать как синоним команды `\pagecolor`. Обе эти команды устанавливают цвет фона, на котором изображается текст на экране. Если вы хотите, чтобы фон HTML-окна соответствовал стандартному «серому цвету навигатора», мы рекомендуем использовать

Рис. 5.9. Заголовок раздела в рамке с градиентной заливкой.

`\pagecolor{silver}`. Ограничение, присущее современному Профессиональному изданию `techexplorer`'а, проявляется в том, что при печати документа цвет фона игнорируется.

Обратите внимание: команды, устанавливающие цвет фона, действуют только на то окно, в котором отображается ваш документ. Если вы хотите поменять цвет фона во всех окнах `techexplorer`'а, то

1. щелкните правой клавишей мыши на свободном месте любого окна, содержащего документ `techexplorer`'а;
2. выберите из появившегося меню раздел **Options...**;
3. щелкните на закладке **Colors**;
4. нажмите кнопку **Standard background...**;
5. выберите цвет, для чего сначала щелкните на нем, а потом нажмите кнопку **OK**.

Чтобы проверить свой выбор, нажмите кнопку **Apply**. Если вам не понравится то, что вы увидите, вернитесь к шагу 3 и повторите дальнейшие действия заново. Для сохранения своего выбора нажмите кнопку **OK** или кнопку **Cancel**, чтобы вернуться к предыдущему цвету фона. Стандартная вкладка свойств для установки цвета показана на рис. 5.2.

`\rgb{`*redValue*`}{`*greenValue*`}{`*blueValue*`}`

Команда `\definecolor` в настоящее время не поддерживается, но для определения новых цветов вы можете использовать команду `\rgb`. Значениями аргументов *redValue* (*уровень красного*), *greenValue* (*уровень зеленого*) и *blueValue* (*уровень синего*) могут быть целые числа от 0 до 255 включительно. Например[1]

```
\def\paleYellow{\rgb{255}{255}{128}}
\pagecolor{\paleYellow}
```

устанавливает светло-желтый цвет фона для окна документа `techexplorer`'а.

Стандартные команды `\colorbox` и `\fcolorbox` позволяют размещать текст в рамках с заданным цветом фона. В `techexplorer`'е имеется команда `\gradientbox` (см. рис. 5.9), которая позволяет создавать рамку (прямоугольник), в которой цвет фона плавно меняется от одного до другого.

[1] `techexplorer` не в полной мере поддерживает команду `\newcommand`, поэтому мы используем TeX'овский примитив `\def`.

`\gradientbox[`*v*`]{`*startColor*`}{`*endColor*`}{`*text*`}`

По умолчанию изменение цвета происходит в направлении слева направо, начиная с цвета, заданного аргументом *startColor (начальный цвет)*, и кончая цветом, указанным в аргументе *endColor (конечный цвет)*. Если задан необязательный аргумент *v*,[1] то цвет будет изменяться в вертикальном направлении сверху вниз.[2]

`\buttonbox[`*i*`]{`*text*`}`
`\colorbuttonbox[`*i*`]{`*color*`}{`*text*`}`

Для создания рамок, которые выглядят как стандартные кнопки Microsoft Windows с серым цветом фона, применяйте команду `\buttonbox`. Если задан необязательный аргумент *i* (опять буквально), эти кнопки будут изображены в «вывернутом» состоянии. Такая кнопка выглядит так, как будто она была нажата.

```
\buttonbox{Это кнопка}
\buttonbox[i]{А это нажатая кнопка}
```

Если вы хотите, чтобы цвет фона у кнопки отличался от серого, используйте команду `\colorbuttonbox`. Если задан необязательный аргумент *i*, то кнопка будет изображаться как нажатая.

```
\colorbuttonbox{green}{Это зеленая кнопка}
\colorbuttonbox[i]{green}{А это нажатая зеленая кнопка}
```

Будьте осторожны, используя цвет в своих документах. Слишком большое разнообразие цветов может сбивать с толку так же, как и злоупотребление шрифтами из разных семейств. Как и должно быть при правильном подходе к разработке LaTeX'овского документа, создавайте макро или стилевые файлы, сосредоточивая в них свои «конструкторские решения». Впоследствии вы можете модифицировать эти макро или стилевые файлы, чтобы внести глобальные изменения в оформление документа. Ниже приведен пример макро «Section», в котором применяются градиентная заливка и изменение цвета текста:[3]

```
\def\Section#1{\section{%
  \gradientbox{blue}{white}{\mbox{\color{white}#1}}}}
...
\Section{Introduction}
```

[1] Необходимо буквально писать *v*, как, например, в `\gradientbox[v]{red}{white}`.

[2] В современных версиях `techexplorer`'а для системы UNIX градиентная заливка цветом не реализована; поэтому команда `\gradientbox` оказывает такое же действие, что и команда `\colorbox`, заполняя область цветом, указанным в аргументе *startColor*.

[3] Не забывайте, что `techexplorer` не в полной мере поддерживает команду `\newcommand`.

Обратите внимание на то, что в techexplorer'е не надо оставлять пробелы после команды \color. Обычный LaTeX может игнорировать пробелы, но techexplorer — нет.

5.1.8 Создание иерархической структуры документа

Большинство традиционных документов имеет древовидную иерархическую структуру в виде глав и разделов. Для удобства редактирования мы часто делим их на несколько файлов, а при печати эти части некоторым способом вставляются в главный документ. Электронные версии документов также часто разбиваются на ряд файлов в соответствии с их иерархией, но эти файлы обычно просматриваются по отдельности, а соединяются гиперссылками.

Для примера рассмотрим представленную на рис. 5.10 структуру, в которой каждый узел показывает название части документа и имя файла, содержащего разметку этой части.

Оглавление является корнем этого дерева, и из него исходят все расположенные на верхнем уровне части книги. Некоторые части, например глава 1, сами состоят из нескольких разделов.

Навигатор techexplorer предоставляет стандартный способ соединения фрагментов иерархической структуры с помощью гиперссылок. Используйте команду \aboveTopic для указания адреса URL документа, который является родительским по отношению к данной части.[1]

\aboveTopic{*url*}
\nextTopic{*url*}
\previousTopic{*url*}

В примере на рис. 5.10 файл preface.tex должен содержать команду \aboveTopic{toc.tex}.

Используйте команду \nextTopic для указания адреса URL следующего документа, который расположен на одном уровне с данным и имеет с ним общего родителя. Таким образом, файл preface.tex должен содержать команду \nextTopic{1.tex}. И наконец, используйте команду \previousTopic для указания адреса URL предыдущего документа, который расположен на одном уровне с данным и имеет с ним общего родителя.

Файл 1.tex должен содержать команду \previousTopic{preface.tex}.

В корневой части документа не будет команды \aboveTopic. В первом разделе данного уровня не будет команды \previousTopic, а в последнем разделе того же уровня не будет команды \nextTopic. Хотя это и не очень увлекательное занятие — всегда определять предыдущий и последующий в некотором отношении разделы, но ваши читатели, вероятно, могут оказаться в более затруднительном положении, если вы не представите какую-либо схему документа.

[1] *Родительский документ* — это ближайший документ в дереве, находящийся на прямом пути к корню дерева. У корневого элемента дерева родительского документа нет.

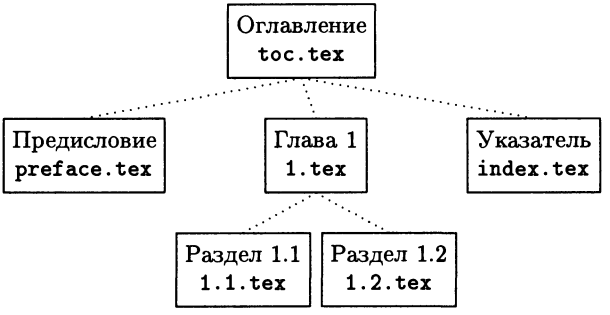
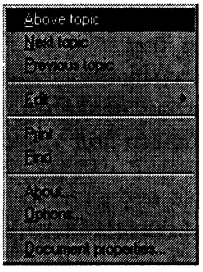


Рис. 5.10. Упрощенная структура книги.

Доступ к этим гиперссылкам можно получить, если щелкнуть правой клавишей мыши в пределах окна `techexplorer`'а. Если вы сделаете это за пределами определенного пользователем всплывающего меню (такие меню описаны в разд. 5.1.6 на с. 250), то увидите меню, похожее на следующее:



В показанном меню доступны все три раздела. Если бы один из разделов — выше (above), следующий (next) или предыдущий (previous) — не был определен в документе, то соответствующий пункт меню был бы «серым», т. е. недоступным.

Итог этому обсуждению подводит рис. 5.11, на котором изображена более подробная схема дерева документа, упоминавшегося в начале данного раздела. В нее включены команды, позволяющие связать воедино ветви иерархического дерева.

5.1.9 Выполнение приложений

`\appLink{`*command*`}{`*text*`}`

Вы можете использовать `techexplorer` в качестве основанного на навигаторе интерфейса для выполнения команд на своем компьютере. *Ссылка на приложение* выполняет заданную команду при щелчке клавишей мыши в заданной области отображаемого на экране текста. Содержимое аргумента *command*

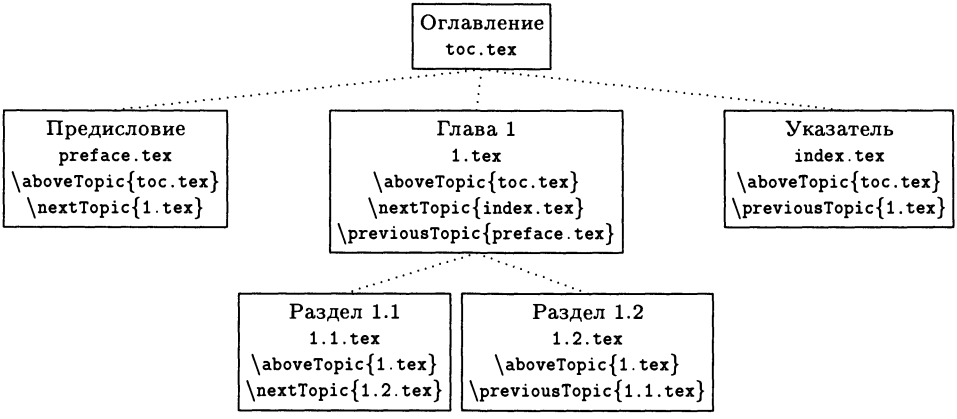


Рис. 5.11. Дерево документа.

определяет ту команду, которая будет выполняться при щелчке на фрагменте текста, задаваемого аргументом *text*.

В следующем примере показывается, как с помощью рассматриваемой команды можно вывести на экран оглавление корневого каталога на диске C: вашего компьютера, работающего под управлением операционной системы Microsoft Windows:

```
Щелкните клавишей мыши \appLink{dir C:\}{здесь}, чтобы увидеть
корневой каталог вашего диска C.
```

Аргумент *command* может ссылаться только на программы, которые постоянно находятся на вашем компьютере, в том числе и на команды операционной системы. Поэтому ссылки на приложения могут оказаться непереносимыми с одной платформы на другую и, возможно, даже между машинами, работающими под управлением одной и той же операционной системы. Однако они могут быть весьма полезны для приложений, в которых `techexplorer` используется как часть более обширной системы электронной публикации. Например, некоторые ссылки на приложения могут использоваться в интерактивных книгах, которые предоставляют полный набор средств, необходимых каждому студенту.

В связи с использованием ссылок на приложения важное значение приобретают вопросы безопасности, более того, эти проблемы возникают при использовании любого сетевого программного обеспечения, которое может выполнять программы на вашем компьютере. Прежде чем разрешить выполнение команды, вам следует знать, что именно она делает, или, по крайней мере, вы должны доверять источнику, из которого получен документ, содержащий ссылку на приложение.

На рис. 5.7 на с. 249 показана стандартная вкладка свойств для задания разрешенных действий techexplorer'а. При стандартной установке свойств ссылок на приложения требуется всегда запрашивать разрешение перед запуском приложения. Если вы более доверчивы, то можете выбрать опцию, которая позволяет всегда запускать приложение при активизации ссылки на него. И, наконец, если вы хотите запретить выполнение любых ссылок на приложения, выберите из предлагаемых вариантов «do not execute».

Для большинства пользователей, вероятно, наилучшим будет стандартный вариант, при котором всегда запрашивается разрешение на выполнение приложения. Однако, если вы не знаете, что именно будет делать конкретная команда, узнайте о ней побольше, прежде чем запускать ее.

5.1.10 Перемещение между двумя отображаемыми выражениями

\altLink{*secondText*}{*firstText*}

Вы можете использовать *переменную ссылку (alternating link)*, чтобы попеременно отображать один из двух фрагментов текста. Первоначально отображается выражение, заданное аргументом *firstText*. При щелчке на этом выражении techexplorer отобразит выражение, заданное аргументом *secondText*. Переменные ссылки могут показаться мелочью, но они могут быть полезны в ситуациях, когда вы хотите заставить читателя подумать, прежде чем он прочитает «ответ». Рассмотрим следующую задачу и ее решение, в котором показывается одно либо другое выражение:

Problem: Solve for x in $ax^2+bx+c=0$.
Solution: x = ?

Problem: Solve for x in $ax^2+bx+c=0$.
Solution: $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$

Основное выражение создается командами

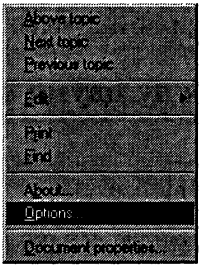
```
\textbf{Problem:} Solve for $x$ in $ax^2+bx+c=0$.

\textbf{Solution:} $x=\altLink{\frac{-b \pm\sqrt{b^2-4ac}}{2a}}{?}$
```

Если второй раз щелкнуть на $\frac{-b \pm \sqrt{b^2-4ac}}{2a}$, то изображение вновь примет форму вопросительного знака.

5.1.11 Печать из techexplorer'а

Только в Профессиональном издании techexplorer'а для системы Windows обеспечивается печать целых документов, отображаемых с помощью динамически подключаемого модуля![1] В любом навигаторе наиболее надежный способ печати документа techexplorer'а состоит в использовании стандартного контекстного меню. Для доступа к стандартному меню подвиньте указатель мыши на свободное место окна techexplorer'а и щелкните правой клавишей. В Профессиональном издании techexplorer'а это стандартное меню выглядит следующим образом:



Выберите пункт **Print...** и после этого пройдите по цепочке меню, чтобы выбрать нужный документ и направить его на принтер. Число страниц в документе остается неизвестным до тех пор, пока не будет завершено формирование файла для печати, поэтому печатайте либо документ целиком, либо только отдельные известные страницы.

В некоторых навигаторах (прежде всего в Netscape Navigator) вы также сможете воспользоваться пунктом **Print...** из меню **File** или кнопкой «печать» на панели инструментов навигатора.

Первой версией techexplorer'а, в которой вообще поддерживался вывод на печать, была версия 2.0. В этой редакции преследовалась цель распечатывать LaTeX-овские документы с качеством, сравнимым с качеством печати HTML-документов в навигаторе. Вообще говоря, сам LaTeX, если у вас есть возможность его использовать, лучше справится с задачей вывода на печать. В оперативной документации вы можете найти дополнительную информацию о стилевом файле, предназначенном для того, чтобы помочь вам в разработке документов, которые могут обрабатываться как techexplorer'ом, так и LaTeX'ом.

Возможно, что в будущих редакциях techexplorer'а будет больше возможностей для управления такими параметрами страницы, как поля. Кроме того, фирма IBM работает над созданием PostScript'овской версии с целью сделать возможным вывод на печать в будущих версиях Профессионального издания techexplorer'а для системы UNIX.

[1] Для динамически подключаемых модулей, встроенных в HTML-страницу, печать пока еще не поддерживается.

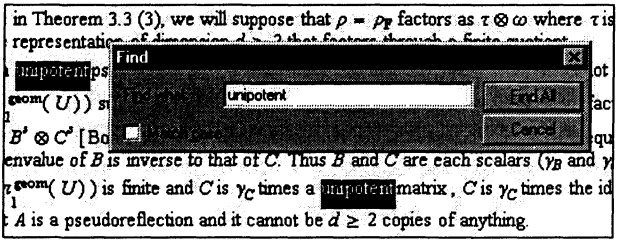


Рис. 5.12. Поиск текста в документе.

5.1.12 Поиск внутри документа

В Профессиональном издании techexplorer'а имеется функция поиска, которая позволяет обнаружить все вхождения некоторой строки в документе. Для того чтобы начать поиск, щелкните правой клавишей мыши внутри документа и из появившегося контекстного меню выберите пункт **Find...**. Введите строку, которую нужно найти, и нажмите кнопку **Find**. В результате в вашем документе будет выбран текст, соответствующий введенной вами строке.

На рис. 5.12 показан пример, в котором в документе ищется слово «unipotent». В иллюстративных целях в тексте, изображенном позади диалогового окна, отмечены результаты предыдущего поиска.

Примечание: для того чтобы определить положение текста в документе techexplorer'а, можно использовать *только* диалог поиска самого techexplorer'а, активизируемый с помощью открывающегося в документе контекстного меню. Диалог поиска навигатора будет выполнять поиск только на HTML-страницах, не спускаясь до уровня вложенных окон динамически подключаемых модулей. Еще хуже то, что некоторые навигаторы будут зависать при попытке вызвать их диалог поиска в то время, когда все окно навигатора находится под управлением некоторого динамически подключаемого модуля.

5.1.13 Оптимизация ваших документов для techexplorer'а

Этот раздел представляет собой «поваренную книгу» для тех, кто хочет сделать существующие LaTeX'овские документы пригодными для использования в techexplorer'е. Попутно мы добавим в электронную версию документов некоторые возможности, способствующие улучшению зрительного впечатления.

- Определите, достаточно ли мал ваш документ, чтобы его можно было обрабатывать динамически подключаемым модулем techexplorer'а. Если нет, то разбейте его на меньшие документы, которые можно будет объединить с помощью гиперссылок.
- Даже если ваш документ достаточно мал, подумайте, не стоит ли все же каким-либо образом разбить его на связанные гиперссылками компонен-

ты. Читателям нравятся документы, которые можно быстро получить и воспроизвести.

- Используйте команды `\aboveTopic`, `\nextTopic` и `\previousTopic` для создания иерархической структуры документа (см. разд. 5.1.8 на с. 256).
- `techexplorer` пока еще не поддерживает перекрестные ссылки, созданные с использованием команды `\ref`. Такие ссылки следует заменить на явные гиперссылки.
- Снабдите метками все команды секционирования документа. При этом используйте стандартные соглашения об именах, с тем чтобы можно было использовать макрокоманды для упрощения остальной разметки. Например, используйте `chapter:2` или `section:3.4`.
- Добавьте гиперссылки в тех местах, где в тексте содержатся ссылки на выражения типа «глава 2» или «раздел 3.4» (хотя следует помнить, что в гипертекстовых документах выражения типа «глава 2» имеют смысл только для статических документов. Для создания таких гиперссылок используйте добавленные вами ранее метки. Используйте команду `\docLink` для гипертекстового перехода за пределы данного документа и команду `\labelLink` для переходов в пределах документа. Подробности см. в разд. 5.1.3 на с. 241.

5.1.14 Разработка сценариев для techexplorer'а на Java и JavaScript

При работе в среде навигатора Netscape Navigator Профессиональное издание `techexplorer`'а позволяет использовать программный интерфейс для написания сценариев на языках Java и JavaScript. При этом используется функция прямого соединения с сервером Netscape (Netscape LiveConnect), и поэтому работа в среде навигатора Microsoft Internet Explorer 4.0 невозможна.

Благодаря этому интерфейсу апплеты и сценарии могут регистрироваться как приемники происходящих в окнах `techexplorer`'а событий, связанных с мышью, клавиатурой и фокусировкой окна. Кроме того, апплеты и сценарии могут обновлять отображаемые в этих окнах документы `techexplorer`'а.

Детальное рассмотрение программного интерфейса выходит за рамки данной книги, но в качестве аннотированного примера (рис. 5.13) предлагается рассмотреть простой LaTeX'овский редактор, написанный на языке Java и использующий `techexplorer` для отображения форматированной разметки. Этот редактор состоит из двух окон внутри HTML-страницы. Верхнее из них — это окно `techexplorer`'а, в котором отображается LaTeX'овская разметка. Нижнее окно содержит исходный LaTeX'овский текст в таком виде, в каком он был введен пользователем. Текст можно вводить независимо от того, в каком окне находится курсор, хотя может понадобиться указать мышью на одно или другое окно, чтобы именно оно реагировало на ввод с клавиатуры. С помощью кнопки **Clear input** можно полностью удалить разметку и полученное из нее

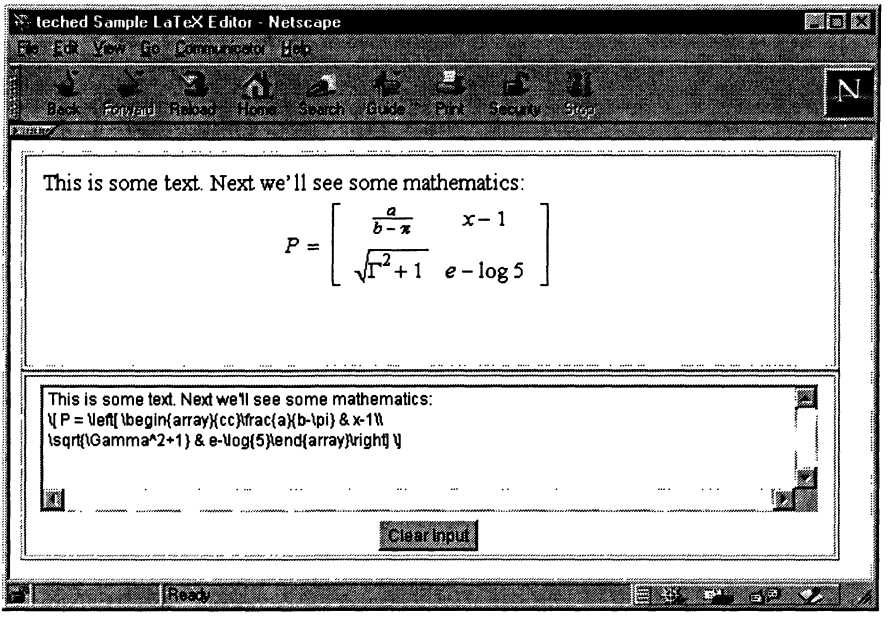


Рис. 5.13. Простой LaTeX'овский редактор, созданный с помощью `techexplorer`'а.

изображение. На рис. 5.13 показаны ввод и воспроизведение текста в этом редакторе.

Исходный код редактора приведен в приложении A.2. Он состоит из двух файлов: `teched.html`, содержащего исходный HTML-код для страницы (разд. A.2.1 на с. 455), и `teched.java`, содержащего исходный код Java-апплета (разд. A.2.2 на с. 456).

5.2 WebEQ

WebEQ [↪WEBEQ] — это набор инструментов, содержащий Java-апплет для отображения математических выражений.[1] В нем обеспечивается поддержка языка математической разметки Mathematical Markup Language (MathML), к которому мы вернемся в гл. 8, и набора команд, которому дано имя WebTEX.

Пакет WebEQ включает в себя редактор, который позволяет создать математическое выражение, а потом сохранить его в виде исходного кода MathML, рисунка в формате JPEG или PNG либо в виде законченного элемента Java-апплета, который можно скопировать и вставить в HTML-страницу. В редакторе предусмотрена также возможность открытия файла, содержащего уже

[1] WebEQ — это торговый знак фирмы Geometry Technologies, Inc.

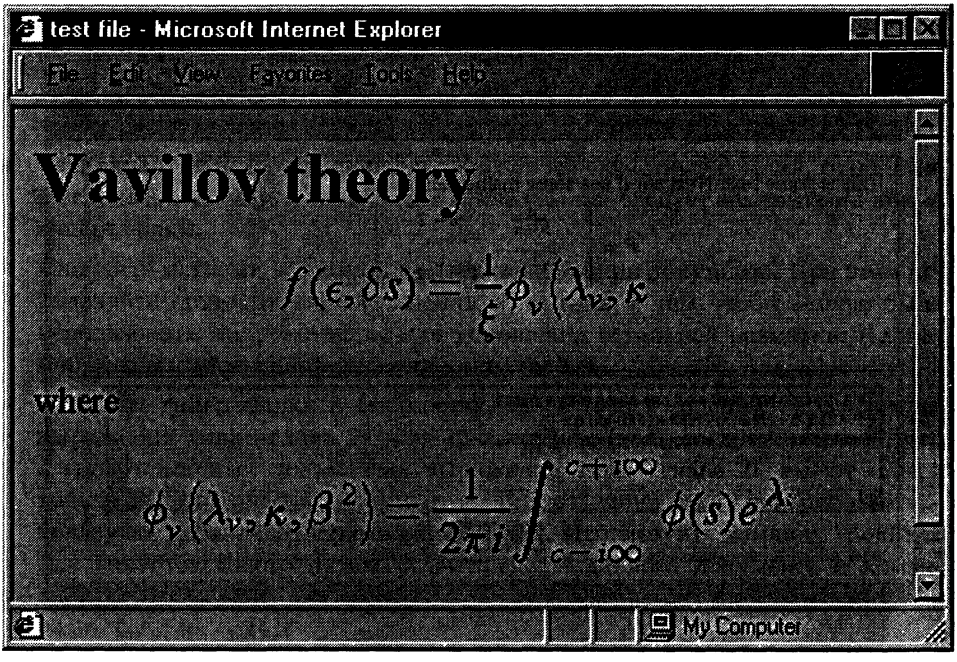


Рис. 5.14. Простой пример WebEQ.

существующее выражение на языке MathML, но он не позволяет сохранить результаты работы в WebTeX'овском формате.

Этот редактор был создан с использованием входящего в состав WebEQ программного интерфейса для языка Java, который позволяет добавлять к вашим собственным апплетам функции отображения математических выражений и создания рисунков, а также дает возможность управлять WebEQ с помощью JavaScript. Вы можете динамически создавать новые математические объекты, а также манипулировать внутренней структурой уже существующих объектов.

Для облегчения вставки математических объектов в Web-страницы разработан Мастер WebEQ. Эта прикладная программа воспринимает в качестве исходных данных файл, содержащий разметку на языке HTML и разметку WebTeX или MathML, и создает новый HTML-файл, содержащий рисунки или элементы апплета WebEQ для математических выражений. На рис. 5.14 показан слегка упрощенный фрагмент нашего тестового файла (см. рис. 5.1 на с. 233), обработанного Мастером WebEQ для создания рисунков. На рис. 5.15 показан пример созданного с помощью WebTeX исходного текста для Мастера WebEQ и полученного в результате HTML-кода. Далее мы более подробно рассмотрим все те функции WebEQ, которые имеют отношение к WebTeX. Мы

Исходный файл Мастера WebEQ:

```
<HTML>
<HEAD><TITLE>test file</TITLE></HEAD>
<BODY>
<H1>Vavilov theory</H1>
\[
f \left ( \epsilon, \delta s \right ) = \frac{1}{\xi} \phi_{v}
\left ( \lambda_{v}, \kappa, \beta^{2} \right )
\]
where
\[
\phi_{v} \left ( \lambda_{v}, \kappa, \beta^{2} \right ) =
\frac{1}{2 \pi i} \int^{c+i\infty}_{c-i\infty}\phi \left( s \right )
e^{\lambda s} ds  c \geq 0
\]
</body>
</html>
```

Результат работы Мастера WebEQ:

```
<HTML>
<HEAD><TITLE>test file</TITLE></HEAD>
<body bgcolor=#C0C0C0>
<H1>Vavilov theory</H1>
<P><CENTER>
<applet code="webeq.Main" width=194 height=48 align=middle>
<param name=eq value="\displaystyle {
f \left ( \epsilon, \delta s \right ) = \frac{1}{\xi} \phi_{v}
\left ( \lambda_{v}, \kappa, \beta^{2} \right )
}">
<param name=color value="#C0C0C0">
<param name=parser value="webtex">
<img src="xxx1.png" alt="xxx1.png" align=absmiddle>
</applet></CENTER><P>
where
<P><CENTER>
<applet code="webeq.Main" width=342 height=68 align=middle>
<param name=eq value="\displaystyle {
\phi_{v} \left ( \lambda_{v}, \kappa, \beta^{2} \right ) =
\frac{1}{2 \pi i} \int^{c+i\infty}_{c-i\infty}\phi \left( s \right )
e^{\lambda s} ds  c \geq 0
}">
<param name=color value="#C0C0C0">
<param name=parser value="webtex">
<img src="xxx2.png" alt="xxx2.png" align=absmiddle>
</applet></CENTER><P>
</body>
</html>
```

Рис. 5.15. Простой пример WebEQ (рис. 5.14): вход и выход Мастера.

начнем с обсуждения WebTEX, а затем покажем, как можно записать элементы APPLET языка HTML для вставки в документ математических выражений. В заключение мы расскажем о Мастере WebEQ и рассмотрим пример его использования для автоматической вставки в документ рисунков или элементов APPLET.

5.2.1 Введение в WebTeX

WebTeX — это не совсем TeX и не совсем LaTeX. Обычно его команды похожи на какие-то TeX'овские или LaTeX'овские команды, но прежде чем создавать новую разметку или преобразовывать существующую в формат WebTeX, нужно все-таки свериться с документацией.

Сначала рассмотрим функции, аналогичные имеющимся в TeX'е или LaTeX'е. Используйте '{' и '}' для группировки (объединения) выражений, а _ и ^ для нижних и верхних индексов.

Для дробей и биномиальных коэффициентов используйте, соответственно, команды `\frac` и `\binom`. Привычный синтаксис имеют команды `\sqrt` и `\root` для символов радикалов. Поддерживаются также все обычные операторы стандартных функций, таких, как `\cos`. Диакритические (надстрочные) знаки можно создавать с помощью команд `\bar`, `\check`, `\dot`, `\ddot`, `\hat`, `\tilde` и `\vec`. С помощью команд `\overbrace` и `\underbrace` можно нарисовать растягивающиеся фигурные скобки (нужного размера) над и под выражением соответственно. Команды `\overset` и `\underset` позволяют разместить выражение над или под другим выражением соответственно. С помощью команды `\text` в выражение можно вставить простой текст. Он может включать в себя встроенное математическое выражение (ограниченное символами `$...$`), но ничего больше.

Поддерживаются стандартные TeX'овские или LaTeX'овские символы, но набор символов AMS не поддерживается. В отличие от `techexplorer`'а, для используемых в WebEQ символов не требуются дополнительные шрифты.[1] Изображения литер и информация о формате для фиксированного набора шрифтов и размеров содержатся в специальном апплете. Для использования семейств шрифтов Roman (прямой шрифт, антиква), italic (курсив), bold (полужирный), fraktur (готический), sans serif (рубленый), typewriter (шрифт пишущей машинки), blackboard bold (ажурный) и calligraphic (рукописный) можно применять команды `\mathrm`, `\mathit`, `\mathbf`, `\mathfr`, `\mathsf`, `\mathtt`, `\mathbb` и `\mathcal` соответственно.

Чтобы изменить размер шрифта в выражении, можно использовать команды `\textsize`, `\scriptsize` или `\scriptscriptsize` (в порядке уменьшения размера шрифта). Стиль форматирования можно явным образом изменить командами `\displaystyle` или `\textstyle`, но такие команды, как `\scriptstyle`, `\scriptscriptstyle`, `\large`, `\small` и т. д., не поддерживаются.

В чем же заключаются действительные *различия* между WebTeX'ом, с одной стороны, и TeX'ом/LaTeX'ом — с другой? Макро создаются с помощью команды `\define`, имеющей такой же синтаксис, как и LaTeX'овская команда `\newcommand`. Макро передаются в апплет WebEQ в качестве параметров; для этого используются параметры `macros` или `macrofile`. (Сведения о макрокомандах приведены на с. 270.)

[1] В комплект дистрибутива Профессионального издания `techexplorer` входят, однако, символьные шрифты.

`\fontcolor{`*color*`}{`*expression*`}`

LaTeX'овские команды для работы с цветом не поддерживаются, но для задания цвета какой-либо отдельной части выражения можно использовать команду `\fontcolor`. Первым параметром команды `\fontcolor` является спецификация цвета `#RRGGBB`, а второй параметр — это выражение, которое должно быть изображено этим цветом. Например, команда

```
\fontcolor{#c0c0c0}{x+y}
```

обеспечивает вывод формулы $x + y$ светло-серым цветом.

`\multiscript{`*prescripts*`}{`*base*`}{`*scripts*`}`
`\tensor{`*base*`}{`*scripts*`}`

Имеющаяся в WebTeX команда `\tensor` позволяет описывать тензоры со всеми их нижними и верхними индексами. Команда `\multiscript` является универсальным инструментом для размещения нижних и верхних индексов перед и после основного выражения. Например, в результате выполнения команды

```
\multiscript{_i^j}{H}{_k^l}
```

получается

$$ {}_i^j H_k^l $$

Стандартные ограничители поддерживаются командами `\left` и `\right`, но обратите внимание на то, что не следует использовать точку для указания присутствующего, но пустого ограничителя. Просто пропустите ограничитель. Не забывайте об этом, если вы преобразуете TeX'овские выражения.

WebTeX содержит мощную функцию форматирования матриц, которая также в определенной степени отличается от функции, известной пользователям LaTeX'а. Основной командой является команда `\array`, для разделения столбцов используется символ `&`, а для разделения строк — символ `\\`. Способ выравнивания столбцов задается параметром `\colalign` в команде `\arrayopts`. Сравните LaTeX'овское выражение

$$ \begin{array}{lcr} x & y+1 & z-1 \\ y-z & x^2 & 0 \end{array} $$

заданное разметкой

```
\begin{array}{lcr}
x     & y + 1 & z - 1 \\
y - z & x^2   & 0
\end{array}
```

с соответствующей WebTeX'овской разметкой:

```
\array{
\arrayopts{\colalign{left center right}}
```

```
x     & y + 1 & z - 1 \\
y - z & x^2   & 0
}
```

Для выравнивания элементов каждой строки по вертикали используйте в \arrayopts команду \rowalign. Она позволяет сдвигать элементы заданной строки вниз, с тем чтобы обеспечить выравнивание всех элементов по нижнему краю. Параметры, определяющие способы выравнивания top, bottom, axis или baseline, задаются для каждой строки.

Дополнительные опции массива позволяют нарисовать вокруг массива рамку с выбранным стилем линий; включать в массив элементы, занимающие более одной строки или одного столбца (обобщая таким образом команду \multicolumn); рисовать линии между строками или столбцами; вставлять нужное число пробелов между элементами; а также выравнивать весь массив по вертикали относительно базовой линии. Благодаря столь богатому набору опций WebEQ способен самостоятельно работать с моделью таблиц языка MathML.

Вставить дополнительные пробелы в выражение можно с помощью команд \thinsp, \medsp, \thicksp и \quad. Команда \quad ничем не отличается от такой же TeX'овской команды, а три остальные полностью аналогичны известным командам \thinspace, \medspace и \thickspace. Команда \qquad не поддерживается.

\space{*height*}{*depth*}{*width*}

Команда \space, в отличие от ее стандартного использования в NFSS, создает свободную область с заданными высотой (height), глубиной (depth) и шириной (width).

\rule{*height*}{*depth*}{*width*}

Команда \rule, как и в LaTeX'е, создает сплошной прямоугольник, но ее аргументы имеют другой смысл.

Важно отметить, что WebEQ игнорирует и не отображает в WebTeX'е те команды, которые он не понимает. В частности, если при вводе будет допущена опечатка, то часть выражения не будет изображена. Поэтому всегда следует сравнивать воспроизведенный результат с разметкой, чтобы убедиться в том, что вы получили именно то, что хотели.

5.2.2 Интерактивные расширения

\href{*url*}{*expression*}

С помощью команды \href WebTeX поддерживает гипертекстовые ссылки. Аргумент *expression* определяет то, что будет видно на экране (синим цветом),

а аргумент *url* — это адрес документа, к которому навигатор перейдет, когда читатель щелкнет клавишей мыши на фрагменте текста *expression*.

`\statusline{`*message*`}{`*expression*`}`

Используя команду `\statusline`, можно изменить сообщение (текст которого определяется параметром *message*), которое будет появляться в строке состояния навигатора, когда пользователь поместит указатель мыши над выражением, указанным в аргументе *expression*. Эта возможность оказывается очень полезной для того, чтобы объяснять значения отдельных частей громоздкого выражения. Обратите внимание на то, что сообщение в строке состояния изображается в виде простого текста, поэтому не надо делать его слишком причудливым.

`\fghighlight{`*color*`}{`*expression*`}`
`\bghighlight{`*color*`}{`*expression*`}`

Для лучшего выделения выражения используйте команды `\fghighlight` или `\bghighlight`. Команда `\fghighlight` изменяет цвет переднего плана, т. е. цвет текста, когда указатель мыши перемещается над выражением, заданным параметром *expression*. Аналогично, команда `\bghighlight` изменяет цвет фона, когда указатель мыши находится над тем же выражением. Аргумент *color* задается в виде `#RRGGBB` с использованием стандартного формата RGB определения цвета в языке HTML.

Благодаря тому что эти функции можно использовать совместно, появляется возможность одновременно изменять цвета и обновлять сообщение в строке состояния. Например, в результате выполнения команд

```
\bghilight{#c0c0c0}{
  \fghilight{#ff0000}{
    \statusline{This is y+1}{y + 1}}}
```

цвет фона изменится на серый, цвет шрифта, которым набрано выражение, станет красным, а когда указатель мыши на экране окажется над «$y + 1$», в строке состояния появится сообщение `This is y+1`.

`\toggle{`*expr1*`}{`*expr2*`}{`*message1*`}{`*message2*`}`

Как `techexplorer` имеет команду `\altLink`, так и WebEQ имеет команду `\toggle`, предназначенную для попеременного отображения двух выражений. Вначале на экране отображается выражение, заданное аргументом *expr1*, а когда указатель мыши проходит над ним, в строке состояния выводится содержимое аргумента *message1*. Если щелкнуть клавишей мыши на этом выражении, то оно будет заменено на указанное в аргументе *expr2*, а при движении мыши в строке состояния будет выводиться содержимое аргумента *message2*.

5.2.3 Использование элемента APPLET в WebEQ

Элемент APPLET языка HTML позволяет вставить WebTEX'овскую разметку, которую WebEQ должен воспроизвести на HTML-странице. Например,

```
<applet codebase="classes"
        code="webeq.Main"
        width=100 height=100 align=middle>
  <param name=eq value="\alpha^2-\frac{1}{\beta}">
  <param name=color value="#FFFFFF">
</applet>
```

При использовании элемента APPLET в WebEQ имеют значение следующие пять атрибутов:

CODEBASE — имя каталога, содержащего созданный Java-компилятором байт-код для WebEQ. Чтобы правильно задать его в соответствии с тем, как вы намереваетесь устанавливать и использовать это программное обеспечение, ознакомьтесь с технической документацией.

CODE — это всегда webeq.Main.

HEIGHT — высота (в пикселах) прямоугольника, в котором должна отображаться разметка.

WIDTH — ширина (в пикселах) прямоугольника, в котором должна отображаться разметка.

ALIGN. Используйте ALIGN=MIDDLE, если вы хотите, чтобы выражение могло перемещаться по вертикали для лучшего расположения относительно базовой линии окружающего текста.

Элемент PARAM используется внутри элемента APPLET для передачи в WebEQ дополнительной форматирующей и управляющей информации. Элемент PARAM имеет следующий формат:

```
<param name=TheName value="TheValue">
```

где допустимыми значениями аргумента TheName являются allow_cut, color, controls, eq, linebreak, macrofile, macros, parser, size и src.

Если задано имя параметра eq или src, то второй аргумент value определяет способ получения исходной разметки математического выражения. При имени eq второй аргумент содержит саму WebTEX'овскую разметку. При имени src второй аргумент является адресом URL содержащего разметку документа (относительно URL охватывающей HTML-страницы).

Используйте имя parser для указания вида исходной разметки: WebTEX или MathML. Если этот параметр не задан, то предполагается, что разметка является WebTEX'овской.

Если заданы имена параметра macros или macrofile, то значение второго аргумента задает макро, которые должны применяться в разметке WebEQ.

Используйте параметр `macros` для задания одного или нескольких явных макроопределений. Например,

```
<param name=macros
       value="\define{\a}{\alpha}\define{\b}[1]{\beta_#1}">
```

задает определения для `\a` и `\b`, причем у первого из них нет аргументов, а у второго — один аргумент. Если вы хотите объединить несколько макроопределений в один файл, то используйте параметр с именем `macrofile` для того, чтобы передать в WebEQ адрес URL этого файла. Как и в случае со значением, соответствующим имени `src`, этот адрес URL задается относительно URL охватывающей HTML-страницы. Если будут заданы как `macros`, так и `macrofile`, то все макро будут собраны вместе до того, как будет выполнен анализ математического выражения.

Значение параметра `size` задает начальный размер шрифта (в пунктах) для математического выражения. WebEQ содержит восемь шрифтов нескольких фиксированных размеров. Если заданное вами значение не совпадет ни с одним из доступных, то WebEQ выберет самый близкий по размеру шрифт. С параметром `size` связан параметр `controls`. Если он имеет значение `true` (принимаемое по умолчанию), то при щелчке правой клавишей мыши, когда курсор находится в зоне отображения апплета, появится диалоговое окно, в котором можно изменить размер шрифта. Это поможет добиться соответствия изображений математических формул окружающему HTML-тексту, а также позволит внести поправки, обусловленные различиями в визуализации Java на разных платформах.

Еще одним форматирующим параметром является `color`. Он позволяет задать цвет фона для прямоугольника, в котором WebEQ отображает данное математическое выражение. Его значение задается в форме `#RRGGBB` с использованием стандартного формата RGB определения цвета в языке HTML. Обратите внимание на то, что в данном случае не допускаются используемые в HTML имена цветов, такие, как, например, «red». Используйте этот параметр для того, чтобы цвет фона математического выражения соответствовал цвету фона всей HTML-страницы. Если вы хотите изменить цвет изображаемого с помощью WebEQ-выражения, то используйте WebTEX'овскую команду `\fontcolor` непосредственно внутри разметки.

Для выражений, слишком длинных по сравнению с отведенным им местом, можно использовать параметр `linebreak` со значением `true`. Тогда WebEQ попробует разделить выражение на несколько строк. На рис. 5.16 показано, как WebEQ может разбить выражение $x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$ на две строки. Возможно, при этом потребуется увеличить значение атрибута `HEIGHT`, чтобы учесть увеличение высоты занимаемой выражением области.

Последний параметр — это `allow_cut`. Если он имеет значение `true`, то разрешается выделение подвыражений,ё и, кроме того, созданная разметка на

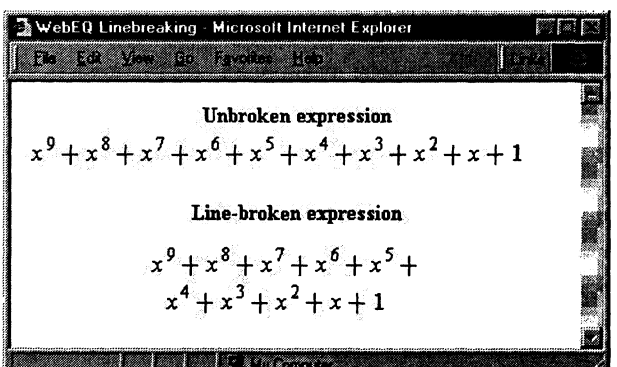


Рис. 5.16. Разбиение строк в WebEQ.

языке MathML может быть помещена во всплывающее окно для вырезания и вставки.

5.2.4 Подготовка HTML-страниц с помощью Мастера WebEQ

Мастер WebEQ предоставляет простой способ подготовки HTML-страниц, содержащих математические выражения. Сначала следует создать исходный HTML-файл, в котором WebTeX'овская математическая разметка заключена внутри ограничителей `$...$` в случае внутритекстовых формул и внутри `\[...\]` для выключных формул.[1] Для этого исходного файла рекомендуется использовать расширение `.src` или `.wiz`, чтобы отличать его от получающегося в итоге `.html`-файла.

После этого запустите Мастер WebEQ, чтобы создать предназначенный для использования в Сети HTML-файл. Мастер преобразует ваши математические выражения либо в рисунки, либо в элементы Java-апплета. Эти элементы инкапсулируют WebTeX'овскую математическую разметку, либо разметку MathML, содержавшуюся в исходном файле, либо разметку MathML, созданную из WebTeX'овского первоисточника. Рисунки могут иметь формат JPEG или PNG, хотя только новейшие Web-навигаторы поддерживают формат PNG и при этом необходима версия Java 1.1 или старше. Элементы апплета будут также содержать предполагаемые значения высоты и ширины прямоугольников, в которых будут отображаться математические выражения.

Выражения, отображаемые в виде рисунков, могут выглядеть более единообразно в разных навигаторах и на платформах разных операционных систем, а кроме того, они могут быть выведены на печать, хотя и с меньшим

[1] Разметку на языке MathML можно также вводить, используя ограничители `<math>` ... `<\math>`.

разрешением, чем окружающий текст. Выражения, отображаемые с помощью апплета, в настоящее время нельзя вывести на печать, кроме того, при этом подходе могут возникнуть проблемы с «форматирующими» прямоугольниками (внутри которых воспроизводятся формулы): WebEQ обрезает изображение формулы по границам этого прямоугольника. С помощью Мастера можно слегка «раздуть» этот прямоугольник, чтобы его предполагаемые размеры наверняка оказались достаточными. Объем этого «воздуха» можно увеличить с помощью опции (проблема размеров прямоугольника, возникающая как в `techexplorer`'е, так и в WebEQ, обсуждается в разд. 5.3.1 на с. 275).

С помощью других форматирующих опций Мастера можно задать исходный размер шрифта для выражений и цвет фона. Дополнительные опции определяют, будет ли включаться режим разбиения длинных математических выражений на несколько строк.

Чтобы показать, как работает Мастер WebEQ, возьмем следующий исходный HTML-документ, содержащий встроенные математические выражения:

```
<HTML>
<HEAD>
  <TITLE>Example WebEQ Wizard Page</TITLE>
</HEAD>
<BODY>
<P>
The answer to Question 1 is
<!-- the following fraction will be shown inline -->
$\frac{x}{y}$
and the answer to Question 2 is
<!-- the following matrix will be shown in display mode -->
\[\left( \array{2&3 \\ 4&\frac{5}{6}} \right).\]
</P>
</BODY>
</HTML>
```

Если выбрать создание элементов Java-апплета, в которых инкапсулируется математический ввод, и при этом использовать стандартные значения опций, то будет создан следующий HTML-документ:

```
<HTML>
<HEAD>
  <TITLE>Example WebEQ Wizard Page</TITLE>
</HEAD>
<body bgcolor=#C0C0C0>
<P>
The answer to Question 1 is
<!-- the following fraction will be shown inline -->

<applet code="webeq.Main" width=12 height=32 align=middle>
<param name=eq value="\frac{x}{y}">
<param name=color value="#C0C0C0">
<param name=parser value="webtex">
```

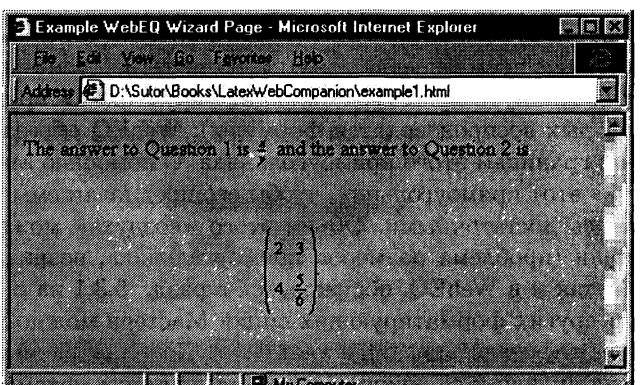


Рис. 5.17. HTML-страница, созданная Мастером WebEQ.

```
</applet>
and the answer to Question 2 is
<!-- the following matrix will be shown in display mode -->

<P><CENTER>
<applet code="webeq.Main" width=71 height=108 align=middle>
<param name=eq
value="\displaystyle {\left( \array{2&3 \\ 4&\frac{5}{6}} \right).}">
<param name=color value="#C0C0C0">
<param name=parser value="webtex">
</applet></CENTER><P>

</P>
</BODY>
</HTML>
```

Окончательный результат, полученный с помощью WebEQ, показан на рис. 5.17. Обратите внимание на то, что благодаря заданию атрибута `align=middle` математические объекты занимают примерно правильное положение по отношению к базовой линии текста.

5.3 Проблемы «внедренного содержимого» и будущее развитие

Как уже упоминалось в разд. 5.1.2.1 на с. 238, при воспроизведении содержимого документа с помощью динамически подключаемых модулей или встроенных в HTML-страницу Java-апплетов могут возникать некоторые проблемы с визуализацией[1]. Все эти проблемы связаны с математическими выражения-

[1] Чтобы отличать такие элементы документа от тех, которые отображаются во всей клиентской области навигатора, будем использовать для них термин «внедренное содержимое».

ми, которые заметно отличаются от окружающего HTML-текста. Эти проблемы обсуждаются в последующих разделах, и там же предлагаются возможные способы их решения в будущих навигаторах.

5.3.1 Размер выражения

Внедренное содержимое отображается на HTML-странице внутри прямоугольника фиксированного размера. Размеры этого прямоугольника явным образом задаются атрибутами `HEIGHT` и `WIDTH` элемента `EMBED` (в случае динамически подключаемых модулей) или элемента `APPLET` (при использовании Java-апплетов).

Рассмотрим, как выражение $\sqrt{1+y-z^2}$ будет воспроизводиться на HTML-странице с помощью `techexplorer`'а и WebEQ. В обоих случаях будем использовать шрифт достаточно большого размера, задавая при этом значение атрибута `HEIGHT` равным 90 пикселов, а значение атрибута `WIDTH` равным 200 пикселов. Ниже приведена разметка HTML для `techexplorer`'а:

```
<EMBED TYPE="application/x-techexplorer"
       TEXDATA="\pagecolor{white}\(\sqrt{1 + y-z^2}\)"
       WIDTH=200 HEIGHT=90>
```

А вот разметка HTML для WebEQ:

```
<APPLET CODEBASE="classes" CODE="webeq.Main" WIDTH=200 HEIGHT=90>
    <PARAM NAME=color VALUE=#FFFFFF>
    <PARAM NAME=size VALUE=18>
    <PARAM NAME=eq VALUE="\sqrt{1 + y-z^2}">
</APPLET>
```

На рис. 5.18 показана таблица с двумя вариантами изображения этой формулы, полученными с помощью Microsoft Internet Explorer 4.0 в операционной системе Windows NT.

На рис. 5.19 показано, что произойдет, если уменьшить значение атрибута `WIDTH` до 105. В этом случае `techexplorer` создает горизонтальную полосу прокрутки, с помощью которой можно перемещать изображение влево и вправо, чтобы увидеть все выражение. Этот способ эффективен, но непривлекателен. Воспроизведенное математическое выражение не вполне сочетается с окружающими элементами HTML-страницы. WebEQ обрезает изображение формулы с правой стороны. Хотя это, может быть, и лучше выглядит, но не забывайте, что исходное выражение было $\sqrt{1+y-z^2}$, а отнюдь не $\sqrt{1+y}$!

Мастер WebEQ (см. разд. 5.2.4 на с. 272) обладает реальными средствами для правильного оценивания размеров прямоугольника, требуемых для отображения выражения. Поэтому опасность того, что выражение может оказаться обрезанным, возникает только в тех случаях, когда пользователь вручную меняет ширину прямоугольника (области отображения) либо изменяет размер используемого шрифта. Использование более крупного шрифта может привести к усечению выражения. Более мелкий шрифт как в `techexplorer`'е, так

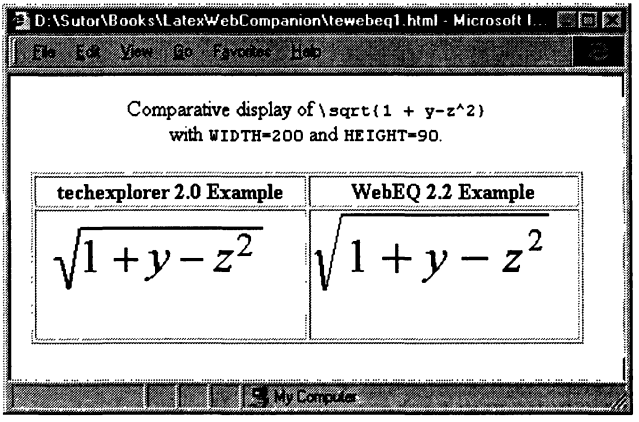


Рис. 5.18. Приемлемые размеры выражения в `techexplorer`'е и WebEQ.

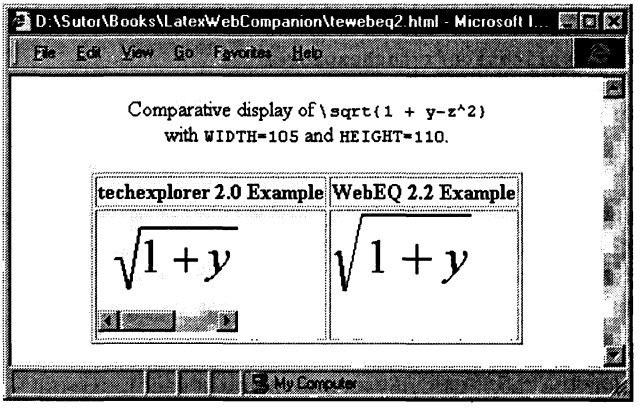


Рис. 5.19. Влияние уменьшения ширины прямоугольной области воспроизведения в `techexplorer`'е и WebEQ.

и в WebEQ приведет к избытку свободного пространства, из-за чего математическое выражение на HTML-странице снова будет выглядеть непривлекательным.

Основная проблема здесь заключается в том, что размер прямоугольной области воспроизведения определяется автором, а не программой — динамически подключаемым модулем или апплетом. Однако точный размер этого прямоугольника известен только программе визуализации. В будущих навигаторах должна быть обеспечена возможность обмена информацией с динамически подключаемым модулем или апплетом для определения оптимального размера области отображения.

Необходимо отметить, что могут существовать внешние ограничения на размер прямоугольника области изображения, поэтому программа визуализации математических выражений должна обладать определенной гибкостью,

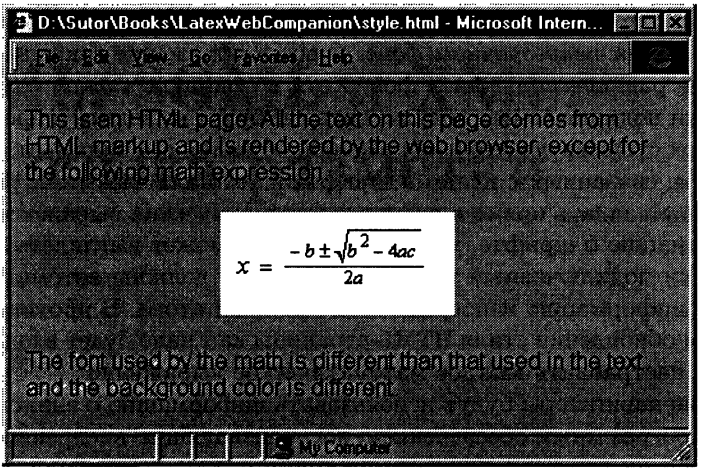


Рис. 5.20. Некоторые проблемы согласования стилей.

чтобы уместить выражение в отведенном для него месте. WebEQ допускает перенос строк в математических выражениях, благодаря чему длинное выражение может поместиться в очень узком прямоугольнике.

5.3.2 Единство стиля

Последняя из рассматриваемых проблем связана со стилями форматирования математических выражений и окружающего HTML-текста. Для того чтобы математическое выражение хорошо гармонировало с текстом, следует использовать одни и те же шрифты (или, по крайней мере, хорошо сочетающиеся друг с другом) и одинаковые цвета шрифтов и фона. Более того, если на HTML-странице используется фоновый рисунок, то такой же рисунок должен присутствовать и в прямоугольной области воспроизведения математического выражения и, более того, он должен быть правильно совмещен с фоном страницы. Некоторые из этих проблем проиллюстрированы на рис. 5.20.

Информация о стиле HTML-страницы была задана в каскадном стиле:

```
P {
  font-family:  sans-serif;
}

BODY {
  background: silver;
}
```

Обратите внимание на то, что математическое выражение воспроизводится на белом фоне шрифтом с засечками (Roman).

Как `techexplorer`, так и WebEQ воспроизводят математические выражения на экране на *непрозрачном фоне*: новое изображение закрывает все, что было в окне раньше, включая и фон. Хотя в принципе можно создать такие динамически подключаемые модули, которые будут выводить информацию на экран поверх существующего изображения, оставляя его видимым, это не снимет проблем, связанных с цветами шрифта и текста. Более того, оказывается недостаточным задать прямо в разметке математических выражений цвет фона и информацию о шрифте, хотя на экране все может выглядеть правильно. Динамически подключаемый модуль или апплет должны автоматически получать эту информацию непосредственно от навигатора. В противном случае при каждом обновлении стиля HTML-страницы вам надо будет входить внутрь и отдельно настраивать каждое математическое выражение.

Будущие навигаторы будут использовать информацию о едином стиле совместно с программным обеспечением, воспроизводящим внедренное содержимое. По мере перехода от HTML к документам XML и создания новых программных интерфейсов для средств визуализации внедренного (возможно, с несколькими уровнями вложения) содержимого присущие навигатору средства форматирования и соответствующая информация будут становиться все более доступными для другого программного обеспечения, используемого для воспроизведения фрагментов документа.

HTML, SGML и XML: три языка разметки

Эта глава освещает взаимосвязи между языком SGML (Standard Generalized Markup Language — Стандартный язык обобщенной разметки), который является прародителем всех существующих в настоящее время открытых языков разметки, языком HTML (HyperText Markup Language — Язык гипертекстовой разметки), представляющим собой сейчас лингва-франка — универсальный язык Сети, а также языком XML (Extensible Markup Language — Расширяемый язык разметки) — упрощенным вариантом языка SGML, используемым в качестве сердцевины целого семейства новых приложений, оптимизированных для применения их в Сети.

Вначале мы объясним, почему не считаем HTML окончательным ответом на вопрос о требуемом средстве информационного обмена в Сети. Затем несколько слов будет сказано об SGML и его истории, чтобы можно было понять контекст последних предложений по языку XML, прежде чем перейти к подробному ознакомлению с XML. Будут описаны различные компоненты этого языка, включая Определение типа документа (DTD— Document Type Definition), а также обзор некоторых из существующих средств работы с документами на языке XML.

6.1 Уйдет ли HTML из Сети?

Популярность HTML можно объяснить присущей ему простотой (он прост в освоении), а также тем, что имеется много *нестандартных* расширений для него, которые предлагаются поставщиками навигаторов и помогают пользователям готовить их Web-страницы так, чтобы они выглядели *профессионально* и *привлекательно*. Однако эта вавилонская башня расширений данного языка, несовместимых друг с другом, представляет собой реальную угрозу для целостности Сети, поскольку разрушает принцип универсальной доступности информации.

Многим HTML нравится потому, что это небольшой, простой и ясный язык, который можно освоить буквально за несколько часов. Этот язык универсален и исполняется в навигаторах повсеместно. Кроме того, множество инструментов включает драйверы для HTML. В *реальном мире*, однако, часто возникают неприятности, связанные с разорванными ссылками, с отсутствием мобильных способов форматирования информации. Почти любому из нас приходилось употреблять такие элементы, как таблицы, фреймы, скрипты на Java и других языках, чтобы получить внешний вид страниц таким, как нам этого хотелось бы, причем это было обусловлено отсутствием подходящего инструмента для формирования Web-страниц, которые отображались бы адекватно везде, где это нужно.

Имеет, по-видимому, смысл бросить взгляд на те проблемные области, где, по нашему мнению HTML можно было бы улучшить.

- *Некорректные HTML-тексты.* Многие общеупотребительные программы допускают ошибки в HTML или же пользуются расширениями данного языка, введенными разработчиками соответствующих программ. Большинство пользователей не проверяют правильность исходного HTML-текста, а навигаторы не возражают против некорректного HTML; обычно они пропускают ту информацию, которую не могут признать грамматически корректной. Эти обстоятельства делают особенно трудным обеспечение согласованности результатов, получаемых при обработке данного HTML-текста различными навигаторами на различных компьютерных платформах.
- *Разорванные ссылки.* При удалении некоторой Web-страницы или перемещении ее на другой компьютер все URL-ссылки на нее становятся неверными. Уже какое-то время ведутся разговоры об *Унифицированных именах ресурсов* (URN— Uniform Resource Names, см. разд. 1.1.2 и [↪URNIETF]), которые позволяли бы адресовать Web-страницы по их именам, что дает возможность перейти на другой уровень выполнения операции переадресации, сводя ее к построению отображения имен на таблицу физических адресов (как это делают серверы имен для интернетовских адресов).
- *Фиксированная грамматика.* Наборы элементов и атрибутов в HTML фиксированы. Поэтому можно сказать про HTML, что он имеет жесткую грамматику, как это описано в *Определении типа документа* (DTD), представляющем собой формальную спецификацию, которая описывает синтаксис некоторого SGML-приложения (см. разд. 6.3.2 и 6.4.3). Из этого вытекает, что нет возможности адаптировать язык с целью улучшить его соответствие новому специфическому набору прикладных проблем или же для расширения его функциональных возможностей, чтобы можно было иметь дело с новыми Web-технологиями, не прибегая к расширению DTD. В последнее время разработчики навигаторов добавили к HTML ряд их собственных расширений, и в результа-

те Web-страницы оптимизированы для некоего единственного навигатора, а остальные не в состоянии из-за этих расширений отображать информацию полностью. Сравнительно недавно с помощью спецификации CSS (Cascading Style Sheets — Каскадные стили) появилась возможность «расширить» визуальное представление жесткого набора элементов HTML с использованием атрибута `class` и элементов типов `span` и `div` (см. разд 7.4.1.4).

- *Ограниченная поддержка метаданных.* Для метаданных существует лишь примитивная поддержка — информация, описывающая содержание документа, в частности, ключевые слова, автор и дата создания. Элемент `<meta>` — это шаг в верном направлении, но стандартного пути для его использования не существует; пользовательские агенты могут просто игнорировать его. Следовательно, для поисковых машин остается нерешенной нетривиальная задача извлечения важной ключевой информации из исходного текста документов.
- *Отсутствие структурных элементов.* Хотя ряд элементов HTML, таких как `<h[1-6]>`, `<div>` и `<p>`, можно было бы использовать для структуризации информации, большинство Web-приложений и Web-авторов пренебрегают этими возможностями и используют элементы HTML только для того, чтобы управлять визуальным представлением документа. Этот неструктурированный подход затрудняет навигацию по дереву (или сети) документов.
- *Трудности с обменом данными.* Вследствие замкнутости набора элементов, нацеленного на представление информации в Сети, почти невозможно извлечь данные, соответствующие размеченным полям данных. Более того, доступен обычно лишь набор символов Latin 1, который не поддерживает полностью даже западноевропейские языки. Использование HTML для других языков основывается на его расширениях, что затрудняет обмен документами. Подумайте только, как же будут выглядеть тогда страницы на русском или японском языке, если ваш навигатор не обеспечивает поддержку кириллицы или каны, или же не располагает правильными кодировками.
- *Отсутствие современных возможностей.* Как это имеет место для любых стандартов (даже тех, что находятся в ведении Web Consortium, довольно быстро откликающегося на изменения в сложившейся практике), язык HTML не включает в себя многих *современных* ингредиентов. Среди них — способы обновления информации на клиентской машине, выявление информации, содержащейся в динамических объектах, таких, как апплеты, отсутствие объектной модели.

Был выполнен ряд работ, призванных помочь преодолеть отмеченные недостатки. Один из подходов — повышение функциональных возможностей языка

HTML, что нашло свое отражение в создании языка HTML 4. Чтобы лучше отделить друг от друга форму и содержание документа, был разработан и рекомендован для использования язык стилей страниц, получивший наименование *Cascading Style Sheets* (CSS). Работы по языку XML (*Extensible Markup Language*) ориентировались на улучшение учета специфики приложений, а также на улучшение организации данных. В динамическом HTML (DHTML) реализуется еще один подход к введению средств динамического представления в Web-страницы. Стандартизованная независимая от языка продукта версия DHTML получила наименование *Document Object Model*, или, кратко, DOM. DOM призвана играть важную роль, позволяя программам получать доступ к элементам языка HTML (XML) как к структуризованным наборам объектов, каждый из которых имеет набор присущих ему свойств и методов.

В этой и следующих главах мы рассмотрим некоторые из упомянутых разработок. При этом следует иметь в виду, что многие новые средства в существующем поколении навигаторов реализованы лишь частично. Пройдет еще какое-то время, прежде чем хотя бы основные из навигаторов станут отвечать тем спецификациям и стандартам, которые будут описаны ниже.

6.2 HTML 4: более богатый и гармоничный язык

Описание языка HTML 4.0 выпущено комитетом W3C (World Wide Web Consortium) 18 декабря 1997 г. в виде Рекомендаций W3C, что примерно эквивалентно по статусу в рамках Сети стандарту ISO или ANSI. Спецификация HTML 4 представляет собой документ объемом свыше 360 страниц и доступна как файл в формате HTML, PostScript или PDF по адресу [↪HTML4][1]. Хотя в языке HTML 4 часть из перечисленных в разд. 6.1 недостатков преодолена, он все еще предлагает фиксированный набор элементов, в нем также отсутствует возможность подстройки языка разметки применительно к потребностям конкретного реализуемого приложения. Поэтому HTML 4 — последний из семейства стандартов HTML; в дальнейшем усилия будут концентрироваться на формировании более модульного подхода, основанного на концепциях языка XML (см. разд. B.5).

[1] Вполне возможно, что та спецификация HTML 4, которую вы найдете по адресу URL, будет слегка отличаться от рассматриваемой здесь. Когда писалась эта книга, ссылочный номер соответствующего документа был `REC-html40-19980424`.

6.2.1 Преимущества HTML 4

Ниже рассматриваются некоторые из наиболее важных изменений, появившихся в языке HTML 4 в сравнении с его предыдущей версией 3.2 (выпущенной в январе 1997 г.):

- Реализована более полная модель для таблиц (основанная на CALS[1] DTD).
- Сделан первый шаг в направлении более четкого разделения содержания и формы на фоне неодобрительного отношения к элементам и атрибутам, которые управляют параметрами представления документа (например, цвет, размер шрифта), с заменой этих элементов на элементы, удовлетворяющие требованиям CSS (*Cascading Style Sheets*) (см. *CSS Specification* [↪CSS2] и гл. 7).
- Любой элемент может быть идентифицирован с помощью (уникального) атрибута ID. Он может адресоваться как параметр привязки для ссылки, как это можно видеть из следующего примера:

```
<H2 id="mysect">Это уникально идентифицируемый заголовок раздела.
<P id="mypara">Это мой адресуемый пункт.
...
<P>Как показано в  <A HREF="#mypara">пункте</A>,
представляющем собой часть  <A HREF="#mysect">раздела</A>...
```

- Осуществляется поддержка многоязычной среды благодаря введению таблицы кодов языков (см. табл. C.1 на с. 527) и возможности указывать направление письма, что позволяет упростить формирование документов практически на любом из языков мира. Разумеется, обеспечить универсальную читаемость документа — весьма непростая задача для программного обеспечения.
- Появилась возможность встраивать с помощью элемента OBJECT объекты общего характера (рисунки, апплеты, другие документы). Если заданный ресурс недоступен, тогда вместо него можно задать некоторый заменяющий. В приводимом ниже HTML-коде вначале предпринимается попытка исполнить апплет на языке Python, демонстрирующий движение электронов в ускорителе LEP (строки 1–2). Если это невозможно, взамен производится попытка показа этого же сюжета в виде фильма в формате MPEG (строка 3) или же в виде статического рисунка в формате GIF (строка 4).

[1] CALS обозначает *Continuous Acquisition and Life-Cycle Support* (непрерывная поддержка процесса приобретения и жизненного цикла) и представляет собой стратегию, выработанную Министерством обороны США для обеспечения эффективного порождения, обмена и использования цифровых данных в оборонных системах и в различных видах оборудования. Стратегия CALS, принятая еще, по крайней мере, полудюжиной оборонных предприятий США, является движущей силой продвижения SGML в качестве языка разметки общего назначения. Более подробную информацию можно получить на домашней странице CALS в [↪CALS].

Наконец, если ни один из этих вариантов не удалось реализовать, печатается текстовая строка (строка 5).

```
1 <OBJECT title="Electrons going round and round"
2         classid="http://www.cern.xxx/CirculatingElectrons.py">
3   <OBJECT data="CirculatingElectrons.mpeg" type="application/mpeg">
4     <OBJECT data="CirculatingElectrons.gif" type="image/gif">
5       Electrons circulating in the LEP tunnel.
6     </OBJECT>
7   </OBJECT>
8 </OBJECT>
```

Отметим, что элемент типа `OBJECT` заменяет (и тем самым способствует исключению из употребления) такие элементы, как `APPLET` и `IMAGE`. Это позволяет более общим способом и значительно более аккуратно работать с событиями, файлами, просмотром рисунков и т. п.

- Введен также ряд более перспективных элементов. В их число входят *элементы носителей*, которые позволяют использовать аппаратно-зависимые стили, а также *атрибуты событий*, которые в сочетании со сценариями позволяют исполнять некоторый программный код в качестве реакции на появление определенного события (таким событием может быть, например, загрузка документа, щелчок клавиши мыши и т. п.). К числу нововведений принадлежат также элементы `DIV` и `SPAN`, которые при использовании их вместе с атрибутами `ID` и `CLASS`, а также стилями, дают в распоряжение авторов механизм для «расширения» HTML путем подгонки его под их потребности и вкусы.

К настоящему времени нет навигаторов, которые полностью реализовывали бы HTML 4, хотя программы Netscape Communicator версии 4 и MS Internet Explorer версий 4 и 5 идут уже именно в этом направлении. Более того, чтобы получить полную отдачу от потенциальных возможностей HTML 4, необходима поддержка Каскадных стилей (CSS версии 2). С навигаторами еще предстоит много поработать, чтобы добиться соответствия их указанным требованиям.

Описание и сопоставление различных версий языка HTML было выполнено Яном Грэмом с использованием модифицированной версии программы `dtd2html`, разработанной Эрлом Худом, и программы `dtddiff` (см. разд. 6.6.2). Этот документ можно найти по адресу [↪HOOD].

6.2.2 HTML 4 — конец старого пути

22 июня 1998 г. Консорциум W3C выступил с заявлением, касающимся перспектив языка HTML [↪W3CFUTURE]. Это заявление обсуждалось специалистами в ходе ряда рабочих совещаний. Было общее ощущение, что надо начинать разработку языка HTML нового поколения, основанного на полноценном наборе элементов XML и модульном подходе. Это позволило бы получить язык, более удобный для использования, обеспечивающий непосредственную интеграцию HTML и уже существующих приложений на XML. Для получения

более подробных сведений относительно XHTML — языка, который реализует переформулирование HTML как XML-приложения, — см. разд. B.5.2.

6.3 Почему SGML?

С начала 80-х годов мы стали свидетелями все убыстряющегося процесса перехода от книгоиздания исключительно в его «бумажной» форме к различным видам электронного представления публикуемых документов. Эта эволюция — очевидное следствие проникновения компьютеров и электроники буквально во все сферы человеческой деятельности. В мире, где нам приходится иметь дело со все возрастающими объемами данных, мы во все большей степени зависим от компьютеров для подготовки телефонных справочников, словарей, юридических текстов — и это лишь небольшая часть возможных примеров. Важен, однако, не столько объем хранимых и обрабатываемых данных сам по себе, сколько простота и удобство доступа к данным, манипулирования ими, просмотра, обмена и распространения данных.

Как только данные сохранены в электронной форме, возникает возможность порождения многих видов одного и того же исходного документа. К примеру, список адресов можно оформить в виде справочника, распечатанного на бумаге, поместить на CD-ROM, занести в базу данных и сделать его доступным в диалоговом режиме или через электронную почту в Интернете, или же печатать адреса из этого списка на конвертах. Аналогичным образом, набор юридических текстов или серию статей по истории, промаркированных с помощью общего языка разметки, можно опубликовать в виде книги, в которой содержатся в полном объеме указанные юридические тексты, или же использовать в качестве основы для подготовки исторической энциклопедии. Благодаря стратегии обобщенной разметки, непосредственно обеспечивается регулярное обновление текстов или извлечение подмножества статей по заданной теме. На основе тех же самых электронных источников может быть организован также консультационный сервис в Интернете, через службу Gopher или через WWW, или же разработана гипертекстовая система, размещенная на CD-ROM.

Во всех перечисленных вариантах мы исходили из предположения, что используемая в них информация не хранилась в виде, пригодном лишь для отображения на экране дисплея или для печати (например, посредством некоторой WYSIWYG-ориентированной системы). Иерархическая структура документа и логические взаимосвязи между различными его компонентами должны быть четко и ясно обозначены. Этот подход, базовый для LaTeX'а и языков на основе SGML/XML, имеет следующие преимущества:

- Улучшается качество рассматриваемого исходного документа благодаря упрощению доступа к данным, повышению удобочитаемости, более полному контролю корректности ввода.
- Работу с документами можно организовать на более рациональной основе, в результате чего улучшается жизненный цикл документа.

- Снижаются издательские расходы.
- Упрощается повторное использование информации, что увеличивает ценность документов (их можно печатать, представлять в виде гипертекста, запоминать и использовать в качестве содержимого баз данных).

6.3.1 Различные типы разметки

Сейчас на каждом компьютере есть какой-нибудь текстовый процессор, чаще всего относящийся к классу WYSIWYG (What You See Is What You Get — что видите, то и получаете). Из-за этого многие пользователи постоянно путают информацию и структуру документа с его отображением посредством соответствующих форматных характеристик для различных компонент текстового документа.

Данная ситуация похожа на ту, что преобладала в 70-х гг. применительно к ранним языкам форматирования; специфические коды смешивались с текстом документа для управления выводом его на печать на микроуровне. Например, для формирования страниц специально указывались разрывы строки и страницы, явное выравнивание по вертикали или горизонтали, пропуск строк. Большая часть из этих управляющих символов была сильно связана с конкретными прикладными программами, поэтому повторное использование исходного текста, размеченного для одной системы, в рамках другой системы практически исключалось. Однако разметка такого типа позволяла управлять с очень высокой точностью физическим представлением конкретного документа и давала важные преимущества при чистовой отработке документа в его окончательном варианте для просмотра на экране дисплея или для печати. Ниже дается фрагмент TEX'овского текста в качестве примера разметки конкретной страницы. Он начинается с новой страницы, название главы в нем печатается жестко заданным способом (полужирными буквами большого размера, при этом слова «Chapter» (Глава) и номер «2» вводятся вручную).

```
\vfil\eject
\par\noindent
{\large\bf Chapter 2: Title of Chapter}
\par\vskip\baselineskip
```

Должно быть ясно, что модификация какого-либо документа, содержащего такую явную разметку, или попытки оценить «значение» рассматриваемых команд будут сопряжены со значительными трудностями. Следовательно, если планируется вносить изменения в документ или же использовать его не единственным способом, размечать его надо так, чтобы его логическая структура и физическое представление четко разделялись.

В случае *логической* или *обобщенной* разметки логическая функция всех элементов документа — заголовка, разделов, абзацев, рисунков, таблиц, библиографических ссылок или математических формул, а также структурные взаимосвязи между ними должны быть четко и ясно определены. LATEX представляет собой огромный шаг в этом направлении, как можно видеть из по-

казанного ниже фрагмента исходного текста, где начало новой главы задается единственной строкой:

```
\chapter{Title of Chapter}
```

При таком подходе сведения о том, каким образом должна начинаться глава и какой вид должен иметь ее заголовок, содержатся в заранее подготовленной спецификации «стиля» (или «класса») документа. В то же время, способ нумерации глав, ввод наименования главы в состав оглавления, если это требуется, находятся под общим контролем автора соответствующего стиля. Нам остается только задать логическое наличие главы и указать ее заглавие.

Аналогичный подход можно видеть также и в следующем фрагменте HTML-текста, хотя надо подчеркнуть, что HTML в настоящее время используется преимущественно как язык разметки визуальных представлений документов. Например, в показанном ниже примере элемент <H1> обычно используется только для маркирования присутствия «заголовка уровня 1» без порождения начала некоторого нового раздела в данном документе.

```
<H1>Title of Chapter</H1>
<P>
```

Только недавно посредством использования каскадных стилей CSS в HTML стал доступен более высокий уровень абстракции для отношений между элементами разметки и визуальным представлением. (Эта концепция будет объяснена в гл. 7; см., в частности, разд. 7.4.)

6.3.2 Обобщенная логическая разметка

Некоторое количество документов может принадлежать к одному и тому же «классу», поскольку они имеют одну и ту же глобальную логическую структуру. В качестве примера рассмотрим две статьи — A и B — с явно заданной структурой, показанной на рис. 6.1.

Из рис. 6.1 видно, что обе статьи структурно построены по одному и тому же логическому шаблону: заголовок, за которым идут один или несколько разделов (section), каждый из которых может состоять из нескольких подразделов (subsection), а также списка литературы (bibliography). В LaTeX'е можно было бы сказать, что данные образцы документов относятся к одному и тому же *классу документов* article (статья).

Чтобы полностью реализовать преимущества структурированных документов, схема разметки должна задаваться неким четким набором правил. Тогда при использовании SGML эти правила фиксируются в виде Определения типа документа DTD. Это определение не только задает допустимые типы элементов (синтаксис), но и описывает также структурные отношения между этими элементами. Программа разбора проверяет затем, соответствуют ли документы с разметкой требованиям DTD.

```
      Article A                Article B
      =========                =========
Title                    Title
Section 1                Section 1
   Subsection 1.1           Subsection 1.1
   Subsection 1.2           Subsection 1.2
Section 2                   Subsection 1.3
Section 3                Section 2
   Subsection 3.1           Subsection 2.1
   Subsection 3.2           Subsection 2.2
   Subsection 3.3
   Subsection 3.4
Bibliography             Bibliography
```

Рис. 6.1. Два экземпляра класса article.

Данный подход напоминает тот, что принят в LaTeX'е, где синтаксис (принятая разметка документа) задается в описании LaTeX'а, приведенном в книге Лесли Лэмпорта; однако в реализациях TeX'а, являющегося для LaTeX'а базовым уровнем, проверка «допустимости» используемой иерархии (вложенности) элементов в большинстве случаев не производится. Например, можно использовать подподраздел (subsubsection), не задавая явно элементы предыдущих уровней — раздел и подраздел, совсем как в HTML. Если определение типа документа DTD построено корректно, этот вид двусмысленности в SGML допускаться не должен. Кроме того, в LaTeX'е можно расширять используемый язык, определяя новые команды и окружения; это также невозможно в случае SGML.

Следовательно, для наших примеров документов типа «статья» спецификация DTD должна определять элементы «заголовок» («title»), «раздел» («section»), «подраздел» («subsection») и «библиография» («bibliography»). Она должна также выражать тот факт, что заголовок предшествует разделам, которые могут содержать подразделы, а также что статья может включать библиографию, располагающуюся в конце статьи. Рассматриваемый DTD приписывает некоторое имя каждому из структурных элементов, часто это фрагмент слова или аббревиатура, отвечающая функциям соответствующего элемента (например, «`sec`» для раздела (section) в строке 3 и «`stit`» для заголовка раздела или подраздела (section title) в строках 3 и 5). Используя DTD, можно теперь начинать разметку собственно исходного текста документа (статьи A или статьи B), применяя при этом «короткие» имена, введенные для каждого из элементов документа. В частности, имея элемент «`sec`», можно использовать открывающую *метку* `<sec>`, отмечающую начало раздела (строка 3), а также метку конца раздела `</sec>` (строка 8); другие компоненты документа действуют похожим образом (см. разд. 6.4.2, где эти вопросы освещены более подробно).

```
1  <article>
2  <tit>SGML and XML</tit>
3  <sec><stit>Why SGML?</stit>
```

```
4       <para>  ...             </para>
5     <ssec><stit>Different types of markup</stit>
6       <para>  ...             </para>
7     </ssec>
8   </sec>
9   </article>
```

6.3.3 Отношения между SGML в HTML и XML

Идея обмена структурированными документами и манипулирования ими посредством публикации их в стандартном открытом формате появилась еще в начале 60-х гг. В частности, одним из комитетов Ассоциации графических коммуникаций (GCA) была предпринята попытка создания формата GenCode для разработки печатающих программ общего назначения, позволяющих клиентам издательских фирм пересылать данные, которые затем можно было бы вывести с помощью различных печатающих устройств. GenCode давал возможность этим пользователям работать с интегрированным набором архивов, несмотря на то что физически требуемые записи могли находиться в различных местах.

Еще одну попытку предприняла фирма IBM, которая разработала Обобщенный язык разметки (GML). Этот язык понадобился IBM из-за значительных издательских проблем внутреннего характера, с которыми столкнулась фирма, — издание и последующее сопровождение документации всех видов, от руководств и сводок новостей до юридических документов и проектных спецификаций. Язык GML разрабатывался в расчете на режим пакетной обработки при подготовке книг, отчетов и электронных вариантов соответствующих изданий на базе одних и тех же файлов.

Язык GML обеспечивал «простой» формат ввода для машинисток, печатающих документ, включал элементы тэгового синтаксиса, который вполне актуален и сегодня. Этот язык был оптимизирован для максимальной скорости ввода данных. Кроме того, поскольку существующее число типов документов было невелико, разрабатывались отдельные программы для интерпретации элементов данных, отвечающих каждому из этих типов документов — едва ли такой подход можно признать достаточно общим. Поэтому, когда число требуемых типов документов стало увеличиваться, сообщества, поддерживающие GenCode и GML, пришли к соглашению о начале совместной работы по стандартизации используемых средств в рамках Американского национального института стандартов (ANSI). Идеи DTD, разметки, синтаксиса были формализованы, и на этой основе родился язык SGML. Он был принят в качестве стандарта ISO — Международной организацией по стандартизации — в 1986 г. (ISO:8879, 1986).

SGML — сложный стандарт, и его использование оставалось ограничено главным образом сферой крупных фирм и организаций, а также несколькими исследовательскими институтами. В частности, Андерс Берглунд, в течение многих лет отвечающий за организацию текстовой обработки в CERN'е — Европейской лаборатории физики элементарных частиц (Женева, Швейца-

рия), ввел там в употребление предварительную версию языка SGML (основанного на языке GML фирмы IBM) за много лет до того даже, как был опубликован стандарт на данный язык. В связи с этим у нас уже не вызывает удивления, что, работая в CERN'е, изобретатель Сети Тим Бернерс-Ли оказался под влиянием *буквы и духа* SGML, когда разрабатывал язык HTML. Его основной целью было (по крайней мере первоначально) не следовать формально определению DTD, а обеспечить, чтобы его ранняя программа-навигатор воспроизводила исходный материал простым и очевидным образом.

Прошло несколько лет до того момента, как разработчики HTML, в частности Дэн Конноли и Дейв Раггет, обнаружили необходимость дать HTML более прочную (и более формальную) основу. Было разработано первое определение DTD для HTML, и язык HTML превратился в приложение SGML, обеспечивая возможность контроля корректности HTML-документов формально, в соответствии с HTML-моделью, представленной посредством DTD?

Язык HTML в его современном варианте HTML 4, который обсуждается кратко в разд. 6.2, до сих пор предлагает лишь ограниченный набор элементов для разметки документов. Должно быть также ясно, что HTML, даже расширенный, никогда не будет в состоянии представлять все документы, которые люди захотят хранить в Сети или в электронной форме вообще.

С другой стороны, поскольку SGML имеет довольно большую и сложную спецификацию, автору не слишком просто ориентироваться во всех его деталях, а для компьютерных программ — осуществлять грамматический разбор и обработку сложных SGML-документов. Тем не менее, если мы хотим обращаться с электронными документами неким оптимальным образом, то *должны* размечать их достаточно общим образом, чтобы показать логическую структуру документа. Необходимо также выбрать для такого рода обработки некоторый согласованный стандартный язык, чтобы гарантировать возможность взаимного обмена текстами. Что же можно сделать, понимая что HTML слишком ограничен по своим возможностям, а SGML слишком сложен и не располагает согласованным набором «стандартных» общедоступных DTD?

В работе *XML, Java, and the future of the Web* [↪BOSAKXML] Ион Босак выявил три области, где HTML страдает серьезными недостатками: *расширяемость* (способность определять новые элементы и атрибуты для конкретных документов), возможность глубокого *структурирования* (способность допускать элементы произвольной глубины вложенности) и *проверка достоверности* (контроль корректности данных до их использования).

6.4 Расширяемые языки разметки

Хотя HTML 4 и представляет собой, вне всякого сомнения, шаг в правильном направлении, если ставить целью обеспечение стандартизации работы в Сети, этот язык остается все же слишком ограниченным по своим возможностям и слишком статичным, чтобы отвечать всем потребностям тех приложений, которые необходимы для обеспечения работы Сети (базы данных, поисковые

машины, оптимальное представление документов, профессиональная печать, верификация данных).

По этой причине в середине 1996 г. Web Consortium образовал рабочую группу по SGML, призванную обеспечить решение указанных проблем. Под руководством Иона Босака данная группа разработала язык XML — расширяемый язык разметки. Эта работа была завершена к концу 1997 г. и опубликована в качестве Рекомендаций W3C 10 февраля 1998 г. под наименованием *Extensible Markup Language (XML) 1.0* [↪XMLSPEC]; см. также список ошибок и опечаток в этом документе [↪XMLERRATA]. Язык XML разработан как подмножество языка SGML;[1] так что любой XML-документ будет также и SGML-документом. Существующие синтаксические анализаторы и системы для SGML могут, вообще говоря, использоваться и с XML. В настоящее время идет работа по расширению набора гипертекстовых возможностей данного языка (см. *XML Linking Language (XLink)* [↪XLINKSPEC] и *XML Pointer Language (XPointer)* [↪XPTR]) и по формированию языка определения стилей для XML, именуемого XSL (см. *Extensible Stylesheet Language XSL* [↪XSL97]). Язык XSL будет рассмотрен в разд. 7.6.

Хорошим источником информации об XML может служить работа Питера Флинна Frequently Asked Questions about the Extensible Markup Language [↪XMLFAQ]. К числу других интересных Web-сайтов относятся: Cafe con Leche XML News and Resources Э. Р. Харольда [↪LECHE] и SGML and XML News [↪SGMLNEW], который ведет Робин Ковер, координирующий также страницу Extensible Markup Language (XML) [↪XMLPAGE]. Это очень полезная страница, собирающая много информации, относящейся к XML. Ссылки-указатели на ряд статей вводного характера по XML доступны по адресу [↪XMLINTRO]. Эти статьи интересны тем, кто только приступает к изучению и использованию XML. Тем, кто до сих пор больше любит читать книги, можно порекомендовать несколько книг по XML, опубликованных в последнее время. Среди тех книг, которыми мы пользовались, были Bradley:1998:TXC, Goldfarb:1998:XH, Harold:1998:XM, Jelliffe:1998:XSC, Leventhal:1998:DXI, McGrath:1998:XEB, Megginson:1998:SXD и StLaurent:1997:XP[2].

6.4.1 Что такое XML?

Консорциум W3C обнаружил, что область применения языка SGML очень широка, а сам язык весьма сложен как для изучения, так и для реализации. В связи с этим W3C было принято решение создать более простую («облегченную») версию данного языка — язык разметки XML, разработанный как подмноже-

[1] Потребовалось некоторое число добавлений к первоначальному варианту стандарта ISO на язык SGML, выпущенному в 1986 г. (ISO:8879:1986). Эти добавления собраны в работах Annex K (*Web SGML Adaptations*) и Annex L (*Additional Requirements for XML*), см. [↪ISO8879TC2].

[2] Число книг по XML, выпущенных за 12 месяцев, намного больше, чем число книг по SGML, опубликованных за последние 12 лет, — еще одно доказательство важности разработки XML.

ство языка SGML за счет удаления из него редко используемых или слишком сложных элементов. Утверждается, что язык XML унаследовал от SGML около 90% его функциональных возможностей, в то время как по сложности он оценивается как 10% от сложности SGML. Таким образом, можно утверждать, что *«десять заповедей»* языка XML (его *проектные цели*, установленные Специальной группой интересов W3C SGML в начале ее деятельности) вполне удовлетворены.

6.4.1.1 Десять заповедей XML

Спецификация для языка XML разрабатывалась исходя из следующих проектных целей:

1. *Язык XML должен быть пригоден для простого и удобного использования во всей сети Интернет.*

   Надо, чтобы документы, написанные с помощью XML, можно было просматривать так же быстро и просто, как HTML-документы (если есть в наличии навигаторы и прикладные программы для XML).

2. *XML должен поддерживать широкую совокупность приложений.*

   XML должен оптимизироваться не только для просмотра документов, он должен быть ориентирован на более широкую совокупность приложений, не ограничиваться лишь рамками Сети. Соответствующие прикладные области должны включать: процессы авторской работы при подготовке документов, представление данных, анализ и оценку корректности содержимого документов, базы данных.

3. *XML должен быть совместим с SGML.*

   Поскольку XML изначально был задуман людьми, исторически вовлеченными в работу по SGML, как «удобное» подмножество языка SGML , мы можем быть уверены, что взаимодействие между XML и SGML будет осуществляться без каких-либо затруднений.

4. *Написание программ, обрабатывающих XML-документы, должно быть несложным.*

   Привлекая для экспертизы членов SGML-сообщества, которые хорошо знают, какие элементы SGML трудны для реализации и, следовательно, должны быть исключены, мы создаем уверенность, что язык XML может быть реализован «в течение нескольких недель» студентом среднего уровня подготовки, специализирующимся на информатике. Фактически, как это будет показано ниже, сейчас уже свободно доступны более десятка синтаксических анализаторов для XML.

5. *Число необязательных средств в языке XML должно быть сведено к минимуму, в идеале — к нулю.*

   Без особых разговоров ясно, что необязательные средства (функции) на самом деле необязательными не являются: как только кто-либо начинает пользоваться ими, кто-то должен реализовать программно интерпретацию этих средств. Следовательно, чтобы минимизировать несовместимости и неразбериху, необязательных средств в XML допускать нельзя, и тогда программы грамматиче-

ского разбора XML во всем мире будут в состоянии интерпретировать любой XML-документ.

6. *XML-документы должны быть удобочитаемыми и понятными.*
   Опыт показывает, что всегда намного лучше, если содержимое документа можно прочитать, не прибегая к услугам специальных программ для его декодирования. Для пользователя также было бы очень хорошо, если бы он мог отобразить документ непосредственно на экране дисплея или производить в нем какие-то мелкие изменения с помощью привычного редактора.

7. *Проект языка XML должен быть подготовлен быстро.*
   В прошлом разработчики навигаторов добавляли в свои программы средства, несовместимые со средствами других разработчиков. Консорциум W3C пытался отслеживать эти разработки, вводя в употребление все более и более сложные спецификации HTML. Несмотря на это, как мы уже видели ранее, реальные проблемы так и не были решены, и весьма важно было заняться этим вопросом *немедленно*. Рабочая группа XML за минимальное время проделала невероятную по объему работу.

8. *Проект языка XML должен быть формальным и лаконичным.*
   В п. 4 содержится пожелание, чтобы XML был прост для программирования. Здесь же мы утверждаем, что язык сам по себе должен быть прост для его формального описания, что позволяло бы анализировать его обычными компьютерными средствами, и в то же время прост в освоении для среднего по уровню подготовки разработчика документа.

9. *XML-документы должны быть просты для создания.*
   Интеллектуальные редакторы или специализированные прикладные программы не должны требовать слишком много времени для формирования корректных данных в формате XML. Необходимо также, чтобы можно было трактовать XML как авторское средство подготовки текстов.

10. *Лаконичность стиля разметки в XML наименее важна.*
    Чтобы упростить реализацию всех тех задач, которые упоминались выше, а также сделать тексты на XML более удобочитаемым (как для программы грамматического разбора, так и для человека), не допускается минимизация объема разметки. Ясность изложения всегда должна иметь более высокий приоритет, чем лаконичность.

6.4.1.2 XML открывает новое окно в Сеть

Из того, что уже было сказано выше, должно быть ясно, что цель разработки XML — пойти дальше, чем просто создать «супер-HTML» как «облегченный вариант SGML для Интернета». Язык XML открывает совершенно новое окно в Сеть: он дает возможность проектировщикам и программистам представлять нужные им данные различными способами, используя методы программирования, встроенные в стандартный синтаксис XML. В принципе, нет необходимости далее комбинировать различные методы и языки, такие, как HTML, Java, Perl, CGI, ActiveX, сценарии и другие методы и динамически подключаемые средства. Все они будут скрыты уровнем абстракции, принятым в XML.

Как уже отмечалось, каждый элемент *корректного* (valid) XML-документа доложен быть объявлен в DTD. Это обеспечивает получение некоторого формального определения для языка XML применительно к рассматриваемому классу документов. Такой подход позволяет программам грамматического разбора для XML контролировать корректность конкретных документов, размеченных в соответствии с данным DTD. Они могут проверять, например, правильно ли соотносятся уровни вложенности элементов документа, все ли требуемые для данного класса документов элементы определены в рассматриваемом конкретном документе и т. д. Отметим, однако, что, строго говоря, спецификация XML *не требует* наличия DTD. Применительно к навигаторам было бы слишком трудоемко загружать для каждого документа DTD и проводить для этого документа его грамматический разбор. В идеале для XML-приложений должна быть уверенность, что в момент их создания все документы соответствуют DTD, так что навигаторы в своей работе исходят из предположения, что все они грамматически корректны. XML требует лишь, чтобы документ был *«правильно оформленным»* (см. разд. 6.4.2.1).

6.4.2 Элементы языка XML

Язык XML основан на концепции *документов*, состоящих из последовательностей *сущностей* (сейчас их модно именовать *объектами*). Каждый объект включает один или более *элементов*, а каждый элемент может характеризоваться нулем или более *атрибутов* (свойств), которые описывают способ обработки каждого из элементов. Соотношения между элементами и списком их возможных атрибутов задаются в DTD.

Особое преимущество XML (SGML) состоит в том, что, используя этот механизм определения некоторого языка с помощью DTD, каждый институт, группа, фирма, организация и т. д. могут определить свой собственный *язык* для всех видов документов, с которыми им приходится иметь дело. Поскольку есть возможность выбрать понятные и удобные для пользователя *элементы разметки*, приспособленные к конкретной прикладной области или культурной среде, использование этих элементов будет много проще, а частота появления ошибок — много меньше, чем применительно к случаю более общих схем разметки. Более того, с помощью *интеллектуальных редакторов*, которые умеют скрывать разметку или показывают пользователю только элементы, допустимые в данном контексте, задача составления синтаксически корректных документов становится тривиальной.

Хотя *базовый синтаксис* SGML предлагает использовать некоторые символы в качестве разделителей и т. п., в принципе их можно выбирать свободно. Кроме того, на уровне документа могут быть определены различные параметры для конкретных текстов на SGML. Литерный набор (по умолчанию это ASCII) также может быть определен в конкретном документе. Эта сложность затрудняет написание программ грамматического разбора, и по этой причине было решено разрешить лишь фиксированный синтаксис элементов XML, ссылок на объекты и т. п.

Элементы и их атрибуты, если таковые необходимы, вводятся между парами согласованных скобок (<...>), а значения атрибутов *всегда* окружаются одинарными или двойными кавычками:

```
<ename attr1="val1" attr2='val2' ...>
```

Ссылки на объекты начинаются знаком & и заканчиваются точкой с запятой:

```
&eref;
```

Имена элементов, атрибутов и объектов должны вводиться с учетом регистра, т. е. метки `<HEAD>`, `<Head>` и `<head>` соответствуют трем различным типам элементов, а `<head lev="val" Lev="val2">` показывают два различных атрибута — `lev` и `Lev` — для элемента типа `head`. Ясно, что применять тонкие возможности надо с большой осторожностью, чтобы у пользователя не возникло путаницы при вашем наборе элементов. Простое правило, вроде «все буквы — строчные», может сделать ваш DTD менее подверженным ошибкам. Что касается объектов, в таком ограничении необходимости нет, поскольку, например, объекты `&Aacute;` и `&aacute;` могут порождать Á и á соответственно (в наборе объектов Latin 1).

Комментарии записываются между `<!--` и `-->`:

```
<!-- Внутри комментария можно писать <e>&</e> -->
```

По своему характеру XML представляет собой подмножество языка-родителя SGML, так что все программное обеспечение для грамматического разбора, проверки корректности или какой-либо другой обработки текстов на SGML должно также работать и с XML-файлами.[1]

Примером синтаксически корректного XML-текста может служить следующий тривиальный документ:

```
<coolxml>XML is a cool idea!</coolxml>
```

Этот XML-документ не может быть, по-существу, подвергнут проверке корректности, поскольку не задано DTD. Он, однако, правильно оформлен и полон.

6.4.2.1 Корректные и правильно оформленные документы

В конце разд. 6.4.1.1 отмечалось, что XML-документы должны быть *правильно оформленными*, однако они еще должны быть и *корректными*. Давайте рассмотрим эти вопросы более подробно.

Чтобы XML-документ можно было считать *правильно оформленным*, он должен отвечать набору простых правил, обеспечивающих XML-процессору

[1] Это строго верно только лишь для программ, подчиняющихся *Web SGML Adaptations* [↪ISO8879TC2].

возможность корректно выполнить грамматический разбор соответствующего файла.

Во-первых, должен существовать *корневой* элемент, который охватывает данный конкретный документ в целом и не появляется в качестве содержимого любого другого элемента (`docu` в строках 1 и 6 в примере, приводимом ниже). Все элементы должны быть сбалансированы, т. е. все элементы должны иметь как открывающий, так и закрывающий элемент (`<el1>...</el1>`), если это только не пустой элемент, для которого можно использовать нотацию `<.../>` (закрывающейся скобке предшествует косая черта, как, например, `<empe1/>` в строке 4, хотя допускается и более многословный синтаксис вида `<empe1></empe1>`). Для элементов должны соблюдаться правила вложенности (`el2` начинается и кончается внутри `el1` в рассматриваемом примере). Все значения атрибутов, которые обязательно должны быть типа `CDATA` (символьные данные), должны быть заключены в двойные кавычки (как в строке 2), а символы `<` и `&` следует записывать как `<` и `&` соответственно. Простой правильно оформленный документ имеет, например, следующий вид:

```
1  <docu>
2  ... <el1 att="val">...
3           <el2>...</el2>
4       ...<empe1/>
5       </el1>
6  </docu>
```

В разделе 6.6.5 мы приведем примеры документов, не являющихся правильно оформленными для различных программ грамматического разбора XML, и посмотрим, каким образом появляющиеся диагностические сообщения помогают устранять ошибки в документе.

Корректные XML-документы являются правильно оформленными и должны отвечать DTD. Корректный XML-документ должен задавать в разделе деклараций в начале конкретного документа, какое DTD следует использовать, а также, каким образом процессор XML может получить его в рамках локальной системы или через сеть. В примере, приводимом ниже, можно видеть, как декларируется тип документа `docu` путем задания некоторого URL:

```
1  <?xml version="1.0"?>
2  <!DOCTYPE docu SYSTEM "http://www.bla.org/mydocu.dtd">
3  <docu>
4  ... <el1>...<empe1/>...</el1>
5  </docu>
```

Некоторое DTD (полностью или частично) может быть также включено в состав раздела деклараций рассматриваемого документа. Если все содержится внутри данного конкретного документа, то можно объявить документ как standalone (автономный).

```
1  <?xml version="1.0" standalone="yes"?>
2  <!DOCTYPE coolxml [
```

```
<!ELEMENT coolxml (#PCDATA)>
]>
<coolxml>XML is a cool idea!</coolxml>
```

Таким образом, добавляя DTD, мы преобразуем правильно оформленный тривиальный документ, введенный в конце предыдущего раздела, в корректный документ.

Различие между правильно оформленными и корректными документами облегчает работу с XML-документами повсюду в Интернете, не обременяя этими функциями «пользовательские» прикладные программы (такие, как навигаторы и программы обработки запросов к базам данных). Клиентское приложение может исходить из того, что корректность документа проверена и обеспечена «серверной стороной». Тогда нам надо лишь удостовериться, что документ является правильно оформленным, т. е., как пример, не поврежден ли документ при передаче его по линиям связи. Разумеется, клиентское приложение также может выполнить проверку полученного документа. Здесь следует сделать одно важное замечание — XML-совместимые приложения должны выдавать сообщение о *серьезной* ошибке и прекращать обработку данных клиентским приложением, если обнаруживается, что полученный документ не является правильно оформленным, хотя можно и продолжить при этом поиск ошибок. Данный подход резко отличается от того, что принят в большинстве HTML-приложений, которые в настоящее время большей частью игнорируют ошибки в HTML-документах и обрабатывают их так, как будто в них никаких ошибок нет, что позже может привести к возникновению проблем.

6.4.2.2 Более сложный документ

Давайте теперь рассмотрим более сложную и интересную задачу — дадим определение языка, с помощью которого будем формировать тексты приглашений, посылаемых нашим друзьям. Можно было бы написать что-нибудь вроде[1]:

```
<invitation>
<to>Anna, Bernard, Didier, Johanna</to>
<date>Next Friday Evening at 8 pm</date>
<where>The Web Cafe</where>
<why>My first XML baby</why>
<par>
I would like to invite you all to celebrate
the birth of Invitation, my first XML document child.
</par>
```

[1] Этот пример неоднократно используется в гл. 6 и 7 при обработке различными средствами. Приведем его перевод:

```
Анне, Бернару, Дидье, Йоханне
Следующая пятница, 8 вечера
Интернет-кафе
Мой XML-первенец
Я имею честь пригласить всех вас отметить
рождение моего XML-первенца - документа Invitation.
Будьте столь любезны прийти в следующую пятницу
и разделить мою радость. Не забудьте взять с собой друзей.
Я с нетерпением жду встречи с вами!
```

— *Прим. ред.*

```
<par>
Please do your best to come and join me next Friday
evening. And, do not forget to bring your friends.
</par>
<par>
I really look forward to see you soon!
</par>
<signature>Michel</signature>
</invitation>
```

Этот документ размечен совершенно очевидным образом. Все его элементы ограничены парами из начальной и конечной меток, они вложены друг в друга соответствующим образом и существует *корневой* элемент самого верхнего уровня (`invitation`). Это значит, что наш элемент в самом деле является *правильно оформленным*. В этом документе остается, однако, по крайней мере один недостаток, а именно: структуру его трудно воспринимать и оценивать. Явным образом видна семантическая функция используемых текстовых строк, однако неясны отношения между различными элементами рассматриваемого документа.

Чтобы прояснить отношения между различными элементами данного документа, разделим его на три части: `front` (заголовок документа), `body` (тело документа — его основная часть) и `back` (завершающая часть документа), отвечающие, соответственно, вводной информации, тексту собственно сообщения и заключающему фрагменту документа. Выделим также несколько слов в данном тексте, заключая их между парами меток `<emph>...</emph>`. Кроме того, добавим некоторые комментарии.

```
<?xml version="1.0"?>
<!DOCTYPE invitation SYSTEM "invitation.dtd">
<invitation>
<!-- ++++ Заголовок документа ++++ -->
<front>
<to>Anna, Bernard, Didier, Johanna</to>
<date>Next Friday Evening at 8 pm</date>
<where>The Web Cafe</where>
<why>My first XML baby</why>
</front>
<!-- +++++ Основная часть документа +++++ -->
<body>
<par>
I would like to invite you all to celebrate
the birth of <emph>Invitation</emph>, my
first XML document child.
</par>
<par>
Please do your best to come and join me next Friday
evening. And, do not forget to bring your friends.
</par>
<par>
I <emph>really</emph> look forward to see you soon!
</par>
</body>
<!-- +++ Завершающая часть документа ++++ -->
<back>
<signature>Michel</signature>
</back>
</invitation>
```

Важно отметить, что до сих пор мы ничего не говорили о том, как обеспечить воспроизведение данного документа. Конкретный XML-документ, показанный выше, описывает лишь соответствующую информацию и отношения между ее составными частями. Каким образом некоторое XML-приложение должно обращаться с этими данными, никак не задано. Можно выполнить некоторое действие, когда в обрабатываемом документе встречается тот или иной элемент. Можно воспроизвести информационное содержимое данного документа в некоторой «выходной» среде, запомнить его в базе данных, преобразовать или же объединить с какими-либо еще данными и т. д. В главе 7 мы достаточно подробно изучим, каким образом языки описания стилей (CSS, DSSSL или XSL) позволяют преобразовывать XML-информацию в форму, необходимую для вывода на принтер или же для отображения на экране.

6.4.3 Декларирование элементов документа

В предыдущем примере был введен небольшой язык для разметки приглашений удобным, ясным и просто обрабатываемым образом. Если надо, чтобы XML-приложения проверяли корректность документов, оформленных согласно этой схеме, то следует дать формальное определение введенному языку. Как объяснялось ранее, это делается с помощью так называемого Определения типа документа DTD, которое формально определяет грамматику вводимого языка; другими словами, это определение описывает структурные отношения между элементами документа и их возможными *атрибутами*. Для случая нашего «языка приглашений» `Invitation` можно было бы задать такое определение для DTD:

```
1  <!-- invitation DTD   -->
2  <!ELEMENT invitation (front, body, back)>
3  <!ELEMENT front      (to, date, where, why?)>
4  <!ELEMENT date       (#PCDATA)>
5  <!ELEMENT to         (#PCDATA)>
6  <!ELEMENT where      (#PCDATA)>
7  <!ELEMENT why        (#PCDATA)>
8  <!ELEMENT body       (par+)>
9  <!ELEMENT par        (#PCDATA|emph)*>
10 <!ELEMENT emph       (#PCDATA)>
11 <!ELEMENT back       (signature)>
12 <!ELEMENT signature  (#PCDATA)>
```

Хотя синтаксис DTD описывается на специальном языке, не основанном на XML[1], он очень прост для применения и понимания (см. разд. 6.5.4, где содержатся дополнительные подробности). Дадим теперь словесное описание значений различных строк в приведенном выше описании. В строке 2 устанавливается, что любой документ, в котором используется элемент `invitation`,

[1] XML-Data [↪XMLDATA] определяет словарь для схем, который может быть использован для определения и документирования объектных классов XML и их отношений. На основе этой работы есть ряд предложений по описанию DTD-данных с использованием XML-синтаксиса: Описание информационного содержимого документа [↪DCD], Язык разметки определений документов [↪DDML], а также Схема для объектно-ориентированного XML [↪SOX].

всегда состоит из трех частей: `front`, затем `body`, а завершается документ частью `back`. Строка 3 определяет, что элемент `front` представляет собой последовательность из элементов `from`, `to`, `where` и, возможно, `why` (тот факт, что элемент `why` является необязательным, обозначается символом `?`). Можно еще ужесточить требования к структуре элемента `to`, если потребовать, чтобы все имена были бы отдельными элементами. В этом случае строка 3 принимает вид

```
<!ELEMENT front (to+, date, where, why?)>
```

и тогда можно было бы кодировать `<to>`-информацию таким образом

```
<to>Anna</to><to>Bernard</to><to>Didier</to><to>Johanna</to>
```

Можно было бы даже потребовать, чтобы элемент `to` в явном виде содержал подчиненные элементы для действий с именами (см. разд. B.4.4, где описывается DTD для BIBTEX). Тогда, заменяя строку 5 в первоначальном варианте DTD на

```
<!ELEMENT to      (name+)>
<!ELEMENT name    (#PCDATA)>
```

можно было бы кодировать `<to>`-информацию таким образом:

```
<to><name>Anna</name><name>Bernard</name>
    <name>Didier</name><name>Johanna</name></to>
```

И дело разработчика DTD решить, какой из подходов больше годится для того или иного приложения.

Продолжая производимый нами грамматический разбор DTD, обнаруживаем, что строка 8 говорит о том, что средняя часть (`body`) документа-приглашения состоит из одного или более абзацев (символ `+` означает *один или более*, тогда как `*` означает *нуль или более*). Согласно строке 9, каждый элемент `par` может содержать, в свою очередь, *синтаксически разобранные символьные данные* (`#PCDATA`) или же выделенный текст (помеченный элементами `<emph>`). Наконец, строка 11 завершающей части (`back`) содержит только элемент `signature` (подпись). Каждый из элементов в терминальных узлах структурного дерева рассматриваемого документа (`date` в строке 4, `to` в строке 5, `where` в строке 6, `why` в строке 7, `emph` в строке 10 и `signature` в строке 12) может содержать лишь `#PCDATA`. Такие данные анализируются XML-приложением, а также подвергаются проверке на разрешенность всех ссылок, содержащихся в них.

6.5 Детализированная структура XML-документа

Теперь, когда у нас есть представление о том, как выглядит и на что похож XML-документ, самое время более подробно обсудить структуру таких документов. Это значит, что необходимо рассмотреть различные компоненты, которые могут входить в конкретный документ, а также DTD.

Детальное комментированное изложение рекомендаций по XML содержится в *Annotated XML Specification* [↪AXML] Тима Брея. Этот документ представляет собой гипертекстовый вариант спецификации XML, дополненный исторической и технической аннотациями, а также примерами и советами по использованию. Знакомство с ним *обязательно* для тех, кто изучает XML.

Чтобы изложение было возможно более точным, мы часто будем ссылаться на спецификацию XML Specification Document, определяющую язык XML, используя 89 «продукций»[1].

6.5.1 XML истинно интернационален

С самого начала язык XML разрабатывался так, чтобы быть интернациональным и его можно было использовать для обработки документов везде в мире; это возможно благодаря тому, что данный язык базируется на стандарте Unicode или ISO/IEC 10646 (см. разд. C.2).

В самом деле, XML позволяет использовать любой символ Unicode. Он разделяет символьный набор Unicode на *буквы* (XMLPR[84]), *символы-комбинаторы* (различные виды диакритических символов, которые можно комбинировать с буквами, XMLPR[87]), *цифры* (римские, арабские, бенгальские и т. д., XMLPR[88]), а также на *символы-расширители* (такие, как центрированная точка (middle dot), XMLPR[89]).

Имена в языке XML начинаются с *буквы*, символа подчеркивания или двоеточия, за которыми могут следовать любая *буква*, *цифра*, точка, знак переноса, символ подчеркивания, двоеточие, *символ-комбинатор* или *символ-расширитель* (XMLPR[4-5]). Имена не могут начинаться с зарезервированной символьной строки «XML» (в прописном или строчном написании).

Язык, с помощью которого должны записываться части документа, может быть определен посредством *языкового кода* (XMLPR[33-38]). Для любого элемента документа можно определить (декларировать) специальный атрибут `xml:lang`, после чего его можно использовать в тексте документа. Пусть для элемента типа p определен атрибут следующим образом (см. разд. 6.5.4.2 для описания синтаксиса деклараций атрибутов в DTD):

```
<!ATTLIST p xml:lang  NMTOKEN  #IMPLIED>
```

В этом случае конкретный документ можно было бы записать примерно следующим образом:

```
<p xml:lang="en-GB">A favour, certainly!</p>
<p xml:lang="en-US">A favor, sure!</p>
<p xml:lang="fr">Une faveur, avec plaisir !</p>
<p xml:lang='de'>Eine Gunst, sicher!</p>
```

[1] В дальнейшем продукционные правила XML будут идентифицироваться как XMLPR[xx], где xx — номер в XML Specification Document.

Каждый грамматически выявленный объект XML-документа может использовать кодировку, отличающуюся от тех, что используются в других частях документа (это может быть удобный путь для включения кусочков русских или китайских текстов в англоязычный текст). Чтобы разрешить эту возможность, надо задать кодировку посредством ключевого слова `encoding` в *текстовой декларации* конкретного документа (XMLPR[77]), например

```
<?xml encoding='UTF-8'?>          <-- одна из кодировок по умолчанию -->
<?xml encoding='ISO-8859-1'?>     <-- кодировка Latin 1 (Западная Европа) -->
<?xml encoding="ISO-2022-JP''?>  <-- японская кодировка -->
```

Препроцессоры XML должны распознавать UTF-16 (16-битовые коды, присущие Unicode) и кодировки UTF-8 («фокус» для использования 7-битовых ASCII-символов «как есть», а всех других символов — как мульти-байтовых последовательностей, занимающих от двух до пяти байтов). Все другие кодировки должны задаваться явным образом (XMLPR[80-81]).

6.5.2 Компоненты XML-документа

XML-документы должны быть *правильно оформленными*. Правильно оформленный XML-документ состоит из *пролога*, одного или более *элементов* и, возможно, некоторой завершающей информации, включающей *инструкции по обработке* документа и комментарии (XMLPR[1]).

Простой пример некоторого корректного документа выглядит следующим образом:

```
1  <?xml version="1.0" standalone="yes"?> <!-- xml declaration      | -->
2  <!-- =======   Начало пролога    ================================+ -->
3  <!DOCTYPE coolxml [                    <!-- DOCTYPE declaration  | -->
4    <!ELEMENT coolxml (#PCDATA)>         <!-- ELEMENT declaration  | -->
5  ]>                                  <!-- end of DOCTYPE declaration  | -->
6  <!-- =======   Конец пролога     ================================+ -->
7
8  <!-- =======   Начало элементов документа   ====================+ -->
9  <coolxml>                            <!-- start tag (root element) | -->
10 XML is a cool idea!                  <!-- element content          | -->
11 </coolxml>                           <!-- end   tag (root element) | -->
12 <!-- =======   Конец элементов документа    ====================+ -->
```

Пролог этого документа может включать *декларации XML* (XMLPR[22-27]) и *декларацию типа документа XML*, за которыми следует фактическое информационное содержимое рассматриваемого документа (используя конкретные экземпляры элементов различных типов и объектов).

Эти компоненты достаточно подробно описываются в следующих разделах.

6.5.3 XML-декларация

XML-декларация — самый первый фрагмент, который обнаруживает в начале документа анализатор XML. Этот фрагмент объявляет для данного документа такие факты, как версия XML, использованная для разметки этого документа

(XMLPR[24-27]) и кодировка (XMLPR[80-81], см. выше). Такого рода объявления вносятся в документ тогда, когда он выполняется как *автономный (standalone)* (XMLPR[32]), т. е. если имеются в наличии внешние декларации. (Ничего не предполагается относительно внешних ссылок, например на рисунки, которые могут присутствовать в «автономном» документе, если они определены внутри него.) Ниже приводится пример, показывающий все три декларации:

```
<?xml  version="1.0" encoding='ISO-8859-1' standalone="yes"?>
<!DOCTYPE racine [<!ELEMENT racine (#PCDATA)>]>
<racine>Salut à vous de la racine du document !</racine>
```

Интерпретация этих трех строк с помощью препроцессора XML не приводит к обнаружению ошибок:

```
java EventDemo racine.xml
Start document:  pubid=null, sysid=file://localhost/home/racine.xml
Resolving entity:  name=[document], pubid=null,
                   sysid=file://localhost/home/racine.xml
Doctype declaration:  name=racine, pubid=null, sysid=null
Start element:  name=racine
Data:  Salut à vous de la racine du document !
End element:  name=racine
End document:  errors=0
```

Предположим, однако, что кодировка не была задана, и подадим такой документ на вход программы грамматического разбора:

```
<?xml  version="1.0" standalone="yes"?>
<!DOCTYPE racine [<!ELEMENT racine (#PCDATA)>]>
<racine>Salut à vous de la racine du document !</racine>
```

Если никакая кодировка явно не задана, выбирается UTF-8 или UTF-16 в зависимости от значения первого байта обрабатываемого файла. В рассматриваемом примере препроцессор XML отвергнет документ с выдачей соответствующего сообщения об ошибке:

```
java EventDemo racineno-enc.xml
Start document:  pubid=null,
                 sysid=file://localhost/home/racineno-enc.xml
Resolving entity:  name=[document], pubid=null,
                   sysid=file://localhost/home/racineno-enc.xml
FATAL ERROR: malformed UTF-8 sequence
  at file://localhost/home/racineno-enc.xml: line 1
java.lang.Error: malformed UTF-8 sequence
```

Поскольку было предположено, что имеет место кодировка UTF-8, акцентированная буква «à» вызвала появление состояния ошибки. Старший бит для этого символа будет ненулевым, поэтому при обработке текста будут допускаться только 7-битовые ASCII-символы, однобайтовые последовательности в

кодировке, принятой по умолчанию. Следовательно, для входного текста, состоящего из однобайтовых акцентированных символов (французских, немецких, русских и т. д.), *должна* быть задана кодировка или же документ должен быть транслитерирован в UTF-8 или UTF-16.

6.5.4 Декларирование типа документа

Вторая часть пролога — это *декларация типа документа* XML (XMLPR[28]). Она начинается литерной строкой `<!DOCTYPE`, за которой следует тип корневого элемента для данного документа. Данная часть пролога содержит также *декларации разметки*, заданные внутри рассматриваемого документа (*внутреннее подмножество*) или во внешних объектах (*внешнее подмножество*), или и то и другое вместе. Внутреннее подмножество читается первым, что позволяет задать элементы, объекты или атрибуты, которые будут использованы при обработке соответствующего внешнего подмножества.

В предыдущих примерах строка

```
<!DOCTYPE racine [<!ELEMENT racine (#PCDATA)>]>
```

представляет собой декларацию типа документа, причем корневой элемент здесь — «`racine`»; данный документ содержит только внутреннее подмножество с одноэлементной декларацией.

Внутреннее и внешнее подмножества, взятые вместе, задают грамматику для данного класса документов, известную как DTD — *Определение типа документа*. Это определение может содержать (XMLPR[29]) следующие элементы:

- *Декларации элементов*, которые начинаются литерной строкой `<!ELEMENT` и содержат *тип (type)* рассматриваемого элемента, а также *модель содержания (content model)* для него.
- *Декларации атрибутов*, начинающиеся со строки `<!ATTLIST`; для данного объявленного элемента они содержат один или более *атрибутов (attributes)* вместе с их типами и значениями по умолчанию.
- *Декларации объектов*, которые начинаются строкой `<!ENTITY` и содержат имя и определение *объекта (entity)*.
- *Декларации нотаций*, начинающиеся со строки `<!NOTATION` и содержащие имя, а также внешний или общеупотребительный идентификатор, поставленный в соответствие некоторой специальной *нотации* для использования с объектами или атрибутами.
- *Инструкции по обработке*, которые ограничиваются с помощью символьных строк `<?` и `?>`. Они содержат данные, не относящиеся к собственно документу (его содержимому), передаваемые в приложение, осуществляющее обработку рассматриваемого документа.
- *Комментарии*, ограниченные символьными строками `<!--` и `-->`, добавляемые в качестве пояснений для повышения удобочитаемости документа.

6.5.4.1 Декларации элементов

Каждый из элементов, принадлежащих логической структуре некоторого документа, должен быть объявлен. Декларация для него (XMLPR[45-46]) задает определенный *тип* элемента, а также *модель информационного содержимого (content model)* для него. Спецификация содержимого может быть одной из четырех:

1. Строка EMPTY означает, что элемент не имеет *никакого* содержимого.

```
<!ELEMENT linebreak EMPTY>
```

2. Строка ANY означает любой другой элемент или символьные данные до тех пор, пока содержимое остается совместимым с правилами XML.

```
<!ELEMENT container ANY>
```

3. *Дочерние элементы* означают, что не допускается никаких символьных данных (XMLPR[47-50]). Их содержимое ограничено списком (возможно, вложенным) элементов, заданных между скобками, за которыми может следовать символ ?, * или +. Внутри скобок может находиться список элементов, разделенных символами , или |, или же возможен следующий уровень вложения. Значение этих символов таково:

,	элементы использовать в экземпляре документа в указанном порядке;
|	выбор одного элемента из списка;
+	одно или более вхождений;
*	нуль или более вхождений;
?	нуль или одно вхождение.

Ссылаясь на наш пример «invitation» в разд. 6.4.3, повторим здесь часть его DTD и обсудим значение деклараций, содержащихся в нем.

```
1 <!ELEMENT invitation (front, body, back)>
2 <!ELEMENT front      (to, date, where, why?)>
3 <!ELEMENT body       (par+)>
```

Каждый элемент invitation (строка 1) имеет в точности одну вступительную (front), основную (body) и завершающую (back) части, и каждый такой элемент следует ввести именно в этом порядке в экземпляр документа. Элемент front (строка 2) должен содержать, последовательно, по одному элементу to, date и where, за которыми может идти необязательный элемент why. Элемент body (строка 3) содержит по крайней мере один элемент par.

4. *Смешанное информационное содержимое* (XMLPR[51]) представляет собой символьные данные (символьная строка #PCDATA), перемежаемые дочерними элементами. Кроме того, строка #PCDATA должна идти первой в модели объявления содержимого, и можно ограничить только тип порожденных элементов, но не порядок числа их вхождений, так как допускается только оператор выбора. Пример из DTD для invitation предста-

вляет собой модель содержимого элемента par, который может включать символьные данные, перемежаемые нулем или более элементов emph.

```
<!ELEMENT par          (#PCDATA|emph)*>
```

6.5.4.2 Декларации атрибутов

Атрибуты, которые могут быть заданы в начальных метках элементов или в пустых элементах, связывают пары «имя-значение» с экземпляром некоторого элемента.

Для проверки их корректности атрибуты должны быть явно декларированы во внутренних или внешних подмножествах DTD. Определение DTD задает, какие атрибуты с какими типами элементов можно использовать, каковы ограничения типов имеют место для этих атрибутов, а также какими будут значения по умолчанию для них (XMLPR[52-53]). По соображениям ясности и удобства декларации атрибутов часто следуют сразу же за декларациями тех элементов, к которым они относятся. Можно, однако, добавлять атрибуты или переопределять путем помещения их во внутреннее подмножество. Как и в случае с объектами, берется первое из вхождений определения атрибута, все последующие его определения игнорируются.

Атрибут *types* подразделяется на три категории (XMLPR[54]):

1. *Строковые типы* (XMLPR[55]) принимают в качестве данных любую символьную строку и определяются с помощью ключевого слова CDATA.

   ```
   <!ATTLIST director name CDATA #REQUIRED>
   ```

2. *Типы, представленные в виде лексем* (XMLPR[56]) вводят лексические и семантические ограничения (условия), которые можно разделить на следующие четыре категории:
   (a) ID представляет собой XML-имя, которое должно идентифицировать уникальным образом данный тип элемента.
   (b) IDREF или IDREFS есть ссылки на элементы, определенные с помощью атрибута ID. При проверке документа на корректность XML-процессор должен проверить, соотносятся ли значения IDREF со значением некоторого атрибута ID.
   (c) ENTITY и ENTITIES представляют собой ссылки на имена объектов, определенных в каком-либо месте DTD.
   (d) NMTOKEN и NMTOKENS ссылаются на множество имен лексем (XMLPR[7-8]) без дальнейших ограничений; например, они не обязаны соответствовать некоторому атрибуту или декларации для объекта.

   Чтобы пояснить сказанное, рассмотрим следующие декларации атрибутов:

   ```
   <!ELEMENT image  EMPTY>
   <!ATTLIST image
        name        ID       #REQUIRED
        size        ENTITY   #REQUIRED
   ```

```
bordercolor NMTOKEN 'red'
title       CDATA   #IMPLIED>
```

Для элемента `image` здесь объявлены два обязательных атрибута (`name` и `size`), а также два необязательных (`bordercolor` и `title`). Атрибут `name` имеет тип `ID` и позволяет ссылаться на изображения в тексте документа. Для атрибута `size` типа `ENTITY` следует использовать определение объекта, чтобы задать его значение. Атрибут `bordercolor`, представляющий собой лексему-имя, будет, если его значение не задано в явном виде, рисовать красную (red) рамку. Наконец, атрибут `title` позволяет при необходимости связать с изображением некоторые символьные данные.
Чтобы завершить эти определения, можно определить другой тип элемента — `imgref`, чтобы ссылаться на рисунки.

```
<!ELEMENT imgref EMPTY>
<!ATTLIST imgref name IDREF #REQUIRED>
```

С помощью такого определения атрибута `name` элемента `imgref` можно использовать его значение, чтобы ссылаться на экземпляры изображения `image`, находящиеся в любом месте исходного текста документа.

3. *Перечислимые типы* (XMLPR[57]) могут принимать один из списков значений, как это задано в соответствующей декларации. Имеется две разновидности:

   (a) *тип нотации* (XMLPR[58]) состоит из ключевого слова `NOTATION`, за которым следует одно или несколько имен обозначений, которые должны быть декларированы в DTD.

   ```
   <!ATTLIST image  type NOTATION (gif|eps|tiff) #REQUIRED>
   ```

   Здесь к элементу `image` добавляется атрибут `type`. Это дает возможность задавать изображение в трех форматах, а именно: GIF, EPS или TIFF. Отметим, что каждый из них должен быть декларирован с помощью некоторого оператора `NOTATION` (см. разд. 6.5.4.4).

   (b) *перечисление* (XMLPR[59]) представляет собой явным образом заданный список лексем-имен, которые следует связать с данным значением атрибута.

   ```
   <!ATTLIST quotation type (inline|display) "inline">
   ```

   Здесь декларируются две возможности — `inline` и `display` — для атрибута `type` элемента `quotation`. Если данный атрибут не задан, то используется значение `inline`.

Чтобы завершить данный раздел об атрибутах, следует объяснить формат записи значений атрибутов по умолчанию, которые задаются в конце каждого элемента декларации (XMLPR[60]). Имеется четыре возможных варианта:

`#REQUIRED` значение *должно* быть задано для атрибута всякий раз, когда используется данный элемент.

`#IMPLIED` атрибуту можно (хотя и необязательно) присвоить значение экземпляра рассматриваемого элемента.

`'defval'` будет использовано значение по умолчанию для рассматриваемого атрибута, если в стартовом элементе не задано другого значения для экземпляра элемента.

`#FIXED` *`'defval'`* атрибут должен *всегда* иметь значение по умолчанию *`defval`*, заданное в DTD.

Примеры для всех четырех случаев можно видеть в декларации для атрибутов элемента `pict`, показанного ниже. Здесь задается значение по умолчанию для атрибута `title`, а значение атрибута `bordercolor` *фиксируется* со значением `blue`.

```
<!ATTLIST pict name        ID      #REQUIRED
               size        CDATA   #IMPLIED
               title       CDATA   'Default title'
               bordercolor NMTOKEN #FIXED 'blue'>
```

6.5.4.3 Декларации объектов

«Внешние» материалы (текстовые фрагменты, специальные символы, изображения, внешние файлы) могут быть включены в исходный XML-текст с помощью ссылок на объекты. XML различает два их типа (XMLPR[70-74]):

- *Общие* объекты
  Декларации, которые могут появляться только в DTD, имеют вид

  ```
  <!ENTITY GEName GEDef>
  ```

  Ссылки на общие объекты (XMLPR[68]) могут появляться как в DTD, так и в экземпляре документа, они состоят из амперсанда (`&`), за которым следует имя объекта с точкой с запятой (`;`) за ним. Эти ссылки *не* раскрываются в DTD.

  ```
  &GEName;
  ```

- *Параметрические* объекты
  Декларации, которые могут появляться только в DTD, имеют вид

  ```
  <!ENTITY % PEName PEDef>
  ```

  Ссылки на параметрические объекты (XMLPR[69]) могут появляться только в разделе DTD рассматриваемого документа. Они состоят из знака процента (`%`), за которым следует имя объекта с точкой с запятой (`;`) за ним.

  ```
  %PEName;
  ```

Определяющая часть `GEDef` и `PEDef` может быть *внутренним* объектом, определения для которого задаются в DTD и для которого нет связанного с ним отдельного физического объекта памяти, или же *внешним* объектом.

Внутренние объекты

Внутренний объект получает свое значение внутри декларации документа; для нее нет отдельного объекта памяти, связанного с этим объектом. Все внутренние объекты разбираются грамматически. Они используются для различных целей, в том числе с их помощью могут быть заданы:

- Определения сокращенных обозначений, чтобы упростить ввод повторяющихся вхождений текстовых строк (общих объектов), например,

```
<!ENTITY XML "Extensible Markup Language">
<!ENTITY MML "Mathematical Markup Language">
```

- Определения сокращенных обозначений для ввода специальных знаков, акцентов или символов (общие знаковые объекты). В качестве иллюстрации дадим набор из пяти предопределенных общих объектов, которые должны распознаваться XML-процессорами[1] Они выражаются как числовые *символьные ссылки* (XMLPR[66]), которые представляют литеры в символьном наборе Unicode в десятичном или шестнадцатеричном представлении. Символьные ссылки немедленно раскрываются при обработке и трактуются как символьные данные.

```
<!ENTITY lt     "&#60;"> <!-- "<" -->
<!ENTITY gt     "&#62;"> <!-- ">" -->
<!ENTITY amp    "&#38;"> <!-- "&" -->
<!ENTITY apos   "&#39;"> <!-- "'" -->
<!ENTITY quot   "&#34;"> <!-- '"' -->
```

Необходимо определить ряд стандартных наборов объектов, таких? как национальные символьные таблицы (серии ISO 8859, Unicode, ISO 10646), графические символы, математические символы и т. д. Ниже приведены еще несколько примеров на основе кодировки Unicode, где используется шестнадцатеричное представление.

```
<!ENTITY cyrya "&#x044f;"> <!-- Кириллическая малая буква "я"       -->
<!ENTITY ggg   "&#x22d9;"> <!-- Математический символ ">>>"
                                                  много больше, чем -->
<!ENTITY U4E0A "&#x4e0a;"> <!-- CJK-идеограмма (китайское "свыше") -->
```

Обычно полные, предопределенные множества таких объектов становятся доступными для определенного экземпляра документа путем включения их как ссылок на внешние объекты. В настоящее время рабочая группа W3C по математике, которая разрабатывает определение MathML, обсуждает с Консорциумом по Unicode пути включения общего набора общеупотребительных математических символов.

- Определения переменных для использования внутри DTD (пáраметрические объекты). Это очень полезное средство для того, чтобы придать мо-

[1] Эти пять объектов, строго говоря, доступны только в правильно оформленных документах, грамматический разбор которых можно осуществить без DTD. Корректные документы *должны* включать определения этих объектов в их DTD, если есть необходимость ссылаться на них.

дульный характер DTD, повышая тем самым удобство использования и сопровождения их. Это положение можно проиллюстрировать такими двумя примерами:

```
<!ENTITY % inline "link | image | object | break | q">
<!ENTITY % align  "align (left|center|right|justify)  #IMPLIED">
```

Первый из них можно было бы использовать, чтобы сделать более короткими модели информационного содержимого за счет того, что допускается глобальное использование «встроенных» элементов. Во втором случае имеет место предварительно отформатированный объект, который может быть использован для объявления атрибута «align» (выровнять) способом, не входящим в противоречие с различными типами элементов.

Все внутренние объекты должны быть объявлены во внутреннем или внешнем подмножестве DTD. Соответствующие декларации в исходном тексте документа должны предшествовать ссылкам на те или иные объекты.

Ссылки на объекты, находящиеся в данной точке XML-файла, запускают операцию подстановки вместо данной ссылки *содержимого* соответствующих объектов. Определения объектов сами могут ссылаться на другие внутренние и уже определенные объекты. Например, используя определение XML-объекта в начале данного раздела, можно следующим образом объявить другой XMLS-объект:

```
<!ENTITY XMLS "&XML; и другие расширяемые языки">
```

Внешние объекты

Внешние объекты — это все те, которые не являются внутренними. Они используются для ссылок на данные, внешние по отношению к рассматриваемому экземпляру документа. Данные, включаемые с помощью такого рода объектной ссылки, могут либо быть подвергнуты грамматическому разбору, либо объявлены с помощью ключевого слова NDATA, причем в последнем случае данные остаются необработанными процедурой грамматического анализа (это будет иметь место, например, в случае рисунка или двоичного файла).

Декларации внешних объектов могут записываться в одной из трех форм:

1. Внешнему идентификатору может предшествовать ключевое слово «SYSTEM», а за ним может следовать *системный литерал*. Он известен также как *системный идентификатор*, который используется для доступа к рассматриваемому объекту (XMLPR[75]). Системный литерал (XMLPR[11]) представляет собой внешний идентификатор в форме URI (*Универсальный идентификатор ресурсов*; см. разд. 1.1.2), который в состоянии идентифицировать любой ресурс данной компьютерной системы.

```
<!ENTITY % subdtd SYSTEM
          "http://www.mysys.org/XML/dtds/headings-xml.dtd">
<!ENTITY chapter1 SYSTEM "chapter1.xml">
```

Здесь первая декларация определяет параметрический объект `subdtd`, который отвечает части некоторого DTD. На него можно сослаться из внутренних или внешних подмножеств DTD. Вторая декларация определяет общий объект `chapter1`, указывающий на исходный файл XML с именем `chapter1.xml` в том же самом каталоге, что и ссылающийся на него главный файл. Такого рода декларации могут быть полезными для разделения больших документов на меньшие по размерам фрагменты с целью упрощения работы с такими документами. Эти фрагменты документов включаются затем в главный документ посредством объектных ссылок.

2. Внешнему идентификатору может предшествовать ключевое слово «`PUBLIC`». В этом случае он должен включать также публичный идентификатор-литерал, за которым в свою очередь будет идти системный литерал в форме URI (XMLPR[75]). Публичный идентификатор (ISO/IEC:9070, 1991) представляет собой имя, которое должно быть осмысленным при работе в разных системах и разнообразных пользовательских окружениях (средах). Формально публичный идентификатор представляет собой композицию из нескольких полей, разделенных двойной косой чертой «//» (см., например, [↪FPISYNTAX] или стандарт (ISO/IEC:9070, 1991)). Короче говоря, первая часть здесь — это *идентификатор владельца*. Объекты в примере, приводимом ниже, отмечены знаком переноса -. Это означает, что рассматриваемые идентификаторы не были формально зарегистрированы, а организация, в которой создан данный файл, — это Консорциум W3C. Вторая часть публичного идентификатора (за ней следует двойная косая черта) называется *текстовым идентификатором*. Первое слово показывает на *общедоступный текстовый класс* (такой, как `DTD` или `ENTITIES`), а за ним следует *описание общедоступного текста* (такое, как `HTML`, `Latin1`). Затем могут следовать (необязательно) после другой двойной косой черты описание *общедоступного текстового языка* и некоторый код из стандарта ISO 639 (см. табл. С.1), который в нашем случае есть `EN`, для английского языка. Наконец, может быть еще одна составляющая — *версия отображения*, если это необходимо. Сервер, помогающий разрешать публичные идентификаторы, создан Питером Флинном [↪FPISERVER].
Рассмотрим теперь, каким образом эти общедоступные идентификаторы используются объектами:

```
<!ENTITY % html4-strict PUBLIC "-//W3C//DTD HTML 4.0//EN"
                        "http://www.w3.org/TR/REC-html40/strict.dtd">
<!ENTITY % ISOlat1      PUBLIC "-//W3C//ENTITIES Latin1//EN//HTML"
                        "d:\home\dtds\iso-lat1.ent">
```

В первом примере мы определили параметрический объект `html4-strict`, который становится известным посредством публичного идентификатора `-//W3C//DTD HTML 4.0//EN`. Используя этот публичный идентификатор, XML-приложение попытается сконструировать некий URI, чтобы получить доступ к требуемому файлу (например, используя файл `catalog`, предлагаемый Консорциумом OASIS; см. разд. 6.6.5.1). Если такой URI не может

быть сформирован, ссылка на внешний объект будет разрешена с использованием явного URI, заданного в конце примера. Второй пример определяет таблицу `ISOlat1` как параметрический объект для определения символов Latin 1. В этом случае URI, задаваемый в конце, представляет собой абсолютный путь (для платформы Microsoft Windows).

3. Для управления данными, не подлежащими грамматическому разбору (такими, как изображения в форматах EPS, GIF или JPEG; исходные TEX'овские файлы или двоичные файлы), необходимо задать ключевое слово `NDATA`, за которым должно идти имя нотации, известное XML-системе (XMLPR[76]). Это позволяет передавать данные в приложение и работать с ними, если приложение в состоянии интерпретировать такие данные.

```
<!ENTITY xmlfig1 SYSTEM
  "http://www.myserver.edu/book-files/figures/xmlfig1.eps" NDATA EPS>
```

Здесь мы определили изображение в формате Encapsulated Postscript, которое можно загрузить с некоторого Web-сервера. Когда XML-приложение проводит грамматический анализ рассматриваемого документа и находит ссылку на этот общий внешний объект, оно должно *знать*, каким образом следует работать с EPS-изображением. Это объявляется с помощью декларации обозначения (см. разд. 6.5.4.4). Ссылки на некоторый неанализируемый объект могут встречаться только в значениях атрибутов, которые были объявлены как типы `ENTITY` или `ENTITIES` (см. «*Tokenized types*» на с. 306).

Отметим, что как и в случае с декларациями атрибутов, первому появлению некоторого объекта должна предшествовать его декларация. Это позволяет записывать декларации во внутреннем подмножестве DTD, которое читается *до* внешнего подмножества, перекрывая, таким образом, возможные определения для того же самого имени объекта во внешнем подмножестве. Чтобы облегчить жизнь программам грамматического разбора, не контролирующим корректность документа при работе во внутреннем подмножестве, ссылки на параметрические объекты не могут быть использованы *внутри* деклараций разметки. Их можно применять только на верхнем уровне, где появляются и сами декларации разметки. В качестве примера рассмотрим следующий документ:

```
1  <?xml  version="1.0"?>
2  <!DOCTYPE doc [
3  <!ENTITY % a '<!ELEMENT a (#PCDATA)>'>
4  %a;
5  <!ENTITY % doc '(a|b)'>
6  <!ELEMENT doc (%doc;)*>
7  <!ELEMENT b EMPTY>
8  ]>
9  <doc><a>text</a><b/><a>more text</a></doc>
```

Со ссылкой на параметрический объект в строке 4 на верхнем уровне все в полном порядке, тогда как аналогичная ссылка в строке 6 внутри декларации элемента будет неверной во внутреннем подмножестве. Однако если пере-

местить строки 3–7 в файл `penext.dtd` и в исходном XML-тексте поместить ссылку на этот файл, то мы обнаружим, что данное ограничение распространяется лишь на рассмотренное внутреннее подмножество. Отсюда вытекает, что следующий документ будет корректным:

```
1 <?xml  version="1.0"?>
2 <!DOCTYPE doc SYSTEM "penext.dtd">
3 <doc><a>text</a><b/><a>more text</a></doc>
```

6.5.4.4 Декларации нотаций

Декларации нотации (XMLPR[82-83]) имеют отношение к способу обращения с неанализируемыми объектами. Они связывают имя

- с форматом некоторого неанализируемого объекта;
- с форматом некоторого элемента с атрибутом «notation»;
- с приложением, которое будет работать с данными внутри обрабатывающей инструкции.

Декларация нотации, следовательно, состоит из имени для требуемого обозначения, как это уже обсуждалось выше, а также из внешнего или общедоступного идентификатора, который позволяет XML-процессору найти программу для обработки данных, которая помечается как отвечающая данному обозначению.

```
1 <!NOTATION gif SYSTEM "c:\Program Files\Internet Explorer\Ie4.dll">
2 <!NOTATION eps SYSTEM "/usr/local/bin/X11/gv">
```

В строке 1 объявлено (для случая Microsoft Windows), что изображения в формате GIF должны обрабатываться с помощью программы *Explorer 4*; в строке 2 объявляется, что делать с файлами EPS в системе UNIX.

6.5.4.5 Значащие пробелы

Чтобы сделать документы более удобочитаемыми и более простыми в обслуживании, часто бывает удобно добавить пустые строки или пробелы в строку. Большую часть времени все эти пробелы не имеют никакого значения и не предназначены для включения их в выходной текст документа, формируемого XML-приложением. Иногда, однако, данные пробелы надо сохранить в выходном тексте так, как они есть в исходном тексте (например, при отображении компьютерного кода). В этом случае с рассматриваемым элементом должен быть связан специальный зарезервированный атрибут `xml:space`, например

```
1 <!ELEMENT computercode (#PCDATA)>
2 <!ATTLIST computercode xml:space #FIXED "preserve">
```

Эти строки заставляют XML-процессор приказать XML-приложению сохранить все пробелы (разрывы строк, табуляцию и т. д.) для материала внутри элемента `computercode`. XML-процессоры никогда не исключают никаких символов пробела, поскольку они должны передать соответствующему приложению в

неизменном виде весь материал, относящийся к содержимому документа. Следовательно, все заботы по управлению пробелами ложатся на приложение. Надо отметить, что пока не сказано обратное, большинство приложений сворачивают последовательность из нескольких пробелов в единственный пробел, т. е. нарушают тот внешний вид, который имел исходный материал документа.

6.5.4.6 Обрабатывающие инструкции

Обрабатывающие инструкции (XMLPR[16-17]) позволяют документу передавать информацию и инструкции заданным приложениям; обрабатывающие инструкции не являются частью символьных данных документа. Их содержимое передается XML-процессором соответствующему приложению.

Обрабатывающие инструкции заключаются внутрь пары символов <? и ?>. За открывающим символом <? следует имя, идентифицирующее целевое приложение (xml представляет собой зарезервированное имя, см. далее); далее идут данные для указанного приложения. Для связывания некоторого приложения с этим именем может быть использована декларация NOTATION (см.разд. 6.5.4.4). Пример того, как указать на приложение PRINT, приводится ниже:

```
<?PRINT Clear Page?>
```

Специальная обрабатывающая инструкция — это строка в самом начале XML-документа, например

```
<?xml version="1.0" standalone="yes"?>
```

Хотя это и называется *XML-декларацией*, это на самом деле не что иное, как обрабатывающая инструкция для XML-процессора, указывающая версию, кодировку и т. п.

6.5.4.7 Секции CDATA

Разделы (секции) CDATA (XMLPR[18-21]) используются для обхода текста там, где разметка не должна распознаваться. Такие секции допускаются там, где могут использоваться символьные данные; они начинаются литерной строкой <![CDATA[и завершаются литерной строкой]]>. Внутри этих ограничителей может быть использован любой вид разметки (за исключением завершающей строки]]>, разумеется). Пользователи LaTeX'а могут усмотреть здесь сходство с окружением verbatim. В следующем примере начальный и конечный элементы, а также ссылка на объект трактуются как символьные данные, а не как разметка.

```
<![CDATA[<p>An ampersand sign looks like &amp;.</p>]]>
```

XML-процессор выведет показанную выше строку неинтерпретированной:

```
<p>An ampersand sign looks like &amp;.</p>
```

6.5.4.8 Условные секции

В пределах внешнего подмножества DTD можно использовать условные секции (XMLPR[61-65]), чтобы включать или исключать части документа, основываясь на значении некоторого ключевого слова. Пусть имеются следующие декларации во внешнем подмножестве нашего DTD:

```
1  <!-- все нижеследующее должно быть во внешнем подмножестве -->
2  <!ENTITY % complex "IGNORE">
3  <!ENTITY % simple  "INCLUDE">
4  <![ %complex; [
5  <!ELEMENT table
6       (caption?, (col*|colgroup*), thead?, tfoot?, tbody+)>
7  ]]>
8  <![ %simple; [
9  <!ELEMENT table (caption?, tbody)>
10 ]]>
```

Параметрический объект `complex` (строка 2) присваивает значение `IGNORE` ключевому слову, управляющему условной секцией; следовательно, первое определение (complex) (строки 4–7) будет пропущено, а вторая (simpler) декларация (строки 8–10) для элемента `table` (который «включается» через объект `simple` в строке 3) будет сохранена. Предположим, однако, что во внутреннее подмножество нашего экземпляра документа помещены следующие строки:

```
1  <!ENTITY % complex "INCLUDE">
2  <!ENTITY % simple  "IGNORE">
```

Поскольку есть предшествование первому появлению деклараций объектов, значения параметрических объектов `complex` и `simple` во внешнем DTD будут перекрыты значениями из внутреннего подмножества. В результате (строка 1) ключевое слово `INCLUDE` будет присвоено условной секции, управляющей сложным определением элемента типа `table`, которое будет использовано в DTD.

6.5.5 Элементы документа

Каждый XML-документ содержит один или более *элементов* (XMLPR[39]), состоящих из *информационного содержимого* (XMLPR[43]), заключенного между начальной меткой *starttag* (XMLPR[40]) и завершающей меткой *endtag* (XMLPR[42]), или метки *пустого-элемента* (*empty-element*) (XMLPR[44]). Каждый элемент имеет тип, идентифицируемый его именем (его «родовой идентификатор»), а начальная и пустая метки могут иметь наборы спецификаций атрибутов. Каждой спецификации атрибута (XMLPR[41]) поставлено в соответствие некоторое имя и некоторое значение.

Для того чтобы XML-документ был корректным, каждый элемент и объект должны быть объявлены в DTD, а их содержимое и атрибуты должны отвечать соответствующим моделям содержимого и декларации атрибута.

Рассмотрим в качестве примера часть введенного ранее документа `invitation` (разд. 6.4.2.2):

```
1  <body>
2  <par>
```

```
I would like to invite you all to celebrate
the birth of <emph>Invitation</emph>, my
first XML document child.
</par>
...
</body>
```

Если сравнить этот пример с ранее определенным DTD для документа `invitation` (разд. 6.4.3), то увидим, что рассматриваемые элементы используются согласно соответствующей части этого DTD.

```
<!ELEMENT body       (par+)>
<!ELEMENT par        (#PCDATA|emph)*>
<!ELEMENT emph       (#PCDATA)>
```

Завершим данный раздел, посвященный элементам документа, примером использования пустых элементов.

```
<!DOCTYPE emptyexample [
<!ELEMENT emptyexample (par*)>
<!ELEMENT par (#PCDATA|image|imref)*>
<!ELEMENT image EMPTY>
<!ATTLIST image name    ID     #IMPLIED
                address CDATA  #REQUIRED>
<!ELEMENT imref EMPTY>
<!ATTLIST imref name    IDREF  #REQUIRED>
<!ENTITY  logo-uri "http://www.ucc.ie/xml/xml.gif">
]>
<emptyexample>
<par>The XML logo is seen in the image <imref name="xml-logo"/>.</par>
<par>The image tag shows an alternative syntax for an empty element
<image name="xml-logo" address="&logo-uri;"></image>.</par>
</emptyexample>
```

Этот корректный XML-документ содержит элементы типа `par` внутри корневого элемента `emptyexample` (строка 2). Элемент `par` здесь имеет смешанное содержимое, состоящее из анализируемых символьных данных, а также элементов `image` и `imref` (строка 3), причем оба они объявлены как пустые (строки 4 и 7). В конце внутреннего подмножества DTD мы объявляем общий объект `logo-uri` (строка 9), который используется для того, чтобы задать место расположения требуемого изображения. Содержимое первого элемента `par` (строка 12) показывает один из двух путей представления пустого элемента в XML-документах. Метка ссылки на изображение `<imref name="xml-logo"/>` в рассматриваемом случае имеет атрибут типа `IDREF` со значением «`xml-logo`». Для того чтобы рассматриваемый документ был корректным, *должен* существовать другой элемент с атрибутом, объявленным как атрибут типа `ID`, и значением этого атрибута «`xml-logo`». Такой элемент виден внутри второго элемента `par` (в строке 14). Здесь рассматриваемый идентификатор определяется внутри пустого элемента `image`, который связывает изображение из файла в формате GIF через URI (и ссылку на объект `logo-uri`). Обратим внимание на альтернативный синтаксис `<image...> </image>` для пустого элемента.

6.6 Программы грамматического разбора и инструментальные средства XML

Рядовому пользователю нет необходимости беспокоиться относительно программ грамматического разбора для XML (совсем как в случае языков Fortran, C или C++, когда компиляторы для них служат программисту лишь средством трансляции компьютерных программ в исполнимый код). Тем не менее, определенная информация относительно доступности и полезности такого рода программ помогает понять, как они (и другие программы) используются для формирования, например, HTML- или PostScript-выхода из XML-данных.

Ряд программ грамматического разбора для XML свободно доступен в Интернете. Большинство из них написано на языке C++ или Java, хотя используются и такие языки, как Python, Perl и Javascript. Новые версии навигаторов, такие, как Internet Explorer Version 5 фирмы Microsoft, Mozilla (преемник программы Netscape Version 4), а также другие программные средства, в частности пакет Office фирмы Microsoft, Acrobat фирмы Adobe, программа FrameMaker, пакеты символьной алгебры, в частности Mathematica и Maple, — все они будут иметь встроенную поддержку для XML.

В этом разделе ограничимся рассмотрением этапов редактирования и верификации XML-документов; в оставшихся главах мы будем иметь дело с путями использования грамматически разобранных данных для выполнения действий с ними, просмотра, печати и обмена информацией. В самом деле, XML-разметка в очень малой степени передает семантическое значение данного элемента; в частности, она ничего не говорит относительно того, как следует интерпретировать тот или иной элемент. Основной принцип XML состоит в том, что структура и семантика полностью разделены, а стиль обработки задается с помощью XSL. Эта тема вместе с другими вопросами, относящимися к стилевым разработкам, будет рассмотрена в гл. 7.

Начнем наш обзор рассмотрением двух примеров редакторов для XML-данных и определений DTD. Затем несколько слов будет сказано о средствах обработки DTD, после чего аналогичное рассмотрение будет проведено для некоторых из программ грамматического разбора.

6.6.1 Emacs и psgml

Программа `emacs` — один из наиболее часто используемых редакторов в среде UNIX (в настоящее время он приобрел также популярность и в среде Microsoft Windows). Этот редактор хорошо приспособлен для настройки его применительно к потребностям конкретного пользователя (с использованием языка Lisp), он также легко работает с синтаксисом различных языков. Леннарт Стафлин разработал для `emacs` базовую поддержку SGML в форме «макрорежима» `psgml` [↪PSGML]. Впоследствии Дэвидом Меггинсоном был разработан динамически подключаемый XML-модуль для `psgml` [↪PSGMLXML]. Для редактирования DTD Тони Грэмом был предложен макро `tdtd` [↪TDTD].

Все эти режимы представлены пользователю с помощью меню и команд для вставки меток, т. е. помогают пользователю вводить только контекстно

корректные метки, а также обеспечивают возможность редактирования значений атрибутов в отдельном окне с информацией о типах и значениями по умолчанию. Они идентифицируют также структурные ошибки.

В качестве примера: рис. 6.2 показывает «головную» («front») часть рассматривавшегося документа «invitation». Из ниспадающего меню «SGML» выберем элемент «List valid tags» («Перечислить разрешенные метки»). Курсор позиционируется сразу за элементом `front`, как это показано в верхней левой части рассматриваемого рисунка. Произведя грамматический разбор определения DTD, программа `psgml` определяет, что только элемент `to` является корректным в рассматриваемом случае и информирует нас об этом в нижнем окне; `psgml` включает также[1] интерфейс с программой `nsgmls` Джеймса Кларка, представляющей собой полномасштабную программу грамматического разбора для SGML, описываемую ниже.

На рис. 6.3 представлен пример того, как режим `tdtd` может быть полезен при редактировании DTD. Ключевые слова, комментарии, объекты, литеральные данные, модель содержимого и ограничители легко различимы, поскольку они обозначаются разными цветами. Кроме того, в каждой точке можно попросить `emacs` о содействии. В частности, в нижней половине примера можно видеть список доступных команд DTD. Если выбрать какой-либо из этих элементов, программа проведет пользователя через диалог, с помощью которого можно будет задать все существенные компоненты для рассматриваемого элемента.

Большинство пользователей предпочтут, скорее всего, более наглядный (WYSIWYG) подход к редактированию XML-документов, созданию или модификации определений DTD. В настоящее время доступны несколько наглядных редакторов для XML. Как и в примере, приведенном выше, на рис. 6.4 показан прототип одного из таких WYSIWYG-редакторов. Он был разработан Пьером Морелем и позволяет редактировать древовидные представления XML-документов (см. верхнюю часть рис. 6.4), а также DTD (сошлемся на нижнюю часть того же самого рисунка). Рассматриваемый редактор написан на языке Java и имеет интерфейс, характерный для базовых классов языка Java (Java Foundation Classes) (см. [↪VISXML] для получения более подробной информации по этой теме, а также для получения инструкций по загрузке требуемого программного обеспечения).

Еще один пример — экспериментальный HTML-редактор Amaya консорциума W3C; его описание доступно по адресу [↪AMAYA]. Строго говоря, это не совсем XML-редактор, но поскольку он позволяет экспериментировать с каскадными стилями (CSS) и языком MathML — можно редактировать сложные математические выражения через WYSIWYG-интерфейс, — он определенно является представителем того семейства редакторов, которые мы сможем увидеть в близком будущем. Рис. 6.5 показывает различные виды одного и того же документа, средства, упрощающие ввод документа и работу с его исходным текстом.

[1] Здесь неточность: `psgml` — не интерфейс, а полноценный отдельный анализатор. — *Прим. ред.*

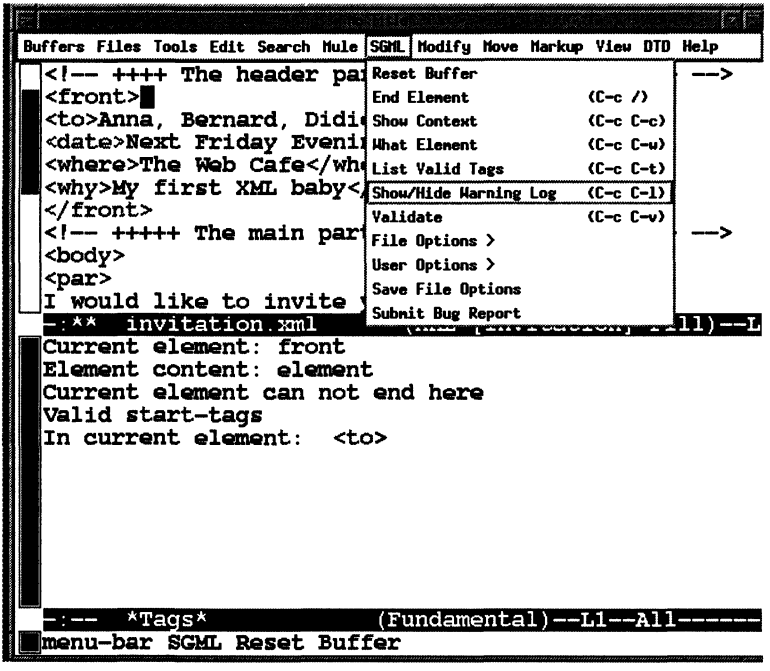


Рис. 6.2. Emacs в режиме psgml при обработке примера invitation.

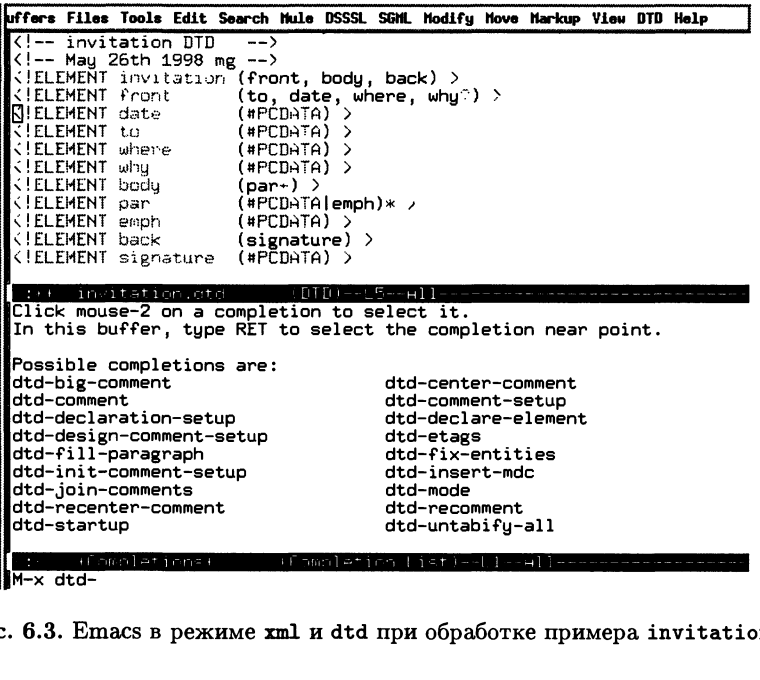


Рис. 6.3. Emacs в режиме xml и dtd при обработке примера invitation.

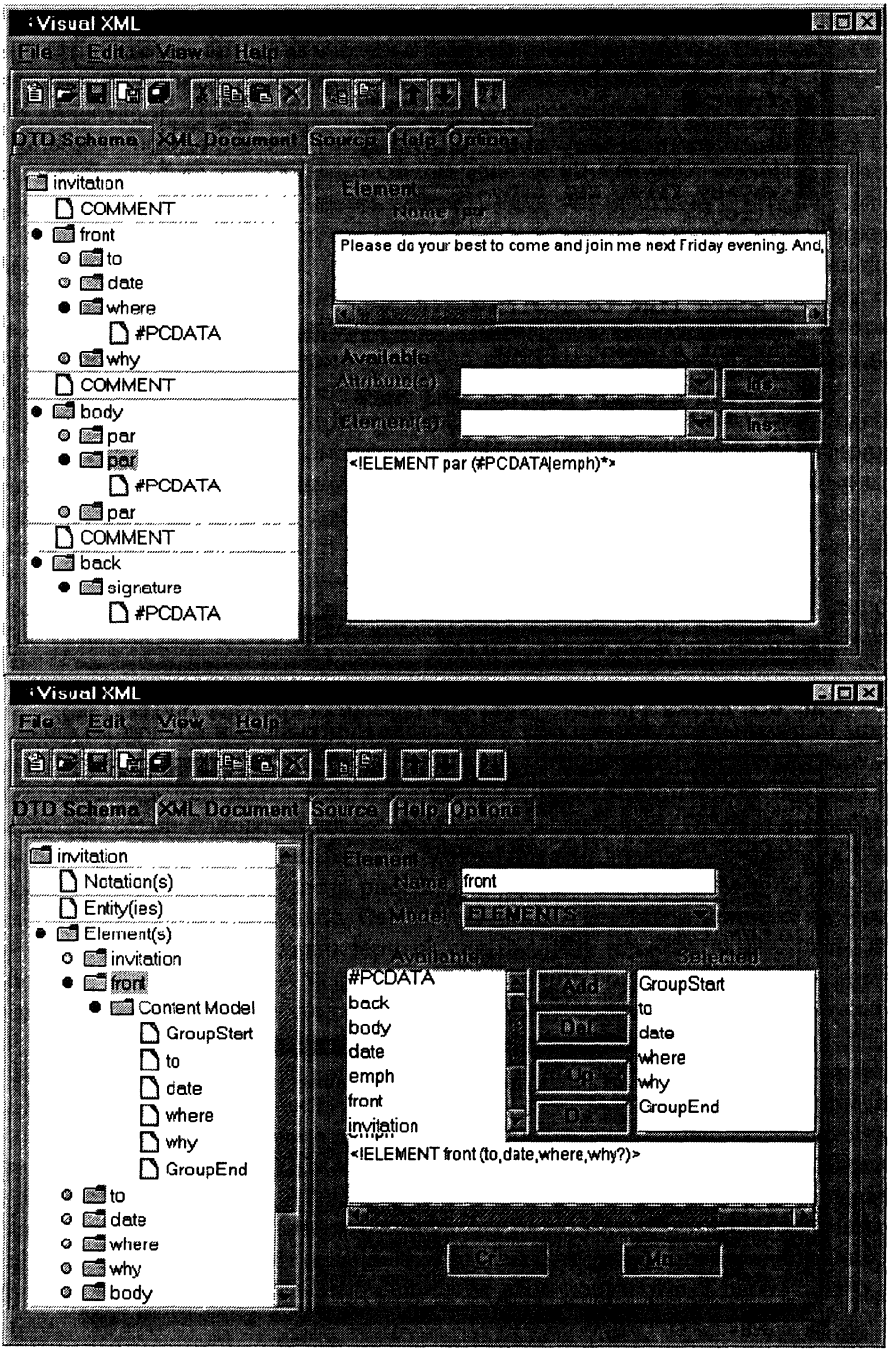


Рис. 6.4. WYSIWYG-редактор для XML.

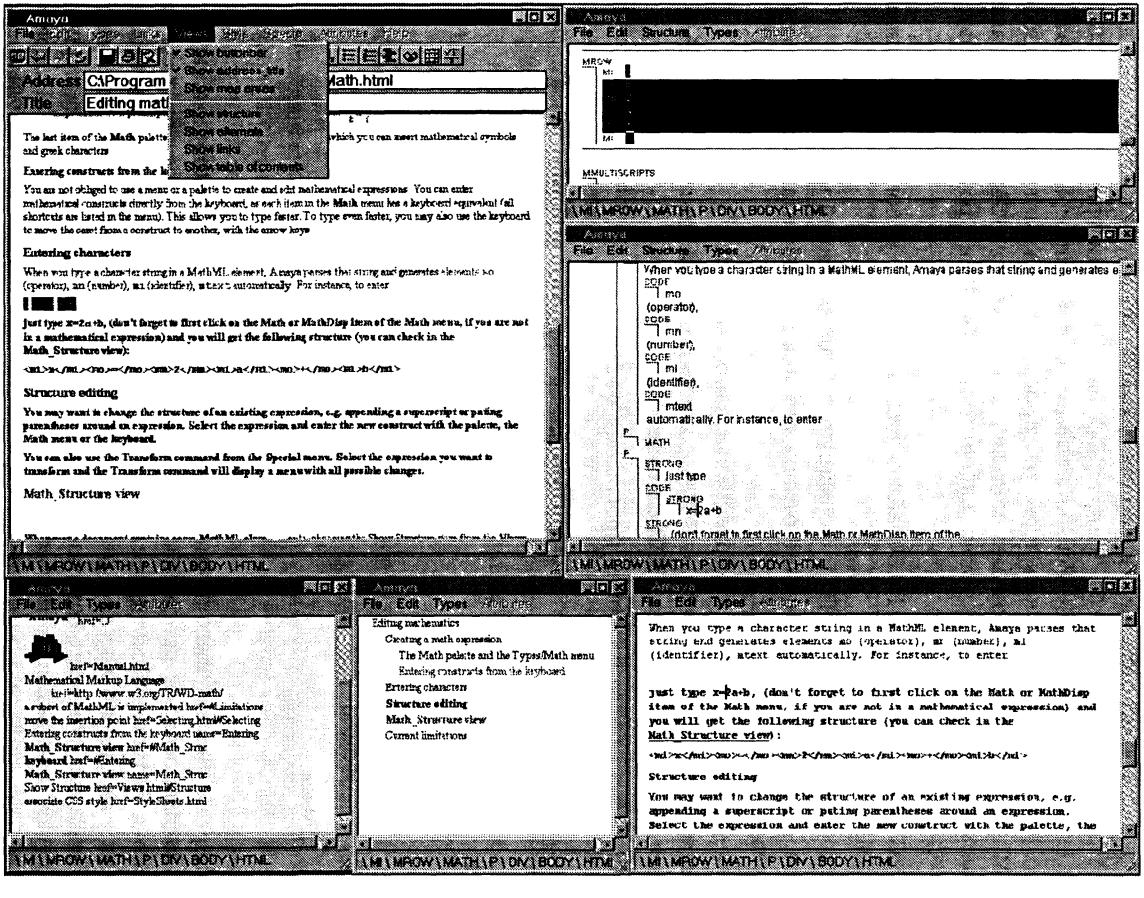


Рис. 6.5. WYSIWYG-редактор Amaya.

6.6.2 Программы perlSGML

Эрл Худ разработал `perlSGML`, набор программ и библиотек на языке Perl для обработки DTD и документов, написанных на SGML. Дистрибутив `perlSGML` [↪PERLSGML] включает две Perl-библиотеки (`dtd.pl`, программу грамматического разбора DTD, а также `sgml.pl`, программу грамматического разбора SGML-документов), набор Perl-модулей для работы с SGML-документами и управления объектами, а также набор программ. Они кратко описываются ниже:

`dtd2html` создает набор HTML-файлов для навигации и документирования SGML DTD.

`dtddiff` сравнивает два DTD и формирует список различий между ними.

`dtdtree` выводит иерархические отношения между элементами DTD в виде древовидной структуры.

`dtdview` позволяет делать интерактивные запросы к DTD.

`stripsgml` освобождает файл от SGML-разметки и заменяет объектные ссылки в нем стандартными ASCII-символами.

Рассмотрим более подробно программу `dtdtree`.

```
 1  dtdtree -dtd invitation.dtd
 2   --------------------------------
 3              INVITATION
 4   --------------------------------
 5  INVITATION
 6   |_(front,
 7   |   |_(to,
 8   |   |   |_(#PCDATA)
 9   |   |
10   |   |__date,
11   |   |   |_(#PCDATA)
12   |   |
13   |   |__where,
14   |   |   |_(#PCDATA)
15   |   |
16   |   |__why?)
17   |       |_(#PCDATA)
18   |
19   |
20   |__body,
21   |   |_(par+)
22   |       |_(#PCDATA |
23   |       |__emph)*
24   |           |_(#PCDATA)
25   |
26   |
27   |
28   |__back)
29       |_(signature)
30           |_(#PCDATA)
```

Как видно из предыдущего примера, представляющего собой отображение дерева, сформированное программой `dtdtree` для DTD в примере `invitation`, программа `dtdtree` обеспечивает удобное визуальное представление иерархических отношений, которые существуют между различными элементами в DTD. Вводимые команды показаны в строке 1. Ясно видна структура различных моделей содержимого. Возьмем, например, элемент `par` (строка 21): сразу видно, что его родительским элементом будет элемент `body` (строка 20), а дочерними элементами являются элементы `#PCDATA` и `emph` (строки 22 и 23).

6.6.3 Инструментальное средство DTDParse

Норман Уолш написал еще одну программу для документирования определений DTD с помощью HTML-файлов. Эта программа входит в состав пакета программ `DTDParse`, доступного по адресу [↪DTDPARSE]. И опять языком реализации здесь является Perl. Уолшем использован двухэтапный подход: вна-

чале производится грамматический разбор DTD, затем осуществляется ввод его в базу данных, что облегчает выполнение операций поиска при последующих обращениях к DTD. Ниже приведен протокол работы утилиты `dtdparse`, формирующей базу данных для DTD в нашем примере `invitation`:

```
dtdparse -v 2 invitation.dtd
Loading catalog: ./CATALOG
Loading catalog: /usr/local/sgml/CATALOG
Loading catalog: ~/sgml/CATALOG
Expanding entities...
Finding children...
Expanding back
Expanding signature
Expanding par
Expanding to
Expanding invitation
Expanding why
Expanding front
Expanding emph
Expanding where
Expanding date
Expanding body
Finding parents...
Finding parents of back
Finding parents of signature
Finding parents of par
Finding parents of to
Finding parents of invitation
Finding parents of why
Finding parents of front
Finding parents of emph
Finding parents of where
Finding parents of date
Finding parents of body
Storing elements...
```

После того как элементы записаны в базу данных, можно преобразовать эту информацию в набор страниц `man` для UNIX или HTML-страниц, используя программу Нормана Уолша `dtd2html` (эта программа *отличается* от программы Эрла Худа, упоминавшейся в разд. 6.6.2) или программы `dtd2man` соответственно. Интересный пример использования этого средства — документация для DocBook DTD, которую можно посмотреть по адресу [↪DBVIEW] (это DTD, используемое часто для разметки технической документации, обсуждается кратко в разд. B.4.1).

6.6.4 Инструментальный XML-пакет, разработанный группой лингвистической технологии

Группа лингвистической технологии Эдинбургского университета занимается созданием средств работы с языком SGML уже в течение многих лет. Недавно этой группой выпущена версия 1 библиотеки LTXML. Она включает инструментальный комплекс для разработчиков, построенный на базе API, реализованного с помощью языка C, для работы с XML-документами, а также набор автономных программ, построенных с использованием указанного API. Группа лингвистической технологии располагает обширным опытом работы с большими *корпусами текстов* — коллекциями текстовых файлов, содержащими от нескольких десятков тысяч до нескольких миллионов знаков. Чтобы работать с такими большими документами, необходимо было разработать чрезвычайно эффективные средства пакетной обработки, ориентированные на взаимодействие с огромными потоками данных. Ниже мы рассмотрим эти средства несколько подробнее. Более детальное их описание можно найти в [↪LTXML].

6.6.4.1 XML-преобразования и поиск регулярных выражений

Программа `sgmltrans` осуществляет трансляцию XML-файлов в некоторые другие форматы, в частности в HTML или LaTeX. Работа ее основывается на

предположении, что начальный и завершающий элементы определяют действия, которые надо предпринять; эти действия в определенной степени зависят также и от контекста. Данная программа позволяет только помещать некоторый текст в выходной поток.

Программа `sggrep` аналогична программе `grep` для среды UNIX. Она позволяет производить поиск файла, включающего некоторые регулярные выражения, принимая во внимание древовидную структуру XML-файлов.

```
sggrep [-h] [-u base-url]   [-d doctype] [-v] [-n] [-r]
           [-m mark-query] [-a element-name] [-q query]
           [-s sub-query] [-t regexp] [--] [inputs...]
```

Наиболее важными аргументами здесь являются следующие:

`-u` *`base-url`* базовый URL, используемый для разрешения относительных адресов (ссылок) URL.

`-n` не печатать newline (переход на новую строку) между смежными фрагментами вывода (по умолчанию newline печатается).

`-q` *`query`* шаблон для отыскиваемых элементов, путь из термов, разделенных знаком «/», где каждый терм представляет собой некоторый элемент SGML.

`-r` значения атрибутов в запросах и в регулярных выражениях.

`-s` *`sub-query`* выбирает подэлементы (вложенные элементы более низкого уровня) из элемента, выбранного по запросу, для сопоставления с образцом *regexp*.

`-t` *`regexp`* регулярное выражение для сопоставления с текстом в элементе, полученном по запросу (если нет вложенного запроса), или во вложенном элементе, полученном по вложенному запросу. Пустая строка успешно сопоставляется с любыми элементами, включая и пустые элементы; на самом деле это единственный путь задания пустых элементов, если в них возникнет необходимость.

Рассмотрим теперь несколько примеров, используя файл `invitation.xml` на с. 298. Строка-запрос заключается в определении «пути», позволяющего достичь выбранных элементов.

```
1  sggrep -q "invitation/body/par" invitation.xml
2  <?xml version='1.0' encoding='UTF-8'?>
3  <!DOCTYPE invitation SYSTEM "invitation.dtd">
4  <par>
5  I would like to invite you all to celebrate
6  the birth of <emph>Invitation</emph>, my
7  first XML document child.
8  </par>
9  <par>
10 Please do your best to come and join me next Friday
11 evening. And, do not forget to bring your friends.
12 </par>
13 <par>
14 I <emph>really</emph> look forward to see you soon!
15 </par>
16
17 sggrep -q ".*/emph"  invitation.xml
```

```
18  <?xml version='1.0' encoding='UTF-8'?>
19  <!DOCTYPE invitation SYSTEM "invitation.dtd">
20  <emph>Invitation</emph>
21  <emph>really</emph>
22
23  sggrep -q "invitation/body/par[0]/emph"  invitation.xml
24  <?xml version='1.0' encoding='UTF-8'?>
25  <!DOCTYPE invitation SYSTEM "invitation.dtd">
26  <emph>Invitation</emph>
27
28  sggrep -q "././.[2]/." invitation.xml
29  <?xml version='1.0' encoding='UTF-8'?>
30  <!DOCTYPE invitation SYSTEM "invitation.dtd">
31  <emph>really</emph>
32
33  sggrep -q "./.[0]/.[2]" invitation.xml
34  <?xml version='1.0' encoding='UTF-8'?>
35  <!DOCTYPE invitation SYSTEM "invitation.dtd">
36  <where>The Web Cafe</where>
```

Первая команда (строка 1) запрашивает все элементы `par` внутри тела документа `body`, внутри элемента `invitation`. Видны три абзаца и их содержимое (строки 4–15). Вторая команда (строка 17) использует синтаксис с групповым символом, чтобы показать все элементы `emph` (строки 20 и 21), тогда как третья команда (строка 23) переходит к элементу `emph` в первом (считая начальный за нулевой) элементе `par` внутри элемента `body`, в свою очередь, находящегося внутри элемента `invitation`. Здесь мы имеем также различные виды синтаксиса с групповым элементом, как в четвертой команде (строка 28), где мы хотим увидеть *любой* элемент внутри третьего элемента, включенного в любой элемент, который входит во внешний (корневой) элемент; последняя команда (строка 33) показывает второй элемент внутри первого элемента, который входит во внешний элемент.

Более сложным средством является программа `sgrpg`, допускающая XML-запросы и общие преобразования. Можно выбрать набор XML-элементов и, если в этом есть необходимость, преобразовать их в некоторый другой формат. Эта программа позволяет обрабатывать довольно сложные запросы, в том числе и вложенные. Это существенно более общий подход, чем реализуемый программой `sggrep`. Средства LTXML очень полезны для простых ad hoc («разовых») запросов. Однако универсальные прикладные программы (приложения), которые должны помещать информацию в XML-файлы основываясь на их иерархической структуре, будут, скорее всего, использовать языки XSL или DSSSL, поскольку они лучше приспособлены для организации сложного поиска.

6.6.4.2 Другие инструментальные средства

Система LTXML располагает еще рядом интересных небольших по размеру средств для работы с XML-файлами. Рассмотрим некоторые из них.

Программа `sgcount` пересчитывает элементы в предъявленном ей XML-файле.

```
sgcount [-o nb] [-t] [inputs...]
```

`-t` считать только элементы верхнего уровня.

`-o` ***nb*** целочисленный параметр, определенный следующим образом:

0 формат по умолчанию для вывода на печать; это таблица, показывающая для каждого элемента документа его тип, частотность, а также частотность, сопровождаемую атрибутом типа `ID`.

1 показывает только имена и счетчики элементов.

2 показывает только полное число элементов.

```
1  sgcount invitation.xml
2  invitation      1       0
3  front           1       0
4  to              1       0
5  date            1       0
6  where           1       0
7  why             1       0
8  body            1       0
9  par             3       0
10 emph            2       0
11 back            1       0
12 signature       1       0
13 *Total*         14      0
14
15 sgcount emptyexample.xml
16 emptyexample    1       0
17 par             2       0
18 imref           1       0
19 image           1       1
20 *Total*         5       1
21
22 sgcount -o 1 emptyexample.xml
23 emptyexample    1
24 par             2
25 imref           1
26 image           1
27 *Total*         5
28
29 sgcount -o 2 emptyexample.xml
30 5
31
32 sggrep -q "invitation/body/par" invitation.xml | sgcount -o 1
33 par     3
34 emph    2
35 *Total* 5
```

Эти примеры показывают число и типы элементов в XML-файлах `invitation` (строки 1–13) и `emptyexample` (строки 15–20), а также влияние опции `-o` (строки 22–30). Особый интерес представляет последняя команда (строка 32); здесь добавлена команда `sggrep` для выбора раздела (секции) из файла `invitation.xml` внутри элементов `par`; здесь подсчитывается также число включенных элементов (строки 33–35).

Утилита `textonly` удаляет всю XML-разметку из заданного файла.

```
textonly [-h] [-u base-url] [-t tag] [-s c] [-x] filename
```

Наиболее значимыми опциями здесь являются следующие:

`-u` ***base-url*** базовый URL.

`-t` ***tag*** выводит только текст, присутствующий внутри элемента ***tag***.

`-s` *`string`* выводит разделитель между следующими друг за другом текстовыми элементами. Специальными значениями для этих разделителей являются `' '` для пробела, `'\n'` для символа начала новой строки newline (он может быть полезен, например, если надо выводить по одному слову в каждой строке), а также `''` для пустой строки.

```
1  textonly -t par invitation.xml
2  I would like to invite you  all to celebrate
3  the birth of Invitation, my first XML document
4  child.
5
6  Please do your best to come and join me next Friday
7  evening. And, do not forget to bring your friends.
8
9  I really look forward to see you soon!
10 textonly -t signature invitation.xml
11 Michel
12 textonly -t front -s '\n' invitation.xml
13 Anna, Bernard, Didier, Johanna
14 Next Friday Evening at 8 pm
15 The Web Cafe
16 My first XML baby
```

Используем еще раз наш пример `invitation.xml`. Первая команда в нем `textonly` (строка 1) получает все данные из элементов `par` (строки 2–9); вторая команда (строка 10) получает данные внутри элемента `signature` (строка 11), тогда как третья команда (строка 12) получает все данные в головной части `front` рассматриваемого документа, добавляя команду newline между различными текстовыми строками, чтобы разделить их (строки 13–16).

Используя LTXML API, можно достаточно просто построить ряд дополнительных средств. Например, рассматриваемый инструментальный пакет содержит также генератор ESIS (см. разд. 6.6.5.2), обеспечивающий выход, похожий на тот, что описывается в следующем разделе.

6.6.5 Проверка корректности документов с помощью программ грамматического разбора XML

Зачастую важно, или уж во всяком случае полезно, иметь возможность проверить, является ли XML-файл правильно оформленным или корректным. Иногда хорошо было бы иметь возможность проанализировать документ и разделить его на составные части, чтобы можно было проводить с ним дальнейшие манипуляции, которые могут потребоваться для получения доступа к DTD. Во всех таких случаях становятся удобны программы грамматического разбора XML-текстов. Полный и современный список XML-средств, включая программы грамматического разбора, можно найти на сайтах [↪XMLPARS] и [↪XMLRES]. В следующих разделах рассмотрим несколько достаточно показательных примеров. Как уже отмечалось, примерно к середине 1999 г.[1] ожидается появление навигаторов и других программных офисных средств, ко-

[1] В настоящее время прогнозы авторов оправдались: поддержка XML входит во многие популярные приложения. — *Прим. ред.*

торые будут располагать встроенными средствами грамматического разбора XML-текстов.

6.6.5.1 Возможность взаимодействия XML-документов

XML-документ должен знать, где найти данные, отвечающие некоторой ссылке на внешний логический объект. Чтобы сделать возможным перемещение документов между компьютерами, необходим некоторый механизм, обеспечивающий такого рода «прозрачное» управление логическими объектами, согласованный в рамках SGML/XML-сообщества.

Вследствие этого, Консорциум OASIS (Организация по развитию структурированных информационных стандартов, до 1997 г. известная как SGML-Open) предложил набор соглашений для отображения идентификаторов внешних логических объектов на имена файлов, URI, или другие объекты памяти. Кроме того, был определен файл-каталог, который содержит публичные идентификаторы, связанные с каждым из файлов, предназначенных для обмена. Детали этого подхода можно найти в [↪OASIS].

Чтобы пояснить идею, согласно которой формируется файл `catalog`, рассмотрим следующий небольшой пример:

```
1  OVERRIDE YES
2
3  SGMLDECL "/afs/cern.ch/user/g/goossens/sgml/dtds/xml.decl"
4
5  PUBLIC "ISO 8879-1986//ENTITIES Added Latin 1//EN" "iso-lat1.gml"
6  PUBLIC "-//LWC//DTD Invitation//EN"
7                                "http://lwc.org/dtds/invitation.dtd"
8
9  ENTITY    "mylogo" "graphicsfiles/mylogo.eps"
10 NOTATION  "gif"    "/usr/local/bin/xv"
11 BASE      "http://wwwinfo.cern.ch/~goossens/sgml/dtds/"
```

Если значение `OVERRIDE` есть `YES` (строка 1), то элементы каталога `PUBLIC`, `ENTITY`, `DOCTYPE` или `NOTATION` будут предшествовать системным идентификаторам, заданным в явном виде для данного внешнего идентификатора. Объект `SGMLDECL` (строка 3) задает положение SGML-декларации, которая определяет синтаксис SGML-документа. (Для HTML или XML эта декларация фиксирована, однако ее надо задавать для общей программы грамматического анализа SGML-текстов наподобие `nsgmls`; см. разд. 6.5.3.) Строки 5–7 устанавливают ссылку URI с некоторым общедоступным идентификатором.

Файл `catalog` может также содержать определения объектов и обозначений (строки 9 и 10). Ключевое слово `BASE` позволяет задавать относительные системные идентификаторы, определенные на базе заданного абсолютного системного пути (строка 11).

Предпринимаются усилия по внедрению каталогов с синтаксисом (ESIS) XML [↪XCATALOG].

6.6.5.2 Информационное множество структур элементов

Некоторые программы грамматического разбора анализируют XML-документы и выводят простое текстовое представление структуры информационных элементов (Element Structure Information Set — ESIS, см. [↪ESIS] для более детального описания). ESIS представляет собой, по-видимому, первое предложение для стандартного формата вывода SGML-данных после их грамматического разбора. Это очень удобно для передачи данных между программой грамматического разбора и приложением, например таким, которое будет выполнять форматирование или же какое-либо другое преобразование входных логических объектов. Поэтому не должно вызывать удивления то обстоятельство, что многие программы грамматического разбора XML обеспечивают возможность формирования выходного потока в формате ESIS, что упрощает сравнение их поведения. Поэтому покажем пример ESIS-вывода, полученного с помощью XML-процессора. Отметим, что большинство XML-процессоров обеспечивает также более современные интерфейсы в форме событийных интерфейсов прикладного программиста (API), например SAX (см. разд. B.6) или API, основанные на использовании древовидных структур, такие, как DOM [↪DOMGEN].

6.6.5.3 Программа грамматического разбора nsgmls

Многие из поставщиков программного обеспечения для SGML провели его доработку, с тем чтобы обеспечить обработку и XML-текстов. В качестве примера рассмотрим пакет программ SGML, разработанный Джеймсом Кларком и принадлежащий к числу наиболее общих и давно находящихся на рынке программного обеспечения средств; из состава этого пакета рассмотрим программу грамматического анализа `nsgmls`. Эта программа может быть использована для проверки корректности XML-документов по отношению к некоторому DTD; она формирует вывод в формате ESIS (см. разд. 6.6.5.2), который можно получить и с помощью других программ грамматического анализа.

Программу `nsgmls` можно скачать с домашней страницы Джеймса Кларка [↪SP]. Данная программа является частью пакета `SP` и написана на языке C++. Двоичный исполнимый код для нее доступен для различных аппаратно-программных платформ, однако можно загрузить также и ее исходный текст, что дает возможность самостоятельно откомпилировать ее в тех редких случаях, когда для требуемой платформы исполнимого кода этой программы нет. Опции командной строки, упоминаемые ниже, подробно объясняются в документации, которой сопровождается рассматриваемая программа. (Ее можно найти также по адресу [↪SPDOC].)

```
nsgmls [-vCegBdlprsu] [-b bctf] [-f error_file] [-c catalog_file]
       [-D dir] [-a link_type] [-A arch] [-E max_errors] [-i entity]
       [-w warning_type] [-m catalog_sysid] [-o output_option]
       [-t rast_file] input_file(s)
```

Обычно среднему пользователю приходится иметь дело лишь с такими опциями, как `-c`, переключающей индикатор расположения файла (файлов) каталога; `-s` (исполнять в «молчаливом» режиме — silently), чтобы выводились только сообщения об ошибках, а также `-w`, чтобы установить *уровень предупреждений* (*warning level*) вместе с именами входных файлов XML. По умолчанию для вводимого документа формируется вывод в формате ESIS (если только не задана опция `-s`). До того как начать грамматический анализ рассматриваемого документа, программа `nsgmls` просматривает «файл-каталог», который описывает, где размещены общеупотребительные SGML-файлы, наборы логических объектов, декларации SGML (см. следующий раздел).

6.6.5.4 Пример вывода ESIS

Как уже объяснялось в разд. 6.6.5.2, программы грамматического анализа для SGML и XML часто обеспечивают возможность получения вывода в формате ESIS, чтобы представить структуру рассматриваемого документа. Ниже будет показан пример вывода, полученного с помощью `nsgmls`. Для ясности рассмотрим вначале значение большинства элементов общего характера в выводном формате ESIS.

`\\` обратная косая черта \.

`\n` *конец записи.*

`\nnn` символ, восьмеричный код которого есть `nnn`.

`(`*gi* начало элемента, родовой (generic) идентификатор которого есть *gi*; атрибуты для этого элемента задаются командой `A`.

`)`*gi* конец элемента, родовой идентификатор которого есть *gi*.

`-data` данные.

`&`*name* ссылка на внешний логический объект данных с именем *name*.

`A`*name key* следующий элемент имеет атрибут *name* с ключевым словом *key* (как описывается в разд. 6.5.4.2).

`?`*pi* обрабатывающие инструкции с данными *pi*.

`N`*nname* обозначение *nname* (ему предшествует команда `p` или `s`, чтобы задать общедоступный или системный идентификатор соответственно).

`s`*sysid* системный идентификатор *sysid*, связанный с некоторыми другими командами (которые ему предшествуют).

`p`*pubid* общедоступный идентификатор *pubid*, связанный с некоторыми другими командами (которые ему предшествуют).

`C` сигналы о том, что рассматриваемый документ представляет собой согласованный (conforming) документ (поэтому такая строка должна быть последней в выводе).

Следующий вывод в формате ESIS получен обработкой файла `emptyexample.xml` (см. с. 316) с помощью программы `nsgmls`. Значение

различных ключевых слов, напечатанных в начале строк, должно быть ясно из списка, приведенного выше.

```
(emptyexample
(par
-The XML logo is shown in the image
Aname TOKEN xml-logo
(imref
)imref
-.
)par
(par
-The image shows an alternative syntax for an empty element \n\012
Aname TOKEN xml-logo
Aaddress CDATA http://www.ucc.ie/xml/xml.gif
(image
)image
)par
)emptyexample
C
```

Аналогично, вывод в формате ESIS, полученный при грамматическом анализе файла `invitation.xml` (см. с. 298), имеет такой вид:

```
?xml version="1.0"
(invitation
(front
(to
-Anna, Bernard, Didier, Johanna
)to
(date
-Next Friday Evening at 8 pm
)date
(where
-The Web Cafe
)where
(why
-My first XML baby
)why
)front
(body
(par
-\n\012I would like to invite you all to celebrate\n\012the birth of
(emph
-Invitation
)emph
-, my\n\012first XML document child.\n\012
)par
(par
-\n\012Please do your best to come and join me next Friday\n\012
evening. And, do not forget to bring your friends.\n\012
)par
(par
-\n\012I
(emph
-really
)emph
- look forward to see you soon!\n\012
)par
)body
(back
(signature
-Michel
)signature
)back
)invitation
C
```

6.6.5.5 Обработка некорректных документов в щадящем режиме

Чтобы показать, как программы грамматического анализа работают с исходными файлами, содержащими ошибки, рассмотрим следующий (неверный) документ — будем называть его для удобства `wrong.xml`.

```
<?xml version="1.0"?>
<!DOCTYPE wrong [
<!ELEMENT wrong (par*)>
<!ELEMENT par  (#PCDATA|emph)*>
<!ELEMENT emph (#PCDATA)*>
]>
<wrong>
<par>This part has wrong entity syntax &lt;par&gt.</par>
<emph>Emph text outside scope.</emph>
<par>Here comes another error <par>a second level
paragraph</par>.</par>
<par>A wrongly nested <emph>construct</par></emph>.
Some more text outside valid scope.
<par>Reserved characters "&" "<" ">" ";" .</par>
</wrong>
```

При обработке этого файла с помощью программы `nsgmls` в режиме XML мы получим следующие сообщения об ошибках (из-за того что некоторые строки получаются слишком длинными, будем переносить их):

```
nsgmls -s -wxml wrong.xml
wrong.xml:8:47:E: general entity "gt." not defined and no default entity
wrong.xml:8:50:W: reference not terminated by refc delimiter
wrong.xml:9:5:E: document type does not allow element "emph" here;
                                  assuming missing "par" start-tag
wrong.xml:10:4:E: document type does not allow element "par" here
wrong.xml:10:34:E: document type does not allow element "par" here
wrong.xml:12:4:E: document type does not allow element "par" here
wrong.xml:12:43:E: end tag for "emph" omitted, but OMITTAG NO was specified
wrong.xml:12:23: start tag was here
wrong.xml:12:50:E: end tag for element "emph" which is not open
wrong.xml:14:4:E: document type does not allow element "par" here
wrong.xml:14:26:W: character "&" is the first character of a delimiter
                                  but occurred as data
wrong.xml:14:30:W: character "<" is the first character of a delimiter
                                  but occurred as data
wrong.xml:15:7:E: end tag for "par" omitted, but OMITTAG NO was specified
wrong.xml:9:0: start tag was here
```

Важно понимать, что программа `nsgmls` продолжает грамматический (синтаксический) анализ (и выдает соответствующую информацию об ошибках) до тех пор, пока не закончится обрабатываемый файл. Некоторые программы грамматического анализа останавливаются при обнаружении первой же *серьезной* ошибки и не дают, соответственно, информации об ошибках в оставшейся части файла.[1] В нашем примере первое сообщение говорит об отсутствии завершающего символа в ссылке на логический объект (точки с запятой ;) для `gt` — в строке исходного текста с номером 8. В строке с номером 9 выявлено

[1] Хотя такая стратегия не выглядит очень полезной, если необходимо обнаружить все ошибки за минимальное время, она совместима с принципом XML — «драконовской» обработкой ошибок. В самом деле, в разделе *Терминология* Спецификации XML [↪XMLSPEC] под заголовком *Фатальная (серьезная) ошибка* утверждается, что согласующийся XML-процессор «не должен продолжать нормальную обработку», как только он обнаружил серьезную ошибку (если он даже и *имеет возможность* продолжить обработку данных, он не должен делать этого).

присутствие пары элементов <emph>...</emph> вне контекста (вне какого-либо элемента par). Программа грамматического анализа пытается «быть умной» и вставляет в строку 9 начальный элемент <par>, однако это означает, что начальный элемент <par> в строке 10 будет (и совершенно правильно) квалифицирован как неверный (элементы par не могут быть вложенными, согласно имеющемуся определению DTD). В строке 12 обнаружено отсутствие завершающего элемента </emph> вблизи элемента par (поскольку элементы опускать в языке XML нельзя, программа-анализатор выдаст сообщение No Omittag). Затем в конце данной строки будет найден элемент завершающего элемента для элемента emph и объявлен неправильным. И опять в строке 14 будет исключен начальный элемент <par>, поскольку (неявно вставленный ранее) начальный элемент <par> в строке 9 все еще остается активным. Строка 13, которая «выпадает из контекста», не вызовет выдачи диагностического сообщения, а «зарезервированные» символы «&» и «<» будут помечены. Затем будет выдано несколько сообщений с жалобами на несбалансированные элементы par. Можно сделать заключение, что большинство из имеющихся в данном тексте ошибок было выявлено, но информация, выходящая за контекст, обнаруживается не всегда. Необходимо попытаться исправить ошибки, начиная с верхней части рассматриваемого файла, и проводить эту работу до тех пор, пока не перестанут появляться сообщения об ошибках и сообщения-предупреждения. Чтобы проследить особенности работы других программ-анализаторов, рассмотрим их работу с тем же самым файлом wrong.xml и покажем сообщение об ошибке, формируемое ими.

6.6.5.6 XML for Java

Программа XML for Java представляет собой программу-анализатор, написанную на языке Java и предназначенную для проверки корректности текстов на XML. Она разработана Кентом Тамурой и Хироси Маруямой из Токийской исследовательской лаборатории фирмы IBM в Японии. Ее можно загрузить из [↪XML4J]. Она осуществляет поддержку DOM и пространств имен, а также предлагает прототип для XPointers [↪XPTR].

Если пропустить файл «wrong.xml» через эту программу-анализатор, получим ряд сообщений, приведенных ниже (следует установить переменную classpath, чтобы информировать интерпретатор языка Java о том, где располагаются требуемые классы). Отметим, что данное приложение использует стандартный вариант языка Java, так что оно может работать на любой компьютерной платформе, где установлена программная среда Java (приводимый ниже вывод был получен в среде Windows NT).

```
set classpath=d:\xml4j\xml4j.jar;d:\jdk1.1.6\src
java trlx wrong.xml
wrong.xml: 8, 51: Reference must end with ';'.
wrong.xml: 8, 51: Undefined entity reference, "&gt.;".
wrong.xml: 11, 22: Element "<par>" is not valid because it does not follow the rule, "(#PCDATA|emph)*".
wrong.xml: 12, 42: "</emph>" expected.
wrong.xml: 12, 43: Element name expected.
wrong.xml: 14, 28: Reference must end with ';'.
wrong.xml: 14, 28: Invalid character, '"', in reference.
```

```
wrong.xml: 14, 31: Element name expected.
wrong.xml: 15, 7: "</par>" expected.
wrong.xml: 15, 8: Element "<par>" is not valid because it does not follow the rule, "(#PCDATA|emph)*".
```

И опять, строка 9, которая «выпадает» из контекста, не обнаружена, а начальный элемент второго уровня `<par>` в строке 10 не помечен. С другой стороны, выявлено неверное информационное содержимое в конце строки 11. Некорректно вложенные элементы `par` и `emph` в строке 12 найдены, и, кроме того, символ точки «.» в конце строки 12 помечен как выходящий за пределы контекста (о строке 13 не сказано ничего). Зарезервированные символы «&» и «<» обнаружены, но вызванное этим обстоятельством сообщение об ошибке оказалось неконкретным. Аналогичным образом, «жалобы», относящиеся к строке 15, не слишком полезны, поскольку они ссылаются на ожидаемый элемент `/par` и на неверный элемент `par`.

6.6.5.7 Обработка событий и программа грамматического разбора Ælfred

Программа Ælfred представляет собой грамматический анализатор, написанный Дэвидом Меггинсоном. Она представляет собой программу-анализатор, написанную на языке Java, не решающую задачу проверки корректности документа, оптимизированную для получения максимальной скорости обработки и минимального размера программы. Она мобильна и может работать с большинством реализаций языка Java. Важное обстоятельство здесь состоит в том, что программа Ælfred поддерживает кодировку Unicode в наиболее полном варианте, доступном для Java. Она корректно работает с XML-документами, представленными с использованием кодовых таблиц UTF-8, UTF-16, Unicode, ISO-10646 и ISO-8859-1, т. е. Ælfred может работать со всеми основными языками.

Программа-анализатор и документация на нее доступны с домашней страницы Ælfred по адресу [↪AELFRED]. Библиотека классов для нее приходит с несколькими примерами классов, демонстрирующими, каким образом некоторое приложение просматривает «события грамматического анализа» для XML. Это значит, что каждый раз, когда анализатор обнаруживает некоторый элемент, атрибут, логический объект, символьные данные и т. д., он выполняет обратный вызов того класса, из которого пользовательское приложение может предпринять требуемое действие (например, записать ESIS-вывод, записать некоторый информационный текст, подготовить вывод на требуемом языке, например на языке TEX). Класс `eventdemo.class`, который поступает в составе дистрибутива программы Ælfred, записывает информацию о каждом «событии» в выходной поток. Ниже дается сформированный вывод для файла-примера `emptyexample.xml` file:

```
1  java -classpath "$CLASSPATH:java/aelfred" EventDemo emptyexample.xml
2  Start document
3  Resolving entity:  [document], pubid=null, sysid=file:emptyexample.xml
4  Starting external entity:  file:emptyexample.xml
5  Doctype declaration:  emptyexample, pubid=null, sysid=null
6  Start element:  name=emptyexample
7  Ignorable whitespace:  "\n"
```

```
 8  Start element:  name=par
 9  Character data:  "The XML logo is shown in the image "
10  Attribute:  name=name, value=xml-logo (specified)
11  Start element:  name=imref
12  End element:  imref
13  Character data:  "."
14  End element:  par
15  Ignorable whitespace:  "\n"
16  Start element:  name=par
17  Character data:
18    "The image shows an alternative syntax for an empty element \n"
19
20  Attribute:  name=name, value=xml-logo (specified)
21  Attribute:  name=address, value=http://www.ucc.ie/xml/xml.gif (specified)
22  Start element:  name=image
23  End element:  image
24  End element:  par
25  Ignorable whitespace:  "\n"
26  End element:  emptyexample
27  Ending external entity:  file:emptyexample.xml
28  End document:  errors=0
```

Чтобы упростить написание кода, который будет осуществлять обработку событий, приходящих от различных программ-анализаторов, Дэвид Меггинсон и Тим Брей (в соавторстве с рядом других авторов) решили, что было бы хорошо определить стандартный API обработки событий для анализаторов XML. Этот API они назвали SAX (*Простой API для XML*, см. [↪SAX]). Эту важную тему мы обсудим в разд. B.6.

6.7 Заключение

В этой главе были рассмотрены некоторые результаты двух десятилетий, предшествовавших приходу XML. Отмечено первое появление понятия разметки текста документа, за которым пришло понимание, что использование более общего подхода сулит много преимуществ. Кульминацией этого процесса через несколько лет стала публикация в 1986 г. стандарта на язык SGML. Затем пришли Сеть и потрясение, вызванное HTML. Наконец, стало понятно, что для того, чтобы Сеть успешно развивалась и оправдывала надежды, возлагаемые на нее как на истинно глобальную среду информационного обмена, хранения данных и работы с ними, необходимо наличие более качественных формальных оснований — в результате родился язык XML.

Мы обсудили некоторые аспекты, связанные с различными компонентами XML. Было показано, как конструировать XML-приложения, оптимизированные для решения определенной задачи, путем определения соответствующего языка, ориентированного на эту задачу, синтаксис которого описывается с помощью определения типа документа (DTD). Обсуждались различные средства, помогающие разрабатывать, анализировать и отлаживать определения DTD. В последнем разделе данной главы было показано, как можно использовать грамматические анализаторы XML для анализа структуры XML-документов. В главе 7 эта информация будет использована для отображения содержимого некоторого XML-документа в различных выходных форматах посредством языков описания стилей страниц.

Глава 7

CSS, DSSSL и XSL: Использование стилей

В этой главе сначала приводится краткий исторический обзор основных языков для описания стилей. Затем объясняется, как можно использовать средства программирования для связывания команд форматирования с XML-документами. В оставшихся разделах данной главы обсуждается ряд вопросов, относящихся к таким языкам описания стилей, как CSS, DSSSL и XSL, и показано, каким образом они связывают стили с XML-элементами. Будет показано, в частности, как сформировать из исходных XML-файлов вывод в форматах HTML и LaTeX.

7.1 Языки описания стилей: краткая история

Как уже говорилось в разд. 6.3.1, когда появились первые языки для автоматизации издательского процесса, команды форматирования и собственно текст документа были перемешаны. Не существовало способа для непосредственного выражения структурных отношений между различными компонентами рассматриваемого документа. Когда были осознаны все неудобства, связанные с этой ситуацией, рядом авторов были разработаны высокоуровневые системы разметки, такие, как GML, LaTeX и Scribe. Структурные команды этих систем представляли собой макро, которые формулировались в терминах команд из низкоуровневого языка для наборных машин (Script, TeX и т. п.). Этот подход представлял собой попытку разделить информационное содержание и форму представления документа, что давало возможность существенно облегчить распространение и повторное использование документов.

Ранее обсуждались языки разметки SGML, XML и HTML, внимание акцентировалось на том, что эти языки не определяют, в принципе, как различные типы элементов документа должны отображаться визуально (на бумаге или экране), в устной форме (например, для людей с ослабленным зрением) и т. п.

Это положение, вообще говоря, не совсем верно для HTML, поскольку этот язык начинал свою жизнь как некий простой язык связывания для информа-

ции, рассеянной исключительно по Сети, в связи с чем и оказались перемешанными структура и форма представления данных. Здесь мы можем иметь дело только с такими элементами, как `FONT`, `B`, или с атрибутами наподобие `ALIGN`, `COLOR` или `WIDTH`, которые сами по себе ничего не делают со структурными отношениями между элементами документа. Такого рода порок подхода, основанного на смешивании средств представления и структурных элементов документа, был обнаружен достаточно быстро, и одним из первых стандартов, который должен был быть разработан Консорциумом W3C, стал CSS1, подготовленный к декабрю 1996 г. ([↪CSS1], Lie and Bos (1997)). Затем данный язык был существенно расширен, и в мае 1998 г. был выпущен его вариант CSS2 [↪CSS2], заменивший предыдущий вариант CSS1. CSS представляет собой декларативный язык, в котором используется набор специализированных обозначений, расширять который у пользователя возможности нет. Истоки такого подхода лежат в HTML, но он может также применяться и в XML, что позволяет сохранить в XML-документе разумную простоту и линейную структуру, которую можно отобразить без утомительных манипуляций.

Кроме того, в SGML-сообществе продолжались работы, нацеленные на поиск путей связывания способов отображения документов с элементами SGML. Важной вехой на этом пути была публикация в 1990 г. спецификации форматирования выходного документа FOSI [↪FOSI]. Этот военный стандарт США, разработанный в рамках проекта CALS, предлагает способ воспроизведения для документа, размеченного с помощью языка SGML. Каждый SGML-документ должен был доставляться вместе с некоторой FOSI-спецификацией, также использующей синтаксис SGML. Эта спецификация содержит значения характеристик для каждого элемента из SGML DTD, в частности, для каждого контекста, в котором определенный элемент использует форматирующие требования, присущие только ему. Задаются также характеристики для атрибутов, влияющих на форматирование.

В рамках ISO в течение многих лет проводились исследования, направленные на разработку стандарта для описания связей общего вида между SGML-документами и их форматированием. Кульминацией этих работ стала публикация стандарта языка спецификаций и семантики стиля DSSSL (ISO/IEC:10179, 1996). Однако спецификация DSSSL была признана слишком сложной для использования ее в документах Интернета, в связи с чем в 1996 г. для использования совместно с XML было предложено подмножество данного языка, получившее наименование DSSSL-online [↪DSSSLONL] и известное также как XS. Однако, как об этом будет сказано в разд. 7.5.2, язык утверждений в DSSSL (и, следовательно, в XS) основан на Scheme — одном из диалектов языка программирования Lisp. Синтаксис языка Scheme выглядит весьма непривычным для большинства пользователей Интернета. В связи с этим, в августе 1997 г. несколькими ведущими «игроками» из мира XML были подготовлены предложения по *расширяемому языку стилей* XSL с синтаксисом, основанным на синтаксисе XML [↪XSL97]. Он включает большую часть функциональных возможностей DSSSL-online и CSS1, однако многие из средств, которые хотелось бы иметь в языке стилей для сложных XML-документов (в противоположность

линейным, обрабатываемым за один проход HTML-документам — цель языка CSS), все еще отсутствуют.

После продолжительного и интенсивного обсуждения, в частности через интернетовские дискуссионные группы, связанные с тематикой XML, такие, как списки рассылки по XSL [↪XSLMAIL] и DSSSL [↪DSSSLLIST], в мае 1998 г. было опубликовано рыезюме требований к XSL [↪XSLREQ]. Вскоре после этого рабочей группой Консорциума W3C была начата работа по формированию первого варианта ядра XSL, реализующего основную часть элементов, введенных в упомянутые выше требования. Ожидается, что завершающий вариант спецификаций для XPath и XSLT будет опубликован в декабре 1999 г.[1], а та часть стилевого языка XSL, которая связана с форматирующими объектами, будет готова несколькими месяцами позже. Более подробно язык XSL в том виде, как он определен сейчас, будет рассмотрен в разд. 7.6.

Если вам потребуется информация о более поздних работах Консорциума W3C в области стилей, в частности по CSS и XSL, ее можно получить на соответствующей Web-странице [↪W3CSTYLE].

Ясно, что CSS и XSL будут, скорее всего, сосуществовать, поскольку они ориентированы на удовлетворение разных потребностей. CSS будет оставаться языком стилей для Web-документов, в особенности для динамического форматирования документов в темпе их поступления применительно к различным средам. С другой стороны, XSL позволит работать со сложными документами, такими, например, что используют верстку в несколько колонок, перекрывающиеся наборы столбцов с многочисленными текстовыми потоками, области размещения сносок, синхронизированные заметки на полях, форматирование математических выражений, смешанные вертикальные и горизонтальные направления письма. Другими словами, XSL представляет собой средство, позволяющее организовать подлинно интернациональный издательский процесс.

Ожидается, что форматирующие части языков XSL и CSS будут совершенно одинаковы. В частности, рабочая группа по XSL гарантирует, что все свойства, основанные на CSS, а также значения в XSL будут иметь то же самое значение, что и в CSS. И наоборот, дополнительные форматирующие функциональные возможности, включенные в XSL, будут показаны и описаны способом, который можно заимствовать из CSS. Важный момент здесь состоит в том, что все форматирующие функциональные средства, описанные в рекомендациях W3C, должны использовать одну и ту же модель нижнего уровня, одну и ту же терминологию. Поэтому рабочей группой Консорциума W3C была подготовлена общая *модель форматирования W3C*. Она будет поддерживать спецификации из всех документов W3C, содержащих описание функциональных средств форматирования, включая средства для HTML, MathML, CSS, XSL, SMIL (Синхронизированный язык интеграции мультимедийных приложений) и SVG (Масштабируемая векторная графика).

[1] В настоящее время Консорциум W3C выпустил спецификации для XPointeq, Xpath и XSLT. — *Прим. ред.*

Более подробную информацию относительно стилей, в особенности применительно к CSS, а также старый синтаксис XSL, ряд материалов по DSSSL-online, можно найти в работе Boumphrey:1998:PSS.

7.2 Программирование или стили — что лучше?

Пользователи SGML-систем с самого начала столкнулись с проблемой печати или просмотра на экране их документов; за прошедшие годы был предложен целый ряд решений этой проблемы. Весьма детальное описание SGML-фильтров для преобразования исходных SGML-документов в формат вывода можно найти в работе SmithNE:1998:SXF, а работа Flynn:1998:USXT представляет собой практическое руководство по значительному числу средств этого рода (как коммерческих, так и доступных свободно), существующих сейчас для SGML и XML. Большая часть этих средств располагает собственным синтаксисом для описания действий, требуемых для воспроизведения элементов SGML в заданном целевом языке. В данной книге рассматриваются коммерческие системы, предоставляющие пользователю развитые среды для работы с SGML; к числу таких систем можно отнести *Balise* [↪BALISE], *Omnimark* [↪OMNIMARK], *SGMLC* [↪SGMLC]. Рассматриваются также свободно доступные средства, основанные на Perl, `awk`, `nsgmls` и т. д. Следует отметить, что в различных текстовых процессорах и средствах для работы с документами, например в программе *FrameMaker* фирмы Adobe [↪FRM], языке PDF этой же фирмы, офисном пакете программ фирмы Microsoft, планируется использовать язык XML как внутреннее средство для сохранения информации о структуре документа. Ожидается, что такая мера существенно упростит обмен электронными документами между приложениями, созданными различными разработчиками.

Большинство фильтров, решающих задачу преобразования XML-документов в некоторый формат вывода для просмотра на экране, печати или каким-то иным способом, доступно в рамках средств, обеспечивающих программирование для Интернета. По этой причине начнем наш обзор с рассмотрения одной простой системы, основанной на языке Perl, с помощью которой можно программировать вид вывода для каждого из типов элементов в исходном XML-документе. В разделе B.6 будет рассмотрен более общий подход, базирующийся на использовании языка Java.

В первых примерах, рассматриваемых ниже, в качестве целевого языка будут приниматься непосредственно LaTeX и HTML, поскольку таким способом проще продемонстрировать, что же надо получить. Однако, как будет разъяснено позже, имеется также и более общий подход, основанный на понятии *форматирующих объектов*, который почти всегда лучше, поскольку на основе какого-либо одного листа стилей можно задавать разные выходные форматы простой заменой программы постпроцессорной обработки, исполняющейся

на машине конечного пользователя. Таким образом, выбор между подходами, основанными на языках программирования или на стилях, а также между использованием потоковых объектов или прямого языкового форматирования в терминах целевого языка надо осуществить до начала реализации некоторого проекта. Ниже будут показаны различные возможности для выполнения этих действий, мы надеемся, что приведенных данных и соображений будет достаточно, чтобы у вас возникло ощущение того, что надо делать в сложившейся ситуации.

7.3 Форматирование с использованием языка Perl

В разд. 6.6.5.2 объяснялось, каким образом программы грамматического анализа и разбора текстов на языке XML формируют ESIS-вывод для представления структуры данного конкретного документа. Дэвид Меггинсон разработал программу SGMLSpm [↪SGMLSPM] — расширяемую библиотеку классов Perl5 для обработки вывода из программ грамматического анализа `sgmls` и `nsgmls`. Дистрибутив этой библиотеки содержит простое модельное приложение `sgmlspl`, показывающее, как следует использовать данную библиотеку классов.

Для преобразования SGML-документов в документы другого формата на основе *файла спецификации* можно использовать прикладную программу `sgmlspl`, где детально определяется, как работать с каждым из типов элементов, внешних объектов, строк `CDATA` и т. п. С дистрибутивом этой программы поставляются два примера SGML-файлов — `sgmlspm.sgml`, описывающий библиотеку классов, и `sgmlspl.sgml`, описывающий само приложение `sgmlspl`. Эти файлы размечены согласно DocBook DTD. Есть также два *файла спецификации*, написанные на языке Perl: `tolatex.pl` и `tohtml.pl`, обеспечивающие преобразование DocBook-разметки, использованной в упомянутых двух SGML-файлах, в разметку LaTeX'а и HTML соответственно.

Чтобы показать, как работает эта процедура, используем приложение `sgmlspl` для подготовки документов в форматах LaTeX и HTML для примера `invitation`, введенного в разд. 6.4.2.2 и использовавшегося неоднократно в гл. 6.

7.3.1 Принципы работы

Приложение `sgmlspl` для извлечения ESIS-представления из некоторого документа использует *событийную* модель. Это значит, что каждый раз, когда программа грамматического разбора XML-документа обнаруживает какое-либо ESIS-«событие», отвечающее некоторой конфигурации в исходном XML-файле, программа `sgmlspl` «предпримет» некоторое действие, выраженное как некий

набор инструкций на языке Perl, определенный в файле *спецификации*. В этом файле, который импортирует модуль классов `SGMLSpm` Perl5, можно определять Perl-пакеты и программы, читать файлы, создавать переменные. Однако в большинстве случаев можно ограничиться добавлением простого Perl-кода. Более того, в интересах эксплуатационной технологичности и ортогональности хорошей практикой будет помещать все низкоуровневые команды в выводные файлы, ориентированные на тот или иной конкретный язык (в частности, для пакетов LaTeX или стилевых файлов CSS).

В состав дистрибутива входит файл-образец `skel.pl`, который формирует файл спецификации, содержащий «заглушки» для всех элементов в исходном XML-документе. Этот модуль на языке Perl работает с любой программой грамматического разбора, порождающей вывод в формате ESIS. Для нашего примера воспользуемся программой `nsgmls`, написанной Джеймсом Кларком, и исполним с ее помощью такую командную последовательность:

```
nsgmls invitation.xml | perl sgmlspl.pl skel.pl > invitation.pl
```

Первая часть этой команды формирует выходной поток ESIS, который направляется на вход Perl-приложения `sgmlspl.pl`. Это приложение, в свою очередь, управляется посредством файла `skel.pl`. Выход этой цепочки программ записывается в выводной файл `invitation.pl`. Точное информационное содержимое этого файла-образца, скорректированного с учетом потребностей конкретной задачи, зависит от конкретного документа, поскольку в файл записываются новые процедурные ссылки для каждого отдельного SGML-события в исходном тексте. Давайте посмотрим, что же именно содержится в данном файле.

```
1  ##############################################################################
2  # SGMLSPL script produced automatically by the script sgmlspl.pl
3  #
4  # Document Type: invitation
5  # Edited by:
6  ##############################################################################
7
8  use SGMLS;                      # Use the SGMLS package.
9  use SGMLS::Output;              # Use stack-based output.
10
11 #
12 # Document Handlers.
13 #
14 sgml('start', sub {});
15 sgml('end', sub {});
16
17 #
18 # Element Handlers.
19 #
20
21 # Element: invitation
22 sgml('<invitation>', "");
23 sgml('</invitation>', "");
24
25 # Element: front
26 sgml('<front>', "");
27 sgml('</front>', "");
28
29    [ ... ]
```

Этот файл содержит строку, в которой вызывается процедура `sgml` для каждого начального дескриптора (строки 22 и 26) и завершающего дескриптора (строки 23 и 27) элементов в исходном XML-документе. В конце данного файла заданы значения по умолчанию для обработчиков других событий, которые могут возникнуть при грамматическом разборе некоторого XML-документа и таких его элементов, как символьные данные, объекты, обрабатывающие инструкции.

7.3.2 Формирование конкретного документа на LaTeX'е

Ограничимся пока LaTeX'ом и подготовим некоторый вариант файла-образца для трансляции XML-файла в LaTeX'овский файл, который можно было бы распечатать. Желательно при этом, чтобы на уровне языка Perl употреблялось как можно меньше низкоуровневого LaTeX'овского кода, так что мы здесь просто сохраним необходимую информацию в некотором наборе переменных и пусть со всеми деталями форматирования имеет дело стилевой файл `invitation.sty`. Ниже дается вариант `inv2lat.pl` файла-примера `invitation.pl` для случая LaTeX'а:

```
1  ##############################################################################
2  # SGMLSPL script produced automatically by the script sgmlspl.pl
3  #
4  # Document Type: invitation --> customization for LaTeX
5  # Edited by: mg (August 14th 1998)
6  ##############################################################################
7
8  use SGMLS;                      # Use the SGMLS package.
9  use SGMLS::Output;              # Use stack-based output.
10
11 #
12 # Document Handlers.
13 #
14 sgml('start', sub {});
15 sgml('end', sub {});
16
17 #
18 # Element Handlers.
19 #
20
21 # Element: invitation
22 sgml('<invitation>', "\\documentclass[]{article}\n" .
23                      "\\usepackage{invitation}\n" .
24                      "\\begin{document}\n");
25 sgml('</invitation>', "\\end{document}\n");
26
27 # Element: front
28 sgml('<front>', "\\begin{Front}\n");
29 sgml('</front>', "\\end{Front}\n");
30
31 # Element: to
32 sgml('<to>', "\\To{");
33 sgml('</to>', "}\n");
34
35 # Element: date
36 sgml('<date>', "\\Date{");
37 sgml('</date>', "}\n");
38
39 # Element: where
40 sgml('<where>', "\\Where{");
41 sgml('</where>', "}\n");
```

```
42
43  # Element: why
44  sgml('<why>', "\\Why{");
45  sgml('</why>', "}\n");
46
47  # Element: body
48  sgml('<body>', "\\begin{Body}\n");
49  sgml('</body>', "\\end{Body}\n");
50
51  # Element: par
52  sgml('<par>', "\\par ");
53  sgml('</par>', "\n");
54
55  # Element: emph
56  sgml('<emph>', "\\emph{");
57  sgml('</emph>', "}");
58
59  # Element: back
60  sgml('<back>', "\\begin{Back}\n");
61  sgml('</back>', "\\end{Back}\n");
62
63  # Element: signature
64  sgml('<signature>', "\\Signature{");
65  sgml('</signature>', "}\n");
66  #
67  # Default handlers
68  #
69  sgml('start_element',sub { die "Unknown element: " . $_[0]->name; });
70  sgml('cdata',sub { output $_[0]; });
71  sgml('re'," ");
72  sgml('pi',sub { die "Unknown processing instruction: " . $_[0]; });
73  sgml('entity',sub { die "Unknown external entity: " . $_[0]->name; });
74  sgml('conforming','');
75
76  1;
```

Ясно видно, как внешний элемент документа `invitation` приводится в соответствие с правилами инициализации LaTeX'овских документов там, где загружается пакет `invitation` (строки 22–24). Элементы `front`, `body` и `back` становятся окружениями `Front`, `Body` и `Back` (строки 28–29, 48–49 и 60–61 соответственно). Большинство остальных элементов преобразуется в высокоуровневые LaTeX'овские команды с тем же самым именем (строки 33–45 и 64–65). Только лишь для XML-элементов `par` и `emph` использованы явные базисные LaTeX'овские эквиваленты `\par` (строки 52) и `\emph` (строки 56–57).

Теперь надо приготовить сценарий на языке Perl, что можно сделать, выполнив такую команду:

```
nsgmls invitation.xml | perl sgmlspl.pl inv2lat.pl > invitation.tex
```

в результате чего будет получен следующий LaTeX'овский файл:

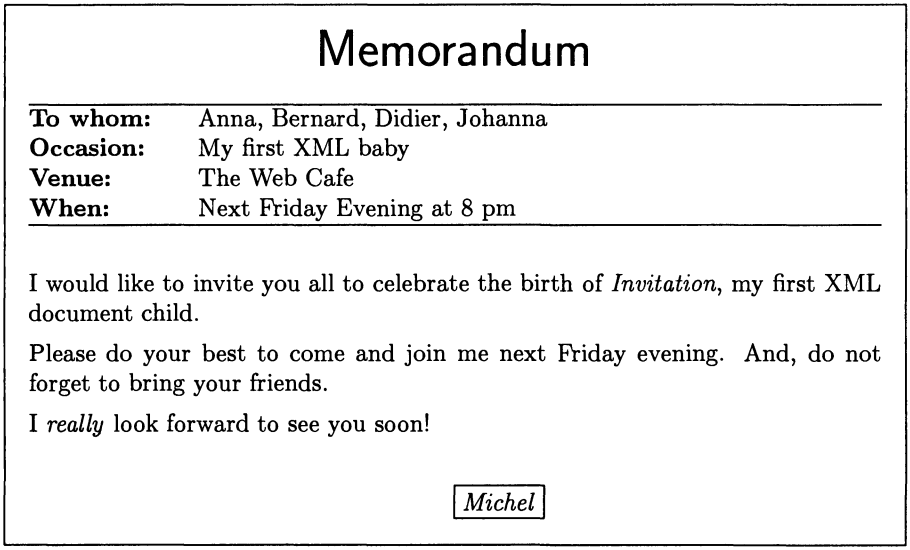
```
1   \documentclass[]{article}
2   \usepackage{invitation}
3   \begin{document}
4   \begin{Front}
5   \To{Anna, Bernard, Didier, Johanna}
6   \Date{Next Friday Evening at 8 pm}
7   \Where{The Web Cafe}
8   \Why{My first XML baby}
9   \end{Front}
10  \begin{Body}
11  \par I would like to invite you all to celebrate
12  the birth of \emph{Invitation}, my
13  first XML document child.
```

```
14  \par Please do your best to come and join me next Friday
15  evening. And, do not forget to bring your friends.
16  \par I \emph{really} look forward to see you soon!
17  \end{Body}
18  \begin{Back}
19  \Signature{Michel}
20  \end{Back}
21  \end{document}
```

Перед тем как отформатировать этот файл с помощью LaTeX'а, надо взглянуть на LaTeX'овский файл-макропакет `invitation.sty`. В принципе, благодаря использованию команд высокого уровня можно форматировать приведенную выше LaTeX'овскую разметку многими способами. В примере конкретного документа, приводимом ниже, будет выбрана одна из возможных реализаций. Небезынтересно, по-видимому, будет прояснить некоторые моменты, связанные с этим примером. Для набора материала из заглавной части `front` рассматриваемого примера здесь используется окружение `tabular` (строки 10–16). Кроме того, в конце примера (строки 25–29) информационное содержимое компонент, входящих во фрагмент `front`, а также в подпись (signature) запоминается в *глобальных* переменных (`\gdef`)[1], так что их значение может быть использовано внутри таблицы (строки 12–15) и в элементе `signature`, заключенном в рамку в конце документа (строка 22). Достаточно ясно, каким образом можно изменить показанный здесь программный код. *Безо всякой модификации* исходного XML-текста или сценария на языке Perl можно распорядиться организовать выдачу документа почти в любом виде, необходимом пользователю.

```
1   % invitation.sty
2   % Package to format invitation.xml
3   \setlength{\parskip}{1ex}
4   \setlength{\parindent}{0pt}
5   \pagestyle{empty}%% Turn off page numbering
6   \RequirePackage{array}
7   \newenvironment{Front}%
8     {\begin{center}\huge \sffamily Memorandum\end{center}
9      \begin{flushleft}
10     \begin{tabular}{@{}>{\bfseries}p{.2\linewidth}@{}p{.8\linewidth}@{}}\hline
11    }
12    {To whom:  & \@To     \\
13     Occasion: & \@Why    \\
14     Venue:    & \@Where  \\
15     When:     & \@Date   \\\hline
16     \end{tabular}
17     \end{flushleft}
18    }
19  \newenvironment{Body}{\vspace*{\parskip}}{\vspace*{\parskip}}
20  \newenvironment{Back}
21    {\begin{flushleft}}
22    {\hspace*{.5\linewidth}\fbox{\emph{\@Sig}}
23     \end{flushleft}
24    }
25  \newcommand{\To}[1]{\gdef\@To{#1}}
26  \newcommand{\Date}[1]{\gdef\@Date{#1}}
```

[1] Здесь `\gdef` TeX — команды, определяющие последовательность инструкций, выдаваемую сразу после значения ее аргумента. Для удобства внутри LaTeX'овских файлов, содержащих макропакеты и описания классов, литера `@` используется как одна из букв при описании имен внутренних команд, связанных с командами, которые может вызывать пользователь (см. Goossens:LC94, с. 15–16).



Memorandum

To whom:	Anna, Bernard, Didier, Johanna
Occasion:	My first XML baby
Venue:	The Web Cafe
When:	Next Friday Evening at 8 pm

I would like to invite you all to celebrate the birth of *Invitation*, my first XML document child.

Please do your best to come and join me next Friday evening. And, do not forget to bring your friends.

I *really* look forward to see you soon!

Michel

Рис. 7.1. XML-файл, отформатированный с помощью LaTeX'а с использованием процедуры `sgmlspl`.

```
27  \newcommand{\Where}[1]{\gdef\@Where{#1}}
28  \newcommand{\Why}[1]{\gdef\@Why{#1}}
29  \newcommand{\Signature}[1]{\gdef\@Sig{#1}}
```

После компиляции рассматриваемого LaTeX'овского файла с помощью TeX'а, используя данный экземпляр пакета, получим результат, показанный на рис. 7.1. Рамка здесь не является частью исходного документа, она добавлена для большей ясности.

7.4 Каскадные стили

Вторая редакция спецификации каскадных стилей CSS2 [↪CSS2], опубликованная в мае 1998 г., представляет собой описание языка стилей, предназначенного для установления связей стилей (визуализации информации) со структурированными документами (первоначально на языке HTML, впоследствии также и на XML). Важно понимать, что модель CSS позволяет как разработчику документа, так и его пользователям вмешиваться в процесс управления представлением документа. С одной стороны, авторы и издатели предпочитают часто определять набор стилевых характеристик, которые обеспечивали бы их публикациям вид узнаваемый и непохожий на другие. В то же время читатели могут иметь на этот счет свое мнение, руководствуясь своими личными вкусами, а также исходя из ограничений, накладываемых имеющимися в их распоряжении аппаратными средствами и программным обеспечением, или же

какими-либо другими физическими ограничениями (дальтонизм, ослабленное зрение и т. п.)

Следовательно, имеется потребность уметь разрешать конфликты между различными стилями. Именно в этой сфере в CSS возникает и используется *каскадирование*. В языке CSS имеются возможности присваивать явным или неявным образом приоритеты каждому из стилевых элементов. Выигрывает тот стиль, который имеет наивысший приоритет. По умолчанию, авторский стиль перекрывает декларации в пользовательском стиле. Однако как автор, так и пользователь могут связать с некоторыми правилами ключевое слово `!important`, и тогда предпочтение будет отдано пользовательским спецификациям.

В дальнейшем будут рассмотрены те аспекты спецификаций CSS2, которые требуются для выполнения простых операций визуализации XML-документов. Более подробно с этими вопросами можно ознакомиться в полной спецификации или же в одной из многих книг, посвященных языку CSS.

7.4.1 Базовая структура стиля в CSS

Лист стиля языка CSS состоит из списка *операторов*. Они могут быть в одной из двух форм: *at-правил* и *наборов правил*.

At-правила начинаются с литеры `@`, за которой непосредственно следует идентификатор (например, `@import`, `@charset`, `@media`), и завершаются точкой с запятой. Программное обеспечение, работающее с CSS, должно игнорировать at-правила, которые оно не распознает. Особенно интересно правило `@import`, которое позволяет стилевым правилам быть импортированными из других листов стилей. Все правила `@import` должны предшествовать всем наборам правил в стиле, они должны содержать ссылку на соответствующий адрес URI. Например,

```
@import "mystyle.css";
```

будет импортировать CSS-стиль с именем `mystyle.css`.

Набор правил состоит из некоторого *селектора*, за которым идет *блок деклараций*. Этот блок начинается с левой фигурной скобки {, за которой следует список *деклараций*, разделенных запятыми, заканчивается он правой фигурной скобкой }. Селектор, синтаксис которого будет более подробно рассмотрен ниже, во всех случаях предшествует открывающейся фигурной скобке блока деклараций. Если CSS-приложение не располагает возможностями осуществить грамматический разбор произвольного селектора, оно должно игнорировать полностью весь набор правил, т. е. все декларации из блока деклараций.

```
selector {declaration 1; declaration 2, ...}
```

Любая декларация состоит из *свойства (property)*, затем идет двоеточие (:), а за ним — некоторое *значение (value)*.

```
selector {property 1: value 1; property 2: value: 2}
```

Любое свойство представляет собой некоторый идентификатор. Синтаксис *значения* зависит от рассматриваемого свойства. CSS-приложения должны игнорировать декларации с неверным именем свойства или с синтаксически некорректным значением.

7.4.1.1 Селекторы

Селектор представляет собой некоторый образец для выбора элементов в дереве документа, к которым должно быть применено данное стилевое правило. Если какой-либо элемент удовлетворяет всем условиям образца, то тогда говорится, что данный селектор *согласуется* с данным элементом.

Любой селектор представляет собой цепочку из одного или нескольких *простых селекторов*, разделенных символами-комбинаторами (пробел, > или +).

Простой селектор — это или селектор типа элемента (селектор, связанный с конкретным типом элемента), или *универсальный селектор*, за которым непосредственно следуют в произвольном порядке нуль или более *селекторов атрибутов*, id-селекторы или *псевдоклассы*.

Процесс выбора может быть конкретизирован путем приписывания дополнительного простого селектора или комбинатора к некоторому существующему селектору. Последний введенный простой селектор также может иметь один приписанный к нему псевдоэлемент.

Несколько деклараций, имеющихся для одного и того же селектора, можно сгруппировать. Например,

```
front {font-size: 12pt}
front {font-style: bold}
front {text-indent: 0pt}
```

эквивалентно следующей записи

```
front {font-size: 12pt; font-style: bold; text-indent: 0pt}
```

Аналогично, декларации, идентичные для нескольких селекторов, также можно сгруппировать. Тогда следующие четыре оператора:

```
date   {text-align: left}
to     {text-align: left}
where  {text-align: left}
why    {text-align: left}
```

можно свернуть в единственное правило:

```
date, to, where, why {text-align: left}
```

Универсальный селектор (*) согласуется с именем *любого* единичного элемента в дереве документа. Некоторый селектор *типа* согласуется с именем по крайней мере *одного* типа элемента в дереве документа.

Отношения предшествования удобно выражать с помощью перечисления.

Например, запись вида «A D» устанавливает, что элемент типа D может быть потомком элемента-предка A.

```
front emph {font-style: italic}
body  emph {font-style: italic; color: blue}
```

Первые правила определяют, что элементы emph с предком front отображаются с использованием курсивного шрифта, тогда как элементы emph с предком body должны выводиться курсивным шрифтом и окрашиваться в синий цвет.

Прямые отношения *предок—потомок* выражаются в нотации >. Например,

```
body > par > emph {font-style: italic; color: red}
```

задает, чтобы элементы emph, имеющие предка par, предком которого, в свою очередь, является элемент body, должны отображаться литерами красного цвета в курсивном начертании.

Смежные вершины дерева, имеющие общего предка, можно выбрать с помощью обозначения +. Например, текст, связанный с элементом why, которому непосредственно предшествует элемент where в дереве документа, будет напечатан шрифтом желтого цвета, если воспользоваться такой декларацией:

```
where + why {background-color: yellow}
```

7.4.1.2 Работа с атрибутами

Можно также установить связь некоторого правила с каким-либо элементом, для которого один из атрибутов имеет некоторое заданное значение. Синтаксис CSS дает несколько способов выбора элементов. Наиболее важный из них иллюстрируется следующим примером:

```
1  invitation[to] {font-size: 14pt}
2  invitation[signature="Peter"] {text-align: right}
3  invitation[why~="birthday"][date~="Friday"] {background-color: green}
```

В строке 1 выбираются все элементы invitation с атрибутом to независимо от значения, установленного для этого атрибута. В строке 2 выбираются элементы invitation, для которых атрибут signature *в точности* имеет значение Peter. Наконец, в строке 3 выбираются элементы invitation с атрибутом why, содержащим строку «birthday», и атрибутом date, который содержит строку «Friday».

Чтобы установить разницу между значениями атрибутов по умолчанию и заданными явно, можно задать общее правило для некоторого элемента и затем специализировать его путем связывания с некоторым атрибутом.

```
1  invitation {text-align: left}
2  invitation[why] {text-align: center}
```

Общее правило в этом примере (строка 1) задает, что элементы invitation должны выравниваться по левому краю, а элементы invitation с атрибутом why следует центрировать (строка 2).

Имеется возможность выбрать требуемые элементы в дереве документа с помощью селектора id, а также связать с ними специфическую стилевую информацию. В этом случае в определении DTD должны содержаться декларации для id-атрибутов рассматриваемых элементов. В частности, в HTML 4 DTD объявлены id-атрибуты для всех их элементов. Таким образом, id-атрибут представляет собой прямой путь для связывания конкретного стиля с некоторым элементом. Синтаксис, используемый в стиле, должен предшествовать соответствующему определению с символом «решетка» (#), например

```
1  *[color~='myblue'] {color="blue"}
2  sect1#specialblue  {color="navyblue"}
```

Строка 1 в этом примере определяет характеристики *всех* элементов (здесь используется универсальный селектор *) с атрибутом color, значение которого myblue. В строке 2 осуществляется выбор между типами элементов sect1, один из которых имеет атрибут id, равный specialblue.

7.4.1.3 Псевдоклассы и псевдоэлементы

Спецификация CSS2 позволяет осуществлять более тонкую селекцию данных, содержащихся в документе, используя *псевдоэлементы* (например, первую букву абзаца) и *псевдоклассы* (выбирая элементы по их характеристикам, которые обычно не могут быть извлечены логическим путем из имени элемента, его атрибута или его информационного содержимого). Ниже показано несколько примеров такого рода:

```
1  body > par:first-child           {text-indent: 0pt}
2  body * emph                      {color: black}
3  section > par:first-child emph   {color: red}
```

В строке 1 элементу par, являющемуся первым потомком — дочерним элементом элемента body, присваивается нулевое значение абзацного отступа. В строке 2 говорится, что все элементы emph, для которых предком является элемент body, должны отображаться в черном цвете. Наконец, строка 3 требует, чтобы все элементы emph внутри «первого потомка — дочернего элемента» — элемента par внутри элемента section должны быть красного цвета.

Другие примеры псевдоклассов:

:link непосещавшаяся ссылка;

:visited посещавшаяся ссылка;

:active элемент, активизированный пользователем;

:focus элемент, обладающий фокусом (обрабатывающий события).

Более детальное рассмотрение затронутых здесь вопросов см. в разд. 5.11 спецификации CSS2.

Аналогично, псевдоэлементы позволяют выделить первую строку или первую букву некоторого элемента документа или же сформировать текст. Подробности см. в разд. 5.12 спецификации CSS2.

7.4.1.4 HTML-элементы span и div, атрибут class

На сегодняшний день XML-модель документа поддерживается лишь небольшим числом навигаторов, так что большинство документов приходится преобразовывать для отображения их в формат HTML. Поскольку набор атрибутов языка HTML фиксирован, HTML-сообщество выработало удобный путь «расширения» данного языка путем введения в него элементов типов `span` и `div`, а также атрибута `class`. Эта комбинация средств будет использоваться в следующих разделах при рассмотрении примеров форматирования на базе CSS, поэтому надо объяснить соответствующие базовые понятия.

CSS—HTML-*классы*[1] позволяют создавать группирующие схемы для стилизованных типов элементов HTML путем специализации определений стилей для определенных классов с учетом общего (родового) определения для одних и тех же типов элементов, которые не принадлежат данному классу. В стилях CSS1 имени класса предшествует точка (.), а в CSS2 этот формат не поощряется и заменен записью в атрибутной нотации вида `[class~=value]` для идентификации команд, специфических для некоторого класса.

Например, в листе стиля можно было бы определить следующее:

```
par {font-family: serif; font-size: 10pt}
to  {font-family: sans-serif; font-weight: bold}
[class="red"]     {color: red}   /* generic class specification    */
to[class="green"] {color: green} /* class specification to element */
.blue {color: blue}  /* Deprecated syntax, HTML only, DO NOT USE!  */
```

Эти спецификации используются аналогично таким (следует напомнить, что в XML DTD должны быть заданы элементы `class` для каждого обрабатываемого элемента):

```
<par>Serif 10pt font.</par>
<par class="blue">Serif 10pt but in blue.</par>
<to class="red">Sans-serif, bold and in red.</to>
<to class="green">Sans-serif, bold and in green.</to>
```

В языке HTML 4 элемент `span` позволяет контролировать стиль встроенных текстовых фрагментов, например

```
<p>Usual style with a <span class="spec">bit of text rendered using
the ''spec'' style rule.</span> Back to the previous style.</p>
```

С другой стороны, элемент `div` в HTML 4 позволяет задать стиль целого блока текста, включающего другие HTML-элементы, как это имеет место, например, в следующем примере:

```
<div class="mydiv">
<p>The style ''mydiv'' controls this whole text block,
including <q>this quote</q> and <cite>this citation</cite>.</p>
```

[1] Этот метод может быть также использован в XML объявлением атрибута `class` для каждого элемента в DTD, как и в случае с HTML 4 DTD.

```
<p>It even extends over several paragraphs, since here also
the same ''mydiv'' style rules.</p>
</div>
```

В HTML атрибуты `class`, `id` и `style` могут использоваться с элементами `<span>` и `<div>`. Это учитывается в механизме неявного расширения для HTML — разрешается определять логические контейнеры (объекты, содержащие другие объекты) и применять к их содержимому специализированный стиль, созданный пользователем.

7.4.2 Связывание стилей с документом

В HTML *элементы* `<LINK>` или `<STYLE>` внутри элемента `HEAD` рассматриваемого документа используются для связывания стиля с конкретным документом. Более того, можно локально использовать *атрибуты* `STYLE` для пользовательской подстройки способа отображения единичного элемента. Такой подход, однако, не одобряется, поскольку ведет к смешиванию формы и содержания в документе. Намного лучше для связывания стилевой информации с одним или более элементов документа будет использование атрибута `id`.

В работе Джеймса Кларка, опубликованной в серии W3C Note [↪XMLSTYLE], предлагается использовать на уровне документа обрабатывающую команду языка XML с целевым объектом `xml-stylesheet`, чтобы обеспечить те же самые функциональные возможности, что и в элементе `<LINK>` языка HTML. Например, семантика элемента HTML 4 `<LINK>`:

```
<LINK href="mystyle.css" rel="stylesheet" type="text/css">
```

отвечает такой обрабатывающей команде языка XML:

```
<?xml-stylesheet href="mystyle.css" type="text/css"?>
```

Разрешается использовать набор обрабатывающих команд посредством атрибута `alternate` для обрабатывающей инструкции `xml-stylesheet`. Атрибут `title` в рассматриваемом примере является необязательным.

```
<?xml-stylesheet alternate="yes"     title="alt1"
                 href="special1.css" type="text/css"?>
<?xml-stylesheet alternate="yes"     title="alt2"
                 href="special2.css" type="text/css"?>
<?xml-stylesheet href="generic.css"  type="text/css"?>
```

7.4.3 Краткий перечень свойств CSS

Язык CSS2 располагает более, чем сотней свойств (см. приложение F в спецификации CSS2, где дан их перечень в табличном виде). В следующих разделах будут бегло рассмотрены наиболее часто используемые свойства — те, с которыми придется близко соприкоснуться большинству пользователей LaTeX'а, и те, что будут использованы в примерах.

7.4.3.1 Установка значений цветов фона и переднего плана

Свойство `color` управляет цветом переднего плана (т. е. цветом собственно изображения) некоторого элемента, а свойство `background-color` — цветом фона, на котором дается изображение некоторого элемента.

Цвета задаются с помощью предопределенных имен каждого из них, или с использованием тройки чисел `rgb` (red-green-blue) — координат в цветовом пространстве. Например, для элемента `box` цвета текста и фона можно было бы задать следующими (эквивалентными) способами:

```
1  box {color: red; background-color: yellow}
2  box {color: #FF0000; background-color: #FFFF00}
3  box {color: rgb(255,0,0); background-color: rgb(255,255,0)}
4  box {color: rgb(100%,0,0); background-color: rgb(100%,100%,0)}
```

В строке 1 используется одно из шестнадцати имен цветов, предопределенных в языках HTML и CSS. В строках 2–4 цвета выражаются в функции цветовой модели `rgb`, строки 2–3 как числа из диапазона от нуля (нет такой компоненты цвета) до 255 (полная насыщенность соответствующего цвета). В частности, строка 2 использует шестнадцатеричную нотацию, а в строке 3 выбрано десятичное обозначение. Строка 4 выражает те же самые цвета в относительных единицах (в процентах от полной насыщенности соответствующего цвета).

7.4.3.2 Шрифты

`font-family` Название гарнитуры или шрифтового семейства. Это может быть какое-либо конкретное имя, такое как Baskerville или Helvetica, или же одно из обобщенных имен: `serif`, `sans-serif`, `monospace`, `cursive` или `fantasy`.

`font-style` Начертание шрифта: `normal`, `italic` или `oblique`.

`font-variant` Разновидность шрифта: `normal` или `small-caps`.

`font-stretch` Мера «сжатости» или «растянутости» литер шрифта.

`font-weight` Насыщенность шрифта. Значение этого параметра — ключевое слово (`bold`, `lighter`, ...) или число из ряда 100, 200, ..., 900 (чем больше число, тем более насыщен шрифт).

`font-size` Размер шрифта. Он может быть задан в абсолютных (12pt, `x-small`) или относительных (`larger`, `120%`, ...) единицах.

Ниже приведены примеры определений шрифтов для набора различных видов абзацев. Видно, что насыщенность шрифта, размеры литер по горизонтали и вертикали для него могут быть выражены в абсолютных единицах или же относительно значений тех же самых характеристик заданного базового родительского элемента.

```
1  P[class="normal"]
2    {font-family: serif}
3  P[class="italic"]
4    {font-family: serif; font-style: italic}
```

```
P[class="bold"]
  {font-family: serif; font-weight: bold}
P[class="explight"]
  {font-family: serif; font-weight: lighter; font-stretch: expanded}
P[class="ttsc14"]
  {font-family: monospace; font-variant: small-caps; font-size: 14pt}
P[class="ssolarger"]
  {font-family: sans-serif; font-style: oblique; font-size: larger}
```

Существует также сокращенное обозначение «font», которое устанавливает значения параметров font-style, font-variant, font-weight, font-size, line-height и font-family за один шаг — обычная практика в традиционной издательской технологии. Все свойства, не заданные явным образом, получают значения, установленные для них по умолчанию. Рассмотрим следующий пример:

```
P {font: normal small-caps bigger/120% Helvetica}
```

Здесь параметру font-variant присваивается значение «small-caps» (капитель), параметру font-size — относительное значение «bigger», т. е. «больше», чем у порождающего (родительского) элемента, параметру line-height — значение «120%» от этого размера шрифта (используя обычное издательское обозначение *font-size/line-height*), а параметру font-family — значение «Helvetica». Двум оставшимся параметрам, font-style и font-weight, присваивается значение, выражаемое ключевым словом «normal».

7.4.3.3 Текстовое и визуальное форматирование

line-height Минимальная высота каждого из сформированных встроенных блоков. Этот параметр используется также, чтобы задать нормальную величину интервала между строками текста.

text-indent Отступ первой строки в текстовом блоке.

text-align Выравнивание встроенного информационного содержимого текстового блока: left (выравнивание по левому краю), right (выравнивание по правому краю), center (центрирование) или justify (выравнивание по левому и правому краю).

text-decoration Оформление текстовых строк: underline (подчеркнуть снизу), overline (линия над текстовой строкой), blink (мерцающая строка), line-through (перечеркнутая строка).

letter-spacing Величина пробела между литерами в тексте.

word-spacing Величина пробела между словами.

text-transform Управление выделением с помощью заглавных букв: capitalize (печатать с прописной), uppercase (печатать прописными буквами), lowercase (печатать строчными буквами), none.

vertical-align Вертикальное выравнивание встроенных блоков: baseline (по базовой линии), top (вверх), middle (центрировать по высоте), bottom (вниз) и т. д.

Возможный пример оформления текста заголовка выглядит следующим образом:

```
title {text-decoration: underline; text-transform: uppercase;
       letter-spacing: .1em; word-spacing: .5em}
```

Текст элемента `title` будет набран заглавными буквами и подчеркнут. Пробелы между отдельными буквами будут *увеличены* на .1 em по сравнению с нормальным значением этого параметра для данного шрифта, тогда как к значению пробела между словами по умолчанию будет *добавлено* .5 em.

7.4.3.4 Блоки

Модель форматирования в языке CSS основывается на прямоугольных блоках, формируемых из элементов дерева документа. У каждого блока есть информационная область, окруженная, возможно, полями, границей или некоторыми символами-заполнителями. Свойства блока управляют размерами и характеристиками этих прямоугольников и определяют, каким образом следует форматировать информационное содержимое рассматриваемого элемента, чтобы вписаться в отведенное элементу место.

Ниже перечислены следующие свойства:

`margin-top`, `margin-bottom`, `margin-left`, `margin-right`, `margin`
: управляют размером верхнего, нижнего, левого, правого и всех полей соответственно;

`padding-top`, `padding-bottom`, `padding-left`, `padding-right`, `padding`
: задают символы заполнения для верхней, нижней, левой, правой и всех сторон рассматриваемого элемента соответственно;

`border-top`, `border-bottom`, `border-left`, `border-right`, `border`
: задают ширину, стиль, цвет границы с верхней, нижней, левой, правой и всех сторон данного элемента соответственно.

Имеются более специфические свойства блока: `border-width` (ширина-границы), `border-style` (стиль-границы) и `border-color` (цвет-границы) для того, чтобы задавать эти характеристики независимо друг от друга.

Ширина и высота информационного поля блока задаются свойствами `width` (высота) и `height` (ширина) для масштабирования изображений, например. Можно также сдвинуть материал блока относительно текущей строки, используя свойство `float`, которое может принимать значения `left` (влево), `right` (вправо) или `none`, т. е. отсутствие данного свойства.

В качестве примера можно взять следующий элемент-список `itemize`, а также элемент-потомок для него `item`, для которого установим два варианта: с границей и без нее.

```
1  itemize {margin: 1em 1em 1em 1em; padding: .3em}
2  item    {margin: .5; padding: .2em 0em .2em .2em}
3  item[class=border]
4          {border-style: dotted; border-width: thin; border-color: red}
```

Строка 1 определяет, что все поля в элементе `itemize` имеют одну и ту же ширину в один квадрат (1 em), с заполнением шириной в .3 em по каждой из сторон[1] Строка 2 задает поле размером в половину квадрата для элемента `item` внутри элемента `itemize` при заполнении шириной в .2 em для всех сторон, за исключением правой стороны, где текст выходит на поле страницы. Наконец, строки 3–4 определяют, что для некоторого элемента `item` класса `border`, в верхней части характеристики строки 2, граница состоит из тонкой красной последовательности точек.

7.4.3.5 Отображения

Свойство `display` определяет, каким образом должен быть показан некоторый элемент уровня данного блока. Это свойство имеет много возможных значений, в частности `inline`, `block`, `run-in` и `compact`. Есть также еще специальное значение `none`, использование которого выливается в данном элементе и его потомках в то, что не будет формироваться *никаких* блоков в выводе (форматированного) дерева, и по этой причине не окажет никакого влияния на вид макета страницы документа. Из этого не следует, что рассматриваемый материал будет невидимым, поскольку эта характеристика управляется через свойство `visibility`, значениями которого могут быть `visible` (видимый), `hidden` (скрытый) и `collapse` (свернутый).

7.4.4 Стили CSS для форматирования XML-документов

Пользуясь материалом, изложенным в предыдущем разделе, можно теперь написать на языке CSS стиль для форматирования XML-документов. Снова вернемся к документу `invitation` и преобразуем его в HTML — единственный язык, с которым успешно работают все существующие на сегодняшний день навигаторы.

Используем ту же самую процедуру, что и в разд. 7.3, на этот раз с таким сценарием на языке Perl, оформленным в виде файла `inv1html.pl`:

```
1  ##############################################################################
2  # SGMLSPL script produced automatically by the script sgmlspl.pl
3  #
4  # Document Type: inv1html.pl (for HTML/CSS formatting)
5  # Edited by: mg (24 Aug 98)
6  ##############################################################################
7
8  use SGMLS;                      # Use the SGMLS package.
9  use SGMLS::Output;              # Use stack-based output.
10
11 #
12 # Document Handlers.
13 #
```

[1] Если задается только одно значение, то оно относится ко всем четырем сторонам; если два — первое значение задает верхнее и нижнее поля, а второе относится к левому и правому полям; если три — первое относится к верхнему полю, второе к левому и правому полям, третье к нижнему полю; при четырех заданных значениях они относятся к верхнему, правому, нижнему и левому полям соответственно.

```
14  sgml('start', "<HTML>\n<HEAD>\n" .
15                "<TITLE> Invitation (sgmlpl/CSS formatting) </TITLE>\n" .
16                "<LINK href=\"invit.css\" rel=\"stylesheet\" type=\"text/css\">\n" .
17                "<!-- 24 August 1998 mg -->\n" .
18                "</HEAD>\n");
19  sgml('end', "</HTML>");
20
21  #
22  # Element Handlers.
23  #
24
25  sgml('<invitation>', "<BODY>\n<H1>INVITATION</H1>\n");
26  sgml('</invitation>', "</BODY>\n");
27
28  sgml('<front>', "<P><TABLE>\n<TBODY>\n");
29  sgml('</front>', "</TBODY>\n</TABLE>\n");
30
31  sgml('<to>', "<TR><TD class=\"front\">To: </TD>\n<TD>");
32  sgml('</to>', "</TD></TR>\n");
33
34  sgml('<date>', "<TR><TD class=\"front\">When: </TD>\n<TD>");
35  sgml('</date>', "</TD></TR>\n");
36
37  sgml('<where>', "<TR><TD class=\"front\">Venue: </TD>\n<TD>");
38  sgml('</where>', "</TD></TR>\n");
39
40  sgml('<why>', "<TR><TD class=\"front\">Occasion: </TD>\n<TD>");
41  sgml('</why>', "</TD></TR>\n");
42
43  sgml('<body>', "");
44  sgml('</body>', "");
45
46  sgml('<par>', "<P>");
47  sgml('</par>', "</P>\n");
48
49  sgml('<emph>', "<EM>");
50  sgml('</emph>', "</EM>");
51
52  sgml('<back>', "");
53  sgml('</back>', "");
54
55  sgml('<signature>', "<P CLASS=\"signature\">");
56  sgml('</signature>', "</P>\n");
57
58  sgml('start_element',sub { die "Unknown element: " . $_[0]->name; });
59  sgml('cdata',sub { output $_[0]; });
60
61  1;
```

В строках 14–18 осуществляется инициализация HTML-документа и определяется его заголовок. В частности, в строке 16 производится связывание листа стиля `invit.css` с HTML-файлом. В строке 25 начинается основная часть (тело) документа с элементом `H1`, который будет печататься особым образом. Для форматирования материала из головной части документа и привлечения механизма класса HTML/CSS для управления форматированием крайних слева ячеек в таблице (строки 31, 34, 37 и 40) здесь решено использовать элемент `TABLE` (строка 28). Элементы-абзацы (строки 46–47) и элементы с эмфатическим текстом (строки 49–50) транслируются в их HTML-эквиваленты. Подпись, содержащаяся в документе, преобразуется в специальный элемент `P` класса `signature` (строка 55). Символьные данные из входного файла передаются в рассматриваемый HTML-файл вместе с оператором `output` в строке 59.

Если исполнить команду:

```
nsgmls invitation.xml | perl sgmlspl.pl inv1html.pl > invcss.html
```

получим HTML-файл, показанный ниже:

```
<HTML>
<HEAD>
<TITLE> Invitation (sgmlpl/CSS formatting) </TITLE>
<LINK href="invit.css" rel="stylesheet" type="text/css">
<!-- 24 August 1998 mg -->
</HEAD>
<BODY>
<H1>INVITATION</H1>
<P><TABLE>
<TBODY>
<TR><TD class="front">To: </TD>
<TD>Anna, Bernard, Didier, Johanna</TD></TR>
<TR><TD class="front">When: </TD>
<TD>Next Friday Evening at 8 pm</TD></TR>
<TR><TD class="front">Venue: </TD>
<TD>The Web Cafe</TD></TR>
<TR><TD class="front">Occasion: </TD>
<TD>My first XML baby</TD></TR>
</TBODY>
</TABLE>
<P>I would like to invite you all to celebrate
the birth of <EM>Invitation</EM>, my
first XML document child.</P>
<P>Please do your best to come and join me next Friday
evening. And, do not forget to bring your friends.</P>
<P>I <EM>really</EM> look forward to see you soon!</P>
<P CLASS="signature">Michel</P>
</BODY>
</HTML>
```

Наконец, рассмотрим файл `invit.css` со стилем, который служит в качестве связки между входным файлом с исходным XML-текстом и способом, задаваемым для отображения выходного HTML-файла.

```
/* invit.css: CSS style-sheet for invitation1 in HTML */
BODY {margin-top: 1em;     /* global page parameters */
      margin-bottom: 1em;
      margin-left: 1em;
      margin-right: 1em;
      font-family: serif;
      line-height: 1.1;
      color: black;
}
H1   {text-align: center;  /* for global title   */
      font-size: x-large;
}
P    {text-align: justify; /* paragraphs in body */
      margin-top: 1em;
}
TD[class="front"] {        /* table data in front matter */
      text-align: left;
      font-weight: bold;
}
EM   {font-style: italic;  /* emphasis in body   */
}
P[class="signature"] {     /* signature          */
      text-align: right;
```

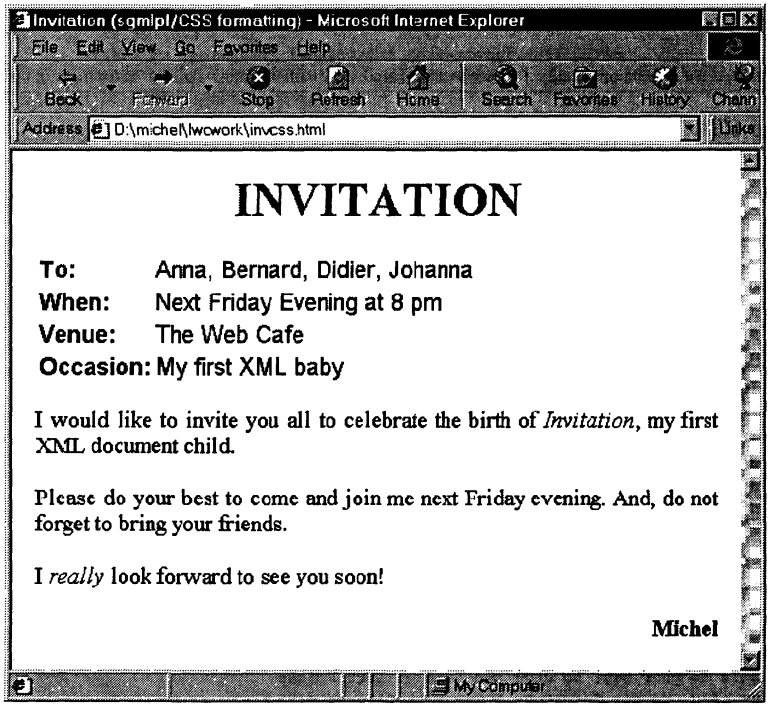


Рис. 7.2. XML-файл, отформатированный под HTML с использованием процедуры `sgmlspl`.

```
24          font-weight: bold;
25          margin-top: 1em;
26      }
```

Строки 2–8 устанавливают некоторые глобальные характеристики для HTML-документа в целом. Далее, для создания большого центрированного заголовка используется элемент `H1` (строки 10–11). Абзацы (строки 13–14) и эмфатический текст (строка 20) трактуются непосредственным образом как текст. Элементы с табличными данными `TD` класса «`front`» (строки 16–18) будут содержать текст, выровненный по левому краю, набранный полужирными буквами, а элементы-абзацы `P` класса «`signature`» также будут набираться полужирным шрифтом, но выравниваться они будут по правому краю. Однако в текущих версиях навигаторов как фирмы Netscape, так и Microsoft до сих пор не реализована поддержка спецификации CSS2 в полном виде. Следовательно, для использования стиля CSS, показанного ранее для отображения HTML-файла, придется использовать старый синтаксис CSS1 при определении деклараций, связанных с классом, — `TD.front` (в строке 16) и `P.signature` (в строке 22).

Результат вывода HTML-файла `invcss.html` с помощью навигатора, применяющего CSS-стиль `invit.css`, показан на рис. 7.2.

7.4.5 Еще раз о примере `invitation`

Пример XML-документа `invitation` и его определение DTD, введенные впервые в разд. 6.4.2.2, содержат только элементы и не используют атрибутов. Дадим вариант DTD для примера `invitation`, где большинство элементов будет заменено атрибутами. В информационном плане это эквивалентно первоначальной версии[1]. Новый DTD, `invitation2.dtd`, имеет следующий вид:

```
1  <!-- invitation2 DTD -->
2  <!ELEMENT invitation (par+)>
3  <!ATTLIST invitation date      CDATA #REQUIRED
4                       signature CDATA #REQUIRED
5                       to        CDATA #REQUIRED
6                       where     CDATA #REQUIRED
7                       why CDATA #IMPLIED >
8  <!ELEMENT par  (#PCDATA|emph)*>
9  <!ELEMENT emph (#PCDATA)>
```

Число элементов теперь сократилось до трех: `invitation` (строка 2), `par` (строка 8) и `emph` (строка 9). Вся дополнительная информация задается в виде атрибутов элемента `invitation` (строки 3–7). XML-документ `invitation2.xml` как таковой маркируется в терминах этого DTD следующим образом:

```
1  <?xml version="1.0"?>
2  <!DOCTYPE invitation SYSTEM "invitation2.dtd">
3  <invitation to="Anna, Bernard, Didier, Johanna"
4              date="Next Friday Evening at 8 pm"
5              where="The Web Cafe"
6              why="My first XML baby"
7              signature="Michel"
8  >
9  <par>
10 I would like to invite you all to celebrate
11 the birth of <emph>Invitation</emph>, my
12 first XML document child.
13 </par>
14 <par>
15 Please do your best to come and join me next Friday
16 evening. And, do not forget to bring your friends.
17 </par>
18 <par>
19 I <emph>really</emph> look forward to see you soon!
20 </par>
21 </invitation>
```

Можно сказать, что структура полученного документа стала не намного яснее, и что атрибуты здесь не внесли ничего дополнительного, однако данный пример показывает, каким образом работать с атрибутами. Подчеркнем еще раз, что информационное содержимое документа из этого примера полностью идентично содержимому документа на с. 298.

[1] См. также разд. B.4.4.2 и B.4.4.3 в приложении, где разработаны два других DTD для BIBTEX'а на основе того же самого подхода.

7.4.6 Формирование HTML-вывода для другого экземпляра документа

Чтобы показать, как в CSS работает принцип повторного использования стилей, рассмотрим другую форму примера «Приглашение» («Invitation»), оформленного в виде файла `invitation2.xml`, как это было сделано в предыдущем разделе. Используем еще раз Perl-модуль SGMLSpm Дэвида Меггинсона, но с другим пользовательским сценарием `inv2html.pl`, как это показано ниже:

```
1  ##############################################################################
2  # SGMLSPL script produced automatically by the script sgmlspl.pl
3  #
4  # Document Type: inv2html.pl (for HTML/CSS formatting)
5  # Edited by: mg (25 Aug 1998)
6  ##############################################################################
7
8  use SGMLS;                      # Use the SGMLS package.
9  use SGMLS::Output;              # Use stack-based output.
10
11 #
12 # Document Handlers.
13 #
14 sgml('start', sub {
15     output "<HTML>\n<HEAD>\n";
16     output "<TITLE> Invitation (sgmlpl/CSS formatting) </TITLE>\n";
17     output "<LINK href=\"invit.css\" rel=\"stylesheet\" type=\"text/css\">\n";
18     output "<!-- 24 August 1998 mg -->\n";
19     output "</HEAD>\n";
20 });
21 sgml('end', "</HTML>");
22
23 #
24 # Element Handlers.
25 #
26
27 # Element: invitation
28 sgml('<invitation>', sub {
29    my ($element,$event) = @_;
30    # First save the information for further use
31    #  Local variables
32    my $date  = $element->attribute('date')->value;
33    my $to    = $element->attribute('to')->value;
34    my $where = $element->attribute('where')->value;
35    my $why   = $element->attribute('why')->value;
36    #  Global variable (saved for end of document)
37    $main::GLsig   = $element->attribute('signature')->value;
38    # Output the HTML commands needed for the front matter
39    output "<BODY>\n<H1>INVITATION</H1>\n";
40    output "<P><TABLE>\n<TBODY>\n";
41    output "<TR><TD class=\"front\">To: </TD>\n<TD>$to</TD></TR>\n";
42    output "<TR><TD class=\"front\">When: </TD>\n<TD>$date</TD></TR>\n";
43    output "<TR><TD class=\"front\">Venue: </TD>\n<TD>$where</TD></TR>\n";
44    output "<TR><TD class=\"front\">Occasion: </TD>\n<TD>$why</TD></TR>\n";
45    output "</TBODY>\n</TABLE>\n";
46 });
47
48 sgml('</invitation>', sub{ # signature and end of document
49      output "<P CLASS=\"signature\">$main::GLsig</P>\n";
50      output "</BODY>\n";
51 });
52
53 # Elements: par and emph
54 sgml('<par>', "<P>");
55 sgml('</par>', "</P>\n");
```

```
56
57   sgml('<emph>', "<EM>");
58   sgml('</emph>', "</EM>");
59
60   sgml('cdata',sub { output $_[0]; });
61   1;
```

Как и в сценарии `inv1html.pl`, показанном в разд. 7.4.4, здесь надо будет работать с атрибутами. Так как атрибуты заданы для элемента `<invitation>` (строки 3–7 файла `invitation2.xml` исходного документа на с. 359), необходимо извлечь эти атрибуты, если произойдет событие «начало элемента `invitation`» (строки 28–46). Во-первых, из строки 29 извлекается дескриптор элемента и как значение присваивается переменной `$element`. Значения атрибутов могут быть получены с использованием метода `value` из класса `SGMLS_Attribute`. Данный класс получается применением метода `attribute` из класса `SGMLS_Element` (строки 32–35 для локальных переменных — им предшествует описатель-спецификатор `my` — и строка 37 для глобальной переменной `$main::GLsig`, значение которой сохраняется до тех пор, пока не будет обнаружен конец элемента `invitation`, где оно и будет использовано для вывода подписи в строке 49). Таким образом, сформирован по сути дела тот же самый HTML-код, что и в разд. 7.4.4. В частности, используется ссылка (строка 17) на тот же самый стиль CSS (`invit.css`), как и в предыдущем примере. Если использовать рассматриваемый исходный XML-текст с предыдущим Perl-сценарием, то получим файл `inv2css.html`, что может быть выполнено с помощью такой команды:

```
nsgmls invitation2.xml | perl sgmlspl.pl inv2html.pl > inv2css.html
```

Просматривая этот HTML-файл с помощью навигатора, можно увидеть, что макет документа остался идентичным представленному на рис. 7.2. Это ясно показывает, что CSS-стили очень удобны для управления характеристиками отображения документов глобальным образом.

7.5 DSSSL — язык описания семантики стиля и спецификации документа

Язык спецификации и описания семантики стиля документа DSSSL ISO:10179:1996 представляет собой международный стандарт ISO/IEC для определения действий по форматированию и преобразованиям документов на языке SGML (и, следовательно, и на языке XML). Как уже объяснялось выше, любой SGML-документ должен полностью игнорировать вопросы визуализации его информационного содержимого, а также любые другие аспекты обработки содержащихся в нем данных. Две основные цели, ради которых создавался язык DSSSL, — это форматирование SGML-документов для вывода их в виде «бумажного» документа или в электронной форме, а также преобразование SGML-документов для различных схем разметки, порожденных различными DTD. Язык DSSSL предлагает стандартизованную основу

(framework) и методы для соответствующей информационной обработки для конкретных экземпляров элементов SGML или общих классов типов элементов.

В этом разделе содержится введение в DSSSL, достаточное для того, чтобы разрабатывать на DSSSL простые стили для XML-документов. Более подробное введение в DSSSL, напоминающее по стилю учебное пособие, написали Даниель М. Жерман [↪DSSSLTUTB] и Поль Преско [↪DSSSLTUTA]. Значительный объем информации, связанной с этой тематикой, имеется на Web-странице по DSSSL, разработанной и сопровождаемой Джеймсом Кларком [↪DSSSLCLARK].

7.5.1 Компоненты языка DSSSL

Язык DSSSL поддерживает четыре отдельные области стандартизации (утверждения, на которые ниже дается ссылка, относятся к спецификации DSSSL (ISO/IEC:10179, 1996)):

1. Языковая модель и модель обработки для преобразования SGML-документов в другие SGML-документы. Здесь используются язык преобразований и спецификация преобразований, представляющие собой список соответствий (Утверждение 11).
2. Язык, позволяющий указать, как применить средства форматирования к SGML-документу. Процесс форматирования управляется с помощью спецификации стиля, которая содержит некоторый список правил конструирования. Язык DSSSL стандартизует только форму и семантику рассматриваемого стилевого языка, *не затрагивая* при этом процесса форматирования как такового (Утверждение 12).
3. Стандартный язык запросов к документам SDQL, позволяющий идентифицировать фрагменты SGML-документа. Язык SDQL позволяет просто организовать навигацию по иерархической структуре SGML и задавать фрагменты документа для их обработки. DSSSL определяет также некоторое подмножество, именуемое *базовый язык запросов* (Утверждение 10).
4. Язык выражений, используемых в предыдущих языках для создания объектов и манипулирования ими. DSSSL определяет также некоторое подмножество, называемое *базовый язык выражений* . Язык выражений из DSSSL использует подмножество языка Scheme, свободное от побочных эффектов (Утверждение 8).

На рис. 7.3 схематически показан полный набор процессов, поддерживаемый языком DSSSL.

7.5.1.1 Стилевой язык из состава DSSSL

В обсуждении, которое последует ниже, нас большей частью будут интересовать свойства DSSSL как языка описания стиля, обеспечивающего мощные стандартизованные средства описания процессов форматирования SGML-документов. В разделе 7.5.3 будет рассмотрен Jade — мощный и свободно до-

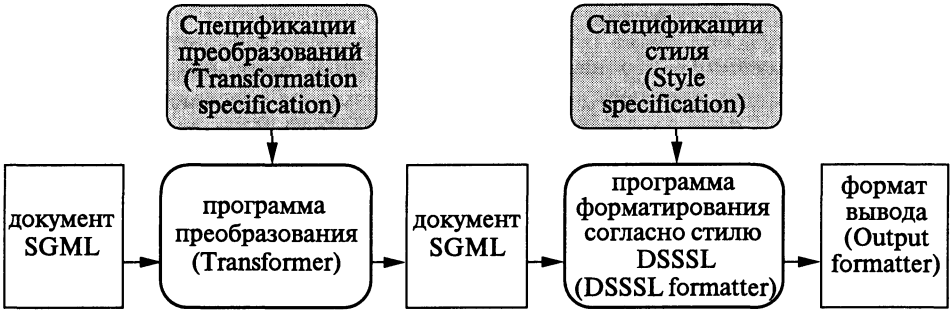


Рис. 7.3. Процессы, поддерживаемые языком DSSSL.

ступный DSSSL-процессор, реализующий стилевой язык. Он и будет «рабочей лошадкой» при рассмотрении всех последующих примеров из данной области.

Центральным элементом стилевого языка является *дерево потоковых объектов*. Это некоторое абстрактное представление соответствия между исходным документом и спецификаций форматирования. В узлах этого дерева находятся *потоковые объекты*, обеспечивающие стандартную основу для описания макета документа с помощью определенных конструкций, таких, например, как последовательности страниц, абзацы, таблицы и иллюстрации. Каждый потоковый объект имеет набор характеристик, таких, как поля страницы, величина отступа первой строки абзаца, граница таблицы, а также высота и ширина рисунка (это аналогично свойствам CSS, упоминаемым в разд. 7.4.3).

Результатом форматирования потокового объекта является последовательность *областей*. Каждая такая область представляет собой прямоугольный блок некоторой фиксированной ширины и высоты. Существует два их вида: отображаемые и встроенные области.

Отображаемые области не являются напрямую фрагментами строк; им присуща абсолютная ориентация, а их положение задается *контейнерами области*. Эти контейнеры имеют свою собственную систему координат с направлениями заполнения, определяющими их начальное ребро. Они могут быть либо фиксированного размера, либо изменяемого. Внутри такого контейнера возможны различные режимы записи.

Встроенные области представляют собой фрагменты строк. Любая встроенная область имеет опорные точки на ее ребрах, так что встроенные области, идущие последовательно, можно позиционировать, чтобы формировать строки. На это позиционирование может оказывать влияние кернинг.

Потоковые объекты, форматируемые для получения последовательности встроенных областей, называются *встроенными*, а потоковые объекты, форматируемые для получения последовательности отображаемых областей, — *отображаемыми*. Будет ли потоковый объект только встроенным, только отображаемым или же одновременно и встроенным, и отображаемым, зависит от характеристик рассматриваемого потокового объекта или потокового объекта, присоединенного к нему.

7.5.1.2 Другие компоненты языка DSSSL

Язык преобразований — это некоторый стандартный язык для преобразования SGML-разметки согласно данному DTD в разметку, отвечающую другому DTD. Это может быть, например, трансляция DocBook-документа в «эквивалентную» TEI-разметку или, что более вероятно, формирование экземпляра рассматриваемого документа в формате HTML, поскольку в настоящее время большинство навигаторов поддерживает только этот язык. Формально язык преобразований, особенно в его части, связанной с выражениями, которая базируется на языке Scheme, имеет много общего с языком стилей.

Спецификации DSSSL работают с деревьями из узлов, чтобы сформировать новые деревья, такие, как «дерево потоковых объектов», которое представляет собой экземпляр форматированного документа или некоторый новый SGML-документ, полученный с помощью какого-либо преобразования. Упомянутые выше узлы организуются в некоторую специализированную структуру данных, «рощу», некий вид дерева деревьев или, выражаясь ближе к техническому языку, в циклический ориентированный граф узлов. Каждый из узлов здесь может иметь некий набор «свойств», которые могут иметь «атомарные» значения (такие, как строки, булевы переменные или целые числа) или быть списками узлов или ссылок на узлы. Эта модель документа с рощей в качестве дерева грамматического разбора как структуры данных в форме некоторого графа узлов со свойствами была разработана для специфических потребностей языка DSSSL и HyTime [↪HYTIME]. Эти два стандарта имеют один и тот же фундаментальный абстрактный подход к данным в SGML-документе (дополнительную информацию по структуре данных типа «роща» можно найти в [↪GROVES]).

Имеется язык запросов для выбора и извлечения компонент документа в форме узлов описанной выше модели документа DSSSL/HyTime. Поскольку для наших целей эти компоненты языка DSSSL не требуются, обсуждать подробнее здесь мы их не будем.

7.5.2 Создание стилей с помощью языка DSSSL

На первый взгляд, синтаксис DSSSL выглядит несколько непривычно, поскольку он основан на языке Scheme — одном из диалектов языка Lisp. Однако, как это можно видеть из примеров, следующих ниже, если отвлечься от необходимости постоянного использования круглых скобок, использование DSSSL достаточно несложно.

7.5.2.1 Правила конструирования

Как и в других языках описания стилей, лист стиля в DSSSL состоит из последовательности операторов (утверждений), именуемых *правилами построения*, которые «конструируют» форматированный документ из некоторого исходного SGML-документа. Рассмотрим вначале строку в примере, следующем ниже.

Она указывает прикладной DSSSL-программе, что все элементы `par` в исходном SGML-тексте должны сформировать на выходе (в выходном дереве потоковых объектов) потоковые объекты-абзацы.

```
(element par (make paragraph quadding: 'justify ))
```

Как уже говорилось выше, потоковый объект соответствует форматирующему объекту. Подробное описание классов потоковых объектов в DSSSL, а также их характеристик дается в разд. 12.6 стандарта ISO:10179:1996.[1] Чтобы упростить ссылки на классы потоковых объектов, дадим список всех таких классов, имеющихся в DSSSL, вместе с их идентификационным номером в соответствующем разделе (например, номер 5 в списке соответствует подразделу 12.6.5 в спецификации DSSSL). Элементы данного списка, помеченные как †, не входят в подмножество языка DSSSL-online, первоначально предложенное в качестве языка описания листов стилей для XML.

1 `sequence` (последовательность)
2 `display-group` (группа-отображения)
3 `simple-page-sequence` (простая-последовательность-страниц)
4 `page-sequence` (последовательность-страниц) †
5 `column-set-sequence` (последовательность-наборов-колонок) †
6 `paragraph` (абзац)
7 `paragraph-break` (конец-абзаца)
8 `line-field` (поле-строка)
9 `sideline` (дополнительная строка)
10 `anchor` (параметр привязки)†
11 `character` (литера)
12 `leader` (служебные данные в начале массива)
13 `embedded-text` (встроенный-текст)†
14 `rule` (линейка)
15 `external-graphic` (внешняя-графика)
16 `included-container-area` (включенная-область-контейнер)†
17 `score` (оценка)
18 `box` (блок)
19 `side-by-side` (поперечное-размещение)†
20 `side-by-side-item` (элемент-поперечного-размещения)†
21 `glyph-annotation` (аннотация-образа-символа)†
22 `alignment-point` (точка-выравнивания)
23 `aligned-column` (столбец-выравнивания)
24 `multi-line-inline-note` (многострочное-встроенное-примечание)†
25 `emphasizing-mark` (маркер-выделения)†
26 классы потоковых объектов для математических формул †
(1) `math-sequence` (математическая-последовательность), (2) `unmath` (текст в математике), (3) `subscript` (нижний индекс), (4) `superscript`

[1] По адресу [↪DSSSLPDF] можно найти вариант спецификации DSSSL в формате PDF, предназначенный для индивидуального использования.

(верхний индекс), (5) `script` (дополнительные шрифты), (6) `mark` (маркер), (7) `fence` (ограничитель), (8) `fraction` (дробь), (9) `radical` (радикал), (10) `math-operator` (математический-оператор), (11) `grid` (сетка), (12) `grid-cell` (ячейка сетки)

27 классы потоковых объектов для таблиц
(1) `table` (таблица), (2) `table-part` (фрагмент-таблицы), (3) `table-column` (колонка-таблицы), (5) `table-row` (строка-таблицы), (6) `table-cell` (ячейка-таблицы), (7) `table-border` (граница-таблицы)

28 классы потоковых объектов для диалогового отображения
(1) `scroll` (прокрутка), (2) `multi-mode` (многорежимность), (3) `link` `flow` (поток ссылок), (4) `marginalia` (поля)

Список характеристик для всех классов потоковых объектов вместе с рядом дополнительных полезных таблиц можно получить на Web-странице, посвященной синтаксису языка DSSSL; автор этой страницы Харви Бингем [↪DSSSLSUM].

Например, если требуется найти все характеристики потокового объекта `paragraph`, который использовался в предыдущем примере, надо, сверившись с приведенным выше списком, просмотреть разд.12.6.6. Здесь содержится описание искомых характеристик, занимающее более восьми страниц. В частности, для характеристики `quadding` устанавливается, что она управляет «выравниванием строк, отличных от последней строки абзаца...». В тексте, имеющем отношение к данному вопросу, перечисляются также все другие значения, которые может принимать параметр `quadding`.

Разработчики программных реализаций DSSSL могут добавлять также свои собственные потоковые объекты. В Jade (см. разд. 7.5.3) эта возможность используется для формирования SGML-вывода; следовательно, имеется возможность использовать Jade для преобразования XML-документа в формат HTML, например. Вместо того, чтобы писать соответствующую программу на языке Perl или Java, как это делалось в предыдущих разделах, можно, если позволяют имеющиеся знания языка DSSSL, использовать Jade, чтобы преобразовать XML-документы в HTML, LaTeX и т. д.

Например, можно взять тот же самый тип элемента `par`, что и в ранее рассмотренном примере, после чего потребовать от Jade провести трансляцию его в элемент `<P>` из HTML. Это может быть проделано следующим образом (здесь используется расширение SGML, имеющееся в Jade):

```
(element par (make element gi: "P")) ;creates an HTML paragraph
```

7.5.2.2 Простая спецификация на DSSSL

Спецификации DSSSL представляют собой настоящие SGML-документы и строятся согласно DTD. В составе дистрибутива Jade поставляется файл `style-sheet.dtd`, имеющий свой собственный публичный идентификатор.

```
-//James Clark//DTD DSSSL Style Sheet//EN
```

На самом деле написать стиль DSSSL не так уж и сложно, и в большинстве случаев приходится иметь дело всего с несколькими потоковыми объектами.

В качестве примера возьмем опять уже использовавшийся неоднократно документ `invitation` и сформируем простой вывод в формате HTML (как это уже проделывалось с CSS в предыдущем разделе), но без записи собственно HTML-кода. Используем вместо этого потоковые объекты DSSSL, а для проведения трансляции воспользуемся программой Jade.

```
1  <!-- invitation.dsl -->
2  <!DOCTYPE style-sheet PUBLIC
3  "-//James Clark//DTD DSSSL Style Sheet//EN"
4  >
5  <style-sheet>
6  <style-specification>
7  <style-specification-body>
8  (define *FontSize* 12pt)
9  (root
10   (make simple-page-sequence
11     left-margin:  5mm
12     page-width:   100mm
13     right-margin: 5mm
14     (make scroll
15       font-size: *FontSize*
16       line-spacing: *FontSize*
17       (process-children))))
18 (element (front date)
19   (make paragraph
20     (literal "When: ")
21     (process-children)))
22 (element (front to)
23   (make paragraph
24     (literal "To: ")
25     (process-children)))
26 (element (front where)
27   (make paragraph
28     (literal "Venue: ")
29     (process-children)))
30 (element (front why)
31   (make paragraph
32     (literal "Occasion: ")
33     (process-children)))
34 (element (body par)
35   (make paragraph
36     quadding: 'justify
37     font-size: *FontSize*
38     space-before: *FontSize*
39     (process-children)))
40 (element emph
41   (make sequence
42     font-posture:   'italic
43     (process-children)))
44 (element (back signature)
45   (make paragraph
46     quadding: 'end
47     space-before: *FontSize*
48     (literal "From: ")
49     (process-children)))
50 </style-specification-body>
51 </style-specification>
52 </style-sheet>
```

Рассмотрим DSSSL-стиль `invitation.dsl`, приведенный выше. Строка 8 показывает, как определяется константа (`*Fontsize*`). По соглашению, константы с обоих сторон выделяются звездочками, так что они хорошо различимы среди программного текста. Далее, путем сборки потоковых объектов, формируется представление документа для его вывода. Определяется корневой объект и создается некоторая простая-последовательность-страниц (`simple-page-sequence`) (строка 10), устанавливаются некоторые характеристики для нее (строки 11–13). Внутри объекта `simple-page-sequence` создается объект `scroll` (строка 14), разрешающий отображение тех потоковых объектов, которые не разделены между несколькими страницами. Отдельно от определения характеристик — внутри выражения `make` и после списка аргументов-ключевых слов — можно задать «выражение, определяющее содержание», которое указывает DSSSL-процессору, что следует поместить внутрь текущего потокового объекта. Например, в строке 17 вызывается процедура `process-children`, обрабатывающая дочерние узлы текущего элемента, который является элементом `root` (т. е. полным документом) в рассматриваемом случае. Если выражение, определяющее содержимое, не задано, DSSSL автоматически включит строку `(process-children)`. Однако если текущий пото-

ковый объект не может иметь потомков, он называется «атомарный потоковый объект», например, литерный (символьный) потоковый объект. Хотя, для ясности, хорошей практикой считается всегда задавать выражение, определяющее содержание, если это только возможно. В частности, требуется составить такого рода выражение, если необходимо подавить некоторые дочерние узлы или же подвергать эти узлы обработке избирательно. В противном случае придется ввести потоковые объекты, которые не отвечают напрямую элементам исходного документа.

Следующая часть стиля имеет дело с созданием потоковых объектов для каждого из элементов, используемых в исходном XML-документе. Например, строки 18–21 относятся к элементам `date`, находящимся внутри элементов `front` (это похоже на синтаксис селекторов `front > element` для CSS-правил, как объясняется в разд. 7.4). В строке 19 создается потоковый объект `paragraph`, который формирует элементы-строки для встроенных областей. Элемент `paragraph` является всегда отображаемым. В рассматриваемом здесь случае он состоит из литерала «`When: `», за которым всегда идет результат обработки узлов-потомков текущего элемента-узла `date`. Аналогично этому строятся потоковые объекты `paragraph` для других элементов типа `front`. В строках 34–39 встречается программный код для обработки элементов `par` внутри элемента `body`, который также превращается в потоковый объект `paragraph`, но в данном случае явно задаются некоторые из его характеристик. В частности, требуется оставить в начале каждого нового абзаца пробел, равный значению переменной `*FontSize*`, определенной в начале рассматриваемого файла. Строки 40–43 объявляют (определяют), каким образом надо работать с элементами `emph`, и здесь же вводится потоковый объект `sequence`, осуществляющий сцепление областей, порожденных каждым из их потомков. В рассматриваемом случае этот объект используется, чтобы установить курсивное начертание шрифта. Наконец, для элемента `back` и входящего в него элемента `signature` (строки 44–49) строится другой потоковый объект `paragraph` с литералом «`From: `» и содержимым данного элемента.

Для получения форматированного вывода этот стиль DSSSL вместе с исходным XML-файлом можно теперь обработать с помощью DSSSL-процессора. Для работы с рассматриваемыми примерами далее будет использоваться программа Jade.

7.5.3 Программа Jade

Первый DSSSL-процессор был разработан Джеймсом Кларком. Его программа Jade (James' Awesome DSSSL Engine) представляет собой быстрый процессор, написанный на языке C++ , рассчитанный на работу на платформах Win32 и UNIX [↪JADE]. Jade реализует обширное подмножество стилевого языка, входящего в DSSSL, однако почти не поддерживает его язык преобразований. Jade может формировать вывод в следующих форматах: RTF (*Rich Text Format*) фирмы Microsoft, TEX (с помощью JadeTEX), XML, HTML (используя

нестандартное расширение языка DSSSL), нестандартный формат «FOT» (Flow Object Tree — в форме потоковых объектов как элементов XML), а также MIF (*Maker Interchange Format фирмы Adobe* для программы FrameMaker).

Для некоторых платформ, включая Microsoft Windows, доступны варианты данной программы в двоичных кодах, что существенно упрощает установку программы, сводя ее к простой распаковке дистрибутивного комплекта. Для других платформ Jade можно получить трансляцией и сборкой из поставляемых исходных программных текстов.

После того как программа Jade установлена способом, указанным в документации, запуск и использование ее довольно просты. Jade использует окружение, аналогичное тому, что имеет место для программы `nsgmls` (см. разд. 6.6.5.1). В частности, следует установить переменную окружения `SGML_CATALOG_FILES`, чтобы информировать программу Jade о том, где размещается файл `catalog`. У Jade многие из опций командной строки такие же, как и у программы `nsgmls`. Другие опции и их значения разъясняются на домашней странице Jade [↪JADE].

```
jade [-vCegG2] [-b encoding] [-f error_file] [-c catalog_sysid]
     [-D dir] [-a link_type] [-A arch] [-E max_errors]
     [-i entity] [-w warning_type] [-d dsssl_spec] [-V variable]
     [-t (fot|rtf|html|tex|mif|sgml|xml)] [-o output_file] input_file(s)
```

Из числа этих опций для наших целей наиболее важны такие новые переключатели, как `-d`, который дает возможность задать имя листа стиля DSSSL; `-t`, который определяет тип вывода (соответствующие возможности перечислены выше); `-G`, инициирующий отладку с показом стека оценок, если что-то пойдет неправильно; `-o`, позволяющий задать имя файла вывода.

7.5.3.1 Исполнение программы Jade

Прежде чем попытаться сделать с помощью Jade что-то конкретное, отметим одну интересную черту этой программы, которая позволяет получить полное текстовое информационное содержание рассматриваемого документа. Если задать пустой лист стиля (`empty.dsl`, см. ниже), система рекурсивно исполняет инструкцию `process-children`, начиная с корневого элемента, и далее вниз — до элементов-потомков, лежащих на наиболее глубоком уровне.

```
1  <!-- empty.dsl -->
2  <!DOCTYPE style-sheet PUBLIC "-//James Clark//DTD DSSSL Style Sheet//EN">
3  <style-sheet>
4    <style-specification>
5      <style-specification-body>
6      </style-specification-body>
7    </style-specification>
8  </style-sheet>
```

Файл `catalog`, который необходим для связывания публичных идентификаторов DSSSL с системными ресурсами, должен содержать следующее:

```
1  PUBLIC "-//James Clark//DTD DSSSL Flow Object Tree//EN" "fot.dtd"
2  PUBLIC "-//James Clark//DTD DSSSL Style Sheet//EN" "style-sheet.dtd"
3  PUBLIC "ISO/IEC 10179:1996//DTD DSSSL Architecture//EN" "dsssl.dtd"
```

Запустим теперь программу Jade с этими файлами и XML-документом `invitation.xml`.

```
jade -t xml -d empty.dsl d:\jade\xml.dcl invitation.xml
```

Файл деклараций XML с именем `xml.dcl` должен быть загружен до файла `invitation.xml`, поскольку по умолчанию Jade допускает только стандартный SGML-синтаксис и не воспринимает элементов, специфических для XML, таких, как синтаксис пустых элементов.

Ниже показан вывод, полученный в данном случае. Можно видеть, что получены символьные данные, составляющие информационное содержимое всех элементов в исходном XML-файле `invitation.xml`.

```
Anna, Bernard, Didier, JohannaNext Friday Evening at 8 pmThe Web
CafeMy first XML babyI would like to invite you all to celebrate the
birth of Invitation, my first XML document child.Please do your best
to come and join me next Friday evening. And, do not forget to bring
your friends.I really look forward to see you soon!Michel
```

Вернемся теперь к нашему листу стиля DSSSL с именем `invitation.dsl`, введенному в разд. 7.5.2.2, и пропустим его через программу Jade вместе с исходным XML-файлом `invitation.xml`, чтобы получить различные выводные форматы. Если выводной формат в явном виде не задавался (с помощью опции `-t`), Jade будет формировать XML-представление дерева потоковых объектов.

```
jade -dinvitation.dsl d:\jade\xml.dcl invitation.xml
```

Ниже приводится сформированный файл `invitation.fot`. Он ясно показывает, как вывод строится согласно инструкциям для каждого элемента из исходного дерева, описанного в листе стиля `invitation.dsl`. Порядок следования потоковых объектов отвечает порядку, в котором элементы конструируются при грамматическом разборе исходного документа `invitation.xml` (см. с. 298).

```
1  <?xml version="1.0"?>                    <!-- invitation.fot -->
2  <fot>
3  <simple-page-sequence left-margin="14.17pt" page-width="283.46pt" right-margin="14.17pt">
4  <scroll font-size="12pt" line-spacing="12pt">
5  <paragraph>
6  <a name="0"/>
7  <a name="1"/>
8  <a name="2"/>
9  <text>To: </text>
10 <text>Anna, Bernard, Didier, Johanna</text>
11 </paragraph>
12 <paragraph>
13 <a name="3"/>
14 <text>When: </text>
15 <text>Next Friday Evening at 8 pm</text>
16 </paragraph>
17 <paragraph>
18 <a name="4"/>
```

```
<text>Venue: </text>
<text>The Web Cafe</text>
</paragraph>
<paragraph>
<a name="5"/>
<text>Occasion: </text>
<text>My first XML baby</text>
</paragraph>
<paragraph space-before="12pt" quadding="justify" font-size="12pt">
<a name="6"/>
<a name="7"/>
<text>
I would like to invite you all to celebrate
the birth of </text>
<sequence font-posture="italic">
<a name="8"/>
<text>Invitation</text>
</sequence>
<text>, my
first XML document child.
</text>
</paragraph>
<paragraph space-before="12pt" quadding="justify" font-size="12pt">
<a name="9"/>
<text>
Please do your best to come and join me next Friday
evening. And, do not forget to bring your friends.
</text>
</paragraph>
<paragraph space-before="12pt" quadding="justify" font-size="12pt">
<a name="10"/>
<text>
I </text>
<sequence font-posture="italic">
<a name="11"/>
<text>really</text>
</sequence>
<text> look forward to see you soon!
</text>
</paragraph>
<paragraph space-before="12pt" quadding="end">
<a name="12"/>
<a name="13"/>
<text>From: </text>
<text>Michel</text>
</paragraph>
</scroll>
</simple-page-sequence>
</fot>
```

С теми же самыми входными файлами можно заставить Jade сформировать файл в формате RTF, это делается с помощью следующей команды:

```
jade -dinvitation.dsl -t rtf d:\jade\xml.dcl invitation.xml
```

В данном примере формируется файл `invitation.rtf`, который можно просматривать с помощью программы Microsoft Word (рис. 7.4). Аналогично, если задать `-t tex` (вместо `-t rtf`), Jade преобразует спецификации рассматриваемого потокового объекта в TEX'овский текст, который затем можно проинтерпретировать с помощью TEX'овского постпроцессора JadeTEX (см.

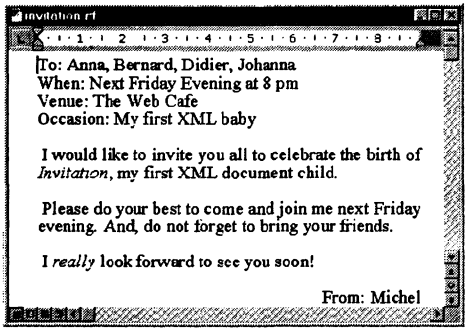


Рис. 7.4. Простой стиль DSSSL для RTF.

To: Anna, Bernard, Didier, Johanna
When: Next Friday Evening at 8 pm
Venue: The Web Cafe
Occasion: My first XML baby

I would like to invite you all to celebrate the birth of *Invitation*, my first XML document child.

Please do your best to come and join me next Friday evening. And, do not forget to bring your friends.

I *really* look forward to see you soon!

From: Michel

Рис. 7.5. Простой стиль DSSSL для TEX'a.

разд. 7.5.4). На рис. 7.5 показан полученный файл вывода, который почти идентичен тому, что представлен на рис. 7.4.

7.5.3.2 Подготовка таблиц в DSSSL

Рассмотрим теперь несколько более изощренный пример и поместим вводный материал в табличную конструкцию DSSSL, чтобы улучшить выравнивание этого материала.

```
<!DOCTYPE style-sheet PUBLIC
"-//James Clark//DTD DSSSL Style Sheet//EN"
>
<style-sheet>
<style-specification>
<style-specification-body>

(define FontSize 12pt)

(root
  (make simple-page-sequence
    left-margin:  2cm
    page-width:   15cm
    right-margin: 2cm
    (make scroll
      font-size: FontSize
      (process-children))))

(element front
  (make sequence
    (make paragraph
      quadding:     'center
      space-before:   20pt
      font-weight: 'bold
      font-size:    24pt
      (literal "INVITATION"))
    (make table
      table-border:   #f  ; no border
      display-alignment: 'start
      space-before:   20pt
      (make table-part
        (make table-column width: 25mm)
        (make table-column width: 10cm)
        (process-children)))))
(element (front date)
  (make table-row
    (make table-cell
      (make paragraph quadding: 'start
        (literal "When:")))
    (make table-cell
        (process-children))))
(element (front to)
  (make table-row
    (make table-cell
     (make paragraph quadding: 'start
        (literal "To:")))
    (make table-cell
        (process-children))))
(element (front where)
  (make table-row
    (make table-cell
      (make paragraph quadding: 'start
        (literal "Venue:")))
    (make table-cell
        (process-children))))
(element (front why)
  (make table-row
    (make table-cell
      (make paragraph quadding: 'start
        (literal "Occasion:")))
    (make table-cell
        (process-children))))

(element (body par)
```

```
65    (make paragraph
66      quadding: 'justify
67      font-size: FontSize
68      space-before: FontSize
69      (process-children)))
70  (element emph
71    (make sequence
72      font-posture: 'italic
73      (process-children)))
74
75  (element (back signature)
76    (make paragraph
77      quadding: 'end
78      space-before: FontSize
79      (literal "From: ")
80      (process-children)))
81
82  </style-specification-body>
83  </style-specification>
84  </style-sheet>
```

Наш подход совершенно аналогичен тому, что используется в LaTeX'е, но, разумеется, таблицу требуется описать более подробно. Если обратиться к списку классов потоковых объектов DSSSL, который начинается на с. 365, можно увидеть, что табличные компоненты (заголовок 27) описываются в разд. 12.6.27 спецификации DSSSL, к которому и надо обратиться, если требуется узнать подробности о способах построения таблиц.

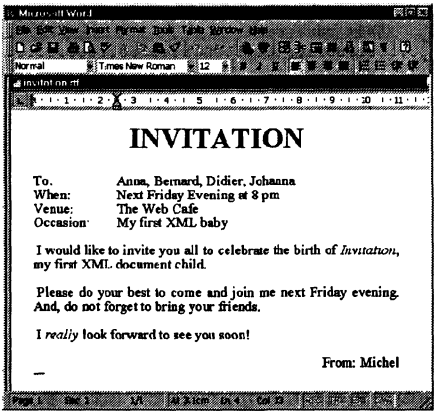
Строки 27–34 определяют несколько новых характеристик рассматриваемой таблицы (отсутствие границы, ширину колонок, выравнивание, промежуток между таблицей и предшествующим материалом) и дают команды DSSSL по поводу того, как работать с потомками элемента `front`. Работа с этими потомками описывается в последующих строках: `date` (строки 35–41), `to` (строки 42–48), `where` (строки 49–55) и `why` (строки 56–62). В каждом из этих случаев строится левосторонняя ячейка, что осуществляется заданием некоторого текста в виде литеральной константы (строки 39, 46, 53 и 60). Затем в правостороннюю ячейку помещается содержимое рассматриваемого потомка элемента `front`. Строки 64–69 относятся к процессу формирования нового абзаца путем создания нового потокового объекта `paragraph`, а строки 70–73 переводят XML-элемент `emph` в потоковый текстовый курсивный объект `sequence`. Наконец, строки 75–80 помещают содержимое элемента `signature` как выровненное по правому краю (строка 77), которому будет предшествовать литерал `From:` (строка 79).

Главное преимущество данного подхода состоит в том, что `invtab1.dsl`, наш DSSSL-лист стиля, можно использовать с программой Jade, чтобы получить форматированный вывод в различных видах. В частности, можно сформировать вывод в формате RTF (для просмотра или редактирования с помощью средств фирмы Microsoft; рис. 7.6 показывает RTF-вывод, просматриваемый с помощью программы Microsoft Word97):

```
jade -dinvtab1.dsl -t rtf d:\jade\xml.dcl invitation.xml
```

Можно сформировать также и вывод в формате TeX'а. Для этой цели используются флаг `-t tex`, а также постпроцессорная обработка полученного TeX'овского файла с помощью JadeTeX, что обсуждается более подробно в разд. 7.5.4. Получаемое представление файла вывода показано на рис. 7.7.

```
jade -dinvtab1.dsl -t tex d:\jade\xml.dcl invitation.xml
jadetex invitation
```



INVITATION

To: Anna, Bernard, Didier, Johanna
When: Next Friday Evening at 8 pm
Venue: The Web Cafe
Occasion: My first XML baby

I would like to invite you all to celebrate the birth of *Invitation*, my first XML document child.

Please do your best to come and join me next Friday evening. And, do not forget to bring your friends.

I *really* look forward to see you soon!

From: Michel

Рис. 7.6. Просмотр вывода в формате RTF с помощью программы Word97.

INVITATION

To: Anna, Bernard, Didier, Johanna
When: Next Friday Evening at 8 pm
Venue: The Web Cafe
Occasion: My first XML baby

I would like to invite you all to celebrate the birth of *Invitation*, my first XML document child.

Please do your best to come and join me next Friday evening. And, do not forget to bring your friends.

I *really* look forward to see you soon!

From: Michel

Рис. 7.7. PostScript-форма TEX'овского вывода.

Следует подчеркнуть, что единственная разница между двумя упомянутыми прогонами программы Jade заключается в значении флага `-t`.

7.5.3.3 Работа с атрибутами

Можно также написать лист стиля для работы с исходным файлом `invitation2.xml`. Здесь требуется организовать доступ к значениям атрибутов элементов, чтобы выполнить форматирование документа. Такого рода лист стиля `invtab2.dsl` имеет вид, показанный ниже:

```
<!DOCTYPE style-sheet PUBLIC "-//James Clark//DTD DSSSL Style Sheet//EN">
<style-sheet>
<style-specification>
<style-specification-body>
(define FontSize 12pt)
(root
  (make simple-page-sequence
    left-margin:  1cm
    page-width:   10cm
    right-margin: 1cm
    (make scroll
      font-size: FontSize
      (process-children))))
(element invitation
  (make sequence
    (make paragraph
      quadding:    'center
      space-before:   20pt
      font-weight: 'bold
      font-size:   24pt
      (literal "INVITATION"))
    (make table
      table-border:   #f  ; no border
      display-alignment: 'start
      space-before:   20pt
```

```
26          (make table-part
27            (make table-column width: 25mm)
28            (make table-column width: 10cm)
29            (make table-row
30              (make table-cell
31                (make paragraph quadding: 'start
32                  (literal "When:")))
33              (make table-cell
34                (make paragraph quadding: 'start
35                  (literal (attribute-string "date")))))
36            (make table-row
37              (make table-cell
38                (make paragraph quadding: 'start
39                  (literal "To:")))
40              (make table-cell
41                (make paragraph quadding: 'start
42                  (literal (attribute-string "to")))))
43            (make table-row
44              (make table-cell
45                (make paragraph quadding: 'start
46                  (literal "Venue:")))
47              (make table-cell
48                (make paragraph quadding: 'start
49                  (literal (attribute-string "where")))))
50            (make table-row
51              (make table-cell
52                (make paragraph quadding: 'start
53                  (literal "Occasion:")))
54              (make table-cell
55                (make paragraph quadding: 'start
56                  (literal (attribute-string "why")))))))
57        (process-children)
58        (make paragraph
59          quadding: 'end
60          space-before: FontSize
61          (literal "From: "
62            (attribute-string "signature")))))
63    (element par
64      (make paragraph
65        quadding: 'justify
66        font-size: FontSize
67        space-before: FontSize
68        (process-children-trim)))
69    (element emph
70      (make sequence
71        font-posture: 'italic
72        (process-children-trim)))
73    </style-specification-body>
74    </style-specification>
75    </style-sheet>
```

Здесь используется та же самая стратегия построения таблицы, что и в разд. 7.5.3.2. Однако вместо правила `process-children` информация извлекается с помощью процедуры `attribute-string`, для которой как символьная строка задается имя атрибута (см. строки 35, 42, 49, 56 и 62). Уменьшим здесь также несколько поля и ширину страницы (сравните строки 8–10 в двух этих вариантах).

Исполним ту же самую команду `jade`, что и в приведенном выше примере, изменив при этом DSSSL-лист стиля на `invtab2.dsl`, а входной XML-файл на `invitation2.xml`. После этого пропустим полученный TEX'овский файл через

Рис. 7.8. PostScript-форма TEX'овского вывода (альтернативный вариант DSSSL-форматирования).

`jadetex` и получим, наконец, файл в формате EPS, показанный на рис. 7.8, который следует сравнить с рис. 7.7.

7.5.4 Постпроцессор TEX'а для Jade и макропакет JadeTEX

Как уже отмечалось выше, в состав Jade также входит постпроцессор для TEX'а, что предоставляет следующие преимущества:

1. TEX является свободно доступным, понятным и работающим на всех компьютерных платформах.
2. TEX разработан для пакетной издательской обработки, основанной на типографских правилах.
3. TEX является хорошим средством для верстки страниц и очень хорошим средством для верстки абзацев.
4. TEX понимает весь диапазон типографских тонкостей (переносы, шрифты, математическая символика и т. д.).

Имеется также перспектива использования Unicode непосредственно в TEX'е, что делается с помощью программного комплекса Omega [↪OMEGA]. Он позволяет работать со сложными письменностями, разными направлениями письма и языковой спецификой там, где TEX сталкивается с трудностями.

Таким образом, при использовании стилей Jade появляется возможность написания спецификаций, независимых от конкретной компьютерной платформы, а также использования всей мощи TEX'а для непосредственной их обработки.

TEX'овский постпроцессор для Jade первоначально был написан Дэвидом Меггинсоном, а позднее модифицирован Себастианом Ратцем и Кэтлин Маршалек. В этом постпроцессоре используется очень простая модель: он выдает TEX'овскую команду в начале и при завершении каждого потокового объекта, определяя любые изменившиеся характеристики при запуске данной команды. Эта абстрактная TEX'овская разметка может быть затем «материализована» путем написания определений для каждой из команд для потоковых объектов, и это именно то, что делает макропакет JadeTEX.

JadeTEX реализован как надстройка над широко используемым макропакетом LATEX в первую очередь потому, что LATEX обеспечивает стандартизованную поддержку работы со шрифтами (New Font Selection System — Новая система выбора шрифтов), сходную с той, что применяется в DSSSL. Удобна здесь также поддержка работы со многими языками, с цветом, включением в документ графики, гипертекстовых возможностей, табличных пакетов. Это значит, что JadeTEX обеспечивает существенное сокращение времени, требуемого на реализацию документа, и позволяет видеть, в какой степени TEX может реально удовлетворить запросам DSSSL. Для улучшения характеристик можно было бы переписать средства работы со шрифтами, имеющиеся в постпроцессоре Jade, а также оптимизировать работу с метками и ссылками, чтобы минимизировать расход памяти для перекрестных ссылок.

Для тех пользователей, которые регулярно работают с LATEX'ом, важно понимать, что рассматриваемый постпроцессор не может использовать LATEX'овские конструкции высокого уровня, а также привычные для них файлы классов (стилевые файлы). Язык DSSSL выражает все в терминах потоковых элементов и не имеет обозначений для таких концептуальных элементов, как разделы, списки, перекрестные ссылки, библиографии. Таким образом, прерогативой TEX'а является лишь завершение страниц и строк; все остальное задается программным кодом DSSSL.

Можно было бы рассмотреть некоторую систему, которая транслирует саму по себе стилевую спецификацию DSSSL в файл класса LATEX'а и преобразует конкретный экземпляр документа в LATEX'овский файл, используя языковые конструкции высокого уровня, однако, по имеющимся у нас данным, попыток такого рода пока не предпринималось.

7.5.4.1 Установка и использование

После загрузки макрокоманд JadeTEX из архива CTAN [↪JADETEX] требуется сделать так, чтобы они были доступны для TEX'а. Наиболее удобным было бы построить новый форматный файл для TEX'а. Последовательность команд, которую надо ввести, чтобы выполнить эту работу, могла бы выглядеть так

(если используется современная система TEX, основанная на Web2c 7.2 или более поздних вариантах):

```
tex jadetex.ins
pdftex -ini "&pdflatex" -progname=pdfjadetex pdfjadetex.ini
tex -ini "&hugelatex" -progname=hugetex jadetex.ini
```

В результате будет получено два форматных файла, `pdfjadetex.fmt` и `jadetex.fmt`, которые надо переместить в каталог, где TEX сможет найти их. Необходимо установить «огромную» версию LATEX'а (`hugelatex` в третьей строке), если ее еще нет на данном компьютере, поскольку JadeTEX чрезвычайно «прожорлив» в отношении памяти. Пользователи системы Web2c 7.2 (или какой-либо более поздней ее версии) могут сделать это, добавив следующие строки в файл `texmf.cnf`:

```
main_memory.hugetex = 1100000
hash_extra.hugetex = 15000
pool_size.hugetex = 500000
string_vacancies.hugetex = 45000
max_strings.hugetex = 55000
pool_free.hugetex = 47500
nest_size.hugetex = 500
param_size.hugetex = 1500
save_size.hugetex = 5000
stack_size.hugetex = 1500
```

и исполнив команду

```
tex -ini -fmt=hugelatex -progname=hugetex latex.ltx
```

которая построит форматный файл `hugelatex.fmt`. Другие реализации TEX'а используют свои собственные методы для построения форматных файлов, позволяющих работать с расширенными объемами памяти, поэтому при их использовании следует обратиться к соответствующей документации по ним.

Если все перечисленные выше действия выполнены успешно, можно теперь формировать TEX'овские файлы и пропускать их через JadeTEX:

```
jade -dinvtab1.dsl -t tex xml.dcl invitation.xml
jadetex invitation   или   pdfjadetex invitation
dvips -E invitation -oinvitation
```

Первая строка в приведенном выше тексте формирует TEX'овский файл, который содержит команды, отражающие структуру потоковых объектов DSSSL. Эти команды интерпретируются с помощью макро JadeTEX и pdfTEX или LATEX'а (строка 2), формируя, соответственно, напрямую файл в формате PDF или DVI, который можно преобразовать в EPS-файл. Эта процедура была использована для получения рис. 7.7.

7.5.4.2 JadeTEX вблизи

Чтобы пояснить, каким образом JadeTEX работает с информацией DSSSL о характеристиках потоковых объектов, рассмотрим следующий пример:

```
1  (root (make simple-page-sequence
2          center-footer: (page-number-sosofo)   \SpS{%
3          font-family-name: body-font-family      \def\fFamName{iso-serif}
4          page-n-columns: 2                       \def\PageNColumns{2}
5          page-column-sep: 16pt                   \def\PageColumnSep{16\p@}
6          header-margin: .5in                     \def\HeaderMargin{36\p@}
7          footer-margin: .5in                     \def\FooterMargin{36\p@}
8          left-margin: 1in                        \def\LeftMargin{72\p@}
9          right-margin: 1in                       \def\RightMargin{72\p@}
10         top-margin: 1in                         \def\TopMargin{72\p@}
11         bottom-margin: 1in                      \def\BottomMargin{72\p@}
12         page-width: 211mm                       \def\PageWidth{598.11\p@}
13         page-height: 297mm))                    \def\PageHeight{841.89\p@}}
```

В левой части этого примера видны спецификации DSSSL, которые задают параметры для простой последовательности страниц. В правой части видно как TEX'овский постпроцессор Jade переводит эти спецификации в некоторую промежуточную форму, которая без труда может быть переработана макрокомандами JadeTEX (в [↪JADETEXB] дается подробное описание этих параметров и процесса трансляции).

Рассмотрим тело некоторого документа с простой XML-разметкой:

```
some <it>go italic</it> others not...
```

Рассмотрим также следующую декларацию DSSSL:

```
1  (element it
2    (make sequence
3      font-posture: 'italic
4      (process-children-trim)))
```

Постпроцессор TEX'а из состава Jade сформирует следующий текст:

```
1  some \Node{\def\Element{11}}%
2  \Seq{\def\fPosture{italic}}%
3  go italic
4  \endSeq{}\endNode{}  others
5  not...\endSeq{}\endNode{}
```

Обратим внимание, как содержимое элемента «курсивного» типа `it` было обработано как DSSSL-последовательность, которая транслируется в TEX'овские макро `\Seq`...`\endSeq`. Требуемые изменения в характеристике `font-posture` выражаются с помощью TEX'овского макроопределения как параметр для `\Seq`. Почти каждый объект, выходящий из Jade, имеет идентификатор «Element» (строка 1), который может быть использован для установления перекрестных ссылок.

7.5.4.3 JadeTEX и математика

А как обстоит дело с математикой? Это традиционно одна из наиболее сильных сторон TEX'а, и мало какие другие издательские системы справляются с этой задачей так же успешно. Рассмотрим соответствующий пример; XML-разметка и отображаемый результат должны ясно показать имеющиеся здесь возможности.

```
<fd><fr><nu>X</nu><de>Y></de></fr></fd>
```

$$\frac{X}{Y}$$

DSSSL-спецификацию можно было бы записать так:

```
; displayed equation
(element fd
 (make display-group
 (make math-sequence
   math-display-mode: 'display
   min-leading: 2pt
   font-posture: 'math
   (process-children-trim))))
; fraction
(element fr
 (make fraction
   (process-children-trim)))
(element nu
      (make math-sequence
         label: 'numerator
   (process-children-trim)))
(element de
      (make math-sequence
         label: 'denominator
   (process-children-trim)))
```

Здесь используется слегка измененная DSSSL-концепция «портов» (это характеристика `label`), которая позволяет задать числитель и знаменатель плавающего объекта типа «дробь». TEX'овский постпроцессор из состава Jade преобразовал бы этот текст в следующий:

```
\DisplayGroup{}
  \MathSeq{\def\MathDisplayMode{display}
           \def\MinLeading{2\p@}
           \def\MinLeadingFactor{0}
           \def\fPosture{math}}
    \FractionSerial{}
      \insertFractionBar{}
      \FractionNumerator{}
        \MathSeq{}X\endMathSeq{}
      \endFractionNumerator{}
      \FractionDenominator{}
        \MathSeq{}Y\endMathSeq{}
      \endFractionDenominator{}
    \endFractionSerial{}
  \endMathSeq{}
\endDisplayGroup{}
```

Ясно видно, как сохраняется структура первоначальной формулы. Благодаря командам TEX'а, которые вызываются на каждом уровне, имеется возможность достаточно просто модифицировать способ отображения различных математических элементов исходя из потребностей пользователя. Реализация по умолчанию (упрощенная) этих макро выглядит следующим образом:

```
\def\FractionSerial#1{#1\bgroup}
\def\endFractionSerial{\egroup}
```

```
\def\FractionDenominator{}
\def\endFractionDenominator{}
\def\FractionNumerator{}
\def\endFractionNumerator{\over }
\def\insertFractionBar{}
```

Интересной инициативой является работа Дэвида Карлайла по DSSSL-стилю для математических выражений на основе MathML DTD [↪DSSSLMML]; данная работа будет рассмотрена более подробно в разд. 8.2.4.

7.5.4.4 Пригоден ли JadeTEX для практического использования?

Обрабатывать простые тексты с помощью Jade нетрудно, получая более или менее идентичный вывод из постпроцессоров для форматов RTF и TEX (рис. 7.6 и 7.7). Для обычных научных публикаций, однако, простая модель страницы, которую реализует Jade, недостаточна. В связи с этим постпроцессор TEX'а реализует некоторое число расширений[1], которые могут быть активизированы с помощью следующего DSSSL-кода:

```
(declare-flow-object-class page-float
      "UNREGISTERED::Sebastian Rahtz//Flow Object Class::page-float")
(declare-flow-object-class page-footnote
      "UNREGISTERED::Sebastian Rahtz//Flow Object Class::page-footnote")
(declare-characteristic page-n-columns
      "UNREGISTERED::James Clark//Characteristic::page-n-columns" 1)
(declare-characteristic page-column-sep
      "UNREGISTERED::James Clark//Characteristic::page-column-sep" 4pt)
```

Эти декларации позволяют задать страницу с набором материала в ней в две колонки, со спецификацией вроде следующей:

```
(make simple-page-sequence
            page-n-columns: 2
            page-column-sep: 16pt
            ..... )
```

и создать новый потоковый объект типа «примечание» (`page-footnote`) или новый плавающий потоковый объект (`page-float`). Получаемый при этом результат показан на рис 7.9; видно, что DSSSL-спецификация, Jade и JadeTEX могут формировать вполне приемлемый вывод для научных текстов. С другой стороны, обнаруживается, что поддержка действий с элементами математического характера в формате RTF слишком бедна, так что для математики (на данный момент) единственный реалистичный путь при форматировании вывода из XML состоит в использовании TEX'а и JadeTEX.

[1] Поддержка работы с многоколонным набором обеспечивается также и постпроцессором для формата RTF.

Test file for math, multicolumns, and footnotes

Sebastian Rahtz

Abstract: Altera C. Caesaris, qui illos publicatis bonis per municipia Italiae distribueudos ac vinculis sempiternis tenendos existimabat. Cum acautem plures senatores ad C. Caesaris quam ad D. Silani sententiam inclinare viderentur, M. Cicero ea, quae infra legitur, oratione Silani sententiam commendare studuit.

1. Maths tests

0. Simple fraction

$$\frac{X}{Y}$$

1. display equation with radical 123 and fraction

$$\frac{(x+y)+\sqrt{123}}{2}$$

2. Display equation with super and subscripts

$$1I/I_0 = (1-\Theta)^{\sum_1^g}.$$

3. Matrix with braces

$$\{ \begin{array}{cccccc} a & b & c & d & e & f \end{array} \}a = 0b = 2$$

4. Line with | (after matrix)

$$\{ \begin{array}{cccccc} a & b & c & d & e & f \end{array} \}|a = 0b = 2$$

5. Line with | (before matrix)

$$|\{ \begin{array}{cccccc} a & b & c & d & e & f \end{array} \}a = 0b = 2$$

6. Nested matrix with braces

$$(X\{ \begin{array}{cccccc} a & b & c & d & e & f \end{array} \}a = 0b = 2)$$

7. Nested fraction

$$\frac{(x+y)+\frac{X}{Y}}{2} \qquad \frac{(x+y)^2-4a}{2}$$

8. Fence

$$(\{ \begin{array}{cccccc} aaa & b & c & d & e & f \end{array} \}|_{\sin\alpha}^{b\times 5=\sqrt{49202}} a = 0b = 2$$

9. Boxing $\boxed{A+B}$

10. Some operators: summation, product, and integral. First, display math:

$$\sum_a^b \quad \prod_c^d \quad \int_e^f \quad \frac{h}{\rightleftharpoons g}\sin\alpha \quad \sum_{1111_{aaa}^{bbb}}^{2222_{8888}^{3333}}$$

Now inline math: $\sum_a^b \prod_c^d \int_e^f \frac{h}{\rightleftharpoons g}\sin\alpha \sum_{1111_{aaa}^{bbb}}^{2222_{5555}^{3333}}$

11. A radical with a radix

$$\sqrt[3]{123}$$

1.1. Second-level header

2. Footnotes

13. A footnote, number 63[63] A footnote, number 65[65] A footnote, number 64[64]

3. Special character entities

AElig	Æ
And	∧
Cap	⋒
Colon	∷
Cup	⌣
Dagger	‡
Delta	Δ
ETH	Ð
Gamma	Γ
Gt	≫
Lambda	Λ
Larr	↞
Lt	≪
OElig	Œ
Omega	Ω
Or	
Oslash	Ø

[63] Never leave home without rope, Sam could have told you.
[65] It always pays to be polite to trees that walk and talk
[64] Little can beat stewed rabbit in the heather.

Рис. 7.9. Математический текст, сформированный с помощью SGML, DSSSL и TEX'a.

Потенциальная мощь SGML/XML, DSSSL и TEX'a, работающих совместно, выглядит многообещающе. Хотя при этом и возникает ряд проблем, реализация DSSSL в Jade уже поддерживает значительное число полезных преобразований и кодов спецификаций, при этом TEX близок к тому, чтобы стать средством форматирования, имеющим возможность работы с DSSSL.

7.5.5 Интерфейс с SGML-преобразованиями в Jade

Jade не реализует язык преобразований (Transformation Language) из DSSSL. Однако эта программа обеспечивает некоторые простые нестандартные расширения языка стилей (Style Language) из DSSSL, что дает возможность использовать данную программу для SGML-преобразований.

Эти расширения Jade доступны для применения, если задать опции `-t sgml` и `-t xml`; во втором из этих двух случаев используются синтаксис XML для пустых элементов и обрабатывающие инструкции.

Ниже приводится список главных классов потоковых объектов SGML, которые определяются в Jade, чтобы дополнить стандартный набор, присутствующий в DSSSL. Эти расширения включают коллекции классов потоковых объектов и их ненаследуемых характеристик. Они используются вместо стандартных классов потоковых объектов, определенных в DSSSL.

`element` Составной потоковый объект, который может иметь потомков в виде потоковых объектов, таких, что для объектов данного типа формируются начальный и завершающий дескрипторы.

`empty-element` Атомарный потоковый объект, у которого не может быть потомков в виде потоковых объектов, поэтому для объекта типа `empty-element` формируется только стартовый дескриптор. Главным образом он предназначен для работы с элементами, для которых в качестве содержимого объявлено `EMPTY`.

Как объект типа `element`, так и объект типа `empty-element` имеют две характеристики:

- `gi` Строка, задающая родовой идентификатор данного элемента (по умолчанию: родовой идентификатор текущего узла).
- `attributes` Атрибуты элемента как некоторый список списков, каждый из которых содержит в точности две строки: первая из них задает имя атрибута, вторая — значение данного атрибута (по умолчанию: пустой список).

`processing-instruction` Атомарный потоковый объект, реализующийся в виде обрабатывающих инструкций. Его характеристика:

- `data` Строка, задающая информационное содержимое обрабатывающей инструкции (по умолчанию: пустая строка).

`document-type` Атомарный потоковый объект, формирующий декларацию `DOCTYPE`. Возможные характеристики для него:

- `name` *Требуемая* строка, задающая имя типа документа, которое должно быть идентично имени типа элемента документа.
- `system-id` Строка, задающая системный идентификатор для типа документа (по умолчанию: пустая строка).
- `public-id` Строка, задающая публичный идентификатор для типа документа (по умолчанию: пустая строка).

`entity` Составной потоковый объект, запоминающий его содержимое в отдельном объекте. Характеристика, возможная для него:

`system-id` Системный идентификатор для объекта. Это должно быть некоторое имя файла.

Этот потоковый объект не порождает никаких деклараций или ссылок на объекты.

`entity-ref` Атомарный потоковый объект, создающий ссылку на объект. Он поддерживает одну характеристику:

`name` Имя объекта.

`formatting-instruction` Атомарный объект, который вставляет некоторые литеры в вывод, не меняя их. Он имеет единственную характеристику:

`data` Вставляемая строка. Перед литерами `&`, `<` и `>` не требуется ставить escape-символ.

В любой DSSSL-спецификации, где используются эти классы потоковых объектов, нужно следующим образом объявлять их с помощью класса `declare-flow-object-class`:

```
(declare-flow-object-class element
  "UNREGISTERED::James Clark//Flow Object Class::element")
(declare-flow-object-class empty-element
  "UNREGISTERED::James Clark//Flow Object Class::empty-element")
(declare-flow-object-class document-type
  "UNREGISTERED::James Clark//Flow Object Class::document-type")
(declare-flow-object-class processing-instruction
  "UNREGISTERED::James Clark//Flow Object Class::processing-instruction")
(declare-flow-object-class entity
  "UNREGISTERED::James Clark//Flow Object Class::entity")
(declare-flow-object-class entity-ref
  "UNREGISTERED::James Clark//Flow Object Class::entity-ref")
(declare-flow-object-class formatting-instruction
  "UNREGISTERED::James Clark//Flow Object Class::formatting-instruction")
```

Ниже приводится стиль, который преобразует исходный XML-файл с именем `invitation.xml` в HTML-файл, используя интерфейс преобразований Jade:

```
<!DOCTYPE style-sheet PUBLIC "-//James Clark//DTD DSSSL Style Sheet//EN">
<style-sheet>
<style-specification>
<style-specification-body>
(declare-flow-object-class element
  "UNREGISTERED::James Clark//Flow Object Class::element")
(declare-flow-object-class empty-element
  "UNREGISTERED::James Clark//Flow Object Class::empty-element")
(declare-flow-object-class document-type
  "UNREGISTERED::James Clark//Flow Object Class::document-type")

(define FontSize 12pt)
(root
  (make simple-page-sequence
    left-margin:  25mm
    page-width:   205mm
    right-margin: 25mm
    (make sequence
      font-size: FontSize
      line-spacing: FontSize
```

```
21          (make document-type
22            name: "HTML"
23            public-id: "-//W3C//DTD HTML 3.2//EN")
24          (make element gi: "HEAD"
25            (make element  gi: "TITLE"
26              (literal "Invitation (XML to HTML transformation)"))
27            (make empty-element  gi: "LINK"
28                          attributes: (list (list "href" "invit.css")
29                                            (list "rel"  "stylesheet")
30                                            (list "type" "text/css"))))
31          (make element gi: "BODY"
32            (make sequence
33              (make element gi: "H1"
34                (literal "INVITATION"))
35              (process-children))))))
36
37    (element (front)
38      (make element gi: "TABLE"
39                    attributes: (list (list "border" "5")
40                                      (list "frame"  "hsides")
41                                      (list "rules"  "none")
42                                      (list "width"  "100%"))
43        (process-children)))
44    (element (front date)
45      (make element gi: "TR"
46        (make sequence
47          (make element gi: "TD"
48                        attributes: (list (list "class" "front"))
49            (literal "When: "))
50          (make element gi: "TD"
51            (process-children)))))
52    (element (front to)
53      (make element gi: "TR"
54        (make sequence
55          (make element gi: "TD"
56                        attributes: (list (list "class" "front"))
57            (literal "To: "))
58          (make element gi: "TD"
59            (process-children)))))
60    (element (front where)
61      (make element gi: "TR"
62        (make sequence
63          (make element gi: "TD"
64                        attributes: (list (list "class" "front"))
65            (literal "Venue: "))
66          (make element gi: "TD"
67            (process-children)))))
68    (element (front why)
69      (make element gi: "TR"
70        (make sequence
71          (make element gi: "TD"
72                        attributes: (list (list "class" "front"))
73            (literal "Occasion: "))
74          (make element gi: "TD"
75            (process-children)))))
76    (element (body par)
77      (make element gi: "P"
78        (process-children)))
79    (element emph
80      (make element gi: "EM"
81        (process-children)))
82    (element (back signature)
83      (make element gi: "P"
84                    attributes: (list (list "class" "signature"))
85        (make sequence
```

```
86          (literal "From: ")
87            (process-children))))
88    </style-specification-body>
89    </style-specification>
90    </style-sheet>
```

Теперь, когда мы рассмотрели довольно много примеров кода на DSSSL, этот стиль DSSSL должен быть простым и легко понимаемым. Обратим внимание на потоковые объекты «SGML» в строках 21 (`document-type`), 24, 25, 31, 33, ... (`element`) и 27 (`empty-element`); эти специальные потоковые объекты были объявлены в строках 5–10. Обращаясь к характеристикам этих потоковых элементов, обсуждавшимся ранее, обнаруживаем в строках 22 и 23 характеристики `name` и `public-id`, связанные с типом документа. Для упомянутых элементов (пустых или нет) необходимо задать родовой идентификатор (тип элемента в терминологии XML) `gi`, надо также задать атрибуты, если они здесь требуются. В строках 28–30 видно, как атрибуты задаются в виде списка списков, каждый из которых состоит из пары строк: первая из них есть имя атрибута, вторая — значение этого атрибута. Например, спецификация элемента `LINK` в строках 27–30 дает результат, который можно наблюдать в строке 4 приводимого HTML-файла. Аналогичным образом, обратим внимание на способ, которым задаются атрибуты для элемента типа `TABLE` (строки 39–42), дающий результат в виде строки 8 в соответствующем HTML-файле.

Получающийся в результате HTML-файл `invitation.html`, который приводится ниже, получен путем исполнения такой команды:

```
jade -dinvhtml.dsl -t xml -oinvitation.html xml.dcl invitation.xml
```

```
1   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
2   <HEAD>
3   <TITLE>Invitation (XML to HTML transformation)</TITLE>
4   <LINK href="invit.css" rel="stylesheet" type="text/css">
5   </HEAD>
6   <BODY>
7   <H1>INVITATION</H1>
8   <TABLE border="5" frame="hsides" rules="none" width="100%">
9   <TR><TD class="front">To: </TD>
10      <TD>Anna, Bernard, Didier, Johanna</TD></TR>
11  <TR><TD class="front">When: </TD>
12      <TD>Next Friday Evening at 8 pm</TD></TR>
13  <TR><TD class="front">Venue: </TD>
14      <TD>The Web Cafe</TD></TR>
15  <TR><TD class="front">Occasion: </TD>
16      <TD>My first XML baby</TD></TR>
17  </TABLE>
18  <P>I would like to invite you all to celebrate the
19  birth of <EM>Invitation</EM>, my first XML document child.</P>
20  <P>Please do your best to come and join me next Friday
21  evening. And, do not forget to bring your friends.</P>
22  <P>I <EM>really</EM> look forward to see you soon!</P>
23  <P class="signature">From: Michel</P>
24  </BODY>
```

Рис. 7.10 показывает результат просмотра данного HTML-файла с помощью некоторого HTML-навигатора, который также использует наш стиль в формате CSS (`invit.css`) для модификации характера форматирования различных HTML-элементов. Обратите внимание на способ задания атрибутов для

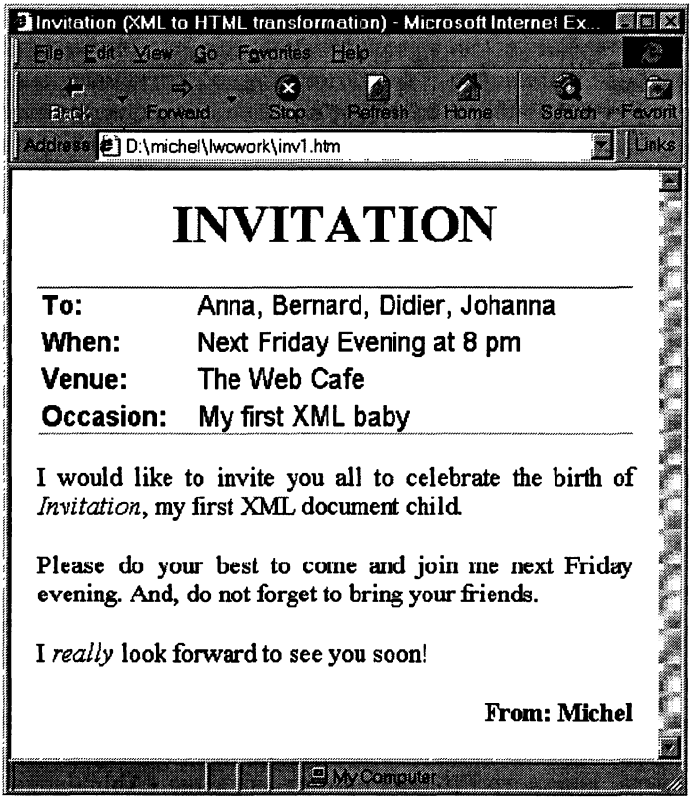


Рис. 7.10. Преобразование из XML в HTML с помощью DSSSL.

элементов `TABLE`: в стиле XSL определено, что во всю ширину экрана формируются горизонтальные строки над табличным материалом и под ним.

7.5.6 Форматирование реальных документов с помощью языка DSSSL

Разумеется, приведенные в данном разделе простые примеры дают лишь очень ограниченное представление о возможностях системы DSSSL. Поэтому надо показать более сложный случай, прежде чем говорить что-либо относительно решений, основанных на использовании DSSSL и рассчитанных на реальные практические задачи.

Пусть мы работаем в многоязычной среде и, как и в случае LaTeX'овского пакета `babel`, требуется, чтобы некоторые символьные строки правильно выводились в каждом из языков, представляющих для нас интерес. Рассмотрим следующий фрагмент программного кода:

```
1  (define (FIGNAME)
2    (case (inherited-attribute-string "xml:lang")
```

```
3       (("de") "Abbildung ")
4       (("nl") "Figuur ")
5       (else "Figure ")))
6   (define (TABNAME)
7     (case (inherited-attribute-string "xml:lang")
8       (("de") "Tabelle ")
9       (("nl") "Tabel ")
10      (else "Table ")))
11  (element (caption)
12    (make paragraph
13      space-before: 6pt
14      space-after: 10pt
15      (make sequence
16        font-weight: 'bold
17        (literal (if (have-ancestor? "figure") (FIGNAME) (TABNAME)))
18        (literal (format-number
19          (if (have-ancestor? "figure")
20            (element-number (ancestor "figure"))
21            (element-number (ancestor "table")))
22          "1"))
23        (literal ". "))
24      (process-children-trim)))
```

В разделе 6.5.1 был введен атрибут `xml:lang`, который задает язык, на котором написано информационное содержимое элемента данного типа. Это значение атрибута наследуется всеми потомками рассматриваемого элемента. Итак, если имеется часть нашего документа, для которого упомянутый атрибут получил определенное значение, операторы `case` в строках 2 и 7 активизируют одну из строк в диапазоне 3–5 и еще одну строку из диапазона 8–10, в зависимости от результата выполнения процедуры «`(inherited-attribute-string "xml:lang")`». Например, при значении атрибута `xml:lang`, равном «`de`» для немецкого языка, параметр `FIGNAME` примет значение «Abbildung» (строка 3), а параметр `TABNAME` — значение «Tabelle» (строка 8). С помощью этих определений констант можно построить элемент `caption` (заголовок), используя символьную строку, соответствующую заданному языку. В строке 17 оператор `if` проверяет, является ли предком элемента `caption` элемент `figure`, если это так, то в этой точке в потоковый объект `paragraph` будет помещена литерная строка `FIGNAME`, в противном случае это будет строка-литерал `TABNAME`. Аналогично этому, функция строк 18–24 — поместить правильный номер рисунка или таблицы следом за текстовой строкой, выбранной ранее. Фактически, чтобы получить значение этого числа, DSSSL подсчитывает число элементов `figure` или `table` (проверка «`if`» и ветвей этого оператора в строках 19–21). Затем это число форматируется, используя процедуру `format-number` (строка 18), и представляется как «десятичное» число (следовательно, `"1"` в строке 22 задает формат, который следует использовать для данного числа)[1], за которым следует точка (строка 23).

Этот пример ясно показывает, какого рода действия можно выполнять с помощью DSSSL, используя значительное количество процедур и потоковых

[1] Раздел 8 Стандарта языка DSSSL описывает язык выражений. Именно здесь можно найти объяснение применению различных процедур, которые использованы в рассматриваемом и других аналогичных примерах. В частности, разд. 8.5.7.24 детально описывает процедуру `format-number`.

объектов, определенных в стандарте DSSSL. Еще одно полезное средство — это использование «режимов», которые позволяют осуществить обработку (и вывод) некоторых элементов более одного раза в различных вариантах. Например, заголовки разделов, подписи для рисунков и таблиц могут быть выведены дважды: один раз при формировании собственно текста документа, а второй — при сборке оглавления документа, а также перечней рисунков и таблиц.

Если надо узнать, каким образом DSSSL используется для работы с приложениями из реальной практики при обработке больших документов, надо ознакомиться с разработанными Норманом Уолшем модульными стилями DocBook [↪DBDSSSL] для форматирования SGML-документов, разметка которых произведена с помощью DocBook DTD. Н. Уолш подготовил два стиля: для печати DocBook-документов с использованием форматов RTF, TEX или MIF и для преобразования этих документов в формат HTML. В упомянутые стили введены специальные средства, дающие возможность пользователю произвести настройку вывода применительно к его собственным нуждам. Помимо практической полезности, эти сложные стили — хороший пример, на котором можно изучать DSSSL.

7.6 XSL — расширяемый язык стилей

XSL [↪XSLSPEC] представляет собой язык для описания стилей, в которых определяются правила для представления XML-элементов. Его синтаксис предполагает работу с двумя подпроцессами:

1. *Преобразование* исходного дерева XML-документа в результирующее дерево.
2. Интерпретация результирующего дерева для получения *форматирующих объектов* для вывода в различных средах, например на компьютерный экран, на бумагу, в звуковой форме.

Преобразующая часть языка XSLT может использоваться независимо от форматирующих объектов. Она надстроена над языком путей XML — XPath для того, чтобы адресовать компоненты требуемого XML-документа. В данном разделе будут рассмотрены некоторые элементы языков XPath и XSLT, тогда как форматирующие объекты будут показаны только на примере, поскольку работа в данной области пока еще не завершена.

7.6.1 XPath — средство для адресации частей документа на языке XML

XPath [↪XPATH] обеспечивает синтаксис и семантику для функциональных средств, общих для языков XSLT [↪XSLT] и XPointer [↪XPTR]. Основное назначение языка XPath состоит в адресации частей XML-документа. XPath использует компактный *не-XML*-синтаксис, чтобы облегчить его использование

при записи Универсальных идентификаторов ресурсов (URI) и значений XML-атрибутов.

XPath рассматривает XML-документ как дерево узлов. Имеется несколько типов узлов, и XPath вычисляет для каждого из этих типов значение в виде символьной строки. Некоторые типы узлов поименованы.

Основная синтаксическая конструкция в XPath — это *выражение*, которое вычисляется как объект, принадлежащий к одному из следующих четырех базовых типов:

- *node-set* (неупорядоченная коллекция узлов без дубликатов среди них);
- *boolean* (true (истина) или false (ложь));
- *number* (число с плавающей точкой);
- *string* (последовательность литер Unicode).

Вычисление выражения производится с учетом *контекста*, который включает в себя:

- контекстный узел;
- пару ненулевых положительных целых чисел (для определения позиции рассматриваемого узла, что касается размера контекста);
- набор связей между переменными, обеспечивающий отображение имен переменных в их значения, где значение представляет собой объект;
- библиотеку функций, состоящую из отображения имен функций в функции, причем каждая функция имеет нуль или более аргументов и возвращает единственный результат;
- множество деклараций для пространства имен в пределах области определения для выражений.

XPath определяет «Location Path» (путь к элементу), который обычно состоит из списка шагов локализации, разделенных символом /, имеющих вид

```
axis-name :: node-test[predicate]*
```

Например, выражение `child::emph` определяет местоположение всех потомков контекстного узла, который относится к типу элементов `emph`, а выражение `child::par/child::emph` определяет местонахождение потомков `emph`, являющихся потомками элемента `par`, который, в свою очередь, будет потомком контекстного узла.

Параметр `axis-name` задает некоторую последовательность местоположений-кандидатов, определяя некоторые ссылки, именуемые «контекстом» и, в частности, текущее местоположение, называемое «контекстный узел». Первоначально контекстным узлом является корень документа, а в более общем случае — результаты априорного шага поиска местонахождения.

Для каждого местоположения-кандидата вдоль предписанной оси производится вычисление предикатов, причем обычно осуществляется проверка типа элемента, атрибутов, координат, и/или других свойств узлов. Можно задавать несколько предикатов, которые будут служить своеобразным набором фильтров.

Например, записанное ниже выражение выбирает второго из потомков типа `emph` для элемента, который является потомком текущего узла, представляющего собой элемент типа `par`.

```
child::par/following-sibling::emph[position()=2]
```

Каждый шаг локализации фиксирует упорядоченный список фрагментов данных, который называется «список контекстных узлов» или «контекст», и любые шаги локализации интерпретируются относительно узлов из списка контекстных узлов. Шаги локализации могут быть использованы в последовательности, интерпретируемой слева направо.

Шаг локализации может быть *абсолютным*; тогда он начинается с литеры `/`, и сам по себе выбирает корневой элемент рассматриваемого документа; или же он может быть *относительным*. В случае относительного шага он состоит из одного или более шагов локализации, разделенных литерой `/`, собранных воедино слева направо. Каждый из поочередно выполняемых шагов выбирает набор узлов относительно контекстного узла. Каждый узел из этого набора используется как контекстный узел для следующего шага. Наборы узлов, идентифицируемые на определенном шаге, соединяются с помощью операции объединения множеств.

7.6.1.1 Оси

Оси в XPath зависят от существования списка контекстных узлов и локализуют другие узлы относительно каждого из узлов в списке. Узлы-кандидаты вдоль заданной оси упорядочены по расстоянию от «контекстного узла». В отсутствие списка контекстных узлов в качестве контекста принимается корневой элемент остающегося ресурса.

Каждая из относительных осей определяет местоположение списка узлов, являющихся кандидатами на членство в результирующем списке контекстных узлов. Фактические результаты выбираются из состава кандидатов с использованием предиката (см. разд. 7.6.1.3). Если не задано никакого предиката, выбираются все кандидаты из оси. Все относительные оси допускают одни и те же формы предикатов в качестве аргументов. Относительными осями в XPath являются:

`child` Определяет местоположение узлов-потомков всех типов из контекстного узла (атрибуты и пространства имен как потомки здесь не рассматриваются).

`descendant` Определяет местоположение для всех типов нисходящих узлов, где бы они ни появились в пределах информационного содержимого контекстного узла.

`descendant-or-self` Идентична оси `descendant`, за исключением того, что сам по себе контекстный узел также включается в качестве кандидата, предшествуя при этом всем нисходящим узлам.

`parent` Определяет местоположение предка, если таковой существует, для узла, содержимое которого рассматривается.

`ancestor` Определяет местоположение восходящих узлов для данного контекстного узла здесь первый узел в списке будет являться непосредственным предком, а последний узел в этом списке — корнем рассматриваемого документа.

`ancestor-or-self` Аналогична оси `ancestor`, но включает также в качестве кандидата и сам контекстный узел, который будет предшествовать всем восходящим узлам.

`preceding-sibling` Определяет местоположение соседних узлов, имеющих общего с контекстным узлом предка и предшествующих контекстному узлу. Эти узлы будут представлены в *обратном порядке*: первый узел в списке будет непосредственно предшествовать контекстному узлу, а последний — будет первым потомком рассматриваемого родительского узла.

`following-sibling` Определяет местоположение соседних узлов (имеющих общего с контекстным узлом предка), которые следуют за контекстным узлом. Эти узлы с списке появятся в *прямом порядке*.

`preceding` Определяет местоположение узлов, которые предшествуют контекстному узлу в целом. Получаемый список выстроен *в обратном порядке*: узел, наиболее близкий к контекстному узлу, идет первым, корневой узел — последним; восходящие узлы исключаются, так же, как узлы-атрибуты и узлы-пространства имен.

`following` Определяет местоположение узлов, которые следуют за контекстным узлом в целом. Список составляется в *прямом порядке*: первый узел в этом списке — для первого узла, для которого открывающая метка появляется после закрывающей метки контекстного узла; восходящие узлы исключаются, так же, как узлы-атрибуты и узлы-пространства-имен.

`self` Определяет местоположение (для каждого контекстного узла в списке контекстных узлов), одноэлементное множество (список узлов), включающее контекстный узел. Эта ось полезна для применения набора предикатов к единичной оси, в частности, когда надо проверить предикат, не являющийся первым, для позиции контекстного узла вдоль всех таких контекстных узлов, которые выбраны с помощью предшествующих предикатов.

`attribute` Атрибуты контекстного узла. Если контекстный узел не принадлежит к тому же типу, что и рассматриваемый элемент, то рассматриваемый список пуст. Порядок узлов по этой оси не определен. Обычно по имени выбирается единственный атрибут.

`namespace` Узлы пространства имен контекстного узла. Если контекстный узел не принадлежит к тому же типу, что и рассматриваемый элемент, то рассматриваемый список пуст.

7.6.1.2 Тестирование узлов

Каждая ось имеет *главный тип узла*, представляющий собой *атрибут* для оси `attribute`, *пространство имен* для действия `namespace` и *элемент* для всех остальных осей.

Проверка узла, запускаемая с помощью составного имени (это комбинация из пространства имен и имени без двоеточия) дает результат «true», если и только если данный узел будет принадлежать к «главному типу узла», с именем, равным тому, что задается с помощью составного имени. Например, `child::par` выбирает потомков элемента `par` из контекстного узла; если рассматриваемый контекстный узел не имеет потомков типа `par`, выбирается пустое множество узлов. Аналогично, `attribute::href` выбирает атрибут `href` контекстного узла; если данный контекстный узел не имеет атрибута `href`, выбирается пустое множество узлов.

Тест узла `text()` является истинным для любого текстового узла. Например, для того чтобы отобрать все текстовые узлы, являющиеся потомками данного контекстного узла, можно записать `child::text()`. Аналогичным образом, тест узла `processing-instruction()` будет истинным для любой обрабатывающей инструкции, а тест `comment()` выбирает узлы-комментарии. Наконец, тест узла `node()` будет истинным для любого узла, к какому бы типу он ни принадлежал.

7.6.1.3 Предикаты

Для чего именно определяется местоположение на данном шаге локализации, определяется используемой осью и предикатом (предикатами) — тесты узлов можно рассматривать как некий специальный случай предиката. Используемая ось определяет упорядоченный список потенциальных кандидатов, а предикаты (если есть хотя бы один из них) осуществляют выбор фактических результатов из представленных кандидатов, причем делается это следующим образом:

1. Конкретная используемая ось определяет упорядоченный список узлов-кандидатов. Если список контекстных узлов состоит более чем из одного элемента, оси применяются с использованием каждого узла в списке контекстных узлов в качестве контекстного узла, а полученные результаты сводятся воедино с помощью операции объединения множеств.
2. Затем осуществляется исполнение тестов узлов и предикатов (если есть хотя бы один), причем каждый из узлов-кандидатов используется в качестве контекста при выполнении этой оценки, т. е. в качестве «контекстного узла». На этом шаге исключаются все узлы-кандидаты, для которых тест узла и предикаты не дали при их выполнении значения «true».
3. Точно такой же процесс надлежащим образом повторяется для каждого дополнительного предиката.

Если не задано ни одного предиката или теста узла, все узлы-кандидаты считаются принятыми (включенными) в результирующий список узлов.

Например, `child::par[position()=2] / following-sibling::text()` выбирает все соседние текстовые узлы второго элемента `par`, являющегося потомком контекстного узла.

Еще один интересный случай — запись `footnote[ancestor::footnote]`, которая находит узлы `footnote`, являющиеся восходящими узлами для другого элемента `footnote`, т. е. здесь мы имеем дело с вложенными друг в друга элементами `footnotes`. Поскольку это не разрешается, например в LaTeX'e, можно воспользоваться этим образцом для выдачи предупреждения о наличии конструкций, которые во всем остальном позволяется использовать в определении DTD (см. разд. B.4.5.1, где содержится обсуждение данного вопроса).

7.6.1.4 Сокращенный синтаксис

Имеется *сокращенный* синтаксис, позволяющий задавать часто встречающиеся комбинации. Например, `child::par[position()=2]` можно переписать в форме `par[2]`. Атрибуты идентичны литере «`@`», так что местоположение атрибута `when` элемента `invitation`, который идет сразу за корневым элементом, может быть определено с помощью выражения `/invitation/@when`.

В более общем случае соотношения «восходящий-нисходящий» («предок-потомок») могут быть выражены с использованием оператора `//`, который допускает запись нуля или более поколений между элементом с левой и с правой стороны оператора. Тогда запись `body//emph` выберет все нисходящие узлы `emph` потомков элемента `body` для текущего узла.

Для представления некоторого единичного типа элемента можно использовать «групповой» символ «`*`», так что шаблон-образец `*` выберет всех потомков для данного текущего узла, а `*/emph` — всех потомков во втором поколении («внуков») `emph` для данного текущего узла. С другой стороны, шаблон `par/*` сопоставляет любой элемент с элементом `par` как с родительским узлом.

Шаблон «`.`» выбирает данный текущий узел. Это может быть использовано, например, для явного представления рассматриваемого текущего узла в отношениях «предок-потомок», как в шаблоне `par//.`, который выбирает всех предков типа `par` для данного текущего узла, или `.//emph`, отбирающий всех потомков типа `emph` для него. Аналогичным образом, образец «`..`» выбирает предка (родительский узел) для данного текущего узла такого, что `../par` выбирает соседние элементы `par` для рассматриваемого узла.

7.6.1.5 Выражения

Внутри предикатов можно организовать некоторые виды тестов. Ниже дается перечень функций и операторов, которые можно использовать для этой цели. В них может быть нуль или более аргументов следующих типов: булев (boolean) (*b*), множество-узлов (node-set) (*ns*), число (number) (*nb*), объект (object) (*o*) или строка (string) (*s*).

- *Функции, принимающие значение «множество-функций»* : `last()`, `position()`, `id(`*o*`)`, `local-name(`*ns?*`)`, `namespace-uri(`*ns?*`)`, `name(`*ns?*`)`.

- Функции для *приведения* подвыражений (sub-expressions) к конкретным типам: `boolean(o)`, `string(o)`, `text()`, `processing-instruction(target)`, and `number(o)`.
- *Сравнения*: инфиксные операторы `=`, `!=`, `<`, `<=`, `>` и `>=`.
- *Булевы функции*: операторы `and`, `or`, функции `not(b)`, `true()`, `false()` и `lang(s)`, которые возвращают (принимают) значение true, если язык контекстного узла, который задан атрибутом `xml:lang`, тот же самый, что задается строкой-аргументом.
- *Строчные функции*: `string(o?)`, `concat(s,s,s*)`, `starts-with(s,s)`, `contains(s,s)`, `substring(s,nb,nb?)`, `substring-after(s,s)`, `substring-before(s,s)`, `string-length(s?)`, `normalize-space(s?)`, `translate(s,s,s)`.
- *Числовые операторы*: `+`, унарные и бинарные `-`, `*`, `/`, `//`, `|`, инфиксные `div`, `mod`.
- *Числовые функции*: `sum(ns)`, `floor(n)`, `ceiling(n)` и `round(n)`.

7.6.1.6 Модель данных

XPath работает с XML-документом, как с деревом, которое может содержать узлы семи типов: корень (root), элемент (element), текст (text), атрибут (attribute), пространство имен (namespace), узлы обрабатывающих инструкций (processing instruction nodes), узлы-комментарии (comment nodes).

Для каждого из типов узлов существует способ определить значение-строку для узла этого типа, которая для некоторых типов является частью рассматриваемого узла, а для ряда других вычисляется на основе значений-строк узлов-потомков.

Существует способ упорядочения, называемый *прямым порядком*, определенный на всех узлах документа, соответствующий порядку, в котором появляется первый символ (литера) XML-представления каждого узла в рассматриваемом документе после раскрытия (подстановки) общих объектов. В таком случае, корневой узел данного документа должен быть первым по порядку узлом. Узлы-элементы появляются до их потомков. Таким образом, при таком упорядочении элементы-узлы выстраиваются в порядке появления их открывающих меток в исходном XML-тексте (после подстановки объектов). Узлы-атрибуты и узлы-пространства имен некоторого элемента появляются при данном упорядочении до потомков рассматриваемого элемента. При этом узлы-пространства имен появляются раньше узлов-атрибутов. Относительный порядок (расположение) среди узлов-пространств имен и узлов-атрибутов не определен и зависит поэтому от конкретной реализации языка. *Обратным порядком* называется упорядочение узлов документа, обратное по отношению к прямому порядку.

Корневые узлы и узлы-элементы располагают упорядоченным списком имен потомков. Каждый узел, отличный от корневого, имеет в точности один родительский узел, который представляет собой или узел-элемент, или корне-

вой узел. Любой корневой узел или узел-элемент будет родительским по отношению к его узлам-потомкам. Потомки некоторого узла являются дочерними узлами для него и потомками дочерних узлов для данного узла.

7.6.2 XSLT — язык преобразований

Язык XSLT [↪XSLT] определяет синтаксис и семантику того фрагмента спецификации языка XSL, который описывает, каким образом некоторый XML-документ преобразуется в другой XML-документ. XSLT напрямую не связан с форматирующим словарем XSL, поэтому он может быть использован независимо от XSL, как это будет показано на примерах, рассматриваемых в последующих разделах.

XML-преобразование, выраженное в терминах языка XSLT, представляет собой правильно оформленный XML-документ, который согласуется с рекомендациями, относящимися к пространствам имен [↪XMLNS]. Это пространство может включать элементы, определяемые средствами XSLT (они тогда будут принадлежать пространству имен XSLT) или же какими-либо другими средствами вне данного языка.

XSLT-преобразование является стилем и состоит из набора правил для преобразования исходного дерева в результирующее дерево. Данное преобразование осуществляется путем связывания образцов с шаблонами. Образец сопоставляется с элементами в исходном дереве. Шаблон подвергаются обработке для создания фрагмента результирующего дерева. Структура результирующего дерева может быть полностью отличной от структуры исходного дерева. При формировании этого результирующего дерева элементы, взятые из исходного дерева, могут фильтроваться и переупорядочиваться, может быть добавлена произвольная структура.

Для получения фрагмента результирующего дерева производится обработка шаблона применительно к конкретному элементу из исходного дерева. Шаблон может содержать элементы, которые определяют буквальным образом фрагмент формируемой элементной структуры. Он может включать также элементы из пространства имен XSLT, которые будут инструкциями для создания фрагментов результирующего дерева. Когда производится обработка шаблона, выполняется каждая из инструкций и заменяется в итоге созданным фрагментом результирующего дерева. Инструкции могут выбирать и обрабатывать элементы-потомки из исходного дерева. Обработка элементов-потомков приводит к созданию фрагмента дерева-результата путем поиска применимого в данном случае правила-шаблона и обработки соответствующего шаблона. Отметим, что элементы обрабатываются только в том случае, если они выбраны в результате выполнения некоторой инструкции. Дерево-результат строится путем поиска соответствующего правила-шаблона для заданного корневого узла и обработки его шаблона.

В процессе поиска шаблонного правила, применимого в рассматриваемом случае, может возникнуть ситуация, когда более чем одно такое правило будет иметь образец, сопоставляющийся с данным элементом. Однако применено

в итоге будет единственное правило-шаблон, что достигается использованием процедуры разрешения конфликтов (см. разд. 7.6.2.4).

Единичный шаблон может создавать структуры произвольной сложности; он может извлекать значения-строки из произвольных мест рассматриваемого исходного дерева; он может формировать структуры, которые повторяются соответственно расположению элементов в данном исходном дереве. Для простых преобразований, где структура дерева-результата не зависит от структуры соответствующего исходного дерева, лист стиля часто содержит только единственный шаблон, который работает как некоторый шаблон для завершенного дерева-результата. Язык XSLT позволяет для такого рода стилей пользоваться упрощенным синтаксисом.

Когда производится обработка некоторого шаблона, она всегда выполняется с учетом текущего узла и текущего списка узлов. Текущий узел в этом случае всегда будет элементом рассматриваемого текущего списка узлов. Многие операции в XSLT «привязаны» к конкретному текущему узлу.

Выбор элементов для обработки, выполнение условной обработки и формирование текста в языке XSLT выполняется с помощью языка выражений XPath.

7.6.2.1 Общая структура стилей в XSL

Ресурс URI `http://www.w3.org/1999/XSL/Transform` представляет собой идентификацию пространства имен для стилей XSLT применительно к процессорам XSLT, которые должны использовать механизм пространства имен XML для распознавания элементов и атрибутов из этого пространства. Расширения (элементы и функции) должны содержаться в отдельном пространстве имен, определяемом независимо.

В XML-документе стиль представляется посредством элемента `xsl:stylesheet`. У него должен быть атрибут `version`, указывающий на версию XSLT, требуемую для работы с данным листом стиля. В настоящее время существует лишь версия 1.0.

В общем случае стиль XSL может содержать нуль или более экземпляров каждого из тринадцати видов элементов верхнего уровня (восемь элементов — пустые, строки 3–10, а пять — могут иметь информационное содержимое, строки 11–15), перечисленных ниже. Многоточие (...) показывает возможное дополнительное информационное содержимое.

```
1  <xsl:stylesheet version="1.0"
2                  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3    <xsl:import href="..."/>
4    <xsl:include href="..."/>
5    <xsl:decimal-format name="..."/>
6    <xsl:key name="..." match="..." use="..."/>
7    <xsl:namespace-alias stylesheet-prefix="..." result-prefix="..."/>
8    <xsl:output method="..."/>
9    <xsl:preserve-space elements="..."/>
10   <xsl:strip-space elements="..."/>
11   <xsl:attribute-set name="...">  ...   </xsl:attribute-set>
```

```
12    <xsl:param name="...">          ...   </xsl:param>
13    <xsl:template match="...">      ...   </xsl:template>
14    <xsl:template name="...">       ...   </xsl:template>
15    <xsl:variable name="...">       ...   </xsl:variable>
16  </xsl:stylesheet>
```

Строки 1–2 объявляют пространство имен `xsl` и номер версии XSLT. Порядок, в котором появляются элементы-потомки элемента `xsl:stylesheet`, не является существенным, за исключением элементов `xsl:import`, которые *должны* всегда задаваться первыми, в начале стиля. В следующих разделах будет более подробно объяснено, как использовать некоторые из этих элементов по мере возникновения потребности в них.

Ниже дается некоторый пример простого стиля XSL, который строит дерево-результат для некоторой последовательности элементов `par`, содержащих элементы `emph`, используя словарь форматирующих объектов XSL:

```
1   <?xml version='1.0'?>
2   <xsl:stylesheet version="1.0"
3                   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
4                   xmlns:fo="http://www.w3.org/1999/XSL/Format">
5     <xsl:template match="/">
6       <fo:basic-page-sequence font-family="Helvetica" font-size="10pt" >
7         <xsl:apply-templates/>
8       </fo:basic-page-sequence>
9     </xsl:template>
10    <xsl:template match="par">
11      <fo:block indent-start="10pt" space-before="12pt">
12        <xsl:apply-templates/>
13      </fo:block>
14    </xsl:template>
15    <xsl:template match="emph">
16      <fo:inline-sequence font-style="italic">
17        <xsl:apply-templates/>
18      </fo:inline-sequence>
19    </xsl:template>
20  </xsl:stylesheet>
```

Строка, которая следует за `xmlns:` в строках 3 и 4, вводит сокращенные обозначения для пространства имен, чтобы дать возможность программе разбора интерпретировать элементы из данного экземпляра документа (см. разд. B.3 для получения более подробных сведений). В рассматриваемом случае объявлены пространства имен для преобразования `xsl` и форматирующего объекта `fo`. Однако можно использовать также и форматирующие объекты CSS (см. документ Консорциума W3C [↪XSLCSS]); в таком случае можно было бы заменить строку 4 следующей декларацией для пространства имен:

```
xmlns:css="http://www.w3.org/TR/NOTE-XSL-and-CSS"
```

Ниже показаны три правила-шаблона. Первое из этих правил (строки 5–9) — для корневого узла рассматриваемого документа, оно определяет, что документ в целом следует форматировать как форматирующий объект «последовательность страниц», шрифтом кегля 10 pt, гарнитуры Helvetica. Второе правило (строки 10–14) объявляет, что каждый элемент `par` должен дать

результат в виде некоторого блокового форматирующего объекта, который отделяется от предыдущего блока промежутком в 12 pt и для которого первая текстовая строка набирается с отступом в 10 pt. Наконец, третье из рассматриваемых правил (строки 15–19) объявляет, что элемент **emph** отвечает форматирующему объекту **sequence**, а также то, что его содержимое должно набираться курсивным шрифтом.

Для стилей, содержащих единственный шаблон для корневого узла документа, разрешается пользоваться упрощенным синтаксисом. Например,

```
1  <html xsl:version="1.0"
2        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3        xmlns="http://www.w3.org/TR/xhtml1/strict">
4    <head>
5      <title>XSL simplified syntax</title>
6    </head>
7    <body>
8      <p>A famous quote <xsl:value-of select="FamousQuote"/></p>
9    </body>
10 </html>
```

эквивалентно следующему

```
1  <xsl:stylesheet version="1.0"
2                  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3                  xmlns="http://www.w3.org/TR/xhtml1/strict">
4  <xsl:template match="/">
5  <html>
6    <head>
7      <title>XSL simplified syntax</title>
8    </head>
9    <body>
10     <p>A famous quote <xsl:value-of select="FamousQuote"/></p>
11   </body>
12 </html>
13 </xsl:template>
14 </xsl:stylesheet>
```

Таким образом, пользователям, работающим с простыми приложениями, совсем нет нужды заботиться о шаблонах, им достаточно написать их HTML-код и извлечь информацию из исходного XML-файла.

7.6.2.2 Модель языка XSLT

Модели данных в XSLT и XPath аналогичны. XSLT трактует исходный документ, результирующий документ и стили одним и тем же образом, а два XML-документа, характеризующиеся одним и тем же деревом, обрабатываются идентично. Комментарии и обрабатывающие инструкции в листе стиля игнорируются.

Для результирующего дерева ослабляются нормальные ограничения на дочерние узлы корневого узла. Дерево-результат может включать любую последовательность узлов в качестве дочерних, которые были бы возможны для узла-элемента. В частности, при записи с использованием XML-метода **output** возможно, чтобы дерево-результат не было правильно оформленным XML-документом.

После того как построено дерево для исходного документа или стилевого документа и до того как продолжить обработку, можно удалить текстовые узлы, которые содержат только символы-пробелы. Работа с пробелами управляется инструкциями `xsl:strip-space` и элементами `xsl:preserve-space`. Оба этих средства имеют атрибут `elements`, где задается список элементов, разделенных пробелами, для которых промежутки (white space) следует удалить или сохранить.

XSLT позволяет также использовать язык выражений из XPath, чтобы отбирать узлы для обработки, задавая условия для различных способов обработки узла, формирования текста, который должен быть вставлен в дерево-результат.

XSLT-выражения появляются в качестве значения некоторых атрибутов и внутри пар фигурных скобок в шаблонах значений атрибутов.

7.6.2.3 Правила для шаблонов

Для создания фрагмента дерева-результата обрабатывается некоторый список исходных узлов. Дерево-результат строится путем обработки списка, содержащего только корневой узел. Список исходных узлов обрабатывается путем объединения структур деревьев-результатов, созданных обработкой по порядку каждого из узлов списка. Обработка узла выполняется нахождением всех шаблонных правил, в которых образцы согласуются с узлом, с последующим выбором лучшего из них — выбранный шаблон для правила затем обрабатывается применительно к данному узлу, рассматриваемому как текущий узел — и со списком исходных узлов как с текущим списков узлов. Шаблон включает обычно инструкции, которые выбирают добавочный список исходных узлов для обработки. Процесс сопоставления с образцом, обработки и выбора продолжается рекурсивно до тех пор, пока не останется необработанных новых исходных узлов.

Правило для шаблона задается с помощью элемента `xsl:template`. При этом атрибут `match` представляет собой образец, позволяющий идентифицировать исходный узел или узлы, к которым можно применить рассматриваемое правило. Атрибут `match` обязателен, за исключением использования *именованного* шаблона. Элемент `xsl:apply-templates` рекурсивно обрабатывает дочерние элементы исходного элемента.

В отсутствие атрибута `select`, инструкция `xsl:apply-templates` производит обработку всех дочерних узлов текущего узла, включая и текстовые узлы. Атрибут `select` можно использовать для обработки узлов, отобранных согласно выражению, вместо того чтобы обрабатывать все дочерние узлы. Такое выражение должно вычисляться для элемента типа node-set. Отобранное множество узлов обрабатывается в прямом порядке, если только не задано, что требуется провести сортировку.

Воспроизведем здесь строки 15–19 из первого примера в разд. 7.6.2.1:

```
15    <xsl:template match="emph">                <!-- match pattern     -->
16      <fo:inline-sequence font-style="italic"> <!-- action template + -->
17        <xsl:apply-templates/>                 <!--                   | -->
```

```
18      </fo:inline-sequence>                    <!-- action template + -->
19    </xsl:template>
```

Этот шаблон будет сопоставляться со всеми элементами типа *emph* (строка 15). Для каждого вхождения элемента `emph` в исходное дерево к дереву-результату должен быть добавлен форматирующий объект `fo:inline-sequence` (строки 16–18). Элемент `xsl:apply-templates` (строка 17) будет рекурсивно обрабатывать дочерние элементы для рассматриваемого исходного элемента.

Атрибут `select` позволяет выбрать, какой шаблон следует применить, разрешая обрабатывать элементы, не являющиеся потомками текущего узла. В показанном ниже примере предполагается, что элемент `faculty` имеет дочерние элементы `department` и элементы-потомки `professor`. Первым делом отыскивается наименование факультета (faculty) для элемента `professor`, а затем обрабатываются дочерние элементы для элементов `department` из `faculty`.

```
1   <xsl:template match="professor">
2     <fo:block>
3       <xsl:apply-templates select="name"/> belongs to the
4       <xsl:apply-templates select="ancestor::faculty/department"/> Department.
5     </fo:block>
6   </xsl:template>
```

В одном шаблоне может встретиться несколько элементов `xsl:apply-templates`, например при простом переупорядочении.

```
1   <xsl:template match="country">
2     <table><xsl:apply-templates select="capital"/></table>
3     <table><xsl:apply-templates select="currency"/></table>
4   </xsl:template>
```

В этом случае создаются две HTML-таблицы, одна с именами столиц, вторая — с наименованиями денежных единиц соответствующих стран.

Итак, видно, что когда срабатывает правило для заданного элемента, осуществляется обработка шаблона, входящего в состав этого правила. Шаблон может добавить литеральные элементы-результаты, символьные данные, а также инструкции для создания фрагментов дерева-результата (копирование, сортировка, нумерация, выполнение макрокоманд).

На практике, как это будет показано ниже в ряде примеров, шаблоны могут выводить множество форматирующих объектов или записывать код в формате HTML, LaTeX, текст Unicode.

7.6.2.4 Разрешение конфликтов между правилами для шаблонов

В случае, когда результат сопоставления шаблона положителен более чем для одного исходного узла, для разрешения возникшего конфликта используется следующий набор правил (для случая, когда применяется единственный шаблон).

- Импортированные шаблоны имеют более низкий приоритет.
- Шаблоны с более низкими приоритетами имеют и более низкий порядок предшествования; приоритеты шаблонам присваиваются с использовани-

ем атрибута `priority`, значением которого является действительное число (положительное или отрицательное).

- По умолчанию приоритеты назначаются следующим образом:
  - образцы, содержащие несколько символов-разделителей |, подобны набору правил-шаблонов, взятых по отдельности;
  - наиболее общий случай дочернего элемента или спецификатора атрибута, сопровождаемого составным именем, имеет приоритет 0;
  - дочерний элемент или спецификатор атрибута, за которым следует выбор с помощью группового символа, имеет приоритет, равный −0.25;
  - тест единичного узла имеет приоритет −0.5;
  - во всех остальных случаях назначается приоритет 0.5.

7.6.2.5 Режимы работы и шаблоны по умолчанию

Любой элемент может обрабатываться неоднократно, давая каждый раз различные результаты, с использованием необязательного атрибута `mode` с элементами `xsl:template` и `xsl:apply-templates`. Если у элемента `xsl:apply-templates` имеется атрибут режима обработки, то он применяется только к таким правилам для шаблонов из элементов `xsl:template`, которые имеют атрибут `mode` с тем же самым значением. Элемент `xsl:apply-templates` без атрибута `mode` применяется только к правилам для шаблонов из элементов `xsl:template`, у которых нет атрибута `mode`.

В случае, если для правила из стиля не получено ни одного успешного сопоставления с образцом, используется встроенное правило для шаблона, объявленное как *правило по умолчанию*, которое применяется неявно к корневому узлу, а также ко всем остальным узлам-элементам. Оно соответствует следующему определению:

```
1  <xsl:template match="*|/">
2    <xsl:apply-templates/>
3  </xsl:template>
```

Это правило по умолчанию указывает, что обработку надо продолжить применительно к дочерним элементам (элемент `xsl:apply-templates` в строке 2). Хотя этот шаблон (строка 1) сопоставляется с корневым и любым другим элементом, он имеет более низкий приоритет, чем любое другое правило для шаблона в листе стиля. Этот механизм дает пользователю удобный способ задать его собственное правило, применяемое по умолчанию, путем подстановки соответствующих XSL-команд в строку 2. При таком подходе пользовательское правило перекрывает встроенное по умолчанию.

Аналогичное встроенное правило для шаблона существует для каждого узла, который допускает рекурсивную обработку в том же самом режиме, даже при отсутствии успеха в сопоставлении с образцом для правил, явно заданных в стиле.

Встроенные правила для шаблонов для текстовых узлов и узлов-атрибутов осуществляют полное копирование текста (строки 1–3), тогда как встроенное

правило для шаблона для обрабатывающих инструкций и комментариев не делает ничего (строка 5).

```
1  <xsl:template match="text()|@*">
2    <xsl:value-of select="."/>
3  </xsl:template>
4
5  <xsl:template match="processing-instruction()|comment()"/>
```

Встроенные правила для шаблонов трактуются, как если бы они были импортированы неявно до листа стиля, и имеют по этой причине более низкий порядок предшествования, чем все другие правила для шаблонов. Таким образом, автор может перекрыть встроенное правило для шаблона путем включения явного правила.

В качестве примера рассмотрим следующий минимальный «пустой» (и тривиальный) лист стиля, который объявляет только лишь пространство имен для `xsl`.

```
1  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
2  </xsl:stylesheet>
```

Этот стиль, которому дадим имя `empty.xsl`, будет использовать по умолчанию встроенное правило для шаблона и обрабатывать все элементы рассматриваемого документа. Это похоже на DSSSL-файл с именем `empty.dsl`, который был введен в разд. 7.5.3.1.

7.6.3 Форматирующие объекты и их свойства

Форматирующие объекты применяются к узлам дерева-результата, будучи размещенными во фрагменте-образце рассматриваемого элемента. Синтаксис их в общем виде выглядит следующим образом:

```
<xsl:template match="pattern">
  <fo:formatting-object (style property="value")*>
    [processing instructions]*
  </fo:formatting-object>
</xsl:template>
```

Мы уже видели пример с таким синтаксисом в начале разд. 7.6.2.1.

Спецификация XSL определяет в той мере, в какой это возможно, модель форматирования, которая была бы совместима как с CSS, так и с DSSSL по именам для объектов, определениям, именам свойств.

Поскольку словарь форматирующих объектов пока еще нельзя считать окончательно отработанным, покажем только один пример в разд. 7.6.6. Более подробно этот вопрос можно изучить по финальной версии спецификации XSL [↪XSLSPEC], когда она станет доступна.[1]

[1] Доступна в настоящее время на сайте W3C. — *Прим. ред.*

7.6.4 XSL-процессоры и инструментальные средства

Имеется несколько программных процессоров, которые могут работать со стилевыми файлами XSL. Для обработки приводимых здесь примеров была выбрана программа `xt` Джеймса Кларка [↪XTPROC], представляющая собой реализацию на языке Java фрагментов XPath и XSLT из XSL. Существуют и другие программы грамматического разбора для XSL, в частности программы `LotusXSL` [↪LOTUSXSL] и `Saxon` [↪SAXON]. Дополнительную информацию по языку XSL, а также по другим программам для работы с ним можно найти в [↪XSLOASIS].

7.6.4.1 Использование процессора `xt`

По умолчанию программа `xt` использует программу `xp` для грамматического разбора XML-текстов ([↪XPPARS]), которая также написана Кларком на языке Java. Поэтому, если на вашем компьютере установлен Java-транслятор, достаточно загрузить архивные zip-файлы `xp` и `xt` с Web-страницы Кларка. Они содержат Java-архивы `sax.jar`, `xp.jar` и `xt.jar`. Надо добавить их содержимое в тот каталог, где Java осуществляет поиск классов. Применительно к операционной системе Windows требуемые для этой цели команды могли бы выглядеть следующим образом (в зависимости от версии доступного комплекса Java-средств):

```
1  set classpath=d:\xml\xt.jar;d:\xml\xp.jar;d:\xml\sax.jar;d:\jdk1.1.6\src;
2  %JAVA_HOME%\bin\java com.jclark.xsl.sax.Driver %1 %2 %3
```

где переменная окружения `JAVA_HOME` (строка 2) определяет каталог, в котором находится установленная на данной машине Java. Аналогично, для операционной системы UNIX (Bourne shell) требуемые команды можно было бы записать так:

```
1  DIR=/afs/cern.ch/asis/src/archive/java
2  CLASSPATH=$DIR/xt.jar:$DIR/xp.jar:$DIR/sax.jar:$CLASSPATH
3  export CLASSPATH
4  java com.jclark.xsl.sax.Driver $1 $2 $3
```

Переменная `DIR` (строка 1) описывает каталог, где содержатся архивы Java, а в строке 4 предполагается, что программа `java` находится на том пути, где система ищет исполняемые программы.

Если сохранить показанные выше строки в командном файле-сценарии `xt.bat` (Windows) или `xt` (UNIX), программу `xt` можно вызвать на исполнение путем ввода с клавиатуры команды:

```
xt XML-source-file XSL-style-sheet Output-file
```

Если третий аргумент здесь не задан, вывод программы будет записываться в «стандартный вывод» (это может быть, например, компьютерный экран). Можно использовать стиль `empty.xsl` и XML-пример `invitation2.xml` из разд. 7.4.5 путем вызова программы `xt` с помощью ввода команды:

```
xt invitation2.xml empty.xsl
```

Получим следующий вывод:

```
I would like to invite you all to celebrate
the birth of Invitation, my
first XML document child.

Please do your best to come and join me next Friday
evening. And, do not forget to bring your friends.

I really look forward to see you soon!
```

Как и ожидалось, мы видим информационное содержимое текстовых узлов для всех элементов рассматриваемого документа. Значения атрибутов (элемента `invitation`) в вывод не копируются. Средство полного копирования текстовых узлов может помочь в пошаговой разработке стиля XSL. В самом деле, для сложного документа это позволяет строить и отлаживать требуемые правила постепенно, поскольку информационное содержимое всех незаданных элементов копируется в вывод. Такой подход дает удобный контекст для разработки и отладки.

Два стиля XSL, [↪SHOWTREE] и [↪FANCYTREE], позволяют получить удобное древовидное представление некоторого XML-документа и могут послужить базисом для изучения способов более изощренного использования языка XSL.

7.6.5 Использование языка XSL для формирования документов в HTML или LaTeX'е

Чтобы продемонстрировать некоторые из возможностей XSL в области форматирования XML-документов, воспользуемся еще раз предложенными выше двумя вариантами XML-файлов для примера `invitation`. Во-первых, сформируем текст на LaTeX'е в значительной степени таким же способом, как это было проделано с помощью языка Perl, и используем тот же самый файл класса. Необходимо определить шаблоны для всех XML-элементов, которые предстоит обрабатывать, это можно сделать так, как это показано в следующем ниже стилевом XSL-файле с именем `invlat1.xsl`:

```
<?xml version='1.0'?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="text" indent="no" encoding="ISO-8859-1"/>

<xsl:strip-space elements="*"/>

<xsl:template match="invitation">
<xsl:text>\documentclass[12pt]{article}
\usepackage{invitation}
\begin{document}
</xsl:text>
<xsl:apply-templates/>
<xsl:text>\end{document}
</xsl:text>
</xsl:template>

<xsl:template match="front">
```

```
19  <xsl:text>\begin{Front}
20  \To{</xsl:text>
21  <xsl:value-of select="to"/>
22  <xsl:text>}
23  \Date{</xsl:text>
24  <xsl:value-of select="date"/>
25  <xsl:text>}
26  \Where{</xsl:text>
27  <xsl:value-of select="where"/>
28  <xsl:text>}
29  \Why{</xsl:text>
30  <xsl:value-of select="why"/>
31  <xsl:text>}
32  \end{Front}
33  </xsl:text>
34  </xsl:template>
35
36  <xsl:template match="body">
37  <xsl:text>\begin{Body}
38  </xsl:text>
39    <xsl:apply-templates/>
40  <xsl:text>
41  \end{Body}
42  </xsl:text>
43  </xsl:template>
44
45  <xsl:template match="par">
46  <xsl:text>
47  \par </xsl:text>
48  <xsl:apply-templates/>
49  </xsl:template>
50
51  <xsl:template match="emph">
52  <xsl:text>\emph{</xsl:text>
53  <xsl:apply-templates/>
54  <xsl:text>}</xsl:text>
55  </xsl:template>
56
57  <xsl:template match="back">
58  <xsl:text>\begin{Back}
59  \Signature{</xsl:text>
60  <xsl:value-of select="signature"/>
61  <xsl:text>}
62  \end{Back}
63  </xsl:text>
64  </xsl:template>
65
66  </xsl:stylesheet>
```

После объявления обычного для XSLT пространства имен (строка 2) указывается, что текстовые файлы следует выводить в кодировке ISO-8859-1 (строка 4). Отмечено также (строка 6), что все пробелы «по умолчанию» необходимо удалить. Это значит, что будут оставлены только те пробелы, которые заданы нами явным образом. Это полезно, поскольку XML и XSL по умолчанию оставляют пробелы нетронутыми, что в LaTeX'е может привести к получению странных результатов.

Первый шаблон (строки 8–16) интерпретирует элемент `invitation`. Он инициализирует LaTeX (строки 9–10), запускает рассматриваемый документ (строка 11), а затем позволяет XSL обработать все дочерние элементы (строка 13) до того, как документ будет завершен (строка 14). Команды LaTeX'а переписываются в выходной поток буквально с помощью элементов `xsl:text`.

Внутри этого элемента пробелы и символы перевода строки являются значимыми и без изменений переписываются в вывод программы.

Второй шаблон (строки 18–34) имеет дело с элементом `front` и его дочерними элементами. Вначале мы открываем окружение `Front` (строка 19), записываем команду `\To`, а затем приказываем XSL пойти и взять содержимое элемента `to` (строка 21) с помощью команды `xsl:value`. Аналогичным образом получаем информацию для команд `\Date` (строки 23–25), `\Where` (строки 26–28) и `\Why` (строки 29–31). На строки 31–33 возложена забота о закрытии окружения `Front`.

Перейдем теперь к элементу `body` в XML-файле, который обрабатывается строками 36–43. Выполняемое действие состоит в том, что записывается начальная строка окружения `Body` (строки 37–38), XSL передается указание обработать дочерние элементы для XML-элемента `body` (строка 39), после чего закрыть окружение `Body` (строки 40–42).

Элемент `par` обрабатывается с использованием образца в строках 45–49; для каждого такого элемента выдается LaTeX'овская команда `\par` (строка 47) до того, как продолжить работу с дочерними элементами для `par` (строка 48).

Содержимое элементов `emph` переносится в аргумент LaTeX'овской команды `\emph` (строки 52–54).

Наконец, мы добрались до элемента `back`, который обрабатывается в строках 57–64. В строке 58 формируется начальная строка LaTeX'овского окружения `Back`, а строки 59–61 осуществляют копирование содержимого элемента `signature` в аргумент LaTeX'овской команды `\Signature`. Строка 62 завершает окружение `Back`.

Используем XML-файл `invitation.xml`, определенный на с. 298, а также ранее введенный стиль XSL совместно с процессором `xt` XSL так, как это показано ниже:

```
xt invitation.xml invlat1.xsl invlat1.tex
```

Полученный в результате LaTeX'овский файл `invlat1.tex` показан ниже. Он полностью идентичен TeX'овскому файлу, который был показан на с. 343.

```
1  \documentclass[]{article}
2  \usepackage{invitation}
3  \begin{document}
4  \begin{Front}
5  \To{Anna, Bernard, Didier, Johanna}
6  \Date{Next Friday Evening at 8 pm}
7  \Where{The Web Cafe}
8  \Why{My first XML baby}
9  \end{Front}
10 \begin{Body}
11 \par
12 I would like to invite you all to celebrate
13 the birth of \emph{Invitation}, my
14 first XML document child.
15 \par
16 Please do your best to come and join me next Friday
17 evening. And, do not forget to bring your friends.
18 \par
19 I \emph{really} look forward to see you soon!
20 \end{Body}
21 \begin{Back}
```

```
\Signature{Michel}
\end{Back}
\end{document}
```

Используем теперь XML-файл `invitation2.xml`, описанный в разд. 7.4.5, и на этом примере покажем, как преобразовать его в формат HTML. Стиль `invhtml2.xsl` для этой цели выглядит следующим образом:

```
<?xml version='1.0'?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="html"/>
<xsl:preserve-space elements="*"/>

<xsl:template match="invitation">
<html>
<head>
<title> Invitation (XSL/CSS formatting) </title>
<link href="invit.css" rel="stylesheet" type="text/css"/>
<!-- 12 November 1998 mg -->
</head>
<body>
<h1>INVITATION</h1>
<table>
<tbody>
<tr><td class="front">To: </td>
<td><xsl:value-of select="@to"/></td></tr>
<tr><td class="front">When: </td>
<td><xsl:value-of select="@date"/></td></tr>
<tr><td class="front">Venue: </td>
<td><xsl:value-of select="@where"/></td></tr>
<tr><td class="front">Occasion: </td>
<td><xsl:value-of select="@why"/></td></tr>
</tbody>
</table>
<xsl:apply-templates/>
<p class="signature"><xsl:value-of select="@signature"/></p>
</body>
</html>
</xsl:template>

<xsl:template match="par">
<p><xsl:apply-templates/></p>
</xsl:template>

<xsl:template match="emph">
<em><xsl:apply-templates/></em>
</xsl:template>

</xsl:stylesheet>
```

Строка 4 определяет, что требуется формировать вывод в формате HTML, откуда вытекает неявное декларирование HTML в качестве пространства имен, используемого по умолчанию, так что нет необходимости задавать префикс для инструкций HTML с явным префиксом пространства имен. Строки 7–32 работают с элементом документа `invitation`. В частности, определяется заголовок для HTML-файла (строки 9–13) с установлением связи с листом стиля CSS с именем `invit.css` в строке 11. Вставляется заголовок первого уровня (строка 15), а в строке 16 начинается таблица. Это необходимо для хорошего представления различных атрибутов элемента `invitation`. Например, в строке 19 можно видеть, как организуется доступ к атрибуту некоторого элемента XML с

использованием XSL-элемента `xsl:value-of` и спецификатора атрибута `@` для его атрибута `select`. В таком случае будет использоваться значение атрибута `to`, который вводится в правую ячейку первой строки таблицы, содержащей символьную строку «To: » в левой ячейке (строка 18). Строки, идущие парами с номера 20 по номер 25, формируют строки для других атрибутов элемента `invitation`. В строках 26–27 таблица завершается до того, как обработать с помощью XSL в первый раз дочерние узлы для `invitation` (строка 28) и затем записать HTML-элемент `p` из класса `signature`, который имеет в качестве содержимого значение атрибута `signature` из стартового дескриптора `<invitation>` (строка 29). Строки 30–31 корректно завершают HTML-документ.

Среди оставшихся задач — обработка элементов `par` и `emph`. Это выполняется, соответственно, в строках 34–36 и 38–40. Сравните выполняемые действия с тем, что делалось в разд. 7.4.6.

Стиль `invhtml2.xsl` и XML-файл `invitation2.xml` передаются на обработку `xt`-процессору.

```
xt invitation2.xml invhtml2.xsl invhtml2.html
```

Получаем в итоге HTML-файл `invhtml2.html`, представленный здесь. Он эквивалентен HTML-файлу, показанному на с. 357.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title> Invitation (sgmlpl/CSS formatting) </title>
<link href="invit.css" rel="stylesheet" type="text/css">
</head>
<body>
<h1>INVITATION</h1>
<table>
<tbody>
<tr><td class="front">To: </td>
<td>Anna, Bernard, Didier, Johanna</td></tr>
<tr><td class="front">When: </td>
<td>Next Friday Evening at 8 pm</td></tr>
<tr><td class="front">Venue: </td>
<td>The Web Cafe</td></tr>
<tr><td class="front">Occasion: </td>
<td>My first XML baby</td></tr>
</tbody>
</table>
<p>I would like to invite you all to celebrate
the birth of <em>Invitation</em>, my
first XML document child.</p>
<p>Please do your best to come and join me next Friday
evening. And, do not forget to bring your friends.</p>
<p>I <em>really</em> look forward to see you soon!</p>
<p class="signature">Michel</p>
</body>
</html>
```

7.6.6 Использование XSL для создания форматирующих объектов

Как уже несколько раз отмечалось, наилучший способ формирования стиля для данного класса документов состоит в построении этого стиля с помощью

обобщенных форматирующих объектов. При таком подходе соответствующий стиль может транслироваться в различные форматы вывода. Мы уже показали, как этот подход работает в случае DSSSL, с которым в качестве обрабатывающей программы использовалась программа Jade (см. разд. 7.5.3). Аналогичным образом можно написать стиль, используя XSL и форматирующие объекты из его состава, после чего все это может быть транслировано в различные выходные представления. Однако к настоящему времени, поскольку стандарт XSL все еще не завершен, для него нет пока общеупотребительного средства вроде программы Jade. Тем не менее можно воспользоваться программой-процессором `fop`, написанной Джеймсом Таубером на языке Java, или программой PassiveTEX Себастиана Ратца [↪PASSIVETEX], которые преобразуют некоторое подмножество форматирующих объектов XSL в формат PDF.

Рассмотрим вначале стиль XSL, представленный как файл `invfo1.xsl`, который был подготовлен для использования с исходным XML-файлом примера `invitation`.

```
<?xml version='1.0'?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                              xmlns:fo="http://www.w3.org/1999/XSL/Format">

<xsl:strip-space elements="*"/>

<!-- Parameterizations -->

<xsl:variable name="PageMarginTop">75pt</xsl:variable>
<xsl:variable name="PageMarginBottom">125pt</xsl:variable>
<xsl:variable name="PageMarginLeft">80pt</xsl:variable>
<xsl:variable name="PageMarginRight">150pt</xsl:variable>
<xsl:variable name="BodyFont">Times-Roman</xsl:variable>
<xsl:variable name="BodySize">12pt</xsl:variable>
<xsl:variable name="TypeWriterFont">Computer-Modern-Typewriter</xsl:variable>
<xsl:variable name="SansFont">Helvetica</xsl:variable>
<xsl:variable name="ListRightMargin">12pt</xsl:variable>
<xsl:variable name="ListAbove">12pt</xsl:variable>
<xsl:variable name="ListBelow">12pt</xsl:variable>
<xsl:variable name="ListNormalIndent">15pt</xsl:variable>
<xsl:variable name="BulletOne">&#x2022;</xsl:variable>

<xsl:template name="listitem">
  <xsl:param name="labeltext">labeltext</xsl:param>
  <xsl:param name="itemid">itemid</xsl:param>
  <xsl:param name="itemtext">itemtext</xsl:param>
  <fo:list-item id="{$itemid}">
    <fo:list-item-label font-style="italic">
      <fo:block>
        <xsl:value-of select="$labeltext"/>
        <xsl:text>:</xsl:text>
      </fo:block>
    </fo:list-item-label>
    <fo:list-item-body>
      <fo:block><xsl:value-of select="$itemtext"/></fo:block>
    </fo:list-item-body>
  </fo:list-item>
</xsl:template>

<xsl:template match='/'>
  <fo:root>
    <fo:layout-master-set>
```

```
43          <fo:simple-page-master
44            page-master-name="allpages"
45            margin-top="{$PageMarginTop}"
46            margin-bottom="{$PageMarginBottom}"
47            margin-left="{$PageMarginLeft}"
48            margin-right="{$PageMarginRight}">
49            <fo:region-body margin-bottom="100pt"/>
50            <fo:region-after extent="25pt"/>
51          </fo:simple-page-master>
52          </fo:layout-master-set>
53          <fo:page-sequence>
54            <fo:sequence-specification>
55              <fo:sequence-specifier-repeating
56                page-master-first="allpages"
57                page-master-repeating="allpages"/>
58            </fo:sequence-specification>
59            <fo:flow font-family="serif">
60              <xsl:apply-templates/>
61            </fo:flow>
62          </fo:page-sequence>
63      </fo:root>
64    </xsl:template>
65
66    <xsl:template match="invitation/front">
67    <fo:block font-family="sans-serif"  font-size="24pt"
68              font-weight="bold"        text-align-last="centered"
69              space-after.optimum="24pt">
70      <xsl:text>INVITATION</xsl:text>
71    </fo:block>
72
73    <fo:list-block provisional-distance-between-starts="2cm"
74                   provisional-label-separation="6pt">
75      <xsl:call-template name="listitem">
76        <xsl:with-param name="labeltext">To</xsl:with-param>
77        <xsl:with-param name="itemid">listto</xsl:with-param>
78        <xsl:with-param name="itemtext"><xsl:value-of select="to"/></xsl:with-param>
79      </xsl:call-template>
80      <xsl:call-template name="listitem">
81        <xsl:with-param name="labeltext">When</xsl:with-param>
82        <xsl:with-param name="itemid">listdate</xsl:with-param>
83        <xsl:with-param name="itemtext"><xsl:value-of select="date"/></xsl:with-param>
84      </xsl:call-template>
85      <xsl:call-template name="listitem">
86        <xsl:with-param name="labeltext">Venue</xsl:with-param>
87        <xsl:with-param name="itemid">listwhere</xsl:with-param>
88        <xsl:with-param name="itemtext" select="where"/>
89      </xsl:call-template>
90      <xsl:call-template name="listitem">
91        <xsl:with-param name="labeltext">Occasion</xsl:with-param>
92        <xsl:with-param name="itemid">listwhy</xsl:with-param>
93        <xsl:with-param name="itemtext"><xsl:value-of select="why"/></xsl:with-param>
94      </xsl:call-template>
95    </fo:list-block>
96    </xsl:template>
97
98    <xsl:template match="invitation/body/par">
99      <fo:block space-before.optimum="{$BodySize}">
100       <xsl:apply-templates/>
101     </fo:block>
102   </xsl:template>
103
104   <xsl:template match="invitation/body/par/emph">
105     <fo:inline-sequence font-style="italic">
106       <xsl:apply-templates/>
107     </fo:inline-sequence>
108   </xsl:template>
109
```

```
110   <xsl:template match="invitation/back">
111   <fo:block space-before.optimum="{$BodySize}"
112             font-weight="bold" text-align-last="end">
113     <xsl:text>From: </xsl:text>
114     <xsl:value-of select="signature"/>
115   </fo:block>
116   </xsl:template>
117
118   </xsl:stylesheet>
```

Укажем вначале (строка 3), что префикс `fo` пространства имен задает адресацию для словаря форматирующих объектов XSL, которые будут использованы для построения дерева-результата. Объявим серию переменных, предназначенных для того, чтобы хранить в них значения различных глобальных параметров форматирования (строки 9-21). В частности, переменная `BodySize` (строка 14) эквивалентна `*FontSize*` в строке 8 DSSSL-листа стиля из разд. 7.5.2.2. Они управляют размером по умолчанию для шрифта, которым будет набираться документ. Для того чтобы показать, каким образом в XSL осуществляется работа с именованными шаблонами, определим в строках 23–38 именованный шаблон `listitem`. У него три параметра: `labeltext` (строка 24), который содержит текст для метки списка (используется в строке 30), `itemid` (строка 25), представляющий собой идентификатор элемента списка (используется в строке 27), и `itemtext` (строка 26), содержащий текст тела элемента (используется в строке 35). Обратите внимание на то, каким образом синтаксис `$name` используется для получения доступа к данным о фактическом значении некоторого параметра или переменной.

Макет рассматриваемого документа определяется в корневом шаблоне (строки 40–64). В частности, определяются размеры страницы (строки 45–48), используя набор переменных в начале листа стиля (строки 9–12).

Возьмем теперь заглавную часть (`front`) рассматриваемого документа (строки 66–96) и начнем ее обработку, записывая текст «INVITATION» шрифтом без засечек, кеглем 24 pt, центрируя этот текст в полосе вывода как объект-блок (строки 67–71). Оставшаяся часть начального (заглавного) фрагмента документа отображается в список (строки 73–95). У этого списка есть метка шириной 2 cm для записи в нее фиксированного текста (строка 73). Для каждого из четырех элементов внутри элемента `front` вызывается именованный шаблон `listitem`, определенный в строках 23–38. Рассмотрим подробнее один из таких вызовов. В строках 75–79 мы имеем дело с элементом `to`. Сначала значение параметра `labeltext` устанавливается равным строке «`To`», параметра-идентификатора `itemid` — равным «`listto`», а параметра `itemtext` — содержимому элемента `to` в исходном XML-документе, доступ к которому организуется с помощью синтаксиса `<xsl:value of select="to"/>`. Строки 80–94 работают с элементами `date`, `where` и `why` аналогичным образом.

Перейдем теперь к телу рассматриваемого документа, где для каждого абзаца (элемент типа `par`) формируется объект-блок (строки 99–101), пропуская промежуток, равный размеру шрифта (строка 99). Элементы `emph` (строки 104–108) преобразуются в последовательность, используя курсивный шрифт (строки 105–107).

Наконец, для завершающей части документа (строки 110–116) создается объект-блок (строки 111–115), где после установки перед ним пробела задаются использование полужирного шрифта и выравнивание текста по его правому краю (строка 112). Выводится символьная строка «From: » (строка 113), за которой следует информационное содержимое элемента типа `signature` (строка 114).

Этот стиль с именем `invfo1.xsl` и исходный XML-файл `invitation.xml` можно следующим образом откомпилировать с помощью программы `xt` :

```
xt invitation.xml invfo1.xsl invfo1.fop
```

Файл `invfo1.fop` содержит форматирующие объекты и может теперь быть использован в качестве входного для программы `fop`, написанной Джеймсом Таубером:

```
java com.jtauber.fop.FOP invfo1.fob invfo1.pdf
James Tauber's FOP 0.11.0
successfully read and parsed temp.fob
successfully wrote invfo1.pdf
```

Эта программа создает PDF-файл `invfo1.pdf`, показанный на рис. 7.11. Покажем также вывод, полученный из созданной Себастианом Ратцем программы PassiveTEX, которая транслирует форматирующие объекты XSL прямо в формат PDF, используя TEX'овскую форматирующую программу. Примечательно, что хотя в этих случаях использовались совершенно различные языки стилей и модели данных, полученные результаты оказались очень похожими на то, что изображено на рис. 7.8. Это показывает, что XSL уже является мощной технологией, которая позволяет использовать синтаксис, основанный на XML, и другие средства на всех этапах обработки документов.

7.6.7 XML, XSL и базы данных

До сих пор мы использовали XSL в основном для визуализации данных оптимальным образом. Однако можно использовать также XSL для выдачи запросов к базе данных и манипулирования данными, получаемыми из базы данных. В качестве примера был подготовлен файл базы данных, включающий список стран, их столиц, принятых в этих странах денежных единиц и т. д. В следующих разделах будет дано описание этих данных и показано, как на их основе можно формировать различные отчеты.

7.6.7.1 Формат данных

Рассматриваемые данные организованы в виде файла с именем `entable.xml`:

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <countries>
3   <country>
4    <shortname>Afghanistan</shortname>
5    <fullname>Islamic State of Afghanistan</fullname>
```

$$\text{XML} \xrightarrow{\text{XSL}} \text{FO} \xrightarrow{\text{FOP}} \text{PDF}$$

INVITATION

To: Anna, Bernard, Didier, Johanna
When: Next Friday Evening at 8 pm
Venue: The Web Cafe
Occasion: My first XML baby

I would like to invite you all to celebrate the birth of *Invitation* , my first XML document child.

Please do your best to come and join me next Friday evening. And, do not forget to bring your friends.

I *really* look forward to see you soon!

From: Michel

$$\text{XML} \xrightarrow{\text{XSL}} \text{FO} \xrightarrow{\text{PassiveTeX}} \text{PDF}$$

INVITATION

To: Anna, Bernard, Didier, Johanna
When: Next Friday Evening at 8 pm
Venue: The Web Cafe
Occasion: My first XML baby

I would like to invite you all to celebrate the birth of *Invitation*, my first XML document child.

Please do your best to come and join me next Friday evening. And, do not forget to bring your friends.

I *really* look forward to see you soon!

From: Michel

Рис. 7.11. PDF-тексты, полученные из форматирующих объектов с помощью `fop` и PassiveTeX.

```
    <isocountry>AF</isocountry>
    <capital>Kabul</capital>
    <citizen>Afghan</citizen>
    <adjective>Afghan</adjective>
    <currency>afghani</currency>
    <isocurrency>AFA</isocurrency>
    <currensubunit>pul</currensubunit>
  </country>
    ...
```

Каждый элемент `country` имеет девять потомков: `shortname`, `fullname`,...

Альтернативное представление этих данных основывается на использовании атрибутов единственного элемента `country`. Преобразуем рассматриваемую форму, используя показанные выше элементы, в форму, основанную на атрибутах, с помощью листа стиля XSL с именем `isotab1to2.xsl` и получим:

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="xml" encoding="ISO-8859-1"/>

<xsl:template match="/">
 <xsl:element name="countries"><xsl:text>&#xA;</xsl:text>
  <xsl:for-each select="countries/country">
   <xsl:element name="country">
    <xsl:attribute name="shortname">
     <xsl:value-of select="shortname"/>
    </xsl:attribute>
    <xsl:attribute name="fullname">
     <xsl:value-of select="fullname"/>
    </xsl:attribute>
    <xsl:attribute name="isocountry">
     <xsl:value-of select="isocountry"/>
    </xsl:attribute>
    <xsl:attribute name="capital">
     <xsl:value-of select="capital"/>
    </xsl:attribute>
    <xsl:attribute name="citizen">
     <xsl:value-of select="citizen"/>
    </xsl:attribute>
    <xsl:attribute name="adjective">
     <xsl:value-of select="adjective"/>
    </xsl:attribute>
    <xsl:attribute name="currency">
     <xsl:value-of select="currency"/>
    </xsl:attribute>
    <xsl:attribute name="isocurrency">
     <xsl:value-of select="isocurrency"/>
    </xsl:attribute>
    <xsl:attribute name="currensubunit">
     <xsl:value-of select="currensubunit"/>
    </xsl:attribute>
   </xsl:element><xsl:text>&#xA;</xsl:text>
  </xsl:for-each>
 </xsl:element><xsl:text>&#xA;</xsl:text>
</xsl:template>

</xsl:stylesheet>
```

Имеется только единственный шаблон, согласующийся с корневым узлом в рассматриваемом случае (строки 6–40). Выведем элемент документа `countries` (строки 7–39), который включает благодаря циклу `xsl:for-each` (строки 8–38) все элементы `country` и их атрибуты (строки 9–37). Будем брать информаци-

онное содержимое для каждого дочернего элемента, связанного с элементом `country` в исходном файле и переносить его в атрибут с тем же самым именем. Например, в строках 10–12 создается атрибут `shortname`, что делается с использованием инструкции `xsl:attribute`, после чего ему присваивается значение дочернего элемента `shortname`, используя инструкцию `xsl:value-of` в строке 11.

Начало результирующего файла `entable-alt.xml` выглядит следующим образом:

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <countries>
3   <country shortname="Afghanistan"
4            fullname="Islamic State of Afghanistan"
5            isocountry="AF"
6            capital="Kabul"
7            citizen="Afghan"
8            adjective="Afghan"
9            currency="afghani"
10           isocurrency="AFA"
11           currensubunit="pul"
12   />
13     ...
```

На самом деле можно было бы написать и более короткий XSL-стиль, чтобы проделать это преобразование с использованием инструкции `xsl:copy`, которая выполняет копирование рассматриваемого текущего узла, пространства имен, но не атрибутов и не дочерних элементов.

```
1  <?xml version='1.0' encoding="ISO-8859-1"?>
2  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3
4  <xsl:output method="xml" encoding="ISO-8859-1"/>
5
6  <xsl:template match="countries">
7   <xsl:copy>
8    <xsl:apply-templates/>
9   </xsl:copy><xsl:text>&#xa;</xsl:text>
10 </xsl:template>
11
12 <xsl:template match="country">
13  <xsl:copy>
14   <xsl:for-each select="*">
15    <xsl:attribute name="{name(.)}">
16     <xsl:value-of select="."/>
17    </xsl:attribute>
18   </xsl:for-each>
19  </xsl:copy>
20 </xsl:template>
21
22 </xsl:stylesheet>
```

Здесь первый шаблон (строки 6–10) полностью копирует содержимое рассматриваемого элемента документа. Поскольку инструкция `xsl:copy` не копирует требуемых дочерних узлов, необходим второй шаблон (строки 12–20), чтобы скопировать содержимое элемента `country`, выбирая каждый из дочерних элементов (строка 14) и создавая некий атрибут с именем рассматриваемого дочернего элемента (значение атрибута `name` помещается в `name(.)` в строке 15). Содержимое текущего дочернего узла передается в атрибут с помощью инструкции `value-of` и ее атрибута `select="."` (строка 16).

7.6.7.2 Использование базы данных

Теперь самое время посмотреть, как можно организовать доступ к данным и использование их применительно к полученной базе данных. Например, можно было бы составить список всех столиц, упорядоченный по алфавиту, вместе с наименованием соответствующей страны и ее международным ISO-кодом. Стиль XSL с именем `isotable1exa1.xsl` для решения этой задачи на основе альтернативного варианта базы данных (`entable-alt.xml` на с. 416), основанного на использовании атрибутов, выглядит следующим образом:

```
1  <?xml version='1.0' encoding="ISO-8859-1"?>
2  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3
4  <xsl:output method="text" encoding="ISO-8859-1"/>
5
6  <xsl:template match="/countries">
7   <xsl:for-each select="country">
8    <xsl:sort select="@capital"/>
9    <xsl:if test="(@capital != '-') and (@capital != ' ')">
10    <xsl:value-of select="@capital"/>
11    <xsl:text> is the capital of </xsl:text>
12    <xsl:value-of select="@shortname"/>
13    <xsl:text> (ISO-code </xsl:text>
14    <xsl:value-of select="@isocountry"/>
15    <xsl:text>)&#xA;</xsl:text><!-- new line -->
16   </xsl:if>
17  </xsl:for-each>
18 </xsl:template>
19
20 </xsl:stylesheet>
```

Организуем цикл по всем странам (`xsl:for-each` в строках 7–17) и укажем, что требуется выполнить сортировку по атрибуту `@capital` (строка 8). Некоторые страны или территории не имеют формальной столицы и помечены в базе данных с помощью знака переноса или пустого ввода для атрибута `capital`. Благодаря оператору `xsl:if` (строки 9–16) гарантируется, что будут использованы только корректные записи из базы данных, для которых записана требуемая информация (строки 11-15). Эти элементы сортируются в алфавитном порядке по именам столиц, входящих в них, так, как это показано в начальном фрагменте результирующего вывода:

```
1  Abu Dhabi is the capital of United Arab Emirates (ISO-code AE)
2  Abuja is the capital of Nigeria (ISO-code NG)
3  Accra is the capital of Ghana (ISO-code GH)
4  Adamstown is the capital of Pitcairn Islands (ISO-code PN)
5  Addis Ababa is the capital of Ethiopia (ISO-code ET)
6  Al aaiun is the capital of Western Sahara (ISO-code EH)
7  Algiers is the capital of Algeria (ISO-code DZ)
8  Alofi is the capital of Niue (ISO-code NU)
9  Amman is the capital of Jordan (ISO-code JO)
```

Существенно более сложный пример, где приходится вводить также и расширения, состоит в следующем. Отметим, что использование расширений ограничит, вероятно, мобильность полученного стиля, что может сказаться при переносе этого стиля между XSL-приложениями. Стиль использует пер-

вый вариант (основанный на дочерних элементах) базы данных по странам (entable.xml со с. 413).

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns:date="http://www.jclark.com/xt/java/java.util.Date"
     xmlns:xt="http://www.jclark.com/xt"
     extension-element-prefixes="xt">

<xsl:template match="/">
 <xsl:call-template name="newfile">
  <xsl:with-param name="abc" select="'ABCDEFGHIJKLMNOPQRSTUVWXYZ'"/>
 </xsl:call-template>
</xsl:template>

<xsl:template name="newfile">
 <xsl:param name="abc" />
 <xsl:variable name="letter" select="substring($abc,1,1)" />
 <xsl:variable name="file" select="concat($letter,'-currency.txt')"/>
 <xt:document method="text" href="{$file}" encoding="ISO-8859-1">
  <xsl:for-each select="//country[starts-with(isocurrency, $letter)]">
   <xsl:sort select="shortname"/>
   <xsl:sort select="currency"/>
   <xsl:variable name="Currency" select="normalize-space(string(currency))"/>
   <xsl:if test="($Currency != '-') and ($Currency != '')">
    <xsl:text>The </xsl:text>
    <xsl:value-of select="currency"/>
    <xsl:text> is used by </xsl:text>
    <xsl:variable name="L1" select="substring(citizen,1,1)"/>
    <xsl:choose>
     <xsl:when test="$L1 = 'A' or $L1 = 'E' or $L1 = 'I' or
                     $L1 = 'O' or $L1 = 'U'">
      <xsl:text>an </xsl:text>
     </xsl:when>
     <xsl:otherwise>
      <xsl:text>a </xsl:text>
     </xsl:otherwise>
    </xsl:choose>
    <xsl:value-of select="citizen"/>
    <xsl:text> living in </xsl:text>
    <xsl:value-of select="shortname"/>
    <xsl:text>.&#xA;</xsl:text><!-- new line -->
   </xsl:if>
  </xsl:for-each>
  <xsl:call-template name="footer"/>
 </xt:document>

 <xsl:if test="string-length($abc) != 1">
  <xsl:call-template name="newfile">
   <xsl:with-param name="abc" select="substring($abc, 2)" />
  </xsl:call-template>
 </xsl:if>
</xsl:template>

<xsl:template name="footer">
 <xsl:text>Last modification : Michel Goossens, </xsl:text>
 <xsl:choose>
  <xsl:when test="function-available('date:to-string') and
                  function-available('date:new')">
   <!-- date format : Fri Dec 31 23:59:59 PDT 1999 -->
   <!--               123456789012345678901234567 8 -->
   <xsl:variable name="datetemp" select="date:to-string(date:new())"/>
   <xsl:variable name="month" select="substring($datetemp,5,3)"/>
   <xsl:variable name="day" select="substring($datetemp,9,2)"/>
   <xsl:variable name="year" select="substring($datetemp,string-length($datetemp)-3,4)"/>
   <xsl:variable name="Date" select="concat($day,' ',$month,'. ',$year)"/>
   <xsl:value-of select="$Date"/>
  </xsl:when>
  <xsl:otherwise>
```

```
68      <xsl:text>11 Nov. 1999</xsl:text>
69     </xsl:otherwise>
70    </xsl:choose>
71   </xsl:template>
72
73   </xsl:stylesheet>
```

Так как предполагается использовать элементы расширения и функции, содержащиеся в программе `xt`, разработанной Джеймсом Кларком, первым делом надо написать декларации для них. Тогда пространство имен `date` для функции расширения объявляется в строке 4, а префикс элементов расширения `xt` и его пространство имен — в строках 5 и 6.

В данном примере вводятся также два новых метода: организация нескольких файлов вывода и использование рекурсии. Поскольку в XSLT нельзя использовать переменные для хранения (запоминания) промежуточных результатов (значение переменной в XSL нельзя изменить), во многих случаях необходимо использование рекурсивных методов.

Шаблон для корневого элемента (строки 8–12) просто вызывает именованный шаблон `newfile` (строки 9–11) и присваивает начальное значение параметру `abc`, равное строке, содержащей все 26 заглавных английских букв (строка 10). Эта строка используется потом для управления рекурсией — извлечением букв из строки по очереди.

Шаблон с именем `newfile` (строки 14-51) записывает отдельный файл для каждого набора денежных единиц (валюты), начинающихся с каждой из букв алфавита. Вначале установим текущий первый символ параметра `abc` (строка 16) и зададим значение переменной `file` как сцепление (конкатенацию) этой буквы и строки `«-currency.txt»` (строка 17). Затем откроем новый файл с именем, совпадающим со значением переменной `file`, используя элемент расширения `xt:document`, объявленный в строках 5 и 6. Организуем цикл по всем элементам `country` и выберем те из них, которые имеют ISO-аббревиатуру (код), равную рассматриваемой букве (т. е. для каждой из букв по очереди надо проделать то, что задано в строках 19–42, а рекурсия гарантирует, что будет вызван шаблон для всех букв, см. ниже). Отобранные элементы сортируются вначале по имени страны (строка 20), а затем по наименованию денежной единицы (строка 21). Поскольку в процессе сортировки надо иметь уверенность, что используются только корректные элементы из базы данных, производится исключение тех, что не удовлетворяют требованию корректности (строки 22-23). Для корректных элементов выводится требуемая информация (строки 24–40), где используется единственная новая конструкция `xsl:choose`, которая осуществляет отбор среди некоторого числа возможных альтернатив. Она состоит из последовательности элементов `xsl:when`, для которых атрибуты `test` оцениваются поочередно, а содержимое первого из них, для которого значение `xsl:when` будет равно true, обрабатывается. Если ни один из тестов не дал значения true, обрабатывается содержимое ветви `otherwise`. В нашем случае необходимо сформировать корректный неопределенный артикль (`«an»` перед гласной, `«a»` — в остальных случаях), и, следовательно, тест в строках 29–30 должен проверять, будет ли начинаться содержимое элемента `citizen` с

гласной, тогда обрабатывается строка 31, а в противном случае обрабатывается строка 34. В конце каждой страницы вызывается шаблон с именем `footer` (строка 43) и закрывается выходной документ (строка 44).

Наконец, перейдем в рекурсии на уровень глубже, вызвав еще раз тот же самый шаблон `newfile`, но удалив при этом крайнюю левую букву из значения параметра `abc` (строки 47–49). Чтобы гарантировать завершение процесса рекурсии, выполняется проверка строки-параметра в шаблоне `abc` — должно оставаться более одной буквы в этой строке (тест `if` в строке 46 перед вызовом рекурсивного шаблона).

Именованный шаблон `footer` (строки 53–71) добавляет строку в нижней части каждого документа. Он использует функцию расширения `date` из программы `xt`, и надо убедиться, что она существует в строках 56–57. Если данная функция доступна, надо получить системные данные и обработать их так, чтобы привести их к желаемой форме (строки 60–65), в противном случае надо вывести некоторую заранее заданную дату (строка 68).

Один из полученных 26 файлов — `A-currency.txt`, группирующий информацию для всех стран, код ISO для денежной единицы которых начинается с буквы «A», — приводится ниже. Обратите внимание на перемену в неопределенном артикле с «an» (строки 1–9, 14) на «a» (строки 10–13, 15–16). Последняя строка формируется именованным шаблоном `footer`, который обсуждался выше.

```
1  The afghani is used by an Afghan living in Afghanistan.
2  The lek is used by an Albanian living in Albania.
3  The readjusted kwanza is used by an Angolan living in Angola.
4  The Argentine peso is used by an Argentinian living in Argentina.
5  The dram (inv.) is used by an Armenian living in Armenia.
6  The Aruban guilder is used by an Aruban living in Aruba.
7  The Australian dollar is used by an Australian living in Australia.
8  The Austrian schilling is used by an Austrian living in Austria.
9  The Azerbaijani manat is used by an Azeri living in Azerbaijan.
10 The Australian dollar is used by a Christmas Islander living in Christmas Island.
11 The Australian dollar is used by a Cocos Islander living in Cocos (Keeling) Islands.
12 The Australian dollar is used by a Kiribatian living in Kiribati.
13 The Australian dollar is used by a Nauruan living in Nauru.
14 The Netherlands Antillean guilder is used by an Antillean living in Netherlands Antilles.
15 The Australian dollar is used by a Norfolk Islander living in Norfolk Island.
16 The Australian dollar is used by a Tuvaluan living in Tuvalu.
17 Last modification : Michel Goossens, 09 Nov. 1999
```

7.7 Заключение

Как это должно быть ясно из материала, приведенного в данной главе, в настоящее время существует ряд интересных подходов к обработке XML-файлов. Большинство из представленных подходов основывается (прямо или косвенно) на интерпретации вывода из программы разбора XML-файлов и применении некоторых правил-«действий» к различным входным элементам XML. Для реализации процесса формирования вывода в формате LaTeX или HTML используются интерпретаторы с таких языков, как Perl, Java или Python. Подчеркнем еще раз, однако, важность программирования процесса трансляции

(перевода) на высоком уровне абстракции посредством использования классов LaTeX'а или листов стиля CSS, которые могут быть использованы различными конкретными документами, даже если они сформированы согласно различным определениям DTD.

Мы рассмотрели в этой главе достаточно подробно три языка для описания стилей: CSS, DSSSL и XSL. *Каскадные стили* тесно связаны с Сетью, в частности, по этой причине они прекрасно приспособлены для отделения формы от содержания при решении задачи отображения документов на экране. С недавнего времени во второй редакции спецификации CSS были добавлены возможности в области мультимедиа и вывод документов на печать.

В настоящее время наиболее завершенная модель форматирования предлагается в рамках языка DSSSL (Document Style Semantics and Specification Language). Он реализован в виде стандарта ISO и позволяет работать со сложными макетами страниц, таблицами, математическими выражениями. Для высококачественной печати и нетривиальных документов данный язык — это единственный приемлемый вариант на сегодняшний день.[1]

Новейшее средство на рынке языков стилей — это XSL (Extensible Style Language). Предполагается предложить, в свое время, объем функциональных возможностей, по крайней мере сопоставимый с таковыми для языков CSS и DSSSL, но с синтаксисом, лучше интегрированным с расширяемыми средствами семейства «X (extensible)». Предпринимаются усилия для того, чтобы модели форматирования покрывали весь диапазон деятельности Консорциума W3C, включая языки CSS, XSL и SVG. Однако XSL-модель форматирования еще не отработана до конца, так что пока нельзя решить окончательно, оправдает ли XSL возлагаемые на него надежды как на средство, позволяющее работать с XML-документами при всех обстоятельствах.

На тех немногих страницах, которые мы могли уделить каждому из упомянутых языков описания стилей, можно было, разумеется, лишь очень бегло коснуться особенностей и возможностей каждого из них. Тем не менее, в каждом из этих случаев была рассмотрена основная идея языка, важнейшие его функциональные возможности, дано общее понятие о синтаксисе соответствующего языка. Такой подход позволяет читать и понимать файлы, написанные на каждом из этих языков, модифицировать их применительно к собственным потребностям, а в ряде случае — даже и писать новые файлы.

Информация и примеры, представленные в этой главе, дают достаточные основания для того, чтобы решить, какая из имеющихся возможностей (CSS, DSSSL и XSL) наиболее подходит для той или иной конкретной задачи. Кроме того, благодаря библиографическим ссылкам, содержащимся в тексте, можно ориентироваться и в путях развития перечисленных средств. XSL и CSS разрабатываются сейчас как полномасштабные языки описания листов стилей; первый из них — как полноценная замена для DSSSL применительно к сложным неинтерактивным документам, а второй — для мультимедийных приложений в Сети.

[1] В настоящее время существует множество развитых средств программирования стилей, базирующихся практически на всех известных языках программирования. — *Прим. ред.*

Глава 8

MathML — язык интеллектуальной разметки математических текстов

Эта глава рассчитана на тех, кто для подготовки текстов пользуется в настоящее время в качестве основного средства LaTeX'ом, но хотел бы также работать и с системами, похожими по стилю на SGML. Очевидно, что HTML не настолько богат соответствующими средствами, чтобы рассматривать его в качестве основного носителя, но, как это было видно из материала, представленного в гл. 6 и 7, язык XML для этой цели вполне подходит. Многие, однако, беспокоятся относительно того, как будет обстоять дело с уже существующими документами, особенно теми, что содержат математическую символику (из-за которой они чаще всего и выбрали в свое время TeX). Рассмотрим возможный ответ на этот вопрос — язык разметки математических текстов MathML, а также способы, с помощью которых можно преобразовать существующие LaTeX'овские тексты в XML.

Эта глава состоит из трех частей. Вначале дается краткое описание нового языка, затем рассматриваются программные средства первого поколения, поддерживающие работу с ним, после чего мы возвращаемся к более детальному рассмотрению примера практического применения системы TeX4ht для трансляции LaTeX → XML (включая MathML).

Следует подчеркнуть, что использование MathML находится пока еще на самой ранней стадии, как и разработка надежных конвертеров LaTeX → XML. В данной главе будут рассмотрены ранние версии программ и предложены некоторые подходы к их использованию, но не надо относиться к этому материалу как к уже устоявшемуся. Нам придется, соответственно, быть здесь менее точными в деталях, чем во всех остальных разделах этой книги.

8.1 Введение в MathML

В течение многих лет математике в Сети не придавалось особого значения, однако в последнее время ей стали уделять все возрастающее внимание. Был предпринят ряд разработок и экспериментов модельного характера, нацелен-

ных на выявление способов включения математической символики в SGML-средства и навигаторы. Однако лишь совсем недавно крупные организации, действующие в той или иной степени в области создания и применения средств обработки математических текстов (в частности, поставщики программ компьютерной алгебры, таких, как Maple, Mathcad, Mathematica; крупные издатели научной литературы, например Elsevier и Американское математическое общество; фирмы-разработчики программного обеспечения, в частности Adobe, а также Консорциум W3C) объединились с целью создания языка для описания структуры и содержимого математических выражений. Первым результатом их усилий стала спецификация для языка разметки математических выражений MathML. Полное описание его доступно по Web-адресу [↪MMLSPEC], а обширный список ресурсов MathML — [↪MMLRES] и [↪MMLGUID].

MathML представляет собой язык разметки математических выражений, предназначенный для использования совместно с языком XML. Среди целей, которые ставились рабочей группой W3C по математике при создании MathML, были:

- обеспечение кодирования материалов математического характера для коммуникаций всех уровней образовательного и научного типа;
- обеспечение кодирования как математической символики, так и ее значений;
- поддержка создания шаблонов и других приемов математического редактирования;
- обеспечение преобразования в другие математические форматы как чисто презентационного, так и семантического характера, а также — из этих форматов в создаваемый язык математической разметки. Форматы вывода должны включать средства для отображения графической информации, синтеза речи, представления текста в форме, пригодной для ввода ее в системы компьютерной алгебры, совместимость с другими языками описания математических текстов, такими, как TEX, возможность отображения «чистого» текста (т. е. не включающего математических символов и выражений), возможность вывода текстов на печать в различной форме, включая вывод по системе Брайля. При этом преобразования между различными форматами могут приводить к потерям информации;
- возможность передачи информации с учетом особенностей конкретных программ визуализации;
- поддержка эффективных процессов просмотра длинных математических выражений;
- обеспечение расширяемости возможностей (способами, которые заранее не известны).

Первоначально ставилась также цель сделать MathML простым для изучения и для редактирования математических выражений *вручную*, однако она отошла на второй план после того, как рабочей группой было принято решение

использовать в качестве основы язык XML, а не полный вариант SGML (работа над проектом была начата еще до появления XML).

MathML реализует две «точки зрения» на математическую разметку: один из ее видов — это *презентационная разметка*, второй — *семантическая разметка*. Презентационная разметка описывает математическую символику с выражениями, которые строятся с использованием некоторых схем вывода, с заданием способов размещения подвыражений, таких, как дроби, верхние и нижние индексы. Семантическая разметка описывает математические объекты и функции, где для каждого узла конструируется дерево выражения согласно некоторой конкретной схеме, а ветви этого дерева отвечают подвыражениям.

Чтобы проиллюстрировать эти два подхода, рассмотрим следующую простую формулу, для которой осуществим разметку согласно упомянутым выше двум схемам:

$$x^2 - 6x + 9 = 0$$

LaTeX'овский код, позволяющий получить эту формулу, имеет такой вид:

```
\[ x^{2} - 6x + 9 = 0\]
```

Первый способ использования MathML состоит в применении *элементов презентации*, которые описывают *визуальную форму представления* математической формулы.

```
1  <mrow>
2    <mrow>
3      <msup>
4        <mi>x</mi>
5        <mn>2</mn>
6      </msup>
7      <mo>-</mo>
8      <mrow>
9        <mn>6</mn>
10       <mo>&InvisibleTimes;</mo>
11       <mi>x</mi>
12     </mrow>
13     <mo>+</mo>
14     <mn>9</mn>
15   </mrow>
16   <mo>=</mo>
17   <mn>0</mn>
18 </mrow>
```

Здесь можно видеть два вида элементов MathML. К первому относятся те, что содержат данные, в частности `<mi>` для идентификаторов (строки 4 и 11), `<mn>` для чисел (строки 5, 9, 14 и 17) и `<mo>`, для операторов (строки 7, 10, 13 и 16). Ко второму виду относятся элементы, содержащие только другие вложенные элементы MathML, примеры их — это `<msup>` (строки 3–6) и `<mrow>` (строки 1–8, 2–15 и 8–12). Вложенные элементы `mrow` обозначают термы — в рассматриваемом случае это часть уравнения, расположенная слева от знака равенства. Задавая тип рассматриваемых данных и размечая термы, можно управлять размещением пробелов для лучшего визуального представления материала, представлением материалов звукового характера, управлением

разрывами строк (переносами), а также автоматической обработкой внешними приложениями. Отметим, что в противоположность LaTeX'у элемент `<msup>` состоит из двух частей: собственно верхний индекс и материал, к которому этот индекс присоединяется — `<msup><mi>x</mi><mn>2</mn></msup>`. Сравните этот способ с LaTeX'овским `x^{2}`, где нет никакой разметки *до* переменной x.

Второй способ использования MathML — выразить *семантическое содержимое* того же самого выражения следующим образом:

```
<reln>
  <eq/>
  <apply>
    <plus/>
    <apply>
      <minus/>
      <apply>
        <power/>
        <ci>x</ci>
        <cn>2</cn>
      </apply>
      <apply>
        <times/>
        <cn>6</cn>
        <ci>x</ci>
      </apply>
    </apply>
    <cn>9</cn>
  </apply>
  <cn>0</cn>
</reln>
```

Элементы содержимого MathML обычно находятся внутри некоторого элемента `<apply>`, который отмечает семантически значимое (осмысленное) выражение (строки 3–19, 5–17, 7–11 и 12–16), и действуют подобно паре скобок. Для построения выражений используется префиксная нотация (элементы `<plus/>`, `<times/>` и `<minus/>`). Для обозначения чисел (`<cn>`) и символов (`<ci>`) используются лексемы.

В состоянии ли MathML удовлетворять поставленным перед ним целям? Некоторые из математиков-практиков считают, что презентационная разметка MathML вполне пригодна для работы, несмотря на чрезвычайное многословие, присущее ей в сравнении с TeX'ом. Наибольшие изменения в методах работы и соответствующем программном обеспечении необходимы для поддержания различий, обеспечивающих удобство авторской работы наряду с эффективностью архивного хранения размеченных документов и их воспроизведения в требуемой форме. Пользователи LaTeX'а приучены писать что-то вроде

```
\newcommand{\Wotsit}{\ensuremath{E_{\beta}^{\infty}}}
\Wotsit{} означает все что угодно
```

Здесь объединены более высокий уровень абстрактной разметки, чем в стандартном LaTeX'е, сокращенные обозначения для ускорения процесса ввода данных, а также детали процесса воспроизведения информации. В среде XML эти операции пришлось бы делать в два этапа — на первом добавить новый элемент

<Wotsit> к используемому Определению типа документа (DTD), а на втором — добавить правило к используемому листу стиля, который отображает данный элемент на

```
<math>
  <msubsup>
    <mi>E</mi>
    <mi>&beta;</mi>
    <mo>&infty;</mo>
  </msubsup>
</math>
```

По сравнению с обычным LaTeX'ом это выглядит очень неудобным, однако в таком подходе есть и ряд преимуществ: лучше условия для синтаксического контроля, возможность контекстно-зависимого редактирования, поиск и возможность использования различных программ визуального воспроизведения информации. Такой подход мог бы заменить в некоторых системах TeX'овскую разметку. Естественно, можно продолжать использовать TeX'овские обрабатывающие средства, шрифты семейства Computer Modern и т. п. для подготовки печатных публикаций. Альтернативный сценарий состоит в том, чтобы LaTeX'у «следующего поколения» сделать доступным синтаксис, намного более близкий к тому, что заложен в MathML, и воспользоваться, соответственно, заложенными в нем возможностями и преимуществами.

Совсем по-иному обстоит дело с семантической математической разметкой. Для фрагментов математического текста такую разметку можно рассматривать в качестве среды обмена. Уровень ее возможностей некоторые квалифицируют как «до уровня университета», хотя другие, не столь благожелательно настроенные, — «до начала XX века»! Пока не очень понятно, будут ли ограничения, присущие MathML, слишком обременительными для математики исследовательского уровня. Когда мы имеем дело с семантической математической разметкой, едва ли будет допустимым использование набора специализированных языков высокого уровня, каждый из которых предназначен для какой-либо частной области математики, с тем чтобы затем объединить и расширить их до заданного уровня семантической разметки MathML, поскольку суть разметки на уровне информационного содержимого документа состоит в том, чтобы дать возможность взаимного обмена между компьютерными программами. Обмен информационным содержимым должен происходить на наивысшем значимом уровне.

8.1.1 MathML, Unicode и XML-объекты

MathML представляет собой приложение XML, так что все используемые символы должны принадлежать только набору Unicode (см. приложение C.2). Следствием этого будут такие две проблемы:

1. Unicode (версия 2) не содержит в своем составе того громадного многообразия специальных символов, что за многие годы создано математиками.

2. При чтении MathML (не говоря уже о записи его вручную) большинство авторов не смогут использовать программное обеспечение, полностью ориентированное на Unicode, или же будут применять кодировку UTF-8 (приложение С.2.3). По этой причине они вместо математических символов могут увидеть нечто неудобочитаемое.

За решение первой из этих двух проблем взялась группа, состоящая из издателей научной литературы и поставщиков программного обеспечения (STIPUB), которая начала реализацию соответствующего проекта в 1997 г. Этой группой было подготовлено исчерпывающее собрание всех символов, используемых в настоящее время при издании научной литературы, а также при создании и сопровождении программного обеспечения. Те из них, что отсутствуют в текущем варианте Unicode, в конце 1998 г. были представлены на рассмотрение Консорциуму Unicode для включения их в следующую версию символьного набора Unicode. До того момента, когда предлагаемые дополнения к Unicode будут одобрены, практическое использование MathML должно основываться на так называемом частном фрагменте Unicode (Unicode private zone) (см. ниже). Даже если определенные символы *присутствуют* в Unicode, все равно остается проблема доступности соответствующих шрифтов. Чтобы исправить это положение, та же самая группа издателей и поставщиков программного обеспечения планирует подготовить полностью обновленный набор математических шрифтов и сделать этот набор свободно доступным.

Вторую проблему можно решить, если использовать ссылки на объекты в определении DTD, т. е. применяя для литер *символические имена* вместо прямого употребления символов Unicode. Тогда ϕ будет записываться и читаться как `&phi;`. При этом, однако, также возникает ряд проблем. Во-первых, объекты, объявляемые в XML, *должны быть* выражены в конечном счете через символы в схеме Unicode. Если требуемый символ в текущей версии Unicode отсутствует, необходимо выбрать некоторую произвольную позицию в «частном фрагменте» Unicode, и все программное обеспечение MathML должно поддерживать это соглашение. Есть вполне очевидная опасность, что какое-то другое приложение воспользуется той же самой позицией в данном частном фрагменте и при этом не сможет сосуществовать с MathML. Во-вторых, программное обеспечение XML *не обязано* читать DTD при обработке документов MathML, поэтому потребуется определять XML-объекты в *документе*, а не в DTD (что весьма неудобно), или же потребуется препроцессорная обработка документов, чтобы удалить ссылки на объекты.

Ситуация еще более запутывается из-за того, что некоторые программы для работы с MathML (особенно `techexplorer` и WebEQ) не обрабатывают документы с помощью обычной программы грамматического разбора XML, а обычно распознают имена объектов из предписанного заранее набора, после чего воспроизводят их в неизменном виде.

Практическое решение этой проблемы, одобренное в рекомендациях MathML (MathML Recommendation), состоит в публикации списка предложенных имен объектов. Это позволяет определить соответствующую позицию

Unicode, если она существует, в противном случае — предлагается позиция в частном фрагменте. Списки объектов представляют собой расширения списков из стандарта ISO 9573 и делятся на следующие группы:

Математические символы	
ISOAMSA	Математические символы: Стрелки
ISOAMSB	Математические символы: Бинарные операторы
ISOAMSC	Математические символы: Ограничители
ISOAMSN	Математические символы: Отношения с отрицанием
ISOAMSO	Математические символы: Порядки
ISOAMSR	Математические символы: Отношения
MMALIAS	Псевдонимы MathML
MMEXTRA	Дополнения к MathML
Служебные символы	
ISOTECH	Служебные символы общего назначения
ISOPUB	Издательские спецзнаки
ISODIA	Диакритические знаки
ISONUM	Числа и специальные графические символы
ISOBOX	Символы для рисования рамок и линий
Дополнительные шрифты	
ISOGRK3	Греческие символы
ISOMSCR	Математический рукописный шрифт
ISOMOPF	Математический ажурный шрифт
ISOMFRK	Математический готический шрифт
ISOGRK1	Греческие буквы
ISOGRK2	Греческий шрифт Monotoniko
ISOGRK4	Альтернативные греческие символы
ISOCYR1	Русские кириллические буквы
ISOCYR2	Нерусские кириллические буквы

Решение проблемы создания набора специальных математических символов и разработки широко доступных шрифтов для отображения этих символов представляет собой одну из ключевых проблем для нарождающегося MathML-сообщества, поэтому в данной области можно ожидать быстрого прогресса.

8.2 Программное обеспечение для MathML

Планы по реализации рекомендаций MathML были объявлены рядом поставщиков программного обеспечения и других организаций. К настоящему времени некоторые из средств, описываемых в данной книге, могут интерпретировать, воспроизводить или выполнять разметку MathML, пользуясь LaTeX'овским файлом в качестве входного. Аналогичным образом, определенный уровень поддержки данного языка уже начинает внедряться в популярные навигаторы.

Программное обеспечение для MathML можно разделить на следующие пять категорий:

1. Пакеты программ для работы с символьными математическими выражениями.
2. Редакторы математических выражений, позволяющие либо осуществлять экспорт в формате MathML, либо использовать его в качестве основного формата для хранения этих выражений.
3. Web-навигаторы, а также динамически подключаемые модули и справочные системы для них.
4. Автономные программы для преобразования текстов из форматов наподобие LaTeX.
5. Системы подготовки текстов, которые могут обрабатывать XML-документы, включающие фрагменты в формате MathML.

Сколько-нибудь подробное рассмотрение программ, относящихся к первой из перечисленных категорий, выходит за рамки данной книги; планируется, что программы Mathematica [↪MATHEMATICA] и Maple [↪MAPLE] получат средства экспорта и импорта MathML с целью обеспечения возможности информационного обмена с другими программами. По-видимому, именно с этих программ начнется работа с *семантической* частью рассматриваемого языка. Несмотря на то что такое использование языка MathML представляется очень важным, практические реализации его пока отсутствуют.

Оставшиеся четыре категории программных средств являются предметом рассмотрения разделов, следующих ниже.

8.2.1 Редакторы математических выражений

В настоящий момент лучший пример диалогового редактора математических выражений, позволяющего работать с MathML, — это коммерческая программа `MathType` (здесь рассматривается ее версия 4; см. [↪MATHTYPE]), представляющая собой полный вариант редактора формул, используемого в программе Microsoft Word. Кратко говоря, экспорт в MathML осуществляется в форме *представления* до тех пор, пока не будет накоплен опыт взаимодействия, необходимый для записи математических выражений в их семантической форме.

Программа `MathType` разработана для того, чтобы ее можно было использовать в качестве «ассистента» для других программ с платформ Windows или Macintosh (таких, как Word), поэтому она не имеет связанного с ней непосредственно типа файла. Вместо этого она помещает компактное представление математического выражения в редактируемой форме в качестве комментария в одном из нескольких ее выводных форматов. Имеется возможность выбрать один из нескольких трансляторов, которые определяют, что за формат был использован, когда фрагмент текста передавался в область сохранения Windows или Mac, при выполнении команды копирования фрагмента («copy»). В чи-

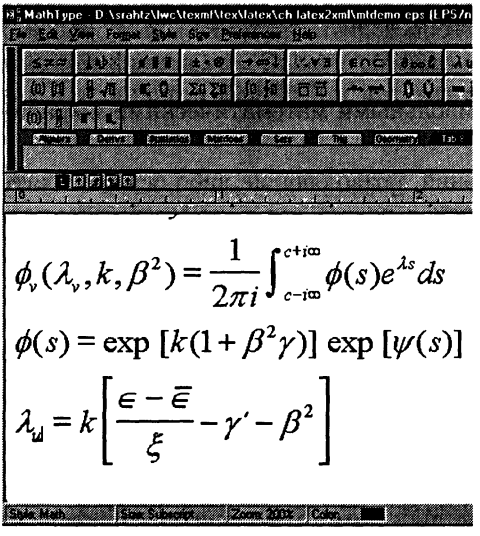


Рис. 8.1. Редактор математических выражений MathType.

$$f(\in, \delta s) = \frac{1}{\xi}\phi_v(\lambda_v, k, \beta^2)$$

$$\phi_v(\lambda_v, k, \beta^2) = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} \phi(s)e^{\lambda s} ds$$

$$\phi(s) = \exp[\kappa(1+\beta^2\gamma)] \exp[\psi(s)]$$

$$\lambda_u = k\left[\frac{\in - \bar{\in}}{\xi} - \gamma' - \beta^2\right]$$

Рис. 8.2. Пример выражений, отпечатанных TEX'ом.

сло этих форматов входят MathML, а также различные варианты TEX'овских форматов (т. е. plain TEX, LATEX и $\mathcal{AMS}$-LATEX); эти трансляторы можно настраивать, а передаваемый фрагмент достаточно просто редактируется. Например, файл правил для LATEX'а содержит следующий текст:

```
paren  = "\left( #1 \right)";                 // parentheses (both)
brack  = "\left[ #1 \right]";                 // brackets (both)
brace  = "\left\{ #1 \right\}";               // braces (both)
abrack = "\left\langle #1 \right\rangle ";    // angle brackets (both)
bar    = "\left| #1 \right|";                 // bars (both)
dbar   = "\left\| #1 \right\|";               // double bars (both)
floor  = "\left\lfloor #1 \right\rfloor ";    // floor
```

из которого видно, что внутреннее представление отображения в MathType (например, brack) в точности соответствует TEX'овскому.

Работа с данным редактором иллюстрируется рис. 8.1, где показан ввод нескольких уравнений из примера текста, использовавшегося в гл. 1. Если задано преобразование в формат $\mathcal{AMS}$-LATEX, получаемый вывод выглядит как фрагмент исходного TEX'овского текста, приводимый ниже. Результат печати этого текста с помощью TEX'а показан на рис. 8.2.

```
\[
\begin{gathered}
\phi _{v} (\lambda _{v} ,k,\beta ^{2} ) =
 \frac{1}{{2\pi i}}\int_{{c - i\infty }}^{{c + i\infty }}
```

```
 {\phi (s)e^{{\lambda s}} ds}  \hfill \\
\phi (s) = \text{exp }[\kappa(1 + \beta ^{2} \gamma )]
 \text{ exp }[\psi (s)] \hfill \\
\lambda _{u}  = k\left[ {\frac{{ \in  - \bar  \in }}
 {\xi } - \gamma ' - \beta ^{2} } \right] \hfill \\
\end{gathered}
\]
```

Причина, по которой правая и левая скобки в первой строке не распространяются на β^2, состоит в том, что использованы обычные символы «(» и «)», а не конструкция типа «bracketed object» из MathType.

Если задать преобразование в MathML, то получим следующий результат (третье уравнение из числа показанных выше опущено для экономии места):

```
<math displaystyle='true'>
 <semantics>
  <mtable columnalign='left'>
   <mtr>
   <mtd>
    <msub>
     <mi>&phi;</mi>
     <mi>v</mi>
    </msub>
    <mo stretchy='false'>(</mo><msub>
     <mi>&lambda;</mi>
     <mi>v</mi>
    </msub>
    <mo>,</mo><mi>k</mi><mo>,</mo><msup>
     <mi>&beta;</mi>
     <mn>2</mn>
    </msup>
    <mo stretchy='false'>)</mo><mo>=</mo><mfrac>
     <mn>1</mn>
     <mrow>
      <mn>2</mn><mi>&pi;</mi><mi>i</mi>
     </mrow>
    </mfrac>
    <msubsup>
     <mo>&int;</mo>
     <mrow>
      <mi>c</mi><mo>-</mo><mi>i</mi><mo>&infin;</mo>
     </mrow>
     <mrow>
      <mi>c</mi><mo>+</mo><mi>i</mi><mo>&infin;</mo>
     </mrow>
    </msubsup>
    <mrow>
     <mi>&phi;</mi><mo stretchy='false'>(</mo>
     <mi>s</mi><mo stretchy='false'>)</mo>
```

```
      <msup>
       <mi>e</mi>
       <mrow>
       <mi>&lambda;</mi><mi>s</mi>
       </mrow>
      </msup>
      <mi>d</mi><mi>s</mi>
     </mrow>
    </mtd>
    </mtr>
    <mtr>
    <mtd>
     <mi>&phi;</mi><mo stretchy='false'>(</mo>
     <mi>s</mi><mo stretchy='false'>)</mo>
     <mo>=</mo><mtext>exp </mtext>
     <mo stretchy='false'>[</mo><mi>k</mi>
     <mo stretchy='false'>(</mo><mn>1</mn><mo>+</mo><msup>
      <mi>&beta;</mi>
      <mn>2</mn>
     </msup>
     <mi>&gamma;</mi>
     <mo stretchy='false'>)</mo>
     <mo stretchy='false'>]</mo><mtext> exp </mtext>
     <mo stretchy='false'>[</mo><mi>&psi;</mi>
     <mo stretchy='false'>(</mo><mi>s</mi>
     <mo stretchy='false'>)</mo><mo stretchy='false'>]</mo>
    </mtd>
    </mtr>
   </mtable>
  </semantics>
 </math>
```

Интересно сравнить эти правила преобразования с теми, что используются для LaTeX'a; если взять те же самые конструкции, что и раньше, правила MathML выглядят следующим образом:

```
paren = "<mrow><mo>(</mo>$+$n#1$-$n<mo>)</mo></mrow>$n";
brack = "<mrow><mo>[</mo> #1 <mo>]</mo></mrow>";
brace = "<mrow><mo>{</mo> #1 <mo>}</mo></mrow>";
abrack = "<mrow><mo>&langle;</mo> #1 <mo>&rangle;</mo></mrow>";
bar = "<mrow><mo>|</mo> #1 <mo>|</mo></mrow>";
floor = "<mrow><mo>&lfloor;</mo> #1 <mo>&rfloor;</mo></mrow>";
```

Экспериментальный Web-навигатор Amaya, созданный Консорциумом W3C, также может быть использован для порождения MathML-текстов. Хотя его нельзя рассматривать как серьезное средство формирования таких текстов, он показывает, что реализовать простые процедуры подготовки, редактирования и отображения совсем не сложно.

8.2.2 Поддержка MathML в Web-навигаторах

Для отображения текстов с разметкой MathML можно использовать программу-навигатор Amaya — простым включением элементов `<math>` (и их потомков) в обычный HTML-текст. Это показано на рис. 8.3 (и на рис. 6.5 на с. 321). Поддержка MathML здесь осуществляется не в полном объеме, причем далеко не последней из нерешенных проблем является отсутствие подходящих символьных шрифтов (см. разд. 8.1.1), но, тем не менее, данная программа полезна потому, что дает возможность предварительного знакомства с рассматриваемой технологией.

Динамически подключаемый модуль для навигатора `techexplorer` и вспомогательная программа WebEQ для языка Java (описаны в гл. 5) также позволяют отображать MathML-тексты, включенные в качестве фрагментов в HTML. Требуемая разметка несколько отличается от той, что принята для элемента `<embed>`, и использует элемент `<applet>`. Тогда, если взять LaTeX'овский фрагмент из нашего тестового файла

```
The Vavilov parameters are simply related to the Landau parameter by
$\lambda_L = \lambda_v/\kappa - \ln\kappa $. It can be shown that as
$\kappa \rightarrow 0$, the distribution of the variable $\lambda_L$
approaches that of Landau.
```

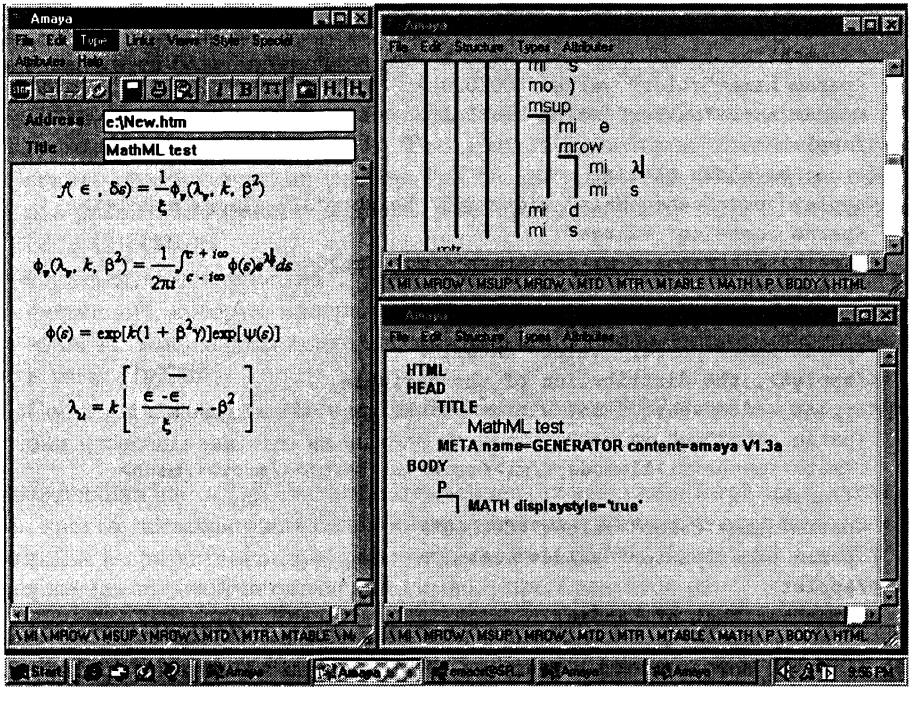


Рис. 8.3. Работа с MathML с помощью Web-навигатора Amaya.

и преобразовать его в смесь элементов HTML и MathML для воспроизведения с помощью программы `techexplorer`, соответствующий текст HTML будет выглядеть так:

```
The Vavilov parameters are simply related to the Landau parameter by
<embed src="htmlmathml71.mml" width="215" height="46" align="middle">.
It can be shown that as
<embed src="htmlmathml72.mml" width="89" height="37" align="middle">,
the distribution of the variable
<embed src="htmlmathml73.mml" width="50" height="42" align="middle">
approaches that of Landau.
```

Каждый математический фрагмент содержится в отдельном файле, ширина и высота данного фрагмента при его воспроизведении должны быть заданы явным образом. Эти вопросы мы уже рассматривали в разд. 5.3.1 на с. 275.

Если надо загрузить Java-код программы WebEQ для обработки математических фрагментов, вместо этого надо написать следующее:

```
The Vavilov parameters are simply related to the Landau parameter by
<applet code="webeq.Main" width="215" height="46" align="middle">
 <param name="eq" value="
  <math><msub><mi>&lambda;</mi><mrow><mi>L</mi></mrow></msub> <mo>=</mo>
   <msub><mi>&lambda;</mi><mrow><mi>v</mi></mrow></msub>
   <mo>/</mo><mi>&kappa;</mi>
   <mo>-</mo> <mi>l</mi><mi>n</mi> <mi>&kappa;</mi>
  </math>">
 <param name="color" value="#C0C0C0">
 <param name="parser" value="mathml">
</applet>.
It can be shown that as
<applet code="webeq.Main" width="89" height="37" align="middle">
 <param name="eq" value="
  <math><mi>&kappa;</mi><mo>&rarr;</mo> <mn>0</mn>
  </math>">
 <param name="color" value="#C0C0C0">
 <param name="parser" value="mathml">
</applet>, the distribution of the variable
<applet code="webeq.Main" width="50" height="42" align="middle">
 <param name="eq" value="
  <math><msub><mi>&lambda;</mi><mrow><mi>L</mi></mrow></msub>
  </math> ">
 <param name="color" value="#C0C0C0">
 <param name="parser" value="mathml">
</applet>
approaches that of Landau.
```

Код MathML встраивается в текст на HTML внутри элементов `<applet>`, однако и в этом случае необходимо явное задание ширины и высоты отображаемого математического фрагмента. Это не слишком приятная черта про-

грамм преобразования[1], и остается только надеяться, что будущие системы будут уметь выполнять передачу требуемых размеров между навигаторами, а также между специализированными справочными системами и динамически подключаемыми модулями.

Представляется маловероятным, что текущее поколение основных Web-навигаторов (Netscape, Internet Explorer и т. д.) будет осуществлять прямую поддержку MathML раньше, чем в 2000 г.[2] До этого момента придется выполнять эти операции с помощью динамически подключаемых программных модулей.

Один из возможных способов поддержки MathML будущими навигаторами состоит в преобразовании к стандартизованному языку векторной графической разметки (эквиваленту языка PostScript) для Сети. Более подробную информацию о разработке этого языка можно найти по адресу [↪W3CGR].

8.2.3 Преобразование LaTeX'а в MathML

Преобразования из LaTeX'овского формата скорее всего в ближайшей и среднесрочной перспективе будут *презентационными* просто по той причине, что ни LaTeX, ни plain TeX не создавались для описания семантики математических выражений. Даже реализация презентационной разметки представляет собой нетривиальную задачу.

Программы-конвертеры из формата LaTeX в формат XML в общем случае могут основываться на одной из следующих базовых идей (см. (Rahtz:1995), где содержится более подробное обсуждение этих вопросов):

1. Автономные программы, написанные на одном из обычных языков программирования наподобие C, Perl или Java, которые пытаются осуществить грамматический разбор TeX'овской разметки. Некоторые примеры реализации такого подхода уже приводились в гл. 1 (TtH) и гл. 3 (LaTeX2HTML).
2. Системы, основанные на TeX'овских макро, добавляющие в DVI-файл информацию, которую программа, использующая этот текст позднее, может извлечь. Такого рода подход достаточно подробно был продемонстрирован в гл. 4 (TeX4ht).
3. Системы, основанные на некотором расширении TeX'а, причем структурно они устроены так, что на выходе дают текст в формате XML.

Значительная часть TeX'овских математических выражений транслируется совсем просто, поскольку для большинства конструкций такого рода имеется отображение в презентационную разметку языка MathML. Тогда дробь в LaTeX'е, содержащая простой нелатинский символ, преобразуется из

```
\frac{\xi}{E_{x}}
```

[1] Эти преобразования были созданы с использованием экспериментальных конфигурационных файлов для TeX4ht.

[2] До сих пор этого не случилось. — *Прим. ред.*.

следующее:

```
<mfrac>
 <mrow><mi>&xi;</mi></mrow>
 <mrow>
   <msub>
    <mi>E</mi>
    <mrow><mi>x</mi></mrow>
   </msub>
 </mrow>
</mfrac>
```

К сожалению, здесь приходится преодолевать два затруднения, причем методы наилучшего преодоления для каждого из них различаются. Первое состоит, очевидно, в том, что авторы, использующие LaTeX, очень часто при написании нетривиальных математических конструкций вводят макроопределения, чтобы реализовать частные схемы разметки, что-то наподобие такого

```
\newcommand{\htheta}{\hat\theta}
```

Программы вроде TtH и LaTeX2HTML справляются с этим без особых затруднений. Рассмотрим, однако, макроопределение в формате plain TeX, в котором содержатся условные выражения, обрабатываемые при раскрытии данного макроопределения, например

```
\def\Fraction:#1/#2:{\ifmmode\frac{#1}{#2}\else #1 divided by #2\fi}
```

Если использовать этот текст с чем-нибудь вроде `\Fraction:36/72:`, можно почти наверняка «сломать» любой из конвертеров, если только они не используют программу грамматического разбора TeX. Конечно, большинство LaTeX'овских пользователей писать такие макроопределения не будут; LaTeX'овская команда `\newcommand` их не поддерживает, но подготовленные пользователи TeX'а не смогут воспользоваться всей его мощью.

Вторая из упомянутых выше проблем возникает тогда, когда в LaTeX'е приходится иметь дело с выражениями типа

```
\[ (a+b)^{2} \]
```

Это должно быть преобразовано в такой MathML-код:

```
<mrow>
 <msup>
  <mrow><mo>(</mo><mi>a</mi><mo>+</mo><mi>b</mi><mo>)</mo></mrow>
  <mrow><mn>2</mn></mrow>
 </msup>
</mrow>
```

Как уже отмечалось, выражения с верхними индексами в MathML надо размечать явным образом. В LaTeX'е подходы, основанные на простом переопределении макро, приводят к возникновению трудностей, поскольку макропрограм-

мист не имеет непосредственного доступа к началу выражения (a+b). Программист, пишущий на языке C или Perl, может достаточно просто организовать возвраты (backtrack) и найти большинство вероятных выражений для включения их внутрь MathML-конструкции <msup>. Хотя конструкций такого рода не так и много, появляются они довольно часто, и транслятор должен уметь находить отвечающее им подходящее решение. Один из возможных вариантов для пользователей LaTeX'а состоит в том, чтобы применять расширенный синтаксис ввода, позволяющий исключить двусмысленности; в таком случае рассмотренный выше пример можно было разметить следующим образом

```
\[ \Sup{(a+b)}{2} \]
```

после чего решение задачи трансляции в MathML становится тривиальным. Такой подход используется в программе WebEQ (см. ниже) Альтернативный подход состоит в том, чтобы переписать всю ту часть LaTeX'а, которая отвечает за работу с математическими конструкциями на нижнем уровне и идентифицировать требуемые подвыражения непосредственно. В настоящее время реализации этой идеи отсутствуют.

В момент написания данной книги было три предварительных реализации конвертеров из LaTeX'а в XML/MathML (они будут описаны ниже), тогда как в `techexplorer` эти возможности в какой-то степени уже присутствуют. Как будет показано в следующих разделах, имеется уже требуемая технология для трансляции из LaTeX'а в MathML, однако ее еще нельзя рекомендовать как надежное, проверенное средство, пригодное для обработки произвольных LaTeX'овских документов.

8.2.3.1 Преобразование LaTeX'а в MathML с помощью WebEQ

Первый из конвертеров был написан как часть программного комплекса WebEQ. Эта программа-ассистент (wizard) показана в действии на рис. 8.4; она обеспечивает работу в диалоговом режиме, в ходе которой TeX'овский код может отображаться в одном окне, а его эквивалент в формате MathML — в другом.

К сожалению, рассматриваемый конвертер может иметь дело *только* с TeX'овскими математическими конструкциями, а не с полноценным LaTeX'овским текстом, и не в состоянии работать с более сложными макроопределениями, как это уже описывалось выше. Система, основанная на LaTeX2HTML Росса Мура [↪L2HMML], обеспечивает трансляцию LaTeX'овского документа в целом. Она использует существующую систему LaTeX2HTML для трансляции нормального текста в HTML. Затем фрагменты, которые она идентифицирует как математические, передаются в программу WebEQ для преобразования их в MathML, после чего они вставляются в HTML.

Недостаток данного подхода состоит в том, что программа-ассистент (WebEQ Wizard) для WebEQ поддерживает средства разметки LaTeX'а не в полном объеме. Осуществляется лишь поддержка его варианта, именуемого

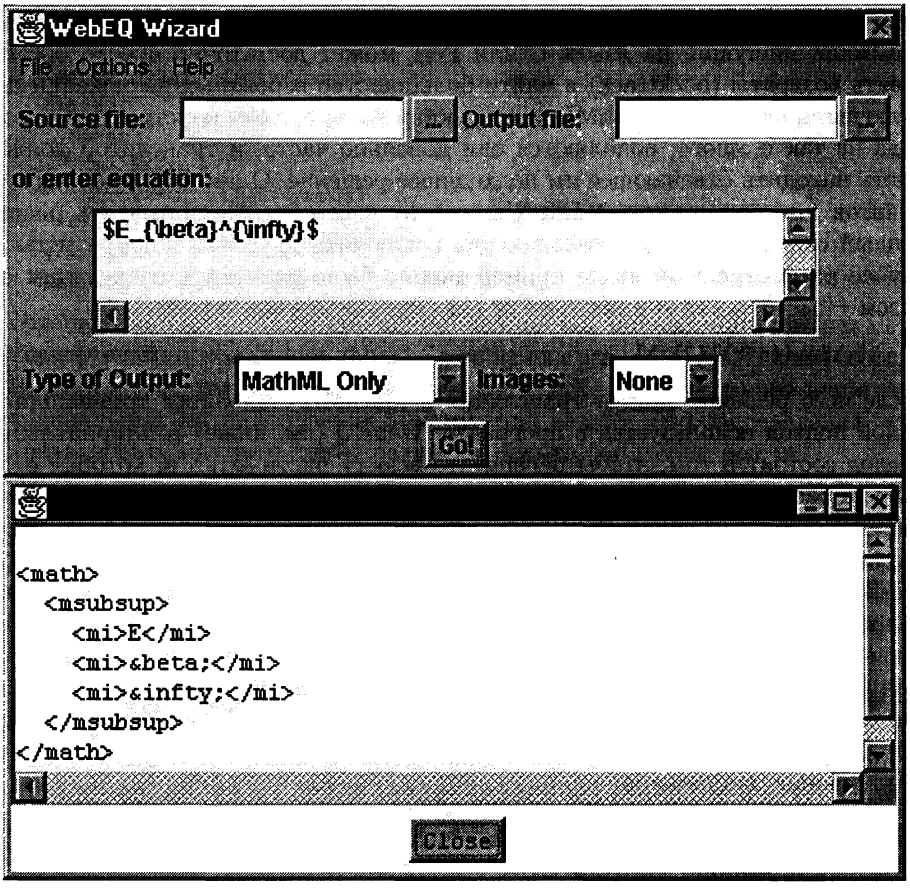


Рис. 8.4. Программа-ассистент из состава WebEQ в процессе преобразования текста из LaTeX'a в MathML.

WebTeX. Из этого вытекает, что при работе с документами потребуется выполнение некоторых дополнительных операций. Более подробно программа WebEQ обсуждается в разд. 5.2.

8.2.3.2 Использование TeX4ht для преобразования LaTeX'a в XML/MathML

В главе 4 можно было видеть, насколько большой объем настроек требуется для TeX4ht при записи вывода в формате HTML. Поскольку HTML представляет собой просто приложение SGML, ясно, что TeX4ht можно не менее успешно использовать и для формирования файла в формате XML. Элементы, формируемые для той или иной конкретной команды LaTeX'a, можно изменить в конфигурационных файлах так, как потребуется. В приложении B.2 описывается, как вернуться от варианта, преобразованного программой TeX4ht,

к исходному и построить затем некоторую новую систему, базирующуюся на простых элементах XML.

Следовательно, полной трансляцией в MathML можно управлять, используя TEX4ht. Эта программа решает проблему «обратного просмотра» («backtracking») применительно к верхним и нижним индексам, используя двухшаговый процесс. «Зацепки» для реализации этого процесса встраиваются в DVI-файл как TEX'овские команды \special, после чего постпроцессор анализирует математическое выражение в целом, чтобы найти и локализовать начало той его части, где, например, начинается подвыражение для нижнего индекса.

Давайте рассмотрим некоторый завершенный пример, для которого сформируем полностью XML-файл. Исходный файл, приводимый ниже, представляет собой фрагмент файла-примера, рассматриваемого в приложении А.1.

```
\documentclass{article}
  \usepackage[x2ldemo]{tex4ht}
  \title{Simulation of  Energy Loss  Straggling}
  \author{Maria Physicist}
\begin{document}
\maketitle
\section{Landau theory}\label{sec:phys332-1}
The Landau probability distribution may be written in
terms of the universal Landau function \cite{bib-LAND}.
\subsection{Restrictions}
The Landau formalism makes two restrictive assumptions:
\begin{enumerate}
\item The typical energy loss is small.
\item The typical energy loss in the absorber should be
  large (see section \ref{urban}).
\end{enumerate}
\section{Urb\'an model}\label{urban}
The following values are obtained:

\begin{tabular}{llcrr}
16   & 16      & & 2000  & 29.63\\
100  & 27.59   & &  100  & 32.00
\end{tabular}

\begin{thebibliography}{10}
\bibitem{bib-LAND} L.Landau. On the Energy Loss of Fast Particles by
Ionisation. Originally published in \emph{J. Phys.}, 8:201, 1944.
\end{thebibliography}
\end{document}
```

Конфигурационный файл в данном случае имеет такой вид:

```
1  \Configure{html}{xml}
2  \Preamble{html,0.0,ref-,fonts}
3  \Configure{HTML} {\IgnorePar\Tg<?xml version="1.0"?>
4      \Tg<document>}
5     {\Tg</document>}
6  \Configure{section}
7     {\EndP \IgnorePar\par  \GetLabel \Tg<section  \PutLabel>}
8     {\EndP \IgnorePar \Tg</section>}
9     {\Tg<stitle>}{\Tg</stitle>}
10 \Configure{subsection}
11    {\EndP \IgnorePar\par  \GetLabel \Tg<subsection \PutLabel>}
12    {\EndP \IgnorePar \Tg</subsection>}
13    {\Tg<stitle>}{\Tg</stitle>}
14 \Configure{likesection}
15    {\EndP  \IgnorePar\par\GetLabel \Tg<section \PutLabel class="star">}
16    {\EndP \Tg</section>}
```

```
17      {\Tg<stitle>}{\Tg</stitle>}
18    \ConfigureList{thebibliography}
19      {\EndP  \GetLabel \Tg<bibliography \PutLabel>
20        \def\EndItem{\def\EndItem{\EndP \Tg</bibitem>}}}
21      {\EndItem \Tg</bibliography>}
22      {\EndItem \DeleteMark}
23      {\Tg<bibitem id="\AnchorLabel">\par}
24    \ConfigureList{enumerate}
25      {\EndP  \GetLabel \Tg<lalist \PutLabel class="enumerate">
26        \def\EndItem{\def\EndItem{\EndP\Tg</item>}}} {\EndItem \Tg</lalist>}
27      {\EndItem \DeleteMark}              {\Tg<item>\par}
28    \Configure{tabular}
29      {\Tg<tabular preamble="\Clr">}   {\Tg</tabular>}
30      {\Tg<row>}{\Tg</row>}  {\Tg<cell \Hnewline>}{\Tg</cell>}
31    \Configure{maketitle}  {}{} {\Tg<title>}{\Tg</title>}
32    \Configure{thanks author date and}
33      {}{} {\Tg<author>}{\Tg</author>} {\Tg<date>}{\Tg</date>} {} {}
34    \Configure{emph}{\Tg<emph>}{\Tg</emph>}
35    \Configure{label}{id="#1"}{\Tg<pagelabel id="#1"/>}
36    \Configure{pageref}{\Tg<pageref refid="#1"/>}
37    \Configure{ref}{\Tg<ref refid="#1"/>}
38    \Configure{cite}{\Tg<cite refid="#1"/>}
39    \begin{document}
40    \EndPreamble
41    \Configure{HtmlPar}  {\EndP\Tg<P>} {\EndP\Tg<P>}  {\Tg</P>} {\Tg</P>}
```

Первая строка этого конфигурационного файла задает расширение в имени файла `xml` вместо `html` для выводного файла.

Опция пакета `fonts` в строке 2 конфигурационного файла определяет «зацепки» для шрифтовых команд LaTeX'а. С другой стороны, опция пакета `ref` требует, чтобы команды работы с перекрестными ссылками LaTeX'а зафиксировали соответствующие метки, не заменяя их на номера разделов и страниц, чтобы использовать эти метки внутри атрибутов для гипертекстовых элементов. Кроме того, эта опция добавляет пару команд

```
\GetLabel
\PutLabel
```

для извлечения метки из следующей команды `\label` и для помещения этой метки в выводной файл соответственно.

Аргументы `id="#1"` и `\Tg{pagelabel id="#1"/}` «крючка» "`label`" в строке 35 показывают как команды `\PutLabel` и `\label`, соответственно, устанавливают их метки.

Команда `\Clr` в строке 29 формирует аргумент окружения `tabular`, который будет сохранен в XML-выводе.

Команды `\ConfigureList` в строках 18 и 24 используют следующую команду в конце их третьего аргумента:

```
\DeleteMark
```

Она удаляет исходные маркеры в начале элементов соответствующих окружений спискового типа.

```
\AnchorLabel
```

Эта команда (используется в строке 23) вставляет затем эти маркеры в вывод.

После того как все это будет обработано программой TEX4ht, получится следующий XML-вывод:

```
<?xml version="1.0"?>
<document>
<title>Simulation of Energy Loss Straggling</title>
<author>Maria Physicist</author>
<date>November 9, 1998</date>
<section id="sec:phys332-1">
<stitle>Landau theory</stitle>
<p>   The Landau probability distribution may be written in
terms of the universal Landau function <cite refid="bib-LAND"/> .
</p>
<subsection>
<stitle>Restrictions</stitle>
<p>The Landau formalism makes two restrictive assumptions:</p>
<lalist class="enumerate">
     <item><p>The typical energy loss is small.</p></item>
     <item><p>The typical energy loss in the absorber should be
     large (see section <ref refid="urban"/> ).</p></item>
</lalist>
</subsection>
</section>
<section id="urban">
<stitle>Urb&aacute;n model</stitle>
<p>   The following values are obtained:
</p><p>   <tabular preamble="llcrr"><row>
<cell>16</cell><cell>16</cell><cell></cell><cell>2000</cell>
<cell>29.63</cell></row><row><cell>100</cell>
<cell>27.59</cell><cell></cell><cell>100</cell><cell>32.00</cell>
   </row></tabular></p>
</section>
<section class="star">
<stitle>References</stitle>
<bibliography >
<bibitem id="bib-LAND">
 <p>L.Landau. On the Energy Loss of Fast Particles by Ionisation.
 Originally published in <emph>J. Phys.</emph>, 8:201, 1944.</p>
</bibitem>
</bibliography>
</section>
</document>
```

По своей природе MathML имеет дело с низкоуровневыми структурными свойствами, которые не всегда реализуются с помощью LATEX'овских макро, а могут представлять собой средства, жестко встроенные в собственно TEX. Это значит, что MathML — естественный претендент на использование особенностей программы TEX4ht, как это описывается в предыдущем разделе и в приложении В.2.

Несколько строк простого LATEX'овского текста, соответственно структурированного, могут служить примером, иллюстрирующим многие из элементов процесса преобразования из LATEX'а в MathML:

```
\begin{eqnarray}
\bar u &=& \int_{I}^E g(x)dx  \nonumber \\
E &=& \frac{I}{1-\mathrm{max}}\label{xx}
\end{eqnarray}
```

При распечатке этого примера получим:

$$\begin{aligned} \bar{u} &= \int_I^E g(x)dx \\ E &= \frac{I}{1-\max} \end{aligned} \tag{1}$$

Теперь требуется сформировать новый конфигурационный файл для TEX4ht, который позволит работать с текстами, включающими математические выражения, получая на выходе правильный текст в формате MathML:

```
1  \Preamble{html,0.0,ref-,fonts,math}
2  \Configure{$$}{\DviMath}{\EndDviMath}{}
3  \Configure{eqnarray}
4     {\GetLabel \Tg<eqnarray \PutLabel>}
5     {\GetLabel \Tg</eqnarray>}
6     {\GetLabel \Tg<subeqn \PutLabel>\Tg<math>}
7     {\Tg</math>\Tg</subeqn>}
8     {\ifnum \Col=4 \Tg<mtext>\PauseMathClass \fi}
9     {\ifnum \Col=4 \EndPauseMathClass\Tg</mtext>\fi}
10 \Configure{label}{id="#1"}{\Tg<pagelabel id="#1"/>}
11 \Configure{SUBSUP}
12    {\Send{BACK}{<msubsup>}\Tg<mrow>}
13    {\Tg</mrow>\Tg<mrow>}
14    {\Tg</mrow>\TG</msubsup>}
15 \Configure{frac}
16    {\Tg<mfrac>\Tg<mrow>} {\Tg</mrow>\HCode{<!--}}
17    {\HCode{-->}\Tg<mrow>}{\Tg</mrow>\Tg</mfrac>}
18 \Configure{mathrm}{\Tg<mi>\PauseMathClass}
19    {\EndPauseMathClass\Tg</mi>}
20 \Configure{accent}\=\bar{{}{}}
21    {}{\Tg<mover accent="true">#2\Tg<mo>\HCode{&OverBar;}\Tg</mo>\Tg</mover>}
22 \Configure{MathClass}{0}{*}{<mi>}{</mi>}{}
23 \Configure{MathClass}{1}{*}{<mo>}{</mo>}{}
24 \Configure{MathClass}{2}{*}{<mo>}{</mo>}{}
25 \Configure{MathClass}{3}{*}{<mo>}{</mo>}{}
26 \Configure{MathClass}{4}{*}{<mrow><mo>}{</mo>}{}
27 \Configure{MathClass}{5}{*}{<mo>}{</mo></mrow>}{}
28 \Configure{MathClass}{6}{*}{<mo>}{</mo>}{}
29 \Configure{MathClass}{7}{*}{<mn>}{</mn>}{0123456789}
30 \begin{document}
31 \EndPreamble
```

За исключением `<eqnarray>`, `</eqnarray>`, `<subeqn>` и `</subeqn>`, все остальные элементы в конфигурационном файле связаны с MathML.

LATEX рассматривает математическое окружение `$$`*formula*`$$` как часть окружения `eqnarray`. Строка 2 в конфигурационном файле предназначена для установки условий обработки этого материала.

Всякий раз, когда программа TEX4ht обрабатывает некоторую таблицу, она сохраняет для нее значения «переменных» `\Row` и `\Col` — номера строки и столбца для текущей «клеточки» этой таблицы. В строках 8 и 9 конфигурационного файла используется команда `\Col`, которая обеспечивает установку номеров выражений.

```
<eqnarray>
<subeqn><math>
   <mover accent="true">
      <mi>u</mi>
```

```
      <mo>&OverBar;</mo>
    </mover>
    <mo>=</mo>
    <msubsup>
      <mo>&int;</mo>
      <mrow><mi>I</mi></mrow>
      <mrow><mi>E</mi></mrow>
    </msubsup>
    <mi>g</mi><mrow><mo>(</mo><mi>x</mi><mo>)</mo></mrow>
    <mi>d</mi><mi>x</mi>
    <mtext></mtext>
</math></subeqn><subeqn><math>
</math></subeqn><subeqn id="xx"><math>
    <mi>E</mi> <mo>=</mo>
    <mfrac>
      <mrow><mi>I</mi></mrow>
      <mrow><mn>1</mn> <mo>-</mo> <mi> max </mi> </mrow>
    </mfrac>
    <mtext>(1)</mtext>
</math></subeqn><subeqn><math>
</math></subeqn>
</eqnarray>
```

8.2.3.3 Преобразование LaTeX'а в MathML с помощью системы Omega

Еще один интересный и рациональный подход реализуется в третьем конвертере, основанном на расширении функциональных возможностей программы TeX. Этот конвертер, именуемый Omega, разработали Джон Плейс и Яннис Хараламбус [↪OMEGA]. Основные расширения программы Omega по сравнению с TeX'ом состоят в использовании Unicode как средства внутренней кодировки, а также набора «процессов трансляции», применяемых к вводу и выводу. Для поддержки MathML программа Omega также осуществляет непосредственный вывод в формате XML в процессе работы TeX'а под управлением пользовательских макро. Тогда LaTeX'овская команда `\frac` переопределяется следующим образом:

```
\renewcommand{\frac}[2]{%
 \SGMLstarttag{mfrac}{#1}{#2}\SGMLendtag{mfrac}}
```

основываясь на некоторых из новых примитивов, введенных в программе Omega[1]:

`\SGMLstarttag{`*имя элемента*`}`
`\SGMLendtag{`*имя элемента*`}`
`\SGMLentity{`*имя объекта*`}`

Имеется новый примитив, запускающий формирование XML-вывода вместо вывода в формате DVI.

[1] Со временем эти имена могут измениться; есть также новые примитивы, задающие атрибуты и формирующие некоторые специальные символы. Более подробное описание всех этих средств можно найти в документации к программе Omega.

Существенное достоинство такого подхода состоит в том, что появляется возможность воспользоваться знанием, которым TEX обладает относительно «внутренних» алгоритмов работы в математическом режиме; эта программа *располагает* некоторыми сведениями относительно того, где, например, начинаются выражения с верхними индексами, но не может сделать эти выражения доступными для макро. Если внести изменения в собственно TEX'овскую обрабатывающую программу, можно получить доступ к требуемым деталям и с полной уверенностью утверждать, что проблем с пользовательскими макро не будет. К недостаткам этого варианта можно отнести то, что TEX не располагает на самом деле таким объемом информации, который запрашивается MathML (см. Knuth (1986, p.129), где обсуждается вопрос о присоединении верхних индексов (индексных выражений)). Может быть, было бы более благоразумным просто представить в некоторой форме всю информацию, которую TEX помещает в DVI-файл, а затем использовать некоторую схему, подобную той, что реализуется в TEX4ht для постпроцессорной обработки DVI-файла и формирования вывода в формате XML.

Подобно программе TEX4ht, программа Omega обеспечивает поддержку редактируемых файлов описания кодировки шрифтов, которые отображают TEX'овские математические символы в объекты MathML, а также позволяют различать типы математических операторов.

8.2.4 Печать MathML-фрагментов

Отвлечемся теперь от проблем, связанных с Сетью, а также от того, как *сформировать* MathML. Пусть надо просто *распечатать* наши новые математические документы, а это задача — более знакомая и понятная. Сам по себе TEX слишком тяжеловесен и монолитен, чтобы его можно было использовать в динамической среде наподобие программы-навигатора, но в качестве форматирующего средства, применяемого в пакетном режиме обработки, он превосходен. Поэтому надо просто уметь преобразовывать имеющуюся XML-разметку в TEX'овский формат, после чего получать искомый результат обычным образом.

Существуют три основных подхода к использованию TEX'а в качестве форматирующего средства для XML, как это уже было продемонстрировано в гл. 7:

1. Можно написать специализированную программу с использованием моделей типа SAX или DOM и конвертировать, например элементы `<mfrac>` непосредственно в LATEX'овскую команду `\frac`.
2. Можно использовать языки стилей XSL или DSSSL, чтобы отобразить элементы MathML в математические форматирующие объекты и затем использовать TEX в качестве постпроцессора для программы обработки стилей.
3. Можно написать TEX'овский макропакет, чтобы интерпретировать и верстать XML-текст непосредственно.

Достоинство первого из этих подходов заключается в том, что программист получает хороший контроль над формируемым результатом; нет необходимости

очень точно описывать формат вывода (макет страницы), поскольку можно воспользоваться преимуществами существующих LaTeX'овских пакетов (наподобие $\mathcal{AMS}$-LaTeX) для выполнения этой непростой работы. К его недостаткам можно отнести то, что он требует некоторой новой программы для каждого определения XML DTD, а трансляция специфична для TeX'а; трансляция для какой-либо другой программы форматирования, вроде FrameMaker, потребовала бы совершенно непохожего конвертера из XML в MIF.

Второй подход основывается на использовании стиля и потенциально намного мощнее. Он устраняет зависимость от конкретной программы форматирования, хотя трудоемкая работа по отображению математических потоковых объектов в TeX остается, ее приходится выполнять хотя бы один раз, при реализации постпроцессора для соответствующего стиля. К сожалению, остаются нерешенными некоторые важные проблемы:

- Существующая версия XSL все еще не включает в себя каких-либо математических форматирующих объектов, поэтому в DSSSL нет возможности их протестировать в достаточной степени, чтобы иметь уверенность в их корректном поведении в приложениях.
- Если результаты недостаточно хороши, практически невозможно как-то пытаться воздействовать на низкоуровневый TeX'овский код, сформированный компьютером.
- С помощью разных программ обработки стилей могут быть получены разные результаты, если в их постпроцессорных модулях есть какие-то ошибки.
- Вся работа должна быть выполнена в стиле; нельзя больше полагаться на поддержку таких развитых и отработанных пакетов, как, например, $\mathcal{AMS}$-LaTeX.

Действительно серьезная поддержка работы с математикой в DSSSL реализована в программе Jade Джеймса Кларка, а также обеспечивается макропакетом `jadetex` [↪JADETEX]. Дэвид Карлайл написал на DSSSL стиль, работающий практически со всеми презентационными и семантическими элементами MathML[↪DSSSLMML]. Большая часть отображений в нем очень проста для реализации, как это видно из примера для дробей:

```
(element mfrac
  (make fraction
    (let ((nl (children(current-node))))
     (sosofo-append
      (make math-sequence
        label: 'numerator
         (process-node-list (node-list-first nl)))
      (make math-sequence
        label: 'denominator
         (process-node-list (node-list-rest nl)))))))
```

$$f(\in, \delta s) = \frac{1}{\xi}\phi_v(\lambda_v, k, \beta^2)$$
$$\phi_v(\lambda_v, k, \beta^2) = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty}\phi(s)e^{\lambda s}ds$$
$$\phi(s) = \exp[k(1+\beta^2\gamma)]\exp[\psi(s)]$$
$$\lambda_u = k[\frac{\in-\in}{\xi} - \gamma' - \beta^2]$$

Рис. 8.5. Пример текста на MathML, обработанного модулем Jade для DSSSL и TEX'ом.

Результат обработки ранее рассматривавшегося примера с помощью Jade и затем `jadetex` показан на рис. 8.5. Остаются, однако, некоторые области MathML, работа с которыми не организована на должном уровне (в частности, полная поддержка операторов растяжения) вследствие ограничений, присущих математическим потоковым объектам в DSSSL.

Интересный метод, занимающий промежуточное положение между рассмотренными выше, это язык TeXML, предложенный и реализованный Дугом Лоувеллом из IBM (см. [↪TEXML]). Это XML-представление TEX'овского синтаксиса, причем печать документа осуществляется в два этапа:

1. Для преобразования исходного XML-документа в более простое XML-представление, в котором присутствуют только элементы TeXML, используются средства преобразования языка, подобного XSL.
2. TeXML-документ преобразуется в TEX посредством простой программы, после чего результат получается запуском TEX'а.

Преимущество этого подхода состоит в том, что можно использовать любую систему преобразования XML→ XML (которых имеется достаточно много) и не беспокоиться относительно таких TEX'овских символов, как обратная косая черта, скобки, другие специальные символы. Пользователь должен, однако, иметь представление о таких TEX'овских средствах, как макрокоманды, однако трансляция в реальный TEX осуществляется достаточно просто.

8.3 Заключение

После краткого объяснения, почему, на наш взгляд, MathML представляет собой важную разработку в области обмена, хранения и преобразования информации математического характера в Сети, мы обсудили проблемы, связанные с соответствующей специальной символикой, а также дали обзор программных пакетов, ориентированных в той или иной степени на работу с MathML. Были рассмотрены средства, позволяющие получить текст в формате MathML из LATEX'овского текста, приведены достаточно подробные примеры того, как программа TEX4ht может преобразовать исходные файлы LATEX'а в тексты в

формате XML и MathML. Было показано затем, каким образом LaTeX можно использовать в качестве постпроцессора вывода на печать для MathML.

Ясно, что предстоит пройти еще долгий путь, прежде чем вся изобразительная мощь LaTeX'а станет доступной для навигаторов Сети в форме MathML. Есть все же надежда, что уже в среднесрочной перспективе пользователи смогут преобразовывать свои исходные LaTeX'овские файлы в MathML для ввода их в программы символьной математики, для просмотра в Сети или для включения в тексты, обрабатываемые различными системами работы с документами. В то же самое время станет возможным сохранить типографское качество, обеспечиваемое обрабатывающими программами ядра TeX'а для воспроизведения Web-документов, содержащих значительную долю математического материала. В следующем тысячелетии мы увидим, возможно, идеальный симбиоз между LaTeX'ом и XML/MathML, что даст возможность пользователям просто переключаться между средствами представления документов, наилучшим образом отвечающими их потребностям в данный конкретный момент. Тогда не придется выбирать между MathML и LaTeX'ом для подготовки соответствующих научных текстов или данных, можно будет использовать такой формат, который наиболее удобен для пользователя и сочетает в требуемой степени преимущества каждого из этих двух подходов, не требуя при этом приложения каких-то дополнительных усилий.

Приложение А

Примеры файлов

А.1 Пример LaTeX'овского файла и его трансляция в XML

На протяжении всей этой книги мы старались привести как можно больше примеров из одного и того же документа (из простой статьи по физике, изданной в CERN'е), для того чтобы читатель имел возможность сравнивать. В этом разделе мы приводим фрагмент распечатки LaTeX'овского исходного документа и подготовленный с помощью TeX4ht перевод на язык XML (см. разд. 8.2.3.2 и приложение В.2) вместе с соответствующим определением типа документа DTD.

Полный текст этих файлов можно найти в директории `info/lwc` в сети TeX'овских архивов CTAN.

А.1.1 Исходный LaTeX'овский документ

```
\documentclass{article}
\usepackage{graphicx}
\usepackage{url}
\title{Simulation of  Energy Loss  Straggling}
\author{Maria Physicist}
\newcommand{\Emax}{\ensuremath{E_{\mathrm{max}}}}
\newcommand{\GEANT}{\texttt{GEANT}}
\begin{document}
\maketitle

\section{Introduction}

Due to the statistical nature of ionisation energy loss, large
fluctuations can occur in the amount of energy deposited by a particle
traversing an absorber element.  Continuous processes such as multiple
scattering and energy loss play a relevant role in the longitudinal
and lateral development of electromagnetic and hadronic
showers, and in the case of sampling calorimeters the
measured resolution can be significantly affected by such fluctuations
in their active layers.  The description of ionisation fluctuations is
characterised by the significance parameter $\kappa$, which is
proportional to the ratio of mean energy loss to the maximum allowed
energy transfer in a single collision with an atomic electron
\[
\kappa =\frac{\xi}{\Emax}
\]
\Emax{}
```

```
28  is the maximum transferable energy in a single collision with
29  an atomic electron.
30
31  .......
32
33  \section{Vavilov theory}
34  \label{vavref}
35
36  Vavilov\cite{bib-VAVI} derived a more accurate straggling distribution
37  by introducing the kinematic limit on the maximum transferable energy
38  in a single collision, rather than using $ \Emax = \infty $.
39  Now we can write\cite{bib-SCH1}:
40  \begin{eqnarray*}
41  f \left ( \epsilon, \delta s \right ) & = & \frac{1}{\xi} \phi_{v}
42  \left ( \lambda_{v}, \kappa, \beta^{2} \right )
43  \end{eqnarray*}
44  where
45  \begin{eqnarray*}
46  \phi_{v} \left ( \lambda_{v}, \kappa, \beta^{2} \right ) & = &
47  \frac{1}{2 \pi i} \int^{c+i\infty}_{c-i\infty}\phi \left( s \right )
48  e^{\lambda s} ds \hspace{2cm} c \geq 0 \\
49  \phi \left ( s \right ) & = &
50  \exp \left [ \kappa ( 1 + \beta^{2}\gamma ) \right ]
51  ~ \exp \left [ \psi \left ( s \right ) \right ], \\
52  \psi \left ( s \right )  & = & s \ln \kappa + ( s + \beta^{2} \kappa )
53  \left [ \ln (s/\kappa) + E_{1} (s/\kappa) \right ] - \kappa e^{-s/\kappa},
54  \end{eqnarray*}
55  and
56  \begin{eqnarray*}
57  E_{1}(z) & = & \int^{\infty}_{z} t^{-1} e^{-t} dt
58  \mbox{\hspace{1cm} (the exponential integral)} \\
59  \lambda_v & = & \kappa \left [ \frac{\epsilon - \bar{\epsilon}}{\xi}
60  - \gamma' - \beta^2 \right]
61  \end{eqnarray*}
62
63  The Vavilov parameters are simply related to the Landau parameter by
64  $\lambda_L = \lambda_v/\kappa - \ln\kappa $. It can be shown that as
65  $\kappa \rightarrow 0$, the distribution of the variable $\lambda_L$
66  approaches that of Landau. For $\kappa \leq 0.01$ the two
67  distributions are already practically identical. Contrary to what many
68  textbooks report, the Vavilov distribution \emph{does not} approximate
69  the Landau distribution for small $\kappa$, but rather the
70  distribution of $\lambda_L$ defined above tends to the distribution of
71  the true $\lambda$ from the Landau density function.  Thus the routine
72  \texttt{GVAVIV} samples the variable $\lambda_L$ rather than
73  $\lambda_v$.  For $\kappa \geq 10$ the Vavilov distribution tends to a
74  Gaussian distribution (see next section).
75  ....
76
77  \begin{thebibliography}{10}
78  \bibitem{bib-LAND}
79  L.Landau.
80  \newblock On the Energy Loss of Fast Particles by Ionisation.
81  \newblock Originally published in \emph{J. Phys.}, 8:201, 1944.
82  \newblock Reprinted in D.ter Haar, Editor, \emph{L.D.Landau, Collected
83    papers}, page 417.  Pergamon Press, Oxford, 1965.
84
85  \bibitem{bib-SCH1}
86  B.Schorr.
87  \newblock Programs for the Landau and the Vavilov distributions and the
88    corresponding random numbers.
89  \newblock \emph{Comp. Phys. Comm.}, 7:216, 1974.
90
91  \bibitem{bib-SELT}
92  S.M.Seltzer and M.J.Berger.
93  \newblock Energy loss straggling of protons and mesons.
94  \newblock In \emph{Studies in Penetration of Charged Particles in
```

```
95    Matter}, Nuclear Science Series~39, Nat. Academy of Sciences,
96    Washington DC, 1964.
97
98  \bibitem{bib-TALM}
99  R.Talman.
100 \newblock On the statistics of particle identification using ionization.
101 \newblock \emph{Nucl. Inst. Meth.}, 159:189, 1979.
102
103 \bibitem{bib-VAVI}
104 P.V.Vavilov.
105 \newblock Ionisation losses of high energy heavy particles.
106 \newblock \emph{Soviet Physics JETP}, 5:749, 1957.
107 \end{thebibliography}
108
109 \end{document}
```

A.1.2 LaTeX'овский текст, преобразованный в XML

```
1   <?xml version="1.0"?>
2   <!DOCTYPE document SYSTEM "latexexa.dtd" []>
3   <document>
4   <frontmatter>
5    <title>Simulation of Energy Loss Straggling</title>
6    <author>Maria Physicist</author>
7    <date>January 14, 1999</date>
8   </frontmatter>
9   <bodymatter>
10  <section id="intro"> <stitle>Introduction</stitle>
11  <par>Due to the statistical nature of ionisation energy loss, large
12  fluctuations can occur in the amount of energy deposited by a particle
13  traversing an absorber element. Continuous processes such as multiple
14  scattering and energy loss play a relevant role in the longitudinal
15  and lateral development of electromagnetic and hadronic showers, and
16  in the case of sampling calorimeters the measured resolution can be
17  significantly affected by such fluctuations in their active
18  layers. The description of ionisation fluctuations is characterised by
19  the significance parameter <inlinemath>
20  <math><mi>&kappa;</mi></math></inlinemath>, which is proportional to
21  the ratio of mean energy loss to the maximum allowed energy transfer
22  in a single collision with an atomic electron
23
24  <displaymath><math><mrow>
25  <mi>&kappa;</mi><mo>=</mo> <mfrac> <mrow>
26  <mi>&xi;</mi></mrow><mrow><msub><mi>E</mi><mrow><mi>max </mi> </mrow>
27  </msub> </mrow> </mfrac> </mrow></math></displaymath>
28
29  <inlinemath><math><msub><mi>E</mi><mrow><mi>max </mi> </mrow> </msub>
30  </math></inlinemath> is the maximum transferable energy in a single
31  collision with an atomic electron.
32
33  ....
34
35  </section>
36  <section id="vavref"><stitle>Vavilov theory</stitle>
37  <par>Vavilov<cite refid="bib-VAVI"/> derived a more accurate
38  straggling distribution by introducing the kinematic limit on the
39  maximum transferable energy in a single collision, rather than using
40  <inlinemath> <math><msub><mi>E</mi><mrow><mi>max </mi> </mrow> </msub>
41  <mo>=</mo><mi>&infin;</mi></math></inlinemath>. Now we can write<cite
42  refid="bib-SCH1"/>: <eqnarray><subeqn><math><mi>f</mi> <mfenced
43  open='('
44  close=')'><mi>&epsi;</mi><mo>,</mo><mi>&delta;</mi><mi>s</mi></mfenced>
45  <mo>=</mo>
46  <mfrac><mrow><mn>1</mn></mrow><mrow><mi>&xi;</mi></mrow></mfrac>
47  <msub><mi>&phi;</mi><mrow><mi>v</mi></mrow>
```

```
48  </msub> <mfenced open='('
49  close=')'><msub><mi>&lambda;</mi><mrow><mi>v</mi></mrow> </msub>
50  <mo>,</mo><mi>&kappa;</mi><mo>,</mo><msup><mi>&beta;</mi><mrow><mn>2</mn>
51  </mrow> </msup> </mfenced> <mtext></mtext> </math></subeqn></eqnarray>
52  where
53  <eqnarray><subeqn><math><msub><mi>&phi;</mi><mrow><mi>v</mi></mrow>
54  </msub> <mfenced open='('
55  close=')'><msub><mi>&lambda;</mi><mrow><mi>v</mi></mrow> </msub>
56  <mo>,</mo><mi>&kappa;</mi><mo>,</mo><msup><mi>&beta;</mi><mrow><mn>2</mn>
57  </mrow> </msup> </mfenced> <mo>=</mo>
58  <mfrac><mrow><mn>1</mn></mrow><mrow><mn>2</mn><mi>&pi;</mi><mi>i</mi></mrow>
59  </mfrac><msubsup><mo>&int;</mo>
60  <mrow><mi>c</mi><mo>+</mo><mi>i</mi><mi>&infin;</mi></mrow>
61  <mrow><mi>c</mi><mo>-</mo><mi>i</mi><mi>&infin;</mi>
62  </mrow></msubsup><mi>&phi;</mi><mfenced
63  open='('
64  close=')'><mi>s</mi></mfenced><msup><mi>e</mi><mrow><mi>&lambda;</mi><mi>s</mi>
65  </mrow> </msup> <mi>d</mi><mi>s</mi><mspace
66  width='2cm'/><mi>c</mi><mo>&geq;</mo><mn>0</mn> <mtext></mtext>
67  </math></subeqn><subeqn><math> </math></subeqn><subeqn
68  ><math><mi>&phi;</mi><mfenced open='(' close=')'><mi>s</mi></mfenced>
69  <mo>=</mo> <mo>exp</mo><mfenced open='['
70  close=']'><mi>&kappa;</mi><mrow><mo>(</mo><mn>1</mn><mo>+</mo>
71  <msup><mi>&beta;</mi><mrow><mn>2</mn>
72  </mrow> </msup>
73  <mi>&gamma;</mi><mo>)</mo></mrow></mfenced><mo>exp</mo><mfenced
74  open='[' close=']'><mi>&psi;</mi> <mfenced open='('
75  close=')'><mi>s</mi></mfenced></mfenced><mo>,</mo> <mtext></mtext>
76  </math></subeqn><subeqn><math> </math></subeqn><subeqn
77  ><math><mi>&psi;</mi> <mfenced open='(' close=')'><mi>s</mi></mfenced>
78  <mo>=</mo> <mi>s</mi><mo>ln</mo>
79  <mi>&kappa;</mi><mo>+</mo><mrow><mo>(</mo><mi>s</mi><mo>+</mo><msup>
80  <mi>&beta;</mi><mrow><mn>2</mn>
81  </mrow> </msup> <mi>&kappa;</mi><mo>)</mo></mrow><mfenced open='['
82  close=']'><mo>ln</mo>
83  <mrow><mo>(</mo><mi>s</mi><mo>/</mo><mi>&kappa;</mi><mo>)</mo></mrow>
84  <mo>+</mo><msub><mi>E</mi><mrow>
85  <mn>1</mn> </mrow> </msub>
86  <mrow><mo>(</mo><mi>s</mi><mo>/</mo><mi>&kappa;</mi><mo>)</mo>
87  </mrow></mfenced><mo>-</mo><mi>&kappa;</mi><msup><mi>e</mi><mrow>
88  <mo>-</mo><mi>s</mi><mo>/</mo><mi>&kappa;</mi>
89  </mrow> </msup> <mo>,</mo> <mtext></mtext> </math></subeqn></eqnarray>
90  and <eqnarray><subeqn><math><msub><mi>E</mi><mrow><mn>1</mn> </mrow>
91  </msub> <mrow><mo>(</mo><mi>z</mi><mo>)</mo></mrow>
92  <mo>=</mo><msubsup> <mo>&int;</mo>
93  <mrow><mi>&infin;</mi></mrow><mrow><mi>z</mi></mrow></msubsup>
94  <msup><mi>t</mi><mrow><mo>-</mo><mn>1</mn>
95  </mrow> </msup> <msup><mi>e</mi><mrow><mo>-</mo><mi>t</mi> </mrow>
96  </msup> <mi>d</mi><mi>t</mi><mspace width='1cm'/><mtext>(the
97  exponential integral)</mtext> <mtext></mtext> </math></subeqn><subeqn
98  ><math> </math></subeqn><subeqn
99  ><math><msub><mi>&lambda;</mi><mrow><mi>v</mi></mrow> </msub>
100 <mo>=</mo> <mi>&kappa;</mi><mfenced open='['
101 close=']'><mfrac><mrow><mi>&epsi;</mi><mo>-</mo><munderover
102 accent='true'><mi>&epsi;</mi><mrow></mrow><mo>&barwed;</mo></munderover>
103 </mrow> <mrow><mi>&xi;</mi></mrow></mfrac>
104 <mo>-</mo><mi>&gamma;</mi><mi>&prime;</mi>
105 <mo>-</mo><msup><mi>&beta;</mi><mrow><mn>2</mn> </mrow> </msup>
106 </mfenced> <mtext></mtext> </math></subeqn></eqnarray>
107 </par>
108 <par>The Vavilov parameters are simply related to the Landau parameter
109 by <inlinemath><math><msub><mi>&lambda;</mi><mrow><mi>L</mi> </mrow>
110 </msub> <mo>=</mo><msub><mi>&lambda;</mi><mrow><mi>v</mi></mrow>
111 </msub> <mo>/</mo><mi>&kappa;</mi><mo>-</mo><mo>ln</mo>
112 <mi>&kappa;</mi></math></inlinemath>. It can be shown that as
113 <inlinemath> <math>
114 <mi>&kappa;</mi><mo>&rarr;</mo><mn>0</mn></math></inlinemath>, the
```

```
115  distribution of the variable <inlinemath> <math>
116  <msub><mi>&lambda;</mi><mrow><mi>L</mi> </mrow> </msub>
117  </math></inlinemath> approaches that of Landau. For <inlinemath>
118  <math>
119  <mi>&kappa;</mi><mo>&leq;</mo><mn>0</mn><mo>.</mo><mn>0</mn><mn>1</mn>
120  </math></inlinemath>
121  the two distributions are already practically identical. Contrary to
122  what many textbooks report, the Vavilov distribution <emph> does
123  not</emph> approximate the Landau distribution for small
124  <inlinemath><math><mi>&kappa;</mi></math></inlinemath>, but rather the
125  distribution of <inlinemath> <math>
126  <msub><mi>&lambda;</mi><mrow><mi>L</mi> </mrow> </msub>
127  </math></inlinemath> defined above tends to the distribution of the
128  true <inlinemath><math><mi>&lambda;</mi></math></inlinemath> from the
129  Landau density function. Thus the routine <texttt> GVAVIV</texttt>
130  samples the variable <inlinemath>
131  <math><msub><mi>&lambda;</mi><mrow><mi>L</mi> </mrow> </msub>
132  </math></inlinemath> rather than <inlinemath> <math>
133  <msub><mi>&lambda;</mi><mrow><mi>v</mi></mrow> </msub>
134  </math></inlinemath>. For <inlinemath> <math>
135  <mi>&kappa;</mi><mo>&geq;</mo><mn>1</mn><mn>0</mn></math></inlinemath>
136  the Vavilov distribution tends to a Gaussian distribution (see next
137  section). </par>
138  </section>
139  .....
140  </section>
141  <section class="star"><stitle>References</stitle>
142  <bibliography>
143  <bibitem id="bib-LAND">
144  <par>L.Landau. On the Energy Loss of Fast Particles by
145  Ionisation. Originally published in <emph>J. Phys.</emph>, 8:201,
146  1944. Reprinted in D.ter Haar, Editor, <emph>L.D.Landau, Collected
147  papers</emph>, page 417. Pergamon Press, Oxford, 1965. </par>
148  </bibitem>
149  <bibitem id="bib-SCH1">
150  <par>B.Schorr. Programs for the Landau and the Vavilov distributions
151  and the corresponding random numbers. <emph>Comp. Phys. Comm.</emph>,
152  7:216, 1974. </par>
153  </bibitem>
154  <bibitem id="bib-SELT">
155  <par>S.M.Seltzer and M.J.Berger. Energy loss straggling of protons and
156  mesons. In <emph>Studies in Penetration of Charged Particles in
157  Matter</emph>, Nuclear Science Series 39, Nat. Academy of Sciences,
158  Washington DC, 1964. </par>
159  </bibitem>
160  <bibitem id="bib-TALM">
161  <par>R.Talman. On the statistics of particle identification using
162  ionization. <emph>Nucl. Inst. Meth.</emph>, 159:189, 1979. </par>
163  </bibitem>
164  <bibitem id="bib-VAVI">
165  <par>P.V.Vavilov. Ionisation losses of high energy heavy
166  particles. <emph>Soviet Physics JETP</emph>, 5:749, 1957.</par>
167  </bibitem>
168  </bibliography>
169  </section>
170  </bodymatter>
171  </document>
```

A.1.3 Определение типа документа на языке XML

```
1  <!-- latex.dtd: XML version of LaTeX + MathML -->
2
3  <!ENTITY % fontchange  "emph|textit|textbf|textsf|textsl|texttt" >
4  <!ENTITY % misc "url|quad|hspace|vspace|includegraphics|footnote|tag|ent">
5  <!ENTITY % xref "ref|cite|pageref">
```

```
<!ENTITY % chunk "lalist|par|tabular|figure|table|align|bibliography">
<!ENTITY % mathobj   "displaymath|inlinemath|equation|eqnarray" >
<!ENTITY % inline "#PCDATA|%fontchange;|%chunk;|%misc;|%xref;|%mathobj;">

<!ELEMENT document (frontmatter?,bodymatter)>
<!ATTLIST document class CDATA "article">

<!ELEMENT frontmatter (title,author,date?,abstract?,keywords?)>
<!ELEMENT bodymatter  ((par|section)*,appendix*)>

<!-- front matter -->
<!ELEMENT title (%inline;)*>
<!ELEMENT author (%inline;)*>
<!ELEMENT date (#PCDATA)>

<!-- structuring -->
<!ELEMENT section (stitle,(%chunk;|subsection)*)>
<!ATTLIST section
 class CDATA #IMPLIED
 id ID #IMPLIED>
<!ELEMENT subsection (stitle,(%chunk;|paragraph)*)>
<!ATTLIST subsection
 class CDATA #IMPLIED
 id ID #IMPLIED>
<!ELEMENT paragraph (stitle,(%chunk;|subparagraph)*)>
<!ATTLIST paragraph
 class CDATA #IMPLIED
 id ID #IMPLIED>
<!ELEMENT subparagraph (stitle,(%chunk;)*)>
<!ATTLIST subparagraph
 class CDATA #IMPLIED
 id ID #IMPLIED>
<!ELEMENT stitle (%inline;)*>

<!-- font changes -->
<!ELEMENT emph   (%inline;)*>
<!ELEMENT textit (%inline;)*>
<!ELEMENT textbf (%inline;)*>
<!ELEMENT textsf (%inline;)*>
<!ELEMENT textsl (%inline;)*>
<!ELEMENT texttt (%inline;)*>

<!-- lists -->
<!ELEMENT lalist (item)*>
<!ATTLIST lalist
 id ID #IMPLIED
 class (enumerate|itemize|description) #REQUIRED>
<!ELEMENT item (%inline;)*>

<!-- bibliography -->
<!ELEMENT bibliography (bibitem)*>
<!ELEMENT bibitem (%inline;)*>
<!ATTLIST bibitem
 id ID #REQUIRED>

<!-- floats -->
<!ELEMENT table (%chunk;|caption|includegraphics)*>
<!ELEMENT figure (%chunk;|caption|includegraphics)*>
<!ELEMENT caption (%inline;)*>
<!ATTLIST caption
 id ID #IMPLIED>
<!ELEMENT includegraphics EMPTY>
<!ATTLIST includegraphics
    width CDATA #IMPLIED
    height CDATA #IMPLIED
    scale CDATA #IMPLIED
    file CDATA #IMPLIED>
```

```
73
74  <!-- tables -->
75  <!ELEMENT tabular (hline|row)*>
76  <!ATTLIST tabular
77     preamble CDATA #REQUIRED>
78  <!ELEMENT row (cell)*>
79  <!ELEMENT hline EMPTY>
80  <!ELEMENT cell (%inline;)*>
81  <!ELEMENT newline EMPTY>
82  <!ATTLIST newline
83   id ID #IMPLIED>
84
85  <!-- low-level bits and pieces -->
86  <!ELEMENT align (%inline;)*>
87  <!ATTLIST align
88    style CDATA #REQUIRED>
89  <!ELEMENT url EMPTY>
90  <!ATTLIST url
91    name CDATA #REQUIRED>
92  <!ELEMENT par (%inline;)*>
93  <!ELEMENT quad EMPTY>
94  <!ELEMENT hspace EMPTY>
95  <!ATTLIST hspace
96    dim CDATA #REQUIRED>
97  <!ELEMENT vspace EMPTY>
98  <!ATTLIST vspace
99    dim CDATA #REQUIRED>
100 <!ELEMENT tag (#PCDATA)>
101 <!ELEMENT ent EMPTY>
102 <!ATTLIST ent
103    value CDATA #REQUIRED
104    name CDATA #REQUIRED>
105
106 <!-- cross-refs -->
107 <!ELEMENT cite EMPTY>
108 <!ATTLIST cite
109   refid IDREF #REQUIRED>
110 <!ELEMENT ref EMPTY>
111 <!ATTLIST ref
112  refid IDREF #REQUIRED>
113
114 <!-- maths. must reduce to <math> elements  for MathML -->
115 <!ELEMENT equation (math)*>
116 <!ATTLIST equation
117  id ID #IMPLIED>
118 <!ELEMENT displaymath (math)*>
119 <!ELEMENT inlinemath  (math)*>
120 <!ELEMENT subeqn (math)*>
121 <!ATTLIST subeqn
122  id ID #IMPLIED>
123 <!ELEMENT eqnarray (subeqn)*>
124 <!ATTLIST eqnarray
125  number (yes|no) "yes"
126  id ID #IMPLIED>
127
128 <!-- sub DTDs and entities -->
129 <!--Added Math Symbols: Arrows-->
130 <!ENTITY % isoamsae.dtd SYSTEM "isoamsae.dtd">
131
132 <!--Added Math Symbols: Binary Operators-->
133 <!ENTITY % isoamsbe.dtd SYSTEM "isoamsbe.dtd">
134
135 <!--Added Math Symbols: Delimiters-->
136 <!ENTITY % isoamsce.dtd SYSTEM "isoamsce.dtd">
137
138 <!--Added Math Symbols: Negated Relations-->
139 <!ENTITY % isoamsne.dtd SYSTEM "isoamsne.dtd">
```

```
140
141   <!--Added Math Symbols: Ordinary-->
142   <!ENTITY % isoamsoe.dtd SYSTEM "isoamsoe.dtd">
143
144   <!--Added Math Symbols: Relations-->
145   <!ENTITY % isoamsre.dtd SYSTEM "isoamsre.dtd">
146
147   <!--General Technical-->
148   <!ENTITY % isoteche.dtd SYSTEM "isoteche.dtd">
149
150   <!--Numbers and Currency symbols-->
151   <!ENTITY % isonume.dtd SYSTEM "isonume.dtd">
152
153   <!--MathML Aliases (From ISO PUB,DIA,NUM)-->
154   <!ENTITY % mmaliase.dtd SYSTEM "mmaliase.dtd">
155
156   <!--Greek Symbols-->
157   <!ENTITY % isogrk3e.dtd SYSTEM "isogrk3e.dtd">
158
159   <!--Math Script Font-->
160   <!ENTITY % isomscre.dtd SYSTEM "isomscre.dtd">
161
162   <!--Math Open Face Font-->
163   <!ENTITY % isomopfe.dtd SYSTEM "isomopfe.dtd">
164
165   <!--MathML Entities-->
166   <!ENTITY % mmlent.dtd SYSTEM "mmlent.dtd">
167
168   <!--Main MathML DTD -->
169   <!ENTITY % mathml.dtd    SYSTEM "mathml.dtd">
170
171   %mathml.dtd;
172   %isoamsae.dtd;
173   %isoamsbe.dtd;
174   %isoamsce.dtd;
175   %isoamsne.dtd;
176   %isoamsoe.dtd;
177   %isoamsre.dtd;
178   %isoteche.dtd;
179   %isonume.dtd;
180   %mmaliase.dtd;
181   %isogrk3e.dtd;
182   %isomscre.dtd;
183   %isomopfe.dtd;
184   %mmlent.dtd;
185   <!ENTITY aacute "&#x00E1;">
186   <!ENTITY OverBar "[OverBar]">
187   <!ENTITY negationslash "/">
188
189
190   <!-- end of latex.dtd -->
```

A.2 Примеры скриптов для techexplorer'a

A.2.1 teched.html

```
1   <HTML>
2   <!-- teched.html -->
3   <!-- (C) Copyright 1998 by Robert S. Sutor. All rights reserved. -->
4   <HEAD>
5      <META HTTP-EQUIV="Content-Type"
6           CONTENT="text/html; charset=iso-8859-1">
7      <META NAME="GENERATOR"
```

```
8           CONTENT="Mozilla/4.01 [en] (Win95; I) [Netscape]">
9       <TITLE>teched Sample LaTeX Editor</TITLE>
10  </HEAD>
11
12  <!-- This is a very simple LaTeX editor built using the     -->
13  <!-- IBM techexplorer Hypermedia Browser and a Java applet. -->
14
15  <BODY>
16  <CENTER>
17
18  <!-- The upper window is controlled by techexplorer. We  -->
19  <!-- give the name 'teInput' to this window. We are      -->
20  <!-- using a table to put a frame around the window.     -->
21
22  <TABLE BORDER=1>
23      <TR>
24          <TD>
25              <EMBED TYPE="application/x-techexplorer"
26              TEXDATA="\(\)"
27              NAME="teInput" WIDTH=600 HEIGHT=150>
28          </TD>
29      </TR>
30  </TABLE>
31
32  <!-- The lower window is handled by the 'teched' Java    -->
33  <!-- applet. Like the techexplorer window, it is 600     -->
34  <!-- pixels wide.                                        -->
35
36  <TABLE BORDER=1>
37      <TR>
38          <TD>
39              <APPLET CODE="teched.class"
40              NAME="teched" ALIGN=CENTER
41              WIDTH=600 HEIGHT=130 MAYSCRIPT></APPLET>
42          </TD>
43      </TR>
44  </TABLE>
45  </CENTER>
46  </BODY>
47  </HTML>
```

A.2.2 teched.java

```
1   // teched.java
2   // (C) Copyright 1998 by Robert S. Sutor. All rights reserved.
3
4   // We first import the classes we need from the standard Java
5   // distribution. This will work with Java 1.0 or higher.
6
7   import java.awt.*;
8   import java.awt.event.*;
9   import java.lang.*;
10  import java.applet.Applet;
11
12  // The following brings in the Netscape LiveConnect classes
13  // that we will use.
14
15  import netscape.javascript.JSObject;
16
17  // These are the classes that we use that are exposed by
18  // techexplorer. The first is the basic interface to the
19  // plug-in. The others are the event and listener classes.
20
21  import ibm.techexplorer.plugin.techexplorerPlugin;
22  import ibm.techexplorer.awt.AWTEvent;
```

```
23  import ibm.techexplorer.awt.event.KeyListener;
24  import ibm.techexplorer.awt.event.KeyEvent;
25
26  public class teched
27      extends java.applet.Applet
28      implements KeyListener
29  {
30      // The JavaScript window object
31      JSObject           Window           = null;
32      // The JavaScript document object
33      JSObject           Document         = null;
34
35      // The techexplorer plug-in instance
36      techexplorerPlugin tePlugin         = null;
37      // The editable text area for the markup source
38      TextArea           markupInputArea  = null;
39      // The 'Clear input' button
40      Button             clearInputButton = null;
41
42      // A utility buffer for holding the markup.
43      StringBuffer       markupString     = new StringBuffer("");
44
45      public boolean action(Event evt, Object arg) {
46          // We only handle the 'Clear input' action.
47
48          boolean result = false;
49
50          if ( evt.target == clearInputButton ) {
51              // Empty the markup edit area.
52              markupInputArea.setText( "" );
53
54              // Reinitialize the techexplorer document. This
55              // needs to be a non-empty string to actually
56              // updated the document, so we give it some
57              // non-visible input.
58
59              tePlugin.reloadFromTeXString( "\\(\\)" );
60
61              result = true;
62          }
63
64          return result;
65      }
66
67      public void init() {
68          // Initialize the components we are displaying
69          // with this Java applet.
70
71          clearInputButton  = new Button("Clear input");
72          markupInputArea   = new TextArea( 5, 80 );
73
74          this.setLayout( new FlowLayout() );
75          this.add( markupInputArea );
76          this.add( clearInputButton );
77      }
78
79      public void keyPressed( ibm.techexplorer.awt.event.KeyEvent e ) {
80          // We don't do anything with this event given us by
81          // techexplorer. But see 'keyTyped'.
82      }
83
84      public void keyTyped( ibm.techexplorer.awt.event.KeyEvent e ) {
85          // This is a naive (but effective!) way of dealing with
86          // keys coming to us from techexplorer. We grab the key
87          // that was pressed and put it on the end of our markup.
88          // Then we update the techexplorer window.
89
```

```
90          if ( e.getSource() == tePlugin ) {
91              markupInputArea.appendText(
92                  ( new Character( e.getKeyChar() )).toString() );
93
94              // This replaces the document within the techexplorer
95              // window with that gotten by parsing the string
96              // passed to it.
97
98              tePlugin.reloadFromTeXString( markupInputArea.getText() );
99          }
100     }
101
102     public void keyReleased( ibm.techexplorer.awt.event.KeyEvent e ) {
103         // This is where we deal with key release events coming to
104         // us from the techexplorer window.
105
106         switch ( e.getKeyCode() ) {
107           case KeyEvent.VK_DELETE:
108             // When we see a 'delete' key, we remove the last character
109             // in the markup.
110
111             if ( e.getSource() == tePlugin ) {
112                 markupString = new StringBuffer( markupInputArea.getText() );
113                 int length = markupString.length();
114                 if ( length > 0 )
115                     --length;
116                 markupString.setLength( length );
117                 markupInputArea.setText( markupString.toString() );
118                 tePlugin.reloadFromTeXString( markupInputArea.getText() );
119             }
120             break;
121
122           case KeyEvent.VK_ENTER:
123             // When we see that the 'enter' key has been pressed, we
124             // insert a newline in the markup. This improves readability.
125
126             if ( e.getSource() == tePlugin ) {
127                 markupInputArea.appendText( "\n" );
128                 tePlugin.reloadFromTeXString( markupInputArea.getText() );
129             }
130             break;
131
132           default:
133             break;
134         }
135     }
136
137     public boolean keyUp( Event evt, int key )
138     {
139         // This key is one from the markup input area.
140         // When a key is released, update the techexplorer
141         // document with the current markup.
142
143         boolean result = false;
144
145         if ( evt.target == markupInputArea ) {
146             if ( evt.id == Event.KEY_RELEASE ) {
147                 int length = markupInputArea.getText().length();
148                 if ( length > 0 )
149                     tePlugin.reloadFromTeXString(
150                         markupInputArea.getText() );
151                 else
152                     tePlugin.reloadFromTeXString( "\\(\\)" );
153             }
154
155             result = true;
156         }
```

```
157
158         return result;
159     }
160
161     public void start() {
162         // Initialize the Netscape JavaScript objects.
163
164         Window   = (JSObject) JSObject.getWindow(this);
165         Document = (JSObject) Window.getMember("document");
166
167         // Try to get the techexplorer plug-in object.
168         tePlugin = (techexplorerPlugin) Document.getMember("teInput");
169
170         if ( tePlugin == null )
171             // If we didn't get it, print a debug message.
172             System.out.println("teched: start(): null teched");
173         else
174             // Otherwise add the listener for techexplorer keys.
175             tePlugin.addKeyListener( (KeyListener) this );
176     }
177
178     public void stop() {
179         if ( tePlugin == null )
180             // If we don't have the techexplorer plug-in object,
181             // print a debug message.
182             System.out.println("teched: stop(): null teched");
183         else
184             // Otherwise remove the listener for techexplorer keys.
185             tePlugin.removeKeyListener( (KeyListener) this );
186     }
187 }
```

Приложение B

Технические приложения

B.1 Стандарт HyperTEX

Важной инициативой в области стандартизации для TEX'а и гипертекста является проект HyperTEX [↪HYPERTEX]. Он реализовывался Тэнмоем Бхаттачариа и Марком Дойлом для электронного архива публикаций [↪EPRINT] в Лос-Аламосской Национальной лаборатории США (Los Alamos National Laboratory) под руководством Поля Гинспарга. В спецификации HyperTEX говорится, что совместимые программы просмотра и трансляторы должны распознавать следующий набор конструкций типа `\special`:

```
href    html:<a href = "href_string">
name    html:<a name = "name_string">
end     html:</a>
image   html:<img src = "href_string">
base    html:<base href = "href_string">
```

Команды `href`, `name` и `end` используются для выполнения основных гипертекстовых операций — установления ссылок между разделами документа. Команда `image` предназначена (как и в HTML-навигаторах) для размещения изображения в некотором графическом формате в текущую позицию на данной странице. Команда `base` используется для передачи программе просмотра DVI-файлов полной информации (URL) о местоположении текущего обрабатываемого документа, что позволяет правильно обращаться к файлам, заданным с помощью относительных URL.

Командам `href` и `name` должны отвечать парные им команды `end`, помещенные в TEX'овском файле ниже их, — TEX'овские команды, находящиеся между двумя командами `end` таких пар, образуют *элемент привязки* в рассматриваемом документе. В случае команды `href`, *элемент привязки* должен подсвечиваться в программе просмотра DVI. Если по этому подсвеченному элементу щелкнуть мышкой, это приведет к переходу в место документа, задаваемое строкой *href_string*. *Элемент привязки*, связанный с командой `name`, представляет собой возможное место, на которое имеют право ссылаться другие гипертекстовые ссылки либо как локальные ссылки (в форме `href="#`*name_string*`"` с атрибутом *name_string*, идентичным атрибуту в данной команде `name`), либо как часть некоторого URL (в форме *URL*`#`*name_string*).

Здесь *href_string* есть корректное значение некоторого URL или локальный идентификатор, тогда как *name_string* вообще может быть любой строкой: единственное ограничение состоит в том, что символам " должна предшествовать обратная косая черта (\); если это выглядит подобно имени URL, могут возникнуть недоразумения.

Команды \special в HyperTEX'е реализованы (по крайней мере) в следующих DVI-драйверах: xdvi для операционной системы UNIX, dviout и dviwindo для Windows, а также OzTeX и Textures для Macintosh. Программа dvips также поддерживает эти команды и может транслировать их в форму, пригодную для преобразования в PDF.

B.2 Настройка конфигурации TEX4ht для формирования вывода в формате XML

Основная идея LATEX'а состоит в организации интерфейсов для идентификации структурных элементов в документе, а затем в связывании с каждым из этих элементов соответствующих форм отображения. При разработке SGML и производных от него языков ставилась цель обобщить такого рода идею, с тем чтобы упомянутые выше интерфейсы представлялись в форматах, пригодных для последующей обработки. Основная цель программы TEX4ht — предложить средство, которое может распознавать интерфейсы, основанные на TEX'е вообще и на LATEX'е в частности, а также преобразовывать эти интерфейсы в представления на основе SGML.

Реализуя этот подход, как уже было показано в гл. 4, TEX4ht передает «родному» компилятору TEX'а задачу обработки исходных файлов, освобождая себя от многочисленных малоприятных забот, связанных с оформлением структурных элементов документов. Чтобы выявить структуру некоторого документа, TEX4ht «засевает» стилевые файлы «зацепками» в критически важных точках внутри макроопределений, а затем предлагает значения для конфигурирования этих зацепок, чтобы сформировать вывод в одной из форм, задаваемых пользователем. Это означает, что имеется возможность создавать преобразования в различные представления, основанные на SGML.

В разделе 4.5 на с. 205 было показано, как можно задать конфигурацию этих зацепок. Теперь опишем некоторый дружественный интерфейс, позволяющий задавать и настраивать зацепки, а затем более глубоко рассмотрим низкоуровневые средства, связанные с TEX'овским компилятором.

Читателю предлагается ознакомиться с наглядной демонстрацией данного интерфейса, обсуждаемого в следующих двух разделах, что лучше всего сделать, если фактически исполнить на компьютере различные этапы данного примера.

B.2.1 Начнем сначала

Быстрое развитие HTML при переходе от версии к версии в последние годы, а также появление языка XML показали, насколько важна возможность внесения обширных модификаций в вывод программы TEX4ht, чтобы удовлетворить

вновь возникающим требованиям. Рассмотрим, каким образом осуществить возврат от TEX4ht к исходному тексту, что дает возможность построить новый конвертер.

B.2.1.1 Загрузка пустых конфигураций

Опция `0.0` рассматриваемого пакета запрашивает для зацепок пустые конфигурации, что приводит к получению вывода в виде текста, не содержащего гипертекстовых элементов. Например, показанный ниже исходный документ выводит просто текст `1 Demo With A List` безо всяких элементов:

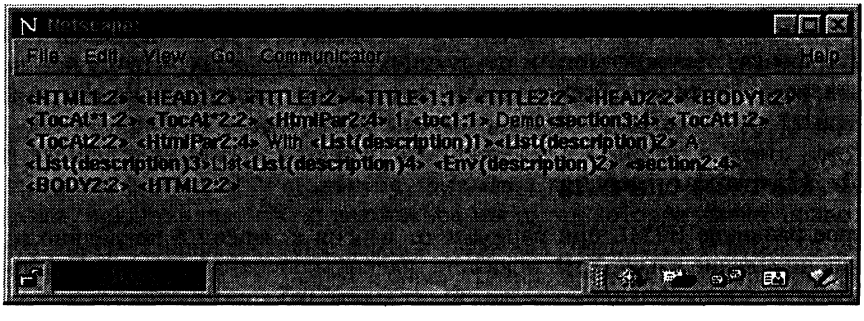
```
\documentclass{article}
  \usepackage[html,0.0]{tex4ht}
\begin{document}
\section{Demo}
With
  \begin{description}
    \item[A] List
  \end{description}
\end{document}
```

Будем иметь в виду этот простой пример при чтении следующих нескольких разделов, чтобы продемонстрировать влияние обсуждаемых средств.

B.2.1.2 Просмотр зацепок

Основная проблема, связанная с изменением конфигурации зацепок, используемых TEX4ht, состоит в том, что надо выяснить, где они находятся и что могут делать. Опция `hooks` дает хорошую возможность запросить для начала *псевдогипертекстовые элементы* для зацепок, которые не имеют других конфигураций. Такого рода *псевдогипертекстовый элемент* показывает имя зацепки, индекс аргумента внутри зацепки и, если это не очевидно, число аргументов для данной зацепки.

Компиляция исходного текста рассматриваемого примера документа с опцией пакета `html,0.0,hooks` приводит к получению вывода, который в навигаторе выглядит следующим образом:



В данном примере элемент <HTML1:2> будет получен из первого аргумента зацепки "HTML". Конфигурация этой зацепки задается следующей командой:

\Configure{HTML} {*first-argument*} {*second-argument*}.

Элемент <Env(description)2> берется из второго аргумента зацепки для окружения description, конфигурация его устанавливается командой

\ConfigureEnv{description} {*first-argument*} {*second-argument*} {*third-argument*} {*fourth-argument*}

Дескриптор <List(description)4> извлекается из четвертого аргумента зацепки для списка description и конфигурируется командой

\ConfigureList{description} {*first-argument*} {*second-argument*} {*third-argument*} {*fourth-argument*}

B.2.1.3 Отображение зацепок

Поскольку используемые сейчас программы-навигаторы пригодны для отображения вывода, получаемого в результате применения опции hooks, в рассматриваемом файле опущены соответствующие HTML-коды в начале и конце данного файла. Такого рода код можно очень просто ввести, задавая соответствующим образом конфигурацию зацепки "HTML". Можно также задать специальные характеристики элементов внутри рассматриваемого вывода-отображения, с тем чтобы привлечь внимание читателя к их использованию. Внешний вид допускается модифицировать в пределах файла CSS, а также путем изменения следующей конфигурации по умолчанию:

```
\Configure{hooks}
  {\HCode{<STRONG CLASS="hooks">&lt;}} {\HCode{&gt;</STRONG>}} {}{}
```

Если ввести файл конфигурации try.cfg так, как это показано ниже, и загрузить его со списком опций try,html,0.0,hooks, HTML становится корректным, а все элементы будут отображаться в зеленом цвете.

```
\Configure{HTML}{\StartHtml}{\EndHtml}
\newcommand\StartHtml{\IgnorePar\HCode{<!DOCTYPE HTML PUBLIC
                "-//W3C//DTD HTML 4.0 Transitional//EN">\Hnewline
                     <HTML><HEAD><LINK REL="stylesheet"
                TYPE="text/css" HREF="\jobname.css">\Hnewline
            <TITLE>\jobname</TITLE></HEAD>\Hnewline <BODY>}
          \Css{.hooks{color:green;}}}
\newcommand\EndHtml{\HCode{</BODY></HTML>}}
\Preamble{}
\begin{document}
\EndPreamble
```

В.2.1.4 Изменение конфигурации зацепок

Начиная с вывода, который состоит из элементов *псевдогипертекста*, можно постепенно установить конфигурацию различных зацепок таким образом, чтобы получить вывод в том виде, который необходим. При выполнении этих действий элементы, не представляющие интереса, можно удалить путем присваивания соответствующим их параметрам непустого невидимого (неотображаемого) кода.

Установкой конфигурации зацепок "BODY", "HEAD", "TITLE+", "TITLE", "TocAt*", "TocAt" и "toc" так, чтобы добавить \empty к их параметрам, рассматриваемый пример сводится к следующему:

```
1 Demo<section3:4> With
   <List(description)1> <List(description)2> A
   <List(description)3> List <List(description)4> <Env(description)2>
   <section2:4>
```

Задавая далее очевидные значения для остающихся зацепок, получим следующую картину:

```
[DOC] [SECTION][NUM]1[/NUM][TITLE]Demo[/TITLE] With [DLIST]
[MARK] A [/MARK]List [/DLIST] [/SECTION][/DOC]
```

Файл конфигурации, который приводит к получению вывода такого вида, выглядит следующим образом:

```
\Configure{HEAD}{\empty}{\empty}
\Configure{HTML}
  {\IgnorePar\HCode{<HTML><HEAD> <TITLE></TITLE></HEAD><BODY>}[DOC]}
  {[/DOC]\HCode{</BODY></HTML>}}
\Preamble{}
   \Configure{BODY}{\empty}{\empty}
   \Configure{TITLE}{\empty}{\empty}
   \Configure{TITLE+}{\empty}
   \Configure{TocAt}{\empty}{\empty}
   \Configure{TocAt*}{\empty}{\empty}
   \Configure{toc}{\empty}
   \ConfigureEnv{description} {[DLIST]}{[/DLIST]}{}{}
   \ConfigureList{description}{}{}{[MARK]}{[/MARK]}
   \Configure{section}   {[SECTION]}{[/SECTION]}
      {[NUM]\arabic{section}[/NUM][TITLE]}{[/TITLE]}
\begin{document}
\EndPreamble
\Configure{HtmlPar}{}{\empty}{}{}
```

В.2.1.5 Завершаем начатую работу

Завершив настройку конфигурации зацепок, представляющих интерес, можно удалить тот код, который был введен ранее для улучшения получаемого вывода, а затем удалить опцию hooks в команде \usepackage.

B.2.2 Добавление элементов XML

Язык XML хорошо подходит для демонстрации того, каким образом осуществить реконфигурацию TEX4ht. Рассмотрим, как добавить для этой цели некоторые элементы XML в формируемый вывод. До тех пор пока вновь составленная конфигурация зацепок не содержит элементов HTML, программы-навигаторы не будут встречать затруднений с отображением результатов произведенных присваиваний.

Программа TEX4ht предоставляет следующие три команды:

```
\Tg<name>
\Tg</name>
\Tg<name/>
```

для получения открывающей метки, закрывающей метки и пустого XML-элемента соответственно. Опция пакета `edit` заставляет эти новые команды порождать видимые (отображаемые) формы рассматриваемых элементов посредством введения кода для *отображения* данных элементов вместо фактического *создания* их.

Например, если заменить [*name*] командой вида `\Tg<`*name*`>` в файле конфигураций для рассматриваемого примера, в HTML-навигаторе получим такую картину:

```
<DOC> <SECTION> <NUM> 1</NUM><TITLE>Demo</TITLE>
With <DLIST> <MARK> A </MARK>List </DLIST> </SECTION> </DOC>
```

при этом был использован следующий конфигурационный файл:

```
\Configure{HTML} {\Tg<DOC>}{\Tg</DOC>}
\Preamble{}
  \ConfigureEnv{description} {\Tg<DLIST>}{\Tg</DLIST>}{}{}
  \ConfigureList{description}{}{}{\Tg<MARK>}{\Tg</MARK>}
  \Configure{section}   {\Tg<SECTION>}{\Tg</SECTION>}
      {\Tg<NUM>\arabic{section}\Tg</NUM>\Tg<TITLE>}{\Tg</TITLE>}
\begin{document}
\EndPreamble
```

Вид по умолчанию для видимых «элементов» задается командой

```
\Configure{edit}{\HCode{<STRONG>&lt;}}{\HCode{&gt;</STRONG>}}
```

Удаление опции `edit` позволяет получить искомый XML-документ.

B.2.2.1 Печать абстрактных элементов

Назначение элементов XML — идентифицировать логические элементы в пределах документа. На этапе редактирования, когда и вводятся эти элементы, может быть полезно выявить природу различных элементов, а также структурные отношения между ними.

Для этого метки `\Tg` могут быть усилены с помощью команд

\Configure<*name*>{*before*}{*after*}
\Configure</*name*>{*before*}{*after*}
\Configure<*name*/>{*before*}{*after*}

которые позволяют получить требуемый вид вывода.

Добавление команд наподобие

```
\Configure<NUM>{\HCode{<DIV ALIGN="CENTER">}}{}
\Configure</TITLE>{}{\HCode{</DIV>}}
\Configure<DLIST>{\HCode{<BR>}}{}
\Configure</DLIST>{}{\HCode{<BR>}}
```

в конфигурационный файл приводит к получению такого вывода в программе-навигаторе:

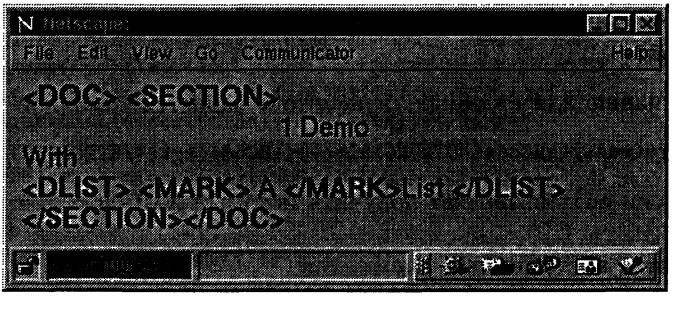
```
<DOC> <SECTION>
     <NUM>1</NUM><TITLE>Demo</TITLE>
With
<DLIST> <MARK> A </MARK>List </DLIST>
</SECTION></DOC>
```

Если первым аргументом является символ переноса -, команды конфигурации сформируют в точности второй аргумент, не показывая при этом собственно метку. В этом случае команды конфигурации определяют некоторый виртуальный навигатор для рассматриваемых новых элементов.

Рассмотрим следующие варианты команд конфигурации:

```
\Configure<NUM>-{\HCode{<DIV ALIGN="CENTER">}}
\Configure</NUM>-{ }
\Configure<TITLE>-{}
\Configure</TITLE>-{\HCode{</DIV>}}
```

Исполнение этих команд приведет к выдаче навигатором на экран монитора следующей картинки:



B.2.2.2 Контроль отношений включения

Опция `verify` требует выдачи предупреждающих сообщений для случая, когда среди элементов, определяемых командой `\Tg`, обнаруживаются неизвестные

отношения включения типа «предок-потомок». Отношения включения могут быть заданы с помощью списка пар имен предков (родительских элементов) и потомков (дочерних элементов) с помощью строки --> между командами

```
\Verify
\EndVerify
```

Различные пары внутри этого перечня (списка) должны быть разделены запятыми. Тогда, если

```
\Verify --> DOC, DOC --> SECTION \EndVerify
```

помещается до \begin{document} в конфигурационном файле try.cfg, а также используется опция пакета verify, будут получены сообщения-предупреждения такого вида:

```
4. --- warning --- SECTION --> NUM ?
4. --- warning --- SECTION --> TITLE ?
4. --- warning --- SECTION --> DLIST ?
4. --- warning --- DLIST --> MARK ?
```

Мы задали только то, что появление элементов SECTION ожидается внутри области действия элемента DOC. Система информирует, что внутри DLIST найдена метка MARK.

Опция пакета verify+ представляет собой функционально расширенный вариант опции verify, в которой обнаруженные отношения включения детально описываются системой в файле протокола.

Аналогично этому, опция пакета hooks+ есть расширенный вариант опции hooks. Кроме того, что она указывает на необходимость вывода информации о том, как используются параметры зацепки, эта опция запрашивает также выдачу в файл протокола информации о том, какая конфигурация зацепок задана. Это полезно, когда просматриваются примеры конфигураций, которые могут помочь определить новые конфигурации. Обычно такого рода примеры состоят из нескольких файлов, и необходимая информация может оказаться затерянной среди не относящегося к делу кода.

Команда \Tg<*argument*> представляет собой специальный случай команды \HCode{<*argument*>}, которая дает возможность на этапе редактирования показать элементы и проверить, удовлетворяют ли они соответствующим отношениям включения. Более слабый вариант этой команды — \TG — предлагается для документов, которые нет необходимости проверять с точки зрения отношений включения. С другой стороны, имеется вариант \tg команды \TG, отвечающий аргументу <*argument*>, взамен команды \HCode{<*argument*>}.

B.2.3 Более подробное рассмотрение дополнительных конфигураций

До настоящего момента наше внимание было сосредоточено на структурных средствах, которые применяются главным образом к достаточно крупным структурным элементам документа, таким, как главы, списки и таблицы. Эти объекты обычно определяются с помощью стилевых файлов; они на самом деле не зависят от низкоуровневых средств, обеспечиваемых TEX'овской наборной программой. Теперь надо познакомиться с некоторыми из средств, доступных в TEX4ht для работы с этими низкоуровневыми командами.

B.2.3.1 Математический режим

Средства набора математических формул составляют значительную часть низкоуровневых средств TEX'a. По умолчанию программа TEX4ht настроена так, что работа с математическими формулами, которые могут иметь сложную структуру, осуществляется как с графическими образами. Опция данного пакета `math` включает альтернативный набор настроек, в котором рассматриваемая формула «засевается» зацепками, конфигурацию которых можно установить так, чтобы задать измененный вариант представления формулы при ее выводе.

Конфигурацию математических окружений \(*formula*\), \[*formula*\], $*formula*$ и $$*formula*$$ можно задать с помощью следующих команд:

```
\Configure{()}{before£at-start}{at-end£after}
\Configure{[]}{before££at-start}{at-end££after}
\Configure{$}{before}{after}{at-start}
\Configure{$$}{before}{after}{at-start}
```

Конфигурация для последней пары окружений может быть применена также для окружений `math` и `displaymath` соответственно.

[equation]a + b[/equation]

```
\Configure{()}{[equation]$}{$[/equation]}
\(a+b\)
```

Некоторые из рассматриваемых средств требуют при их употреблении, чтобы программа TEX4ht была переведена в специальный режим функционирования. Для этой цели можно использовать команды

```
\DviMath
\EndDviMath
```

B.2.3.2 Математические классы для символов

На самом нижнем уровне математическое выражение хранится как набор примитивов-лексем, таких, как числа, переменные, знаки операторов, скобки и т. д. На основе классификации математических символов, принятой в TEX'е, построена и аналогичная (но независимая) классификация в пакете TEX4ht.

А именно, математическим символам могут быть приписаны номера классов, представляющие собой числа в диапазоне от 0 до 9, где согласно начальной классификации большие операторы отнесены к классу 1, бинарные операторы — к классу 2, символы отношений — к классу 3, открывающие ограничители — к классу 4, закрывающие ограничители — к классу 5, знаки пунктуации — к классу 6, а остальные символы — к классу 0.

Для соотнесения выводимых меток с символами некоторого конкретного класса и добавления символа к данному классу можно воспользоваться следующей командой:

```
\Configure{MathClass}{class-number}{string}{before}{after}{symbols}
```

Если аргумент *string* непуст, он должен быть единичным символом, не появляющимся в аргументах *before* и *after*. В этом случае, упомянутые аргументы задают содержимое, которое следует вставить до и после символов параметра *class*.

Аргумент *symbols* должен представлять собой последовательность символов. Когда он непуст, символы, заданные в аргументе, присваиваются предписанному классу.

```
[ordinary]a[/ordinary]
[operator]+[/operator]
[digit]1[/digit]
```

```
\Configure{()}{\DviMath$}{$\EndDviMath}
\Configure{MathClass}{0}{*}
   {[ordinary]}{[/ordinary]}{}
\Configure{MathClass}{2}{*}
   {[operator]}{[/operator]}{}
\Configure{MathClass}{7}{*}
   {[digit]}{[/digit]}{0123456789}
\(a+1\)
```

Когда обрабатываются аргументы команды \HCode, параметры *before* и *after* обрабатываются только для макрорасширений, а не для шрифтов, определений и вычислений.

Команды

```
\PauseMathClass
\EndPauseMathClass
```

можно использовать для временного подавления действия различных классов.

B.2.3.3 Математические классы для ограничителей и слов

Символы, отвечающие открывающим и закрывающим ограничителям, группирующим лексемы для инструкций \Send{BACK} (разд. B.2.3.4), должны быть соотнесены с их классами с помощью команды:

```
\Configure{MathDelimiters}{open}{close}
```

Когда используется опция пакета `math`, значения параметров, принятые по умолчанию, приводят к исполнению команды \Configure{MathDelimiters}{()}{()}, которая соотносит символы скобок «(» и

«)» с математическими классами 4 и 5 соответственно, а также информирует программу TEX4ht относительно парности этих ограничителей.

Пакет TEX4ht реализует команды \mathord, \mathop, \mathbin, \mathrel, \mathopen, \mathclose и \mathpunc для связывания классов, соответственно от 0 до 6 включительно, с подформулами. Следующие ниже инструкции служат для настройки конфигурации этих команд, поскольку непрямым образом связывают упомянутые команды с их номерами классов:

```
\Configure{FormulaClass}{class-number}{string}{before}{after}
```

Если аргумент *string* не является пустым, он должен быть некоторой литерой, не появляющейся в аргументах-параметрах *before* и *after*. Если же параметр *string является* пустым (т. е. в качестве значения имеет пустую строку), тем самым задается, что конфигурация должна использоваться для символов с предписанным значением параметра *class-number*.

open[a + 1]close

```
\Configure{()}{\DviMath$}{$\EndDviMath}
\Configure{FormulaClass}{4}{*}{}{[}
\Configure{FormulaClass}{5}{*}{]}{}
\(\mathopen{open}a+1\mathclose{close}\)
```

B.2.3.4 Группы, основанные на DVI

Вывод из TEX'а описывается на низкоуровневом языке описания страниц DVI, который существенно основывается на механизме иерархического группирования элементов. Используемая иерархия отражает структуру, задаваемую в исходном документе явным или неявным образом. Иногда к этой информации необходимо иметь доступ, например при написании трансляторов.

```
\Trace{GROUP}
\EndTrace{GROUP}
\Configure{GROUP}{string}{open-1}{open-2}{close-1}{close-2}
```

Здесь первые две команды ограничивают область, в которой группы будут доступны (область видимости). Третья команда управляет способом отображения групп.

Аргументы *open-1* и *open-2* задают, что должно быть вставлено в начале групп и, если первый из этих аргументов не имеет в качестве значения пустую строку, включают также уровень вложенности и номер группы. Аргументы *close-1* и *close-2* аналогичным образом определяют информационное содержимое, которое должно быть вставлено в конце группы.

A[3 480][4 481] =[/4 481] [4 482] [/4 482][/3 480] B

```
\Configure{GROUP}{*}{[}{]}{[/}{]}

\noindent \Trace{GROUP}
\(A \stackrel{\string~}{=} B\)
\EndTrace{GROUP}
```

Следующие две команды дают возможность указывать содержимое для входной и выходной точек рассматриваемой текущей группы. Рассматриваемые группы идентифицируются с помощью их уровня вложенности, на котором появилась рассматриваемая команда. Уровень 0 относится к группе, которая непосредственно включает команду \Send. Более высокие уровни относятся к группам, которые следуют за данной командой.

\Send{GROUP}{*level*}{*content*}
\Send{EndGROUP}{*level*}{*content*}

```
\Configure{()}{\DviMath
    \Send{GROUP}{1}{[1]}
    \Send{EndGROUP}{1}{[/1]}
    \Send{GROUP}{2}{[2]}
    \Send{EndGROUP}{2}{[/2]}
  $}{$\EndDviMath}
\(A \stackrel{\string~}{=} B\)
```

A[1][2] =[/2] [/1] B

Команду

\Send{BACK}{*content*}

разрешается использовать, чтобы послать содержимое в обратном направлении через символ, оказавшийся самым последним, группу или область, заключенную между ограничителями, определенными как пара командой \Configure{MathDelimiters}{*open*}{*close*}.

B.2.3.5 Нижние и верхние индексы

Имеется несколько альтернативных способов задать конфигурацию нижних и верхних индексов. Все они основываются на инструкциях \Send{BACK}{*content*}, кроме того, надо задать еще и базу для расстановки этих индексов.

\Configure{SUB}{*before*}{*after*}
\Configure{SUP}{*before*}{*after*}
\Configure{SUBSUP}{*before*}{*between*}{*after*}
\Configure{SUPSUB}{*before*}{*between*}{*after*}
\Configure{SUB/SUP}{*before-1*}{*between-1*}{*after-1*}
{*before-2*}{*between-2*}{*after-2*}

Здесь первая пара команд соответствует случаям, когда нижние и верхние индексы появляются изолированно.

Третья команда относится к случаю соседних индексов, но только если нижние индексы всегда появляются в выводе перед верхними индексами, независимо от порядка, в котором они находились в исходном тексте. Четвертая команда представляет собой разновидность третьей, для случая, когда нижние индексы всегда появляются после верхних индексов. Пятая команда — это

еще один вариант, который требует сохранить в выводе тот порядок нижних и верхних индексов, который они имели в исходном тексте.

Аргументы *before-1*, *between-1* и *after-1* применяются, если нижний индекс предшествует верхнему, а аргументы *before-2*, *between-2* и *after-2* — когда верхний индекс идет первым.

```
\Configure{()}{\DviMath$}{$\EndDviMath}
\Configure{SUBSUP}
   {\Send{BACK}{[base]}[/base][sb]}
   {[/sb][sp]}
   {[/sp]}
\(a^b_c\)
```

[base]a[/base][sb]c[/sb][sp]b[/sp]

B.2.3.6 Акценты

Пакет TEX4ht работает с акцентами путем просмотра таблицы, формируемой при исполнении следующей команды:

```
\Configure{accent}\textcmd\mathcmd
{{in-1}{out-1}{in-2}{out-2}...{}{out-last}}
{do-found} {do-not-found}
```

Пары {*in-1*}{*out-1*}, {*in-2*}{*out-2*},... формируют наборы данных для рассматриваемого акцента. Первое поле последней пары полей данных должно быть пустым.

Таблица сопоставляется команде формирования акцента в набираемом тексте *textcmd* и соответствующей команде для формирования математического акцента *mathcmd*.

Когда в тексте исходного документа встречается команда формирования акцента, в таблице для этого акцента производится поиск записи, первое поле которой совпадает с аргументом исполняемой команды. Если такая запись найдена, активизируется параметр *do-found*. В этом случае второе поле найденной пары становится доступным как макропараметр #1.

Если пары, удовлетворяющей условиям поиска, не найдено, активизируется параметр *do-not-found*. В таком случае параметр #1 представляет команду формирования акцента в том варианте, который имеется в LATEX'е, а параметр #2 представляет собой аргумент данной команды.

```
\Configure{accent}\^\hat{a{a}{}{}}
    {[hat]#1[/hat]}
    {[HAT]#2[/HAT]}
\Configure{accent}\vec\vec{{}{}}
    {} {[vec]#2[/vec]}
$\vec a=(\hat a,\hat b)$
```

[vec]a[/vec] = ([hat]a[/hat] , [HAT]b[/HAT])

Если второй аргумент пуст, используется предыдущая таблица для команд формирования акцентов. Аналогично, если пусты и параметр *do-found*, и параметр *do-not-found*, будет использован предыдущий фрагмент кода.

B.3 Пространства имен XML

Документ на языке XML может включать в себя элементы и атрибуты, ссылающиеся на многие программные модули, каждый из них со своим словарем. Это значит, что одни и те же имена могут использоваться одним или несколькими из этих модулей с разными целями; это приведет к конфликту имен. Следовательно, в высшей степени желательно располагать механизмом, который осуществлял бы управление *видимостью имен*, т. е. позволял бы указывать, на какое из пространств имен ссылается тот или иной тип элемента или атрибут.

Консорциум W3C разрешает эту проблему в рекомендации «Namespaces in XML» [↪XMLNS]. Этот документ определяет *пространство имен XML* как набор имен, идентифицируемый с помощью унифицированного идентификатора ресурсов (URI), которые используются в XML-документах в качестве типов элементов и имен атрибутов.

Имена из пространства имен XML состоят из *префикса пространства имен (namespace prefix)*, который отображается в URI, чтобы выбрать пространство имен[1]; за этим префиксом следует двоеточие, а за ним *локальная часть (local part)*. Поскольку URI может содержать символы, недопустимые в составе имен, надо использовать какой-либо транслятор, который ассоциирует префикс пространства имен и данный URI. Такому сокращению придается префикс в виде строки `xmlns`. Одним из примеров такого рода является стиль XSL, который обсуждался в разд. 7.6.2.1, где содержались такие две декларации:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:fo="http://www.w3.org/1999/XSL/Format"
```

Здесь виден префикс `xmlns`, за которым идет сокращенное имя для пространства имен, которое будет использоваться в документе, чтобы задать области видимости различных типов элементов и атрибутов. Например, в том же самом документе можно обнаружить конструкции вида:

```
1   <xsl:template match="par">
2     <fo:block indent-start="10pt" space-before="12pt">
3       <xsl:apply-templates/>
4     </fo:block>
5   </xsl:template>
```

Строки 1 и 5 ссылаются на пространство имен `xsl`, задаваемое первой из двух деклараций для элемента типа `xsl:stylesheet`, определенной выше. Строки 2 и 4 ссылаются на фрагмент XSL-документа с форматирующими объектами, как это определено второй декларацией элемента `xsl:stylesheet` (для пространства имен `fo`). Фрагмент имени, перед которым стоит двоеточие, называется *локальной* частью данного имени. Эта часть должна иметь значение

[1] Фактически этот "URI" является, строго говоря, только символьной строкой, которая идентифицирует уникальным образом некоторое пространство имен. От него не требуется соответствия какому-либо реальному документу в Интернете, хотя хорошей практикой будет задавать истинный URI, если таковой существует.

в рамках пространства имен, определяемого соответствующим URI. Префикс, определенный в рассматриваемом примере документа (например, префиксы `xsl` и `fo`, задававшиеся выше), не имеет никакого значения вне того документа, в котором он декларирован, поскольку он в явном виде представляет собой «заполнитель», т. е. заменяемый некоторым реальным элементом URI. По этой причине важно всегда экспортировать декларации для префикса вместе с элементами, которые используют их.

Необходимо заметить, что префикс `xml` (в любой комбинации больших и малых букв) зарезервирован для использования спецификациями, связанными с XML.

Имеется специальный случай, когда объявляется пространство имен, используемое *по умолчанию*; в этом случае опускаются двоеточие и префикс пространства имен. Тогда типы элементов без спецификатора пространства имен ссылаются на упомянутое пространство имен, принятое по умолчанию. Например, строка 2 в приведенном ниже примере декларирует, что пространством имен по умолчанию для рассматриваемого документа будет HTML4, как это определено в спецификации HTML4, которую можно найти по данному URI.

```
1  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
2                  xmlns="http://www.w3.org/TR/REC-html40"
```

В примере стиля можно тогда прямо записать:

```
1  <xsl:template match="invitation/par/emph">
2  <em><xsl:process-children/></em>
3  </xsl:template>
```

где дескрипторы `<em>` в строке 2 принадлежат «чистому» HTML.

Пространство имен, принятое по умолчанию, может быть перекрыто.

```
1  <?xml version='1.0'?>
2   ...
3   <particles>
4     <!-- Default namespace is set to HTML -->
5     <table xmlns='http://www.w3.org/TR/REC-html40'>
6      <tr><td>Particle</td><td>Mass</td><td>Details</td></tr>
7      <tr>
8        <td><em>neutron</em></td>
9        <td>939.56 MeV</td>
10       <td>
11         <pdg xmlns="mypdgnamespace">
12          <!-- HTML namespace is no longer used -->
13          <quarks>udd</quarks><lifetime>886.7 s</lifetime>
14          <decay>p,e,anue</decay>
15         </pdg>
16       </td>
17     </tr>
18    </table>
19  </particles>
20  ...
```

Начиная с элемента `table` (строка 5), в качестве значения по умолчанию используется пространство имен HTML 4 до тех пор, пока не будет совершен переход к данным в третьем столбце (строка 10). Здесь пространство имен по

умолчанию для элемента pdg и его дочерних элементов уже не будет более иметь значение HTML, но будет изменено на mypdgnamespace (строка 12). Как только мы покидаем элемент pdg (строка 15), пространством имен по умолчанию автоматически опять становится HTML.

Правила видимости, определяемые пространством имен, распространяются не только на типы элементов; его можно также использовать для атрибутов и DTD-деклараций.

B.4 Примеры сложных определений DTD

Для того чтобы лучше понять и представить, как выглядит определение типа документа DTD для более сложных документов, чем рассматривавшиеся выше, в данном разделе приведем несколько достаточно представительных примеров. Рассмотрение это начнем с того, что будет сказано несколько слов относительно DTD для DocBook, ISO-12083, TEI и HTML. Затем обсудим подробнее, как формировать DTD для документов BIBTEX'а и LATEX'а.

B.4.1 DTD для DocBook

Определение DocBook DTD было разработано специально для подготовки документации на компьютерное программное обеспечение, такой, как руководства пользователя и справочники программиста. Первоначально DocBook поддерживался и сопровождался Davenport Group [↪DAVENPORT], представлявшей собой дискуссионный форум, который спонсировался отдельными представителями из организаций, связанных с разработкой крупных программных комплексов, а также пользователей таких комплексов, работающих с соответствующей документацией. После июля 1998 г. эксплуатационное сопровождение DocBook осуществляет комитет DocBook Technical Committee, который действует под эгидой OASIS. DocBook был принят для инициативы, именуемой Linuxdoc-SGML, которая использует пакет SGML-Tools [↪SGMLTOOLS]. Ранее Linuxdoc-SGML основывалась на qwertz DTD, упоминаемом также в разд. B.4.5.

Найти и получить DocBook DTD можно по адресу [↪DOCBOOK]. Это полное определение SGML DTD, которое уже использовалось многими поставщиками документации для разметки соответствующих руководств. Совсем недавно Норман Уолш полностью переделал данное DTD, чтобы добиться его XML-совместимости [↪DBXML].

Определение DocBook DTD использует модель «книги» для документов, подготавливаемых с его помощью. Книга формируется из элементов типа book, таких, как предисловия, главы, приложения и глоссарии. Можно использовать до пяти уровней разделов, в которых могут содержаться абзацы, списки, элементы для формирования указателей, перекрестные ссылки и ссылки на литературу.

```
book
  meta information
```

```
chapter
  sect1
    sect2
  sect1
chapter
  sect1
appendix
  sect1
appendix
  sect1
  ...
glossary
```

В DTD остается некоторое пространство для локализаций. Пользователь данного DTD свободен в выборе вариантов модели содержимого для приложений, глав, уравнений, указателей и т. п.

```
1  <!ENTITY % local.appendix.class "">
2  <!ENTITY % appendix.class "appendix %local.appendix.class;">
3
4  <!ENTITY % local.chapter.class "">
5  <!ENTITY % chapter.class  "chapter %local.chapter.class;">
6
7  <!ENTITY % local.index.class "">
8  <!ENTITY % index.class    "index|setindex %local.index.class;">
9
10 <!ELEMENT book ((%div.title.content;)?, bookinfo?, dedication?,
11                 toc?, lot*, (glossary|bibliography|preface)*,
12                 (((%chapter.class;)+, reference*) | part+ | reference+ ),
13                 (%appendix.class;)*, (glossary|bibliography)*,
14                 (%index.class;)*, lot*, toc?)>
```

Элемент типа `book` (строки 10–14) представляет собой некоторую сложную комбинацию из разнотипных элементов, но в нескольких случаях (`appendix`, `chapter` и `index`, см. строки 1–8) модель содержимого явным образом определяет параметрические объекты, которые можно использовать для более удобной подстройки их пользователем. В самом деле, поскольку первое определение некоторого объекта имеет преимущество, все такие параметрические объекты (начинающиеся с `local.`) могут быть определены до того, как производится чтение полного определения DTD. В разделах B.4.4 и B.4.5 показано, как этот метод применяется на практике.

Аналогичным образом, в случае атрибутов для элемента типа `book` параметрический объект `local.book.attrib` (он определен в строке 1, а ссылка на него содержится в строке 8 ниже) обеспечивает средство для дополнения атрибутов, задаваемых в DTD.

```
1  <!ENTITY % local.book.attrib "">
2  <!ENTITY % book.role.attrib "%role.attrib;">
3
4  <!ATTLIST book  %label.attrib;
5                  %status.attrib;
6                  %common.attrib;
7                  %book.role.attrib;
8                  %local.book.attrib;
9  >
```

Фактически в дистрибутиве Нормана Уолша содержится файл `dbgenent.ent`, где содержатся пустые определения для обсуждавшихся выше объектов, которые разрешается доопределять пользователю. Поэтому достаточно отредактировать данный файл, чтобы задать ваши собственные добавочные объекты общего характера, обозначения, а также параметрические объекты.

B.4.2 Вклад AAP и стандарт ISO 12083

После публикации стандарта SGML Standard в 1985 г. Американской издательской ассоциацией AAP проводилась работа по продвижению SGML в качестве электронного стандарта по подготовке рукописей для целей книгоиздания. После нескольких лет работы был подготовлен AAP Standard, продвижением которого в качестве электронного стандарта Electronic Manuscript Standard занялись группа EPSIG (Electronic Publishing Special Interest Group) и AAP. Этот стандарт AAP обеспечивал достижение двух главных целей. Во-первых, в нем был представлен согласованный способ идентификации и маркировки электронных рукописей с помощью предложенного набора элементов SGML tagset, что обеспечило компьютерам возможность распознавать эти части. Во-вторых, был обеспечен логический способ представления специальных литер, символов и табличного материала, используя только литерный набор ASCII.

На основе работ, упомянутых выше, AAP и Европейское физическое общество EPS выработали соглашение относительно стандартного метода для разметки научных документов. Это соглашение позднее было оформлено как международный стандарт ISO 12083, рассматриваемый в качестве преемника стандарта AAP/EPSIG Standard; четыре определения DTD стали распространяться EPSIG как ISO DTD [↪EPSIG].

Это DTD содержит основную структуру книги, состоящую из глав, разделов, а также подразделов, в общей сложности до шести уровней соподчинения. Интересно, что включено DTD для набора математических выражений, которое совершенно аналогично, хотя и не идентично, модели визуального представления из языка MathML, рассмотренной в разд. 8.1. Например, дескрипторы визуальной разметки математических выражений, введенные AAP, позволяют работать со следующими категориями элементов:

character transformations (преобразования литер) `<bold>`, `<italic>`, `<sansser>`, `<typewrit>`, `<smallcap>`, `<roman>`;

fractions (дроби) `<fraction>`, `<num>`, `<den>`;

superiors, inferiors (точная верхняя и нижняя грани) `<sup>`, `<inf>`;

embellishments (декоративные элементы) `<top>`, `<middle>`, `<bottom>`;

fences, boxes, overlines, and underlines (рамки, линии) `<mark>`, `<fence>`, `<post>`, `<box>`, `<overline>`, `<undrline>`;

roots (радикалы) `<radical>`, `<radix>`, `<radicand>`;

arrays (массивы) `<array>`, `<arrayrow>`, `<arraycol>`, `<arraycel>`;

spacing (пробелы) `<hspace>`, `<vspace>`, `<break>`, `<markref>`;
formulae (формулы) `<formula>`, `<dformula>`, `<dformgrp>`.

Акцент здесь делается на правильное расположение формулы. Простой пример разметки выглядит следующим образом:

```
<formula>
  S = &sum;<inf>n=1</inf><sup>10</sup>
      <fraction>
        <num>1</num>
        <den>
          <radical>3<radix>n</radical>
        </den>
      </fraction>
</formula>
```

B.4.3 Проект Text Encoding Initiative

Проект TEI (Text Encoding Initiative, см. [↪TEIHOME]) акцентирует внимание на процессах обмена текстовой информацией, хотя рассматриваются также и другие формы данных, в частности изображения и звук. Главная цель данного проекта — сделать некоторые свойства текста явными таким образом, чтобы помочь обработке текстов компьютерными программами, исполняемыми на различных программно-аппаратных платформах. Соответствующий документ кодируется путем введения разметки исходных текстов.

Был согласован ряд нормативов разметки документов. Те, кто заинтересуются этими нормативами, могут ознакомиться с ними по адресу [↪TEIGUIDE]. Поскольку предполагается, что TEI рассчитан на интернациональное использование при подготовке текстов из самых разнообразных предметных областей, обязательным становится следование ряду принципов. Перечислим их кратко:

- Общее ядро текстовых средств должно быть просто разделяемым между приложениями.
- Должна быть простая возможность вводить дополнительные специальные средства в текст (или удалять их из текста).
- Должна быть возможность использования множественной параллельной кодировки для одного и того же средства.
- Степень богатства разметки должна определяться пользователем с наложением минимальных ограничений.
- Должно быть обеспечено адекватное документирование текста и его кодирование.

Был подготовлен полный набор принципов, однако на практике часто используется краткий «стартовый набор» *TEI-Lite* [↪TEILITE]. Поэтому едва ли вызовет удивление тот факт, что была начата работа по адаптации TEI DTD к XML именно для *TEI-Lite*. Патрис Боном подготовил XML-вариант для *TEI-Lite*, а также предварительный вариант для полного TEI P3 DTD. Он перевел также несколько справочных документов по TEI в формат XML [↪TEIXML].

B.4.4 Определение DTD для BIBTEX'а

Практика — лучший способ усвоения идей, поэтому интересно было бы попробовать применить методы XML, введенные в гл. 6, к тем областям, где большинство пользователей LATEX'а уже имеют какой-то опыт, а именно к собственно LATEX'у, а также к базам данных BIBTEX'а. Поэтому в данном разделе будут построены два варианта DTD для BIBTEX'овской базы данных, при этом даются разъяснения по поводу тех или иных принятых решений. В следующем разделе формируется DTD для подмножества LATEX'а.

B.4.4.1 Трудности при разработке DTD

Рассмотрим следующие важные вопросы, которые возникают, если принято решение написать DTD: какова область применения этого определения; какого рода информацию требуется моделировать; насколько важно, чтобы разметка была понятной для человека, или же разметка и использование документа будут осуществляться автоматически? Аналогично этому, архитектура DTD должна быть проста с точки зрения его сопровождения. Она должна быть сравнительно простой, чтобы облегчить поиск и исправление ошибок, а также в случае, когда это определение DTD надо расширить, причем эти операции должны быть доступны и для того, кто не является разработчиком данного определения, а также не имеет возможности обратиться за разъяснениями к этому разработчику (автору).

Разработка DTD, по-видимому, скорее искусство, чем наука, и только значительный практический опыт дает возможность писать хорошие DTD. Ив Малер и Джинн Эль Андалусси (Maler and Andaloussi (1996)) детально проанализировали множество примеров применения SGML, рассмотрели способы моделирования данных, реализации DTD, а также сопровождения их. Последняя книга Дэвида Меггинсона (Megginson (1998)) более ориентирована на применение XML, здесь обсуждаются различия между XML и SGML. Он посвятил также отдельную главу различным DTD, упоминавшимся нами в разд. B.4.1–B.4.3.

Одно из важных решений, которые приходится принимать при формировании модели, состоит в том, что надо определить, что будет «контентом» (информационным содержимым) и что будет «характеристиками», или, другими словами, какие компоненты нашего документа будут отвечать *элементам*, а какие — *атрибутам*. Существуют различные подходы к решению этой проблемы, по этому предмету много написано, а вывод, который можно сделать из всего написанного таков: практически не существует какого-то единого, универсального ответа на поставленные вопросы (см. [↪ELEMATTR], где содержится обзор данной проблемы).

Сильно упрощая, можно сказать, что сложную информацию, в которой содержится разметка, почти всегда лучше включать внутрь элементов, тогда как атрибуты удобны, когда выбирать приходится только между несколькими ключевыми словами (например, цвет принимает одно из значений — «красный», «голубой» или «зеленый»), или если существует некий набор характери-

стик рассматриваемого элемента, который может быть определен каким-либо простым способом (например, высота, ширина, наименование месяца, страны или языка).

Все же, поскольку в состав XML входит значительно меньше конструкций, чем в SGML (см. анализ, проведенный Норманом Уолшем, применительно к задаче конвертирования SGML в XML DTD [↪SGML2XML] и Megginson:1998:SXD), мы существенно более ограничены в выборе подходящих моделей документа, чем в случае полномасштабного SGML. Например, если надо выразить в модели информационного содержимого тот факт, что некий набор типов элементов можно вводить в формируемый конкретный документ в любом порядке, — в XML это представляет собой чрезвычайно трудоемкую задачу. Аналогично, не существует способа задать включение или исключение элементов некоторых определенных типов по отношению к модельным группам (см. статью Ива Малера относительно управления исключительными ситуациями в SGML и XML [↪MALEREX]). По этим причинам разработка DTD для XML, обеспечивающих максимальную гибкость при решении заданного круга задач, является отнюдь не тривиальной задачей, часто приходится идти на различного рода компромиссы и упрощения в сравнении с SGML.

В следующих двух разделах мы представим два совершенно разных DTD, которые моделируют BIBTEX'овскую базу данных. Первый из этих примеров основан почти исключительно на элементах, предназначенных для хранения информации, тогда как во втором предпринимается попытка задать информацию в максимально возможной степени с помощью атрибутов.

B.4.4.2 Первый вариант DTD для BIBTEX'а

В качестве одного из возможных проектных решений можно принять, что используемые элементы и атрибуты носят те же самые имена, что и соответствующие элементы базы данных (entries) и поля в них (fields), как это описано Лесли Лэмпортом в *приложении B* книги Lamport:1994:LDP. Однако связи между элементами базы данных и ключами цитирования в документе будут задаваться через атрибуты типа `ID` из XML. Вызовем этот атрибут `id` (строка 5), тем самым каждый элемент базы данных становится доступным как объект перекрестных ссылок с помощью атрибута `crossref` типа `IDREF` (строка 6). Каждый элемент базы данных имеет эти два атрибута, значения которым присваиваются с помощью параметрического объекта `atype` (определяемого в строках 5–6, ссылки на него содержатся в строках 19, 24 и т. д.).

Приложение B не предписывает какого-либо порядка, согласно которому надо задавать указанные выше поля (как «требуемые» или «необязательные») для различных типов элементов базы данных. Однако поскольку в XML нет простого способа выразить этот факт и требуемые элементы можно задать только в некотором порядке (в XML оператор `&`, в отличие от SGML, нельзя использовать внутри модельных групп), мы *зафиксируем* тот порядок полей, в котором они пронумерованы Лэмпортом. Кроме того, в конце каждого из

элементов базы данных позволим размещать набор необязательных полей с помощью параметрического объекта `info` (строка 14).

Работа с именами осуществляется с помощью элементов `names`, состоящих из одного или более наборов пар элементов `first` и `last`, разделенных пустым элементом `and` (строки 99–102). Большинство простых полей просто декларируются как `#PCDATA`, тогда как поля с более сложным содержимым (бóльшая часть из тех полей, которым нельзя присвоить значения атрибутов в альтернативной версии определения DTD, их мы обсудим в разд. B.4.4.3) могут также включать эмфатические элементы (строка 88).

Ниже дается полный текст DTD `biblioxml1.dtd` (упоминавшиеся выше номера строк относятся именно к этому тексту):

```
1  <!-- biblioxml1.dtd: XML DTD for BibTeX markup: version 1 -->
2
3  <!-- Every biblio entry _must_ have an identifier and
4                          _can_  have a cross-reference to an existing entry -->
5  <!ENTITY % atype  "id       ID    #REQUIRED
6                     crossref IDREF #IMPLIED">
7
8  <!-- Possible types of biblio entries -->
9  <!ELEMENT biblio (#PCDATA | article| book| booklet| inbook| incollection|
10                            inproceedings| manual| mastersthesis| misc|
11                            phdthesis| proceedings| techreport| unpublished)*>
12
13 <!-- Optional annotation, note, ISBN, or key (the latter for sorting) -->
14 <!ENTITY % info "(annote|note|ISBN|key)*">
15
16 <!-- An article from a journal or magazine -->
17 <!ELEMENT article (author, title, journal, year,
18                    volume?, number?, pages?, month?, %info;)>
19 <!ATTLIST article %atype;>
20
21 <!-- A book with an explicit publisher -->
22 <!ELEMENT book ((author|editor), title, publisher, year,
23                 (volume|number)?, series?, address?, edition?, month?, %info;)>
24 <!ATTLIST book %atype;>
25
26 <!-- A work that is printed or bound, but without a named -->
27 <!--   publisher or sponsoring institution                 -->
28 <!ELEMENT booklet (title ,
29                    author, howpublished?, address?, month?, year?, %info;)>
30 <!ATTLIST booklet %atype;>
31
32 <!-- A part of a book, usually untitled;                          -->
33 <!--   (a chapter, a sectional unit, or just a range of pages) -->
34 <!ELEMENT inbook ((author|editor), title, (chapter|pages)*, publisher, year,
35                   (volume|number)?, series?, type?, address?, edition?, month?, %info;)>
36 <!ATTLIST inbook %atype;>
37
38 <!-- A part of a book with its own title -->
39 <!ELEMENT incollection (author, title, booktitle, publisher, year,
40                         editor?, (volume|number)?, series?, type?, chapter?,
41                         pages?, address?, edition?, month?, %info;)>
42 <!ATTLIST incollection %atype;>
43
44 <!-- An article in a conference proceedings -->
45 <!ELEMENT inproceedings (author, title, booktitle, year,
46                          editor?, (volume|number)?, series?, pages?, address?,
47                          month?, organization?, publisher?, %info;)>
48 <!ATTLIST inproceedings %atype;>
49
50 <!-- Technical documentation -->
```

```
<!ELEMENT manual (title,
                  author?, organization?, address?, edition?, month?, year?, %info;)>
<!ATTLIST manual %atype;>

<!-- A master's thesis -->
<!ELEMENT mastersthesis (author, title, school, year,
                         type?, address?, month?, %info;)>
<!ATTLIST mastersthesis %atype;>

<!-- Miscelleneous: use this type if nothing else fits -->
<!ELEMENT misc (author?, title?, howpublished?, month?, year?,%info;)>
<!ATTLIST misc %atype;>

<!-- A Ph. D. thesis -->
<!ELEMENT phdthesis (author, title, school, year,
                     type?, address?, month?, %info;)>
<!ATTLIST phdthesis %atype;>

<!-- The porceedings of a conference -->
<!ELEMENT proceedings (title  , year ,
                       editor?, (volume|number)?, series?, address?,
                       month?, organization?, publisher?, %info;)>
<!ATTLIST proceedings %atype;>

<!-- A report published by a school or other institution -->
<!--   usually numbered within a series                  -->
<!ELEMENT techreport (author, title, institution, year,
                      type?, number?, address?, month?, %info;)>
<!ATTLIST techreport %atype;>

<!-- A document with author and title, but not formally published -->
<!ELEMENT unpublished (author, title, note,
                       month? , key?)>
<!ATTLIST unpublished %atype;>

<!-- For adding typographic emphasis to the information -->
<!ELEMENT emph (#PCDATA)>
<!ATTLIST emph style (textbf|emph|textsf|textsl|texttt|quote) "emph">

<!ENTITY % inline "(#PCDATA|emph)*">
<!-- The basic fields (Lamport (1994), pages 162-164) -->

             <!-- Usually the address of publisher or institution -->
<!ELEMENT address       (#PCDATA|emph)*>
             <!-- Annotation (not used by standard styles)        -->
<!ELEMENT annote        (#PCDATA|emph)*>
             <!-- Author(s) in format described on pp. 157-158    -->
<!ELEMENT author        (names)>
<!ELEMENT names         ((first,last),(and,first,last)*)>
<!ELEMENT last          (#PCDATA|emph)*>
<!ELEMENT first         (#PCDATA|emph)*>
<!ELEMENT and           EMPTY>
     <!-- Title of a book, used in incollection and inproceedings -->
<!ELEMENT booktitle     (#PCDATA|emph)*>
             <!-- Chapter (or sectional unit) number              -->
<!ELEMENT chapter       (#PCDATA|emph)*>
             <!-- Edition of a book (e.g., "third")               -->
<!ELEMENT edition       (#PCDATA|emph)*>
             <!-- Names of editor(s)                              -->
<!ELEMENT editor        (names)>
             <!-- Describe how something strange is published     -->
<!ELEMENT howpublished  (#PCDATA|emph)*>
             <!-- Sponsoring institution of a technical report    -->
<!ELEMENT institution   (#PCDATA|emph)*>
             <!-- The ISBN number (non-standard, but useful       -->
<!ELEMENT ISBN          (#PCDATA)>
             <!-- A journal's name                                -->
```

```
<!ELEMENT journal       (#PCDATA|emph)*>
          <!-- Used for ordering the biblio entries            -->
<!ELEMENT key           (#PCDATA)>
          <!-- Month of publication (or writing, if not published) -->
<!ELEMENT month         (#PCDATA)>
          <!-- Additional information to help the user         -->
<!ELEMENT note          (#PCDATA|emph)*>
          <!-- Number of a journal, magazine, report, etc.     -->
<!ELEMENT number        (#PCDATA)>
          <!-- Sponsor of conference or publisher of manual    -->
<!ELEMENT organization  (#PCDATA|emph)*>
          <!-- page, or page range                             -->
<!ELEMENT pages         (#PCDATA)>
          <!-- Publisher's name                                -->
<!ELEMENT publisher     (#PCDATA|emph)*>
          <!-- Name of the school where thesis was written     -->
<!ELEMENT school        (#PCDATA|emph)*>
          <!-- Name of the series or set of books              -->
<!ELEMENT series        (#PCDATA|emph)*>
          <!-- A work's title                                  -->
<!ELEMENT title         (#PCDATA|emph)*>
          <!-- Type of a technical report, e.g., "Research Note" -->
<!ELEMENT type          (#PCDATA|emph)*>
          <!-- Volume number of a journal or multi-volume book -->
<!ELEMENT volume        (#PCDATA)>
          <!-- Year of publication (or writing, if not published) -->
<!ELEMENT year          (#PCDATA)>
```

Теперь можно использовать это DTD для обработки примера документа; в качестве основы такого примера возьмем исходный BIBTEX'овский текст, показанный на рис. 13.4 книги *The LATEX Companion* (Goossens et al. (1994)). Ниже показывается, каким образом разметить упомянутый документ применительно к построенному DTD. В начале рассматриваемого файла (строки 2–10) дадим определения объектов, работа с которыми осуществлялась с помощью стилевого файла `bibnames.sty`, или определенных в первоначальном файле. Обратите внимание, каким образом мы используем объекты, чтобы ввести сокращенные варианты наименований месяцев (строки 12–23), предопределенные в BIBTEX'е. Исходя из соображений удобства и типографской совместимости, на протяжении всего файла будем использовать эти ссылки на объекты, чтобы получить в итоге соответствующие им текстовые значения. Конечно, если бы ставилась задача построить систему текстовой обработки, отличную от LATEX'а, пришлось бы некоторые из объектов задать другим способом (в частности, строки 4–11, которые являются специфическими для LATEX'а).

Ссылки цитирования могли бы использовать значения, определенные атрибутом `id` для каждого из элементов базы данных. Обратите внимание, как перекрестная ссылка в строке 85 использует возможности `ID/IDREF` из XML. Однако поскольку идентификаторы в некотором экземпляре XML-документа используют совместно некоторое глобальное пространство имен, надо стараться применять ясные и непротиворечивые схемы именования для идентификаторов элементов документа.

```
1  <!DOCTYPE biblio SYSTEM "biblioxml1.dtd" [
2  <!ENTITY AW "Addison-Wesley">
3  <!ENTITY AW:adr "Reading, Massachusetts">
4  <!ENTITY emdash "---">
5  <!ENTITY endash "--">
```

```
<!ENTITY ouml '\"o'>
<!ENTITY j-TUGboat "TUGboat">
<!ENTITY LaTeX "\LaTeX{}">
<!ENTITY TeX "\TeX{}">
<!ENTITY WEB "\textsc{web}">
<!-- abreviations for the months -->
<!ENTITY jan "Jan.">
<!ENTITY feb "Feb.">
<!ENTITY mar "Mar.">
<!ENTITY apr "Apr.">
<!ENTITY may "May">
<!ENTITY jun "Jun.">
<!ENTITY jul "Jul.">
<!ENTITY aug "Aug.">
<!ENTITY sep "Sep.">
<!ENTITY oct "Oct.">
<!ENTITY nov "Nov.">
<!ENTITY dec "Dec.">
]>
<biblio>
<manual id="Dynatext">
  <title>Dynatext, Electronic Book Indexer/Browser</title>
  <organization>Electronic Book Technology Inc.</organization>
  <address>Providence, Rhode Island</address>
  <year>1991</year>
</manual>
  <book id="Eijkhout:1991">
  <author><names><first>Victor</first><last>Eijkhout</last></names></author>
  <title>&TeX; by Topic, a &TeX;nicians Reference</title>
  <publisher>&AW;</publisher>
  <year>1991</year>
  <address>&AW:adr;</address>
</book>,
<techreport id="EVH:Office">
  <author><names><first>Eric</first><last>van Herwijnen</last></names></author>
  <title>Future Office Systems Requirements</title>
  <institution>CERN DD Internal Note</institution>
  <year>1988</year>
  <month>&nov;</month>
</techreport>
<article id="Felici:1991">
  <author><names><first>James</first><last>Felici</last></names></author>
  <title>PostScript versus TrueType</title>
  <journal>Macworld</journal>
  <year>1991</year>
  <volume>8</volume>
  <pages>195&endash;201</pages>
  <month>&sep;</month>
</article>
<techreport id="Knuth:WEB">
  <author><names><first>Donald E.</first><last>Knuth</last></names></author>
  <title>The &WEB; System of Structured Documentation</title>
  <institution>Department of Computer Science, Stanford University</institution>
  <year>1983</year>
  <number>STAN-CS-83-980</number>
  <address>Stanford, CA 94305</address>
  <month>&sep;</month>
</techreport>
<phdthesis id="Liang:1983">
  <author><names><first>Franklin Mark</first><last>Liang</last></names></author>
  <title>Word Hy-phen-a-tion by Com-pu-ter</title>
  <school>Stanford University</school>
  <year>1983</year>
  <address>Stanford, CA 94305</address>
  <month>&jun;</month>
  <note>Also available as Stanford University, Department of
    Computer Science Report No. STAN-CS-83-977</note>
```

```
73  </phdthesis>
74  <article id="Mittelbach-Schoepf:1990">
75    <author><names><first>Frank</first><last>Mittelbach</last><and/>
76            <first>Rainer</first><last>Sch&ouml;pf</last></names></author>
77    <title>The New Font Selection &emdash; User Interface to
78       Standard &LaTeX;</title>
79    <journal>&j-TUGboat;</journal>
80    <year>1990</year>
81    <volume>11</volume>
82    <number>2</number>
83    <pages>297&endash;305</pages>
84  </article>
85  <incollection id="Wood:color" crossref="Roth:postscript">
86    <author><names><first>Pat</first><last>Wood</last></names></author>
87    <title>PostScript Color Separation</title>
88    <booktitle>Real World PostScript</booktitle>
89    <publisher>&AW;</publisher>
90    <year>1988</year>
91    <pages>201&endash;225</pages>
92  </incollection>
93  <book id="Roth:postscript">
94    <editor><names><first>Stephen E.</first><last>Roth</last></names></editor>
95    <title>Real World PostScript</title>
96    <publisher>&AW;</publisher>
97    <year>1988</year>
98    <address>&AW:adr;</address>
99    <ISBN>0-201-06663-7</ISBN>
100 </book>
101 <inproceedings id="Yannis:1991">
102   <author><names><first>Yannis</first><last>Haralambous</last></names></author>
103   <title>&TeX; and those other languages</title>
104   <booktitle>1991 Annual Meeting Proceedings, Part 2, &TeX; Users Group,
105            Twelfth Annual Meeting, Dedham, Massachusetts, July 15--18, 1991
106   </booktitle>
107   <year>1991</year>
108   <editor><names><first>Hope</first><last>Hamilton</last></names></editor>
109   <volume>12</volume>
110   <series>&j-TUGboat;</series>
111   <pages>539&endash;548</pages>
112   <address>Providence, Rhode Island</address>
113   <month>&dec;</month>
114   <organization>&TeX; Users Group</organization>
115 </inproceedings>
116 </biblio>
```

B.4.4.3 Второй вариант DTD для BIBTEX'а

При разработке альтернативного варианта DTD для BIBTEX'а попробуем поэкспериментировать, чтобы задать возможно больший объем информации с помощью атрибутов. Мы знаем уже, что весьма непросто использовать атрибуты для сложных полей, таких как (book)`title`, `author` и `editor`, в которые входят имена, `note` и `annote`, в которых могут содержаться текстовые примечания. По этой причине они определены как элементы (строки 179–194 в конце рассматриваемого DTD), тогда как все прочие поля даются как атрибуты, следуя определениям различных объектов (строки 24–30, 34–43 и т. д.).

В этом варианте мы обеспечим возможность расширения DTD путем введения параметрического объекта `local.info` (определяется как пустая цепочка литер в строке 4) и помещением ссылки на этот объект в определении параметрического объекта `info` (строка 7). Теперь пользователь данного DTD

имеет возможность определять новые элементы, вводя их в в переменную `local.info`, как это продемонстрировано в следующем ниже примере. Таким способом достаточно просто можно обеспечить доступ к новым полям. Аналогичным образом, определяя параметрический объект `local.biblio`, пользователь может ввести дополнительно новые типы объектов (мы добавили ссылку на `local.biblio` в конце определения модели содержимого для корневого элемента `biblio`; см. строку 20). Строка 6 декларирует `local.keys`, где обеспечивается глобальный способ добавления атрибутов ко всем элементам базы данных.

Ниже показан альтернативный вариант DTD для BIBTEX'а, представляющий собой файл с именем `biblioxml2.dtd`:

```
1  <!-- biblioxml2.dtd: XML DTD for BibTeX markup: version 2 -->
2
3  <!-- Supplementary entry types, optional fields and attributes       -->
4  <!ENTITY % local.info "">
5  <!ENTITY % local.biblio "">
6  <!ENTITY % local.keys "">
7  <!ENTITY % info  "note %local.info;">
8
9  <!-- All base elements must have ID and can have cross-reference to  -->
10 <!-- an existing ID and include a  key to sort, if needed            -->
11 <!ENTITY % keys   "id       ID    #REQUIRED
12                   crossref IDREF #IMPLIED
13                   key      IDREF #IMPLIED
14                   %local.keys;">
15 <!ENTITY % month  'month (jan|feb|mar|apr|may|jun|jul|aug|sep|oct|nov|dec) #IMPLIED'>
16 <!-- Possible types of biblio entries -->
17 <!ELEMENT biblio (#PCDATA | article| book| booklet| inbook| incollection|
18                            inproceedings| manual| mastersthesis| misc|
19                            phdthesis| proceedings| techreport| unpublished
20                            %local.biblio;)*>
21
22 <!-- An article from a journal or magazine -->
23 <!ELEMENT article (author,title,(%info;)*)>
24 <!ATTLIST article %keys;
25                   journal  CDATA  #REQUIRED
26                   year     CDATA  #REQUIRED
27                   %month;
28                   number   CDATA  #IMPLIED
29                   pages    CDATA  #IMPLIED
30                   volume   CDATA  #IMPLIED>
31
32 <!-- A book with an explicit publisher -->
33 <!ELEMENT book    ((author|editor),title,(%info;)*)>
34 <!ATTLIST book    %keys;
35                   publisher CDATA  #REQUIRED
36                   year      CDATA  #REQUIRED
37                   address   CDATA  #IMPLIED
38                   edition   CDATA  #IMPLIED
39                   %month;
40                   number    CDATA  #IMPLIED
41                   series    CDATA  #IMPLIED
42                   volume    CDATA  #IMPLIED
43                   ISBN      CDATA  #IMPLIED>
44
45 <!-- A work that is printed or bound, but without a named -->
46 <!--   publisher or sponsoring institution                 -->
47 <!ELEMENT booklet (title,author?,(%info;)*)>
48 <!ATTLIST booklet %keys;
49                   address      CDATA  #IMPLIED
50                   howpublished CDATA  #IMPLIED
```

```
51                  %month;
52                  year         CDATA  #IMPLIED
53                  ISBN       CDATA  #IMPLIED>
54
55  <!-- A part of a book, usually untitled;                          -->
56  <!--   (a chapter, a sectional unit, or just a range of pages) -->
57  <!ELEMENT inbook  ((author|editor),title,(%info;)*)>
58  <!ATTLIST inbook  %keys;
59                  pages     CDATA  #REQUIRED
60                  publisher CDATA  #REQUIRED
61                  year      CDATA  #REQUIRED
62                  address   CDATA  #IMPLIED
63                  chapter   CDATA  #IMPLIED
64                  edition   CDATA  #IMPLIED
65                  %month;
66                  number    CDATA  #IMPLIED
67                  series    CDATA  #IMPLIED
68                  type      CDATA  #IMPLIED
69                  volume    CDATA  #IMPLIED
70                  ISBN      CDATA  #IMPLIED>
71
72  <!-- A part of a book with its own title -->
73  <!ELEMENT incollection  (author,title,booktitle,editor?,(%info;)*)>
74  <!ATTLIST incollection  %keys;
75                        publisher CDATA  #REQUIRED
76                        year      CDATA  #REQUIRED
77                        address   CDATA  #IMPLIED
78                        chapter   CDATA  #IMPLIED
79                        edition   CDATA  #IMPLIED
80                        %month;
81                        number    CDATA  #IMPLIED
82                        pages     CDATA  #IMPLIED
83                        series    CDATA  #IMPLIED
84                        type      CDATA  #IMPLIED
85                        volume    CDATA  #IMPLIED
86                        ISBN      CDATA  #IMPLIED>
87
88  <!-- An article in a conference proceedings -->
89  <!ELEMENT inproceedings (author,title,booktitle,editor?,(%info;)*)>
90  <!ATTLIST inproceedings %keys;
91                        year         CDATA  #REQUIRED
92                        address      CDATA  #IMPLIED
93                        %month;
94                        number       CDATA  #IMPLIED
95                        organization CDATA  #IMPLIED
96                        pages        CDATA  #IMPLIED
97                        publisher    CDATA  #IMPLIED
98                        series       CDATA  #IMPLIED
99                        volume       CDATA  #IMPLIED
100                       ISBN         CDATA  #IMPLIED>
101
102 <!-- Technical documentation -->
103 <!ELEMENT manual (title,author?,(%info;)*)>
104 <!ATTLIST manual  %keys;
105                 address      CDATA  #IMPLIED
106                 edition      CDATA  #IMPLIED
107                 organization CDATA  #IMPLIED
108                 %month;
109                 year         CDATA  #IMPLIED
110                 ISBN         CDATA  #IMPLIED>
111
112 <!-- A master's thesis -->
113 <!ELEMENT mastersthesis  (author,title,(%info;)*)>
114 <!ATTLIST mastersthesis  %keys;
115                        school  CDATA  #REQUIRED
116                        year    CDATA  #REQUIRED
117                        address CDATA  #IMPLIED
```

```
                        %month;
                        type    CDATA  #IMPLIED
                        ISBN    CDATA  #IMPLIED>

<!-- Miscelleneous: use this type if nothing else fits -->
<!ELEMENT misc  (((author,title)|(title,author))?,(%info;)*)>
<!ATTLIST misc  %keys;
                howpublished CDATA  #IMPLIED
                %month;
                year         CDATA  #IMPLIED>

<!-- A Ph. D. thesis -->
<!ELEMENT phdthesis  (author,title,(%info;)*)>
<!ATTLIST phdthesis  %keys;
                     school  CDATA  #REQUIRED
                     year    CDATA  #REQUIRED
                     address CDATA  #IMPLIED
                     %month;
                     type    CDATA  #IMPLIED
                     ISBN    CDATA  #IMPLIED>

<!-- The proceedings of a conference -->
<!ATTLIST proceedings %keys;
                      year         CDATA  #REQUIRED
                      address      CDATA  #IMPLIED
                      %month;
                      number       CDATA  #IMPLIED
                      organization CDATA  #IMPLIED
                      publisher    CDATA  #IMPLIED
                      series       CDATA  #IMPLIED
                      volume       CDATA  #IMPLIED
                      ISBN         CDATA  #IMPLIED>

<!-- A report published by a school or other institution -->
<!--   usually numbered within a series                  -->
<!ELEMENT techreport (author,title,(%info;)*)>
<!ATTLIST techreport  %keys;
                      institution  CDATA  #REQUIRED
                      year         CDATA  #REQUIRED
                      address      CDATA  #IMPLIED
                      %month;
                      number       CDATA  #IMPLIED
                      type         CDATA  #IMPLIED
                      ISBN         CDATA  #IMPLIED>

<!-- A document with author and title, but not formally published -->
<!ELEMENT unpublished  (author,title,(%info;)*)>
<!ATTLIST unpublished  %keys;
                       address      CDATA  #IMPLIED
                       %month;
                       number       CDATA  #IMPLIED
                       type         CDATA  #IMPLIED>

<!-- For adding typographic emphasis to the information -->
<!ELEMENT emph (#PCDATA)>
<!ATTLIST emph style (bf|em|it|sf|sl|tt|qu) "em">

<!-- The basic fields (pages 162-164) -->
<!-- Only fields with names (author, editor) and titles are left as -->
<!-- basic elements -->
                   <!-- Author(s) in format described on pp. 157-158 -->
<!ELEMENT author     (name,(and,name)*)>
<!ELEMENT name       (first,last)>
<!ELEMENT last       (#PCDATA)>
<!ELEMENT first      (#PCDATA)>
<!ELEMENT and         EMPTY>
                   <!-- The names of the editor(s)                  -->
```

```
185  <!ELEMENT editor      (name,(and,name)*)>
186                    <!-- The work's title                                  -->
187  <!ELEMENT title       (#PCDATA|emph)*>
188        <!-- Title of a book, used in incollection and inproceedings -->
189  <!ELEMENT booktitle   (#PCDATA|emph)*>
190  <!-- Optional notes at end of entries -->
191                    <!-- Annotation (not used by standard styles)      -->
192  <!ELEMENT annote      (#PCDATA|emph)*>
193                    <!-- Additional information to help the user       -->
194  <!ELEMENT note        (#PCDATA|emph)*>
```

Этот новый вариант DTD для BIBTEX'а можно использовать, ссылаясь на него в маленьком DTD `mybiblio.dtd`, который дается ниже. Строка 2 определяет параметрический объект `biblio.dtd`, который позволяет включить DTD `biblioxml2.dtd` в строке 17. Строки 4–12 определяют те же самые общие объекты, что и в строках 2–10 в примере документа из разд. B.4.4.2.

```
 1  <!-- mybiblio.dtd (refers to "biblioxml2.dtd" -->
 2  <!ENTITY % biblio.dtd SYSTEM "biblioxml2.dtd">
 3
 4  <!ENTITY AW "Addison-Wesley">
 5  <!ENTITY AW:adr "Reading, Massachusetts">
 6  <!ENTITY emdash "---">
 7  <!ENTITY endash "--">
 8  <!ENTITY ouml '\"o'>
 9  <!ENTITY j-TUGboat "TUGboat">
10  <!ENTITY LaTeX "\LaTeX{}">
11  <!ENTITY TeX "\TeX{}">
12  <!ENTITY WEB "\textsc{web}">
13  <!ENTITY % local.info "| url |annote">
14  <!ELEMENT url (#PCDATA)>
15  <!ENTITY % local.biblio "| webdocument">
16
17  %biblio.dtd;
18
19  <!-- Document published on the Web -->
20  <!ELEMENT webdocument (title,author?,(%info;)*)>
21  <!ATTLIST webdocument  %keys;
22                         organization CDATA  #IMPLIED
23                         %month;
24                         year         CDATA  #IMPLIED>
```

Поскольку на параметрические объекты нельзя ссылаться до того, как они определены, надо тщательно следить за порядком, в котором задаются различные элементы базы данных в DTD. Поэтому определим теперь объект `local.info` (строка 13), где добавим новое поле `url` и включим также поле `annote`, которое определено в рассматриваемом DTD (строка 192), но не включено ни в одну из моделей содержимого. Строка 14 определяет содержимое элемента `url` как `#PCDATA`. Конечно, мы могли бы предпочесть задать адреса URL в виде атрибута. В этом случае надо было бы заменить строки 13–14 такими:

```
<!ENTITY % local.info "|annote">
<!ENTITY % local.key "url CDATA IMPLIED">
```

Однако в примере, который будет рассмотрен ниже, будем использовать определение в виде элемента.

Наконец, поскольку мы хотим определить новый тип элемента базы данных, `webdocument`, включим его в параметрический объект `local.biblio`

(строка 15). Включим первоначальный DTD посредством ссылки на этот объект `biblio.dtd` (строка 17). Данный DTD использует значения всех параметрических объектов `«local»`, а также определения элементов, соответственно адаптированных.

Только теперь (в строках 20–24) можно поместить определение нового типа элемента базы данных `webdocument`, поскольку он ссылается на различные параметрические объекты (`info`, `keys` и `month`), которые определены в DTD. По тем же самым соображениям нельзя определить этот элемент и его атрибуты во внутреннем подмножестве рассматриваемого примера документа. Последнее читается до внешнего подмножества, когда указанные параметры еще не определены.

Используя DTD `mybiblio.dtd`, выполним разметку той же самой BIBTEX'овской базы данных, что рассматривалась в разд. B.4.4.2. Посмотрим, каким образом теперь разделяется информация между элементами и атрибутами. К полям, задаваемым с помощью атрибутов, можно обращаться в любом порядке, что упрощает выполнение операций по доступу к данным. Обратите внимание, как из наименований месяцев получены имена лексем в списке атрибутов (как определено в строке 15 в DTD `biblioxml2.dtd`) и на то, что они более не задаются с помощью общих ссылок на объекты (строки 20, 29 и т. д.). В качестве нововведения в конец файла (строки 102–113) включим добавочный элемент базы данных, которого не было в первоначальном примере. Этот элемент будет иметь тип `xmldocument`, как это определено в DTD `mybiblio.dtd`, показанном ранее в строках 20–24.

```
1  <!DOCTYPE biblio SYSTEM "mybiblio.dtd">
2  <biblio>
3  <manual id           ="Dynatext"
4          organization="Electronic Book Technology Inc."
5          address     ="Providence, Rhode Island"
6          year        ="1991">
7    <title>Dynatext, Electronic Book Indexer/Browser</title>
8  </manual>
9  <book id       ="Eijkhout:1991"
10       publisher="&AW;"
11       year     ="1991"
12       address  ="&AW:adr;">
13   <author>
14     <name><first>Victor</first><last>Eijkhout</last></name>
15   </author>
16   <title>&TeX; by Topic, a &TeX;nicians Reference</title>
17 </book>
18 <techreport id          ="EVH:Office"
19             institution="CERN DD Internal Note"
20             month       ="nov"
21             year        ="1988">
22   <author>
23     <name><first>Eric</first><last>van Herwijnen</last></name>
24   </author>
25   <title>Future Office Systems Requirements</title>
26 </techreport>
27 <article id     ="Felici:1991"
28          journal="Macworld"
29          month  ="sep"
30          year   ="1991"
31          volume ="8"
32          pages  ="195&endash;201">
```

```
33    <author><name><first>James</first><last>Felici</last></name></author>
34    <title>PostScript versus TrueType</title>
35  </article>
36  <techreport id          ="Knuth:WEB"
37              institution="Department of Computer Science, Stanford University"
38              address    ="Stanford, CA 94305"
39              month      ="sep"
40              year       ="1983"
41              number     ="STAN-CS-83-980">
42    <author><name><first>Donald E.</first><last>Knuth</last></name></author>
43    <title>The &WEB; System of Structured Documentation</title>
44  </techreport>
45  <phdthesis id    ="Liang:1983"
46             school="Stanford University"
47             address="Stanford, CA 94305"
48             month  ="jun"
49             year   ="1983">
50    <author><name><first>Franklin Mark</first><last>Liang</last></name></author>
51    <title>Word Hy-phen-a-tion by Com-pu-ter</title>
52    <note>Also available as Stanford University, Department of
53       Computer Science Report No. STAN-CS-83-977
54    </note>
55  </phdthesis>
56  <article id="Mittelbach-Schoepf:1990"
57           journal="&j-TUGboat;"
58           year="1990"
59           volume="11"
60           number="2"
61           pages="297&endash;305">
62    <author><name><first>Frank</first><last>Mittelbach</last></name>
63            <and/>
64            <name><first>Rainer</first><last>Sch&ouml;pf</last></name>
65    </author>
66    <title>
67      The New Font Selection &emdash; User Interface to Standard &LaTeX;"
68    </title>
69  </article>
70  <incollection id       ="Wood:color"
71                crossref ="Roth:postscript"
72                publisher="&AW;"
73                year     ="1988"
74                pages    ="201&endash;225">
75    <author><name><first>Pat</first><last>Wood</last></name></author>
76    <title>PostScript Color Separation</title>
77    <booktitle>Real World PostScript</booktitle>
78  </incollection>
79  <book id       ="Roth:postscript"
80        publisher="&AW;"
81        year     ="1988"
82        address  ="&AW:adr;"
83        ISBN     ="0-201-06663-7">
84    <editor><name><first>Stephen E.</first><last>Roth</last></name></editor>
85    <title>Real World PostScript</title>
86  </book>
87  <inproceedings id          ="Yannis:1991"
88                 series      ="&j-TUGboat;"
89                 volume      ="12"
90                 pages       ="539&endash;548"
91                 organization="&TeX; Users Group"
92                 address     ="Providence, Rhode Island"
93                 month       ="dec"
94                 year        ="1991">
95    <author><name><first>Yannis</first><last>Haralambous</last></name></author>
96    <title>&TeX; and those other languages</title>
97    <booktitle>1991 Annual Meeting Proceedings, Part 2, &TeX; Users Group,
98             Twelfth Annual Meeting, Dedham, Massachusetts, July 15--18, 1991
99    </booktitle>
```

```
100     <editor><name><first>Hope</first><last>Hamilton</last></name></editor>
101   </inproceedings>
102   <webdocument id="xml-latest-news"
103                organization="OASIS">
104     <title>SGML and XML News</title>
105     <author><name><first>Robin</first><last>Cover</last></name></author>
106     <url>http://www.oasis-open.org/cover/sgmlnew.html</url>
107     <note>
108       Information about what is (relatively) new in the 'SGML/XML Web Page'
109     </note>
110     <annote>This page keeps you informed about the latest developments
111        in the area of SGML, XML, and related areas.
112     </annote>
113   </webdocument>
114   </biblio>
```

B.4.4.4 Резюме определений DTD для BIBTEX'а

Из материалов, представленных в разд. B.4.4.2 и B.4.4.3, где были описаны два варианта DTD для BIBTEX'а, становится ясно, что разметку одних и тех же данных можно осуществить несколькими эквивалентными способами, без какой-либо потери информации. Это обстоятельство показывает гибкость подхода, реализуемого в XML, где разработчик DTD может предусмотреть зацепки для расширения модели данных и ее функционального поведения, а также подстройки под потребности пользователя этих элементов, взятых из начального варианта DTD. Разумеется, это очень сильно зависит от решаемой прикладной задачи и, что еще более важно, от предпочтений соответствующего сообщества пользователей, чтобы решить, какая форма представления информации (т. е. вид схемы данных или DTD) будет более полезной.

B.4.5 Разметка документа в стиле LATEX'а: от DTD до вывода на печать

Было предпринято несколько попыток разработать SGML-приложения для LATEX'а. Наиболее успешной из этих попыток можно считать, по-видимому, проект `qwertz` Тома Гордона из Немецкого национального исследовательского центра по информатике [↪QWERTZ]. Наиболее свежая версия этой системы конвертирует HTML-документы в LATEX с помощью встроенной программы грамматического анализа SGML, написанной на языке Java, или же грамматического анализатора `nsgmls`, разработанного Джеймсом Кларком. Основное достоинство упомянутого подхода состоит в том, что используется один и тот же исходный текст для формирования документа как в виде распечатки, так и в Web-форме, но, поскольку DTD имеет статический характер, нельзя добавлять свои собственные макро, а обработка математических выражений и изображений требует специального подхода. Основным потребителем этой системы до недавнего времени был проект Linux Documentation Project. Однако для этого проекта теперь выбран для применения пакет `SGMLtools`, основанный на DocBook DTD [↪SGMLTOOLS].

В данном разделе построим вначале мини-DTD для LaTeX'а, где будут реализованы структурные элементы языка LaTeX. При этом в стороне остается множество элементов типографского характера, а также детали, связанные с математическими выражениями, которые достаточно полно обсуждались в гл. 8. После построения DTD, основываясь на схеме разметки, реализованной в нем, разработаем пример многоязычного документа, который будет отпечатан с использованием стиля XSL посредством трансляции XML в LaTeX.

B.4.5.1 Определение DTD, основанное на LaTeX'е

В нашей DTD-модели (`minilatex.dtd`), полный исходный текст которой приводится ниже, не делается попытки следовать LaTeX'у слишком буквально, поскольку в некоторых случаях в LaTeX'е смешиваются структурные и презентационные элементы (то, что HTML делает *в большинстве* случаев). Например, элементы секционирования становятся реальными контейнерами с заголовком, за которым следуют абзацеподобный материал, плавающие объекты (строка 140) и элементы секционирования более низкого уровня (строки 143–156). Таким образом, команды секционирования будут только структурными, и они будут показывать разделение документа на подразделы, в то время как их заголовок должен задаваться отдельно (строки 141–142).

Иногда трудно бывает решить, в какое место DTD следует включить данный элемент LaTeX'а. В LaTeX'е есть некоторые элементы, которые могут встретиться практически везде, и в то же время наложен запрет использовать их в качестве вложенных конструкций (например, подстрочные примечания и примечания на полях. Придется игнорировать такого рода элементы и оставить их под полным контролем пользователя. Такого рода элементы, если потребуется, можно вставить явным образом путем расширения DTD, и остается только надеяться, что пользователь распорядится этой возможностью должным образом. Наш файл-пример, представленный в разд. B.4.5.3, покажет, каким образом можно применить этот механизм расширения.

Следуя хорошей практике, введенной в DTD для HTML4, большинство элементов снабдим атрибутом `id` типа `ID`, вследствие чего они станут доступными для организации перекрестных ссылок, а также атрибутами `style` и `class` (строки 3–5). Последние два атрибута пригодятся, когда потребуется осуществить визуализацию элементов XML-документа, как это объяснялось в гл. 7. Введем также атрибут `xml:lang` (строка 7), позволяющий задать язык, на котором написана данная часть текста. Это полезно для программ проверки орфографии, алгоритмов переносов слов, типографского воспроизведения и т. п.

Система перекрестных ссылок полностью основана на механизме идентификаторов XML (лексемы типа `ID` и `IDREF` для атрибутов, см. разд. 6.5.4.2). Как уже объяснялось ранее, атрибут `id` допускается для большинства элементов, и, как это видно в строках 44–50, все элементы `cite`, `pageref` и `ref` обладают атрибутом `idref`. Используя все это в комплексе, можно довольно просто сослаться практически на любой элемент документа.

При написании DTD будем пытаться придать ему в какой-то степени модульную структуру за счет определения параметрических объектов, чтобы структуризовать информацию (`fontchange`, `xref` и т. д., в строках 10–27). Чтобы упростить расширение DTD, каждый из этих параметрических объектов имеет «компаньона» — файл с расширением `.new` (`fontchange.new`, `xref.new` и т. д.). Они определяются как пустые строки, но пользователь может переопределить их во внутреннем подмножестве документа или в другом DTD. Таким способом новые элементы могут быть достаточно просто добавлены в различных точках иерархии документа.

Большая часть из элементов в исходном листинге, приводимом ниже, проста для понимания. Работа с математическими выражениями (строки 162–172) ссылается на другие фрагменты DTD, которые обсуждаются отдельно, в разд. B.4.5.2.

```
1  <!-- minilatex.dtd: XML version of a subset of LaTeX -->
2  <!-- Most elements have the following attributes  -->
3  <!ENTITY % all    "id    ID #IMPLIED
4                     class CDATA #IMPLIED
5                     style CDATA #IMPLIED">
6  <!-- Most non-empty elements have this language attribute -->
7  <!ENTITY % i18n   "xml:lang NMTOKEN #IMPLIED">
8  <!ENTITY % basic  "%all; %i18n;">
9  <!-- Declaration of parameter entities for structuring the DTD -->
10 <!ENTITY % fontchange.new "">
11 <!ENTITY % fontchange "emph|textbf|textsc|textsf|textsl|texttt %fontchange.new;">
12 <!ENTITY % misc.new "">
13 <!ENTITY % misc "url|quad|hspace|vspace|includegraphics|tag|ent %misc.new;">
14 <!ENTITY % xref.new  "">
15 <!ENTITY % xref "cite|pageref|ref %xref.new;">
16 <!ENTITY % mathobj.new "">
17 <!ENTITY % mathobj "displaymath|inlinemath|equation|eqnarray" >
18 <!ENTITY % inline.new "">
19 <!ENTITY % inline  "%fontchange;|quote|tabular|%misc;|%xref;|%mathobj; %inline.new;">
20 <!ENTITY % list.new "">
21 <!ENTITY % list    "lalist|description|enumerate|itemize|bibliography %list.new;">
22 <!ENTITY % preformat.new "">
23 <!ENTITY % preformat "verbatim|verse %preformat.new;">
24 <!ENTITY % likepara.new "">
25 <!ENTITY % likepara "%list;|quotation|align|%preformat; %likepara.new;">
26 <!ENTITY % floats.new "">
27 <!ENTITY % floats "figure|table %floats.new;">
28
29 <!-- The top level document declarations                    -->
30 <!ELEMENT document (frontmatter?,bodymatter,backmatter?)>
31 <!ATTLIST document %i18n;
32                    class CDATA "article">
33 <!ELEMENT frontmatter (title,author?,date?,abstract?)>
34 <!ELEMENT bodymatter  ((par|%likepara;|%floats;|part|chapter|section)*,appendix*)>
35 <!ELEMENT backmatter  (index|glossary)*>
36 <!-- fontchanges -->
37 <!ELEMENT emph    (#PCDATA|%inline;)*>
38 <!ELEMENT textbf  (#PCDATA|%inline;)*>
39 <!ELEMENT textsc  (#PCDATA|%inline;)*>
40 <!ELEMENT textsf  (#PCDATA|%inline;)*>
41 <!ELEMENT textsl  (#PCDATA|%inline;)*>
42 <!ELEMENT texttt  (#PCDATA|%inline;)*>
43 <!-- cross-references -->
44 <!ENTITY % idref  "refid IDREF #REQUIRED">
45 <!ELEMENT cite     EMPTY>
46 <!ATTLIST cite     %idref;>
47 <!ELEMENT pageref  EMPTY>
```

```
<!ATTLIST pageref  %idref;>
<!ELEMENT ref      EMPTY>
<!ATTLIST ref      %idref;>
<!-- quoted material inline and displayed -->
<!ELEMENT quote        (#PCDATA|%inline;|par)*>
<!ATTLIST quote        %basic;>
<!ELEMENT quotation    (#PCDATA|%inline;|par)*>
<!ATTLIST quotation    %basic;>
<!-- lists -->
<!ELEMENT description  ((term*,item*)+)>
<!ATTLIST description  %basic;>
<!ELEMENT enumerate    ((term*,item*)+)>
<!ATTLIST enumerate    %basic;>
<!ELEMENT itemize      ((term*,item*)+)>
<!ATTLIST itemize      %basic;>
<!ELEMENT lalist       ((term*,item*)+)>
<!ATTLIST lalist       %basic;>
<!ELEMENT term         (#PCDATA|%inline;)*>
<!ATTLIST term         %basic;>
<!ELEMENT item         (#PCDATA|%inline;|par|%likepara;)*>
<!ATTLIST item         %basic;>
<!ELEMENT bibliography (bibitem)*>
<!ATTLIST bibliography %basic;>
<!ELEMENT bibitem      (#PCDATA|%inline;|par|%likepara;)*>
<!ATTLIST bibitem      %basic;>
<!-- low-level bits and pieces -->
<!ELEMENT url    EMPTY>
<!ATTLIST url    name CDATA #REQUIRED>
<!ELEMENT quad   EMPTY>
<!ELEMENT hspace EMPTY>
<!ATTLIST hspace dim CDATA #REQUIRED>
<!ELEMENT vspace EMPTY>
<!ATTLIST vspace dim CDATA #REQUIRED>
<!ELEMENT tag    (#PCDATA)>
<!ELEMENT ent    EMPTY>
<!ATTLIST ent    value CDATA #REQUIRED
                 name  CDATA #REQUIRED>
<!-- everything that can go into a paragraph -->
<!ELEMENT par     (#PCDATA|%inline;|%likepara;)*>
<!ATTLIST par     %basic;>
<!-- "floats" and their contents -->
<!ELEMENT figure  (#PCDATA|par|%inline;|%likepara;|caption)*>
<!ATTLIST figure  %basic;>
<!ELEMENT table   (#PCDATA|par|%inline;|%likepara;|caption)*>
<!ATTLIST table   %basic;>
<!ELEMENT includegraphics EMPTY>
<!ATTLIST includegraphics %basic;
                          file     CDATA #REQUIRED
                          width    CDATA #IMPLIED
                          height   CDATA #IMPLIED
                          bb       CDATA #IMPLIED
                          scale    CDATA ".5"
                          angle    CDATA #IMPLIED>
<!ELEMENT caption  (#PCDATA|par|%inline;)*>
<!ATTLIST caption  %basic;>
<!-- tabular material -->
<!ELEMENT tabular (hline|row)*>
<!ATTLIST tabular %basic;
                  preamble CDATA  #REQUIRED
                  width    CDATA  #IMPLIED
                  border   CDATA  #IMPLIED>
<!ELEMENT row     (cell)*>
<!ATTLIST row     %basic;>
<!ELEMENT hline   EMPTY>
<!ELEMENT cell    (#PCDATA|%inline;)*>
<!ATTLIST cell    %basic;
                  rowspan CDATA "1"
```

```
                    colspan CDATA "1"
                    align   (left|center|right) "center">
<!-- verbatim material -->
<!ELEMENT verbatim (#PCDATA)>
<!ATTLIST verbatim %basic;
                   xml:space (default|preserve) 'preserve'>
<!ELEMENT verse    (#PCDATA|%inline;)*>
<!ATTLIST verse    %basic;
                   xml:space (default|preserve) 'preserve'>
<!-- things that go in the frontmatter -->
<!ELEMENT newline  EMPTY>
<!ELEMENT title    (#PCDATA|%inline;|thanks|newline)*>
<!ATTLIST title    %basic;>
<!ELEMENT author   (#PCDATA|%inline;|name|thanks|inst)*>
<!ELEMENT name     (#PCDATA|%inline;)*>
<!ATTLIST name     %basic;>
<!ELEMENT thanks   (#PCDATA|%inline;|newline)*>
<!ATTLIST thanks   %basic;>
<!ELEMENT inst     (#PCDATA|%inline;|newline)*>
<!ATTLIST inst     %basic;>
<!ELEMENT date     (#PCDATA)>
<!ATTLIST date     %basic;>
<!ELEMENT abstract (par+)>
<!ATTLIST abstract %basic;>
<!-- structuring -->
<!ENTITY % sect          "stitle, (par|%likepara;|%floats;)* ">
<!ELEMENT stitle         (#PCDATA|%inline;)*>
<!ATTLIST stitle          %basic;>
<!ELEMENT part           (%sect;, (chapter|section|subsection|subsubsection)*)>
<!ATTLIST part            %basic;>
<!ELEMENT chapter        (%sect;, section*)>
<!ATTLIST chapter         %basic;>
<!ELEMENT section        (%sect;, subsection*)>
<!ATTLIST section         %basic;>
<!ELEMENT subsection     (%sect;, subsubsection*)>
<!ATTLIST subsection      %basic;>
<!ELEMENT subsubsection  (%sect;, paragraph*)>
<!ATTLIST subsubsection   %basic;>
<!ELEMENT paragraph      (%sect;, subparagraph*)>
<!ATTLIST paragraph       %basic;>
<!ELEMENT subparagraph   (%sect;)>
<!ATTLIST subparagraph    %basic;>
<!ELEMENT appendix       EMPTY>
<!-- backmatter: index and glossary -->
<!ELEMENT index (par|%likepara;)*>
<!ATTLIST index %basic;>
<!ELEMENT glossary (par|%likepara;)*>
<!ATTLIST glossagry %basic;>
<!-- LaTeX math constructs -->
<!ENTITY % LaTeXmath  "INCLUDE">
<!ENTITY % latexmath.dtd SYSTEM "latexmath.dtd">
<![ %LaTeXmath; [
%latexmath.dtd;
]]>
<!ENTITY % MathML     "IGNORE">
<!ENTITY % latexmml.dtd SYSTEM "latexmml.dtd">
<![ %MathML; [
%latexmml.dtd;
]]>
<!-- Basic XML entities -->
<!ENTITY lt     "&#60;"> <!-- "<" -->
<!ENTITY gt     "&#62;"> <!-- ">" -->
<!ENTITY amp    "&#38;"> <!-- "&" -->
<!ENTITY apos   "&#39;"> <!-- "'" -->
<!ENTITY quot   "&#34;"> <!-- '"' -->
```

B.4.5.2 Математические выражения в LaTeX'е и наше DTD

Как отмечалось в предыдущем разделе, в главном файле DTD с именем `minilatex.dtd` мы фактически не касались вопросов разметки математических выражений.

Вообще говоря, можно либо использовать TeX'овскую разметку для описания математических выражений (как это будет сделано в примере документа, рассматриваемом в следующем разделе), либо решить применять MathML, и в таком случае придется подключить полное определение DTD для этого языка. Фактически, как уже отмечалось, в наш файл `minilatex.dtd` (строки 164–173) были встроены зацепки для работы в каждом из этих двух вариантов. В самом деле, два объекта, «`LaTeXmath`» и «`MathML`», позволяют задать, какая из двух моделей набора математических выражений (разметка LaTeX'а или MathML) должна использоваться. По умолчанию будем использовать LaTeX'овскую разметку (значение для `LaTeXmath` есть `INCLUDE` (строка 164), а для `MathML` устанавливается `IGNORE` (строка 169)).

Посмотрим, что произойдет в каждом из этих двух случаев. Предположим вначале, что имеет место более простая ситуация, когда решено использовать разметку на основе LaTeX'а (по умолчанию для нашего варианта DTD принято именно это значение). Тогда строка 167 будет активной и произойдет подключение файла `latexmath.dtd` к главному DTD. Этот файл имеет следующий вид:

```
1  <!-- latexmath.dtd +++ inline and display math -->
2  <!ELEMENT  inlinemath  (#PCDATA)>
3  <!ATTLIST  inlinemath  %all;>
4  <!ELEMENT  displaymath (#PCDATA)>
5  <!ATTLIST  displaymath %all;>
6  <!ELEMENT  equation    (#PCDATA)>
7  <!ATTLIST  equation    %all;>
8  <!ELEMENT  eqnarray    (#PCDATA)>
9  <!ATTLIST  eqnarray    %all;>
10 <!ELEMENT  align       (#PCDATA)>
11 <!ATTLIST  align       %all;>
```

Файл `latexmath.dtd` включает обычные LaTeX'овские математические контейнеры, содержимое которых декларируется как `#PCDATA`, т. е. никаких ограничений на их внутреннюю структуру не накладывается. Внутри XML-файлов литеры `<`, `>` и `&` должны предваряться обратной косой чертой; отсюда следует, что будет более практичным вводить данные внутрь математических элементов в виде секций `CDATA`, где можно использовать все LaTeX'овские литеры без соблюдения каких-либо специальных предосторожностей (см., в частности, строки 51–53 примера XML-файла в разд. B.4.5.3).

Если, с другой стороны, будет решено использовать разметку языка MathML, для чего значение объекта `MathML` должно быть установлено как `INCLUDE` (а объекта `LaTeXmath` как `IGNORE`), то будет, соответственно, осуществлено подключение файла `latexmml.dtd`, показанного здесь:

```
1  <!-- latexmml.dtd -->
2  <!ELEMENT equation (math)*>
3  <!ATTLIST equation %all;>
4  <!ELEMENT displaymath (math)*>
5  <!ELEMENT inlinemath  (math)*>
```

```
6  <!ELEMENT subeqn (math)*>
7  <!ATTLIST subeqn %all;>
8  <!ELEMENT eqnarray (subeqn)*>
9  <!ATTLIST eqnarray %all;
10                    number (yes|no) "yes">
11 <!ELEMENT align (%inline;)*>
12 <!ATTLIST align %all; >
13
14 <!-- ISO and MathML DTDs and entities -->
15 <!--Added Math Symbols: Arrows-->
16 <!ENTITY % isoamsae.dtd SYSTEM "isoamsae.dtd">
17 <!--Added Math Symbols: Binary Operators-->
18 <!ENTITY % isoamsbe.dtd SYSTEM "isoamsbe.dtd">
19 <!--Added Math Symbols: Delimiters-->
20 <!ENTITY % isoamsce.dtd SYSTEM "isoamsce.dtd">
21 <!--Added Math Symbols: Negated Relations-->
22 <!ENTITY % isoamsne.dtd SYSTEM "isoamsne.dtd">
23 <!--Added Math Symbols: Ordinary-->
24 <!ENTITY % isoamsoe.dtd SYSTEM "isoamsoe.dtd">
25 <!--Added Math Symbols: Relations-->
26 <!ENTITY % isoamsre.dtd SYSTEM "isoamsre.dtd">
27 <!--General Technical-->
28 <!ENTITY % isoteche.dtd SYSTEM "isoteche.dtd">
29 <!--Numbers and Currency symbols-->
30 <!ENTITY % isonume.dtd SYSTEM "isonume.dtd">
31 <!--MathML Aliases (From ISO PUB,DIA,NUM)-->
32 <!ENTITY % mmaliase.dtd SYSTEM "mmaliase.dtd">
33 <!--Greek Symbols-->
34 <!ENTITY % isogrk3e.dtd SYSTEM "isogrk3e.dtd">
35 <!--Math Script Font-->
36 <!ENTITY % isomscre.dtd SYSTEM "isomscre.dtd">
37 <!--Math Open Face Font-->
38 <!ENTITY % isomopfe.dtd SYSTEM "isomopfe.dtd">
39 <!--MathML Entities-->
40 <!ENTITY % mmlent.dtd SYSTEM "mmlent.dtd">
41 <!--Main MathML DTD -->
42 <!ENTITY % mathml.dtd    SYSTEM "mathml.dtd">
43
44 %mathml.dtd;
45 %isoamsae.dtd;
46 %isoamsbe.dtd;
47 %isoamsce.dtd;
48 %isoamsne.dtd;
49 %isoamsoe.dtd;
50 %isoamsre.dtd;
51 %isoteche.dtd;
52 %isonume.dtd;
53 %mmaliase.dtd;
54 %isogrk3e.dtd;
55 %isomscre.dtd;
56 %isomopfe.dtd;
57 %mmlent.dtd;
```

Поскольку MathML DTD использует элемент `math` как глобальный контейнер, в файле `latexmml.dtd` надо декларировать LaTeX'овские контейнеры как состоящие из элементов `math` (строки 2 и 4–6). Оставшаяся часть приведенного выше файла DTD включает набор определений объектов (строки 14–40), а также ссылку на собственно MathML DTD (строка 42). Файлы, определенные этими объектами, подключаются в самом конце (строки 44–57).

Файлы DTD с именами `minilatex.dtd` и `latexmml.dtd` могут быть использованы с XML-файлом, полученным на основе LaTeX'овского тестового файла,

использовавшегося на протяжении всей этой книги и показанного в приложении A.1. Ниже приведено начало этого файла:

```
1  <?xml version="1.0"?>
2  <!DOCTYPE document SYSTEM "minilatex.dtd"[
3  <!ENTITY % MathML     "INCLUDE">
4  <!ENTITY % LaTeXmath  "IGNORE">
5  <!ENTITY % inline.new  "|math">
6  <!ENTITY aacute "&#xOOE1;">
7  <!ENTITY OverBar "[OverBar]">
8  <!ENTITY negationslash "/">
9  ]>
10 <document>
11 <frontmatter>
12 <title>Simulation of Energy Loss Straggling</title>
13 <author>Maria Physicist</author>
14 <date> January 14, 1999</date>
15 </frontmatter>
16 <bodymatter> ...
```

Начало этого файла определяет DTD, которое следует использовать, и устанавливает значения объектов `MathML` и `LaTeXmath` соответственным образом. Определим также элемент `math`, необходимый в качестве контейнера при использовании разметки MathML, введением его в параметрический объект `inline.new`, так что он будет включен в элементы `inline` в файле `minilatex.dtd` (строка 19). Оставшиеся строки (6–8) определяют объекты, на которые есть ссылки в тестовом документе, но которые не определены в уже загруженных наборах ссылок на объекты.

B.4.5.3 Пример документа

Покажем теперь, как можно использовать DTD `minilatex`, введенное в разд. B.4.5.1. В данном разделе рассмотрим пример XML-файла, размеченный в соответствии с этим DTD. Содержимое рассматриваемого документа частично основывается на тексте рисунков 9.2 и 9.3 в главе книги *The LaTeX Companion* (Goossens et al. (1994)), посвященной системе Babel.

В строке 1 видна обрабатывающая инструкция `<?xml`, которая декларирует использование кодировки Latin 1 (ISO-8859-1) в рассматриваемом документе (она используется в немецком тексте в строках 60–99 и 128–131, а также во французском тексте в строках 100–126 и 132–135). Определим также (в строках 3–8) несколько объектов, которые встречаются в тексте нашего документа и не определены во внешнем подмножестве DTD. Они настраиваются применительно к трансляции в LaTeX. Для трансляции в HTML или какой-либо другой формат эти объекты должны быть определены соответственно. То же самое проделывается и для некоторых разделов текста, содержимое которых поместим внутрь секций `CDATA`, поскольку знаем, что в качестве постпроцессора будет использоваться LaTeX (например, строки 40 и 41, а также математические сегменты строк 51–53). Фактически нет необходимости использовать секцию `CDATA` для подготовки вывода непосредственно для LaTeX'а, так как можно использовать XML-объекты для представления специальных символов, как это показано в строках 42–43. Какой из подходов выбрать — диктуется лишь со-

ображениями удобства. Заметим, однако, что символу & предшествует обратная косая (строки 40, 42 и 43), поскольку известно, что амперсанд надо предварять обратной косой чертой, чтобы его можно было использовать в качестве литеры в LaTeX'е.

В конце внутреннего подмножества определим элемент `footnote`, который должен быть включен в параметрический объект `inline` (строка 19 в `minilatex` DTD), чтобы быть доступным для использования в примере документа, приводимом ниже. Поэтому же введем его в параметрический объект `inline.new` (строка 9) и объявим его содержимое как элементы `#PCDATA` и `par` (строка 10). Однако элементы `par`, согласно строке 85 файла `minilatex.dtd`, ссылаются на объект `inline`. Тогда, поскольку наш элемент `footnote` также составляет часть объекта `inline`, обязательно нужно убедиться, что неумышленно не сформированы вложенные элементы `footnote` (определение DTD не содержит запретов на такую конструкцию, но LaTeX не допускает ее; см. разд. 7.6.1.3, где описывается, что и как надо делать в этой ситуации в XSL).

Две основные части документа — его предварительная служебная часть `frontmatter` (строки 13–19) и основной текст `bodymatter` (строки 20–137) — различимы вполне отчетливо. Аналогичным образом, благодаря элементам секционирования (`<section>`, `<subsection>` и т. д.) и возможности их вложения, можно достаточно просто реконструировать и проверить на согласованность иерархическую структуру древовидного элемента секционирования.

Достаточно просто организуется работа с несколькими языками внутри одного и того же документа. Рассматриваемый здесь документ начинается с раздела на английском языке (это язык документа по умолчанию, согласно декларации в строке 12 рассматриваемого примера документа), затем идет раздел на немецком языке (начиная со строки 60 нашего документа), после чего помещается раздел на французском языке (начиная со строки 100). Кроме того, два библиографических элемента `bibitem` имеют различные значения языковых атрибутов (строки 128–131 записаны на немецком языке, тогда как строки 132–135 — на французском; окружающий элемент `thebibliography` имеет значение по умолчанию).

Команда `includegraphics` и ее атрибуты определяются в строках 93–99 в `minilatex` DTD. Она используется в строке 74 рассматриваемого примера, где выполняется подключение изображения в формате Encapsulated PostScript внутри элемента `figure` (строки 73–76).

Оставшаяся часть выполняемой разметки совсем проста, и поэтому соответствующие части файла-примера комментировать мы не будем.

Тем не менее, обратите внимание на следующий важный момент. Как отмечалось в начале данного раздела, в строках 51–53 напрямую используются LaTeX'овские математические обозначения внутри элементов `inlinemath` (это принято по умолчанию в `minilatex` DTD). Используем секции `CDATA`, чтобы избавиться от необходимости применять ссылки на XML-объекты для некоторых литер, таких, как <, > и &. Преобразуя XML-файлы с помощью стиля XSL, следует позаботиться о декларировании корректного типа документа, чтобы

представить такие специальные знаки в файле вывода соответственно потребностям целевого приложения.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE document SYSTEM "minilatex.dtd"[
<!ENTITY LaTeX  "\LaTeX{}">
<!ENTITY TeX    "\TeX{}">
<!ENTITY dots   "\dots">
<!ENTITY endash "--">
<!ENTITY emdash "---">
<!ENTITY nbsp   "~">
<!ENTITY % inline.new "|footnote">
<!ELEMENT footnote (#PCDATA|par)*>
]>
<document class="article" xml:lang="en">
<frontmatter>
<title>The &LaTeX; DTD and multiple languages</title>
<author><name>Michel Goossens</name>
<thanks>Partly from an example in <emph>The &LaTeX; Companion</emph>
</thanks></author>
<date>August 4th, 1998</date>
</frontmatter>
<bodymatter>
<section id="sec-en">
<stitle>The basic principles</stitle>
<par>
This is an example input file. We start in English to show the
principle. You should especially pay attention that we have used
slightly different notation for some of the common &LaTeX; constructs,
such as the dashes, which come in three sizes: an intra-word dash, a
medium dash for number ranges like 1&endash;2, and a punctuation
dash&emdash;like this.  Text can be emphasized as <emph>shown
here</emph>. An ellipsis is made with &dots; Footnotes<footnote>This
is a simple footnote.<par>It can also contain <texttt>par</texttt>
elements.</par></footnote> are tricky constructs, since one must be
careful not to nest them.
</par>
<subsection>
<stitle>Dealing with special characters</stitle>
<par>
XML has a different set of reserved characters than &LaTeX;, in
particular, when you want to use any of the three characters
<texttt><![CDATA[\&]]></texttt>, <texttt><![CDATA[<]]></texttt>,
and <texttt><![CDATA[>]]></texttt>, you should enter them as
<texttt>\&amp;amp;</texttt>, <texttt>\&amp;lt;</texttt>, and
<texttt>\&amp;gt;</texttt>, respectively.
</par>
</subsection>
<subsection id="sec-math">
<stitle>&LaTeX; and mathematical formulae</stitle>
<par>
&LaTeX; and <emph>a fortiori</emph> &TeX; are very good at typesetting
mathematical formulae, like
<inlinemath><![CDATA[x- 3 y + z < 7]]></inlinemath> or
<inlinemath><![CDATA[a_{1} > x^{2n} + y^{2n} > x']]></inlinemath> or
<inlinemath><![CDATA[(A, B) = \sum_{i} a_{i} b_{i}]]></inlinemath>.
Do not forget that for reasons of consistency, if you want to refer to
a variable in one of the formulae, such as the symbol
<inlinemath>x</inlinemath>, you must also use math mode in the text.
</par>
</subsection>
</section>
<section xml:lang="de">
<stitle>Beispiel eines Textes in deutscher Sprache</stitle>
<subsection>
<stitle>Eine EPS Abbildung</stitle>
```

```
<par>
Dieser Abschnitt zeigt, wie man eine PostScript-Abbildung
<cite refid="bib-PS"/> in ein Dokument einbinden kann.
Abbildung <ref refid="fig-psfig"/> wurde mit dem Befehl
<verbatim>
<![CDATA[\includegraphics[width="3cm"]{file="colorcir.eps}]]>
</verbatim>
in den Text aufgenommen.
</par>
<figure id="fig-psfig">
<includegraphics width="3cm" file="colorcir"/>
<caption xml:lang="de">Ein EPS Bild</caption>
</figure>
</subsection>
<subsection>
<stitle>Beispiel einer Tabelle</stitle>
<par>Die Tabelle <ref refid="tab-exag"/> auf Seite <pageref
refid="tab-exag"/> zeigt eine Tabelle.
</par>
<table id="tab-exag">
<caption xml:lang="de">
Eingabe der deutschen Zusatzzeichen in &LaTeX;</caption>
<tabular preamble="ccccccc">
<row>
<cell><texttt>&#34;a</texttt> ф</cell>
<cell><texttt>&#34;A</texttt> -</cell>
<cell><texttt>&#34;o</texttt> I</cell>
<cell><texttt>&#34;O</texttt> </cell>
<cell><texttt>&#34;u</texttt> Ю</cell>
<cell><texttt>&#34;U</texttt> </cell>
<cell><texttt>&#34;s</texttt> &#223;</cell>
</row>
</tabular>
</table>
</subsection>
</section>
<section xml:lang="fr">
<stitle>Continuation du texte en franчais</stitle>
<subsection id="sec-list">
<stitle>Traiter les listes</stitle>
<par>
Les listes sont utilisщes frщquemment pour structurer ou mettre
en щvidence certains щlщments d'un document (voir <cite refid="bib-Liste"/>).
</par>
<itemize>
  <item>Ceci est le premier щlщment d'une liste non-ordonnщe. Chaque щlщment
    de ce type de liste est prщcщdщ d'un signe distinctif, comme une
    puce, un tiret, etc.
  </item>
  <item>Ce second щlщment de la mъme liste contient une liste de
    <emph>description</emph> imbriquщe.
    <description>
      <term>XML</term>
      <item>Meta langage pour dщfinir des classes de documents</item>
      <term>XLL</term>
      <item>Langage pour dщfinir des hyperliens entre diffщrentes
      parties de documents XML</item>
    </description>
    Nous continuons notre texte p l'intщrieur de la premiиre liste.
  </item>
</itemize>
</subsection>
</section>
<bibliography>
<bibitem id="bib-PS" xml:lang="de">
Adobe Inc. <emph>PostScript Handbuch (2. Auflage)</emph>
           Addison-Wesley (Deutschland) GmbH, Bonn, 1991.
```

```
</bibitem>
<bibitem id="bib-Liste" xml:lang="fr">
Michel Goossens. Personnaliser les listes &LaTeX;.
        <emph>Cahiers GUTenberg</emph>, 17:32&endash;48, mai 1994.
</bibitem>
</bibliography>
</bodymatter>
</document>
```

B.4.5.4 Подготовка варианта для печати

Подтвердить правильность файла-примера `minilatexexa.xml`, представленного в предыдущем разделе, можно с помощью одного из многочисленных доступных программ грамматического разбора XML с привлечением DTD `minilatex.dtd`. Например, прогоним его через программу `xml4j`, обсуждавшуюся в разд. 6.6.5.6.

```
xml4j.sh minilatexexa.xml
minilatex.dtd: 18, 25: Warning: Entity name, "inline.new", already
defined. This declaration will be ignored.
```

Этот синтаксический анализатор показывает, что приведенный пример вполне можно считать корректным по отношению к `minilatex` DTD. Вместе с тем, анализатор предупреждает, что определение объекта `inline.new` в DTD пропущено. Это как раз то, что и было необходимо, поскольку, как объяснялось в разд. 6.5.4, внутреннее подмножество DTD читается первым, так что определения во внутреннем подмножестве перекрывают те, что имеются во внешнем DTD (это верно только для определений объектов и атрибутов, *но не* для элементов, которые таким образом нельзя переопределить с помощью внешнего DTD).

Интересный пример применения — формирование варианта нашего файла для вывода на печать. Можно использовать стиль XSL с именем `minilatex.xsl`, который показан ниже. Цель его — транслировать XML-объект `minilatexexa.xml` в файл LaTeX'а.

```
<?xml version='1.0'?>
<!-- minilatex.xsl -->
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns="http://www.tug.org/latex">

<xsl:output method="text" indent="no" encoding="ISO-8859-1"/>

<xsl:strip-space elements="*"/>

<xsl:template name="label">
  <xsl:apply-templates/>
  <xsl:if test="../@id"><xsl:text>\label{</xsl:text>
  <xsl:value-of select="../@id"/><xsl:text>}</xsl:text></xsl:if>
<xsl:text>}
</xsl:text>
</xsl:template>

<xsl:template match="document">
<xsl:text>\documentclass[]{</xsl:text>
<xsl:value-of select="@class"/>
```

```
<xsl:text>}
\usepackage[dvips]{graphicx}
\usepackage[T1]{fontenc}
\begin{document}
</xsl:text>
<xsl:apply-templates/>
<xsl:text>\end{document}
</xsl:text>
</xsl:template>

<!--   ================ Frontmatter element ====================== -->
<xsl:template match="frontmatter">
  <xsl:apply-templates/>
<xsl:text>
\maketitle
</xsl:text>
</xsl:template>
<xsl:template match="frontmatter/title">
<xsl:text>
\title{</xsl:text>
  <xsl:apply-templates/>
<xsl:text>}</xsl:text>
</xsl:template>
<xsl:template match="frontmatter/author">
<xsl:text>
\author{</xsl:text>
  <xsl:apply-templates/>
<xsl:text>}
</xsl:text>
</xsl:template>
<xsl:template match="frontmatter/author/name">
  <xsl:apply-templates/>
</xsl:template>
<xsl:template match="frontmatter/author/thanks">
<xsl:text>
\thanks{</xsl:text>
  <xsl:apply-templates/>
<xsl:text>}</xsl:text>
</xsl:template>
<xsl:template match="frontmatter/author/inst">
<xsl:text>
\thanks{</xsl:text>
  <xsl:apply-templates/>
<xsl:text>}</xsl:text>
</xsl:template>
<xsl:template match="frontmatter/date">
<xsl:text>
\date{</xsl:text>
  <xsl:apply-templates/>
<xsl:text>}
</xsl:text>
</xsl:template>
<xsl:template match="frontmatter/abstract">
<xsl:text>
\begin{abstract}
</xsl:text>
  <xsl:apply-templates/>
<xsl:text>
\end{abstract}
</xsl:text>
</xsl:template>
<xsl:template match="frontmatter/keywords">
<xsl:text>\keywords{</xsl:text>
  <xsl:apply-templates/>
<xsl:text>}</xsl:text>
</xsl:template>

<!--   ==================== Bodymatter element ================== -->
```

```
<xsl:template match="bodymatter|part|chapter|section|subsection|
                     subsubsection|paragraph|subparagraph">
  <xsl:apply-templates/>
</xsl:template>
<!--   ================== Section headings ====================== -->
<xsl:template match="part/stitle">
  <xsl:text>\part{</xsl:text><xsl:call-template name="label"/>
</xsl:template>
<xsl:template match="chapter/stitle">
  <xsl:text>\chapter{</xsl:text><xsl:call-template name="label"/>
</xsl:template>
<xsl:template match="section/stitle">
  <xsl:text>\section{</xsl:text><xsl:call-template name="label"/>
</xsl:template>
<xsl:template match="subsection/stitle">
  <xsl:text>\subsection{</xsl:text><xsl:call-template name="label"/>
</xsl:template>
<xsl:template match="subsubsection/stitle">
  <xsl:text>\subsubsection{</xsl:text><xsl:call-template name="label"/>
</xsl:template>
<xsl:template match="paragraph/stitle">
  <xsl:text>\paragraph{</xsl:text><xsl:call-template name="label"/>
</xsl:template>
<xsl:template match="subparagraph/stitle">
  <xsl:text>\subparagraph{</xsl:text><xsl:call-template name="label"/>
</xsl:template>
<!--   ==================== Font changes ======================== -->
<xsl:template match="emph">
<xsl:text>\emph{</xsl:text>
  <xsl:apply-templates/>
<xsl:text>}</xsl:text>
</xsl:template>
<xsl:template match="textbf">
<xsl:text>\textbf{</xsl:text>
  <xsl:apply-templates/>
<xsl:text>}</xsl:text>
</xsl:template>
<xsl:template match="textsc">
<xsl:text>\textsc{</xsl:text>
  <xsl:apply-templates/>
<xsl:text>}</xsl:text>
</xsl:template>
<xsl:template match="textsf">
<xsl:text>\textsf{</xsl:text>
  <xsl:apply-templates/>
<xsl:text>}</xsl:text>
</xsl:template>
<xsl:template match="textsl">
<xsl:text>\textsl{</xsl:text>
  <xsl:apply-templates/>
<xsl:text>}</xsl:text>
</xsl:template>
<xsl:template match="texttt">
<xsl:text>\texttt{</xsl:text>
  <xsl:apply-templates/>
<xsl:text>}</xsl:text>
</xsl:template>
<!--   ================== Cross-references ====================== -->
<xsl:template match="cite">
  <xsl:text>\cite{</xsl:text>
  <xsl:value-of select="@refid"/>
  <xsl:text>}</xsl:text>
</xsl:template>
<xsl:template match="pageref">
  <xsl:text>\pageref{</xsl:text>
  <xsl:value-of select="@refid"/>
  <xsl:text>}</xsl:text>
</xsl:template>
```

```
158  <xsl:template match="ref">
159    <xsl:text>\ref{</xsl:text>
160    <xsl:value-of select="@refid"/>
161    <xsl:text>}</xsl:text>
162  </xsl:template>
163  <!--   =============== quotes, footnotes, verbatim ================ -->
164  <xsl:template match="footnote">
165  <xsl:text>\footnote{</xsl:text>
166    <xsl:apply-templates/>
167  <xsl:text>}</xsl:text>
168  </xsl:template>
169  <xsl:template match="quote">
170  <xsl:text>
171  \begin{quote}</xsl:text>
172    <xsl:apply-templates/>
173  <xsl:text>\end{quote}</xsl:text>
174  </xsl:template>
175  <xsl:template match="quotation">
176  <xsl:text>
177  \begin{quotation}</xsl:text>
178    <xsl:apply-templates/>
179  <xsl:text>\end{quotation}</xsl:text>
180  </xsl:template>
181  <xsl:template match="verbatim">
182  <xsl:text>
183  \begin{verbatim}</xsl:text>
184    <xsl:apply-templates/>
185  <xsl:text>\end{verbatim}</xsl:text>
186  </xsl:template>
187  <!--   ======================== Lists ============================ -->
188  <xsl:template match="description">
189  <xsl:text>
190  \begin{description}
191  </xsl:text>
192    <xsl:apply-templates/>
193  <xsl:text>
194  \end{description}
195  </xsl:text>
196  </xsl:template>
197  <xsl:template match="description/term">
198  <xsl:text>
199  \item[</xsl:text>
200    <xsl:apply-templates/>
201  <xsl:text>]</xsl:text>
202  </xsl:template>
203  <xsl:template match="description/item">
204    <xsl:apply-templates/>
205  </xsl:template>
206  <xsl:template match="enumerate">
207  <xsl:text>
208  \begin{enumerate}
209  </xsl:text>
210    <xsl:apply-templates/>
211  <xsl:text>\end{enumerate}
212  </xsl:text>
213  </xsl:template>
214  <xsl:template match="itemize">
215  <xsl:text>
216  \begin{itemize}
217  </xsl:text>
218    <xsl:apply-templates/>
219  <xsl:text>\end{itemize}
220  </xsl:text>
221  </xsl:template>
222  <xsl:template match="enumerate|itemize/item">
223  <xsl:text>
224  \item </xsl:text>
225    <xsl:apply-templates/>
```

```
226  </xsl:template>
227  <xsl:template match="bibliography">
228  <xsl:text>
229  \begin{thebibliography}{99}
230  </xsl:text>
231    <xsl:apply-templates/>
232  <xsl:text>
233  \end{thebibliography}
234  </xsl:text>
235  </xsl:template>
236  <xsl:template match="bibliography/bibitem">
237  <xsl:text>\bibitem{</xsl:text>
238  <xsl:value-of select="@id"/>
239  <xsl:text>}</xsl:text>
240  <xsl:apply-templates/>
241  </xsl:template>
242  <!--   ====================== Mathematics ======================== -->
243  <xsl:template match="inlinemath">
244  <xsl:text>$</xsl:text>
245    <xsl:apply-templates/>
246  <xsl:text>$</xsl:text>
247  </xsl:template>
248  <xsl:template match="displaymath">
249  <xsl:text>
250  \begin{displaymath}
251  </xsl:text>
252    <xsl:apply-templates/>
253  <xsl:text>
254  \end{displaymath}
255  </xsl:text>
256  </xsl:template>
257  <xsl:template match="equation">
258  <xsl:text>
259  \begin{equation}
260  </xsl:text>
261    <xsl:apply-templates/>
262  <xsl:text>
263  \end{equation}
264  </xsl:text>
265  </xsl:template>
266  <xsl:template match="eqnarray">
267  <xsl:text>
268  \begin{eqnarray}
269  </xsl:text>
270    <xsl:apply-templates/>
271  <xsl:text>
272  \end{eqnarray}
273  </xsl:text>
274  </xsl:template>
275  <!--   ====================== A paragraph ======================== -->
276  <xsl:template match="par">
277  <xsl:text>
278  \par
279  </xsl:text>
280    <xsl:apply-templates/>
281  </xsl:template>
282  <!--   ======================== Tabular ========================== -->
283  <xsl:template match="tabular">
284  <xsl:text>
285  \begin{tabular}{</xsl:text>
286    <xsl:value-of select="@preamble"/><xsl:text>}
287  </xsl:text>
288    <xsl:apply-templates/>
289  <xsl:text>
290  \end{tabular}
291  </xsl:text>
292  </xsl:template>
293  <xsl:template match="tabular/row">
```

```
294   <xsl:apply-templates/>
295   <xsl:text>\\
296   </xsl:text>
297   </xsl:template>
298   <xsl:template match="tabular/row/cell">
299   <xsl:apply-templates/><xsl:text>&amp;</xsl:text>
300   </xsl:template>
301   <xsl:template match="tabular/row/cell[position()=last()]" priority="2">
302   <xsl:apply-templates/>
303   </xsl:template>
304   <!--   ================== "floats" and their contents ============= -->
305   <xsl:template match="figure">
306   <xsl:text>
307   \begin{figure}\centering
308   </xsl:text>
309     <xsl:apply-templates/>
310   <xsl:text>\end{figure}
311   </xsl:text>
312   </xsl:template>
313   <xsl:template match="table">
314   <xsl:text>
315   \begin{table}\centering
316   </xsl:text>
317     <xsl:apply-templates/>
318   <xsl:text>\end{table}
319   </xsl:text>
320   </xsl:template>
321   <xsl:template match="figure/caption | table/caption">
322     <xsl:text>\caption{</xsl:text><xsl:call-template name="label"/>
323   </xsl:template>
324   <!--   ==================== Includegraphics ====================== -->
325   <xsl:template match="includegraphics">
326   <xsl:text>
327   \includegraphics[</xsl:text>
328   <xsl:if test="@width"><xsl:text>width=</xsl:text>
329       <xsl:value-of select="@width"/><xsl:text>, </xsl:text></xsl:if>
330   <xsl:if test="@height"><xsl:text>height=</xsl:text>
331       <xsl:value-of select="@height"/><xsl:text>, </xsl:text></xsl:if>
332   <xsl:if test="@bb"><xsl:text>bb="</xsl:text>
333       <xsl:value-of select="@bb"/><xsl:text>, </xsl:text></xsl:if>
334   <xsl:if test="@angle"><xsl:text>angle=</xsl:text>
335       <xsl:value-of select="@angle"/><xsl:text>, </xsl:text></xsl:if>
336   <xsl:if test="@scale"><xsl:text>scale=</xsl:text>
337       <xsl:value-of select="@scale"/><xsl:text></xsl:text></xsl:if>
338   <xsl:text>]{</xsl:text><xsl:value-of select="@file"/><xsl:text>}
339   </xsl:text>
340   </xsl:template>
341
342   </xsl:stylesheet>
```

Бóльшая часть кода, приведенного выше, кажется более или менее тривиальной после ознакомления с материалом в разд. 7.6, где объяснялось, каким образом надо писать стили XSL и как они могут формировать вывод в формате HTML или LaTeX. Одна из важных особенностей содержится в строке 7, где объявляется, что требуется записать текстовый файл таким образом, чтобы специальные литеры XML(&, < и >) выводились в неизменном виде. В выводном файле будет использоваться кодировка ISO-8859-1.

Для каждого элемента в рассматриваемом примере документа дается уникальное правило, которое сообщает XSL-анализатору, как транслировать данный элемент в LaTeX. При формировании DTD большинство XML-элементов отображается в их LaTeX'овские эквиваленты по принципу «один-в-один» (правила-шаблоны для элементов, связанных с переключением шрифтов, со-

держащиеся в строках 116–146 — достаточно ясный пример такого использования).

Рассмотрим более детально три области: перекрестные ссылки, материал табличного характера, работу с необязательными атрибутами.

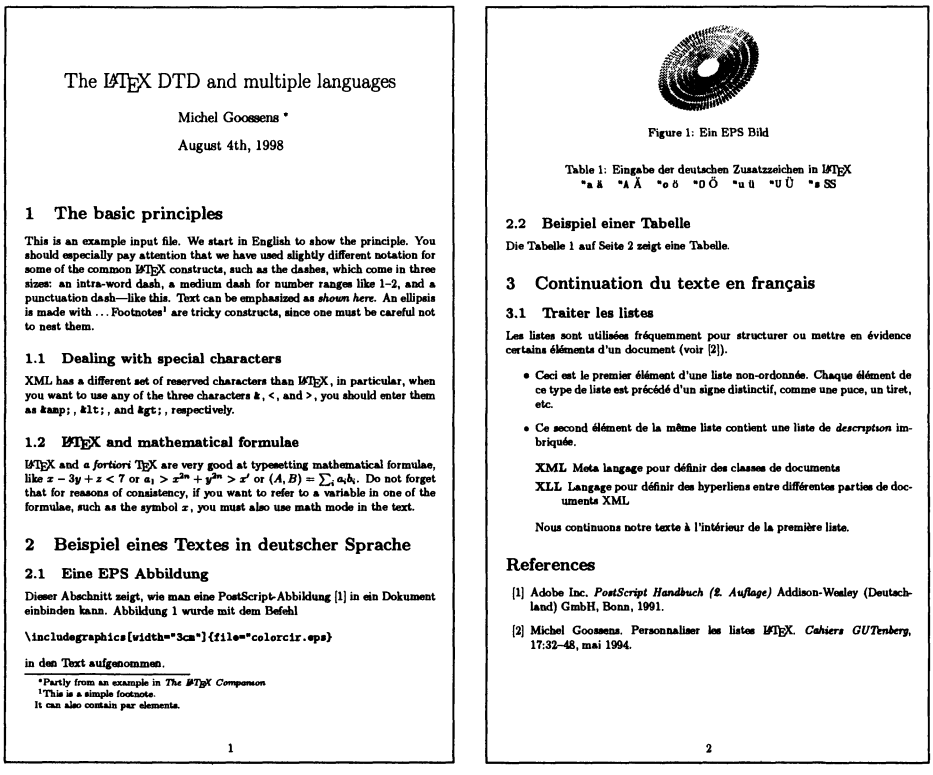
Как отмечалось в разд. B.4.5.1, работа с перекрестными ссылками основывается на XML-атрибутах `id` и `idref`. Поэтому определим именованный шаблон `«label»` (строки 11–17), помещающий LaTeX'овскую команду `\label` внутрь элементов, для которых это имеет смысл в LaTeX'е. Для данного родительского элемента надо посмотреть, имеется ли атрибут `id` (структурный элемент `xsl:if` в строках 13–14). Если такой атрибут найден, вставим команду `\label` с ключом, извлеченным из атрибута `id` родительского XML-элемента (строка 14). Закончим это макроопределение закрывающей скобкой для завершающего элемента (строки 15–16).

В представленном варианте XSL-файла будем ссылаться на этот именованный шаблон для всех элементов `stitle` из команд секционирования (строки 96, 99, 102, 105, 108, 111 и 114) и команды `caption` (строка 322). Команда `bibitem` (строки 236–241) использует свой собственный идентификатор в качестве ключа (строка 238).

Ссылки на эти ключи делаются с помощью элементов `cite` (строки 148–152), `pageref` (строки 153–157) и `ref` (строки 158–162). В каждом из этих случаев просто берется значение атрибута `refid` и вводится в качестве аргумента в соответствующую LaTeX'овскую команду.

Табличный материал (строки 283–292), задаваемый с помощью элемента `tabular`, должен преобразовываться в формат, совместимый с LaTeX'овским окружением `tabular`. Вначале берется преамбула (строка 286), а затем осуществляется обработка строк таблицы (строки 293–297), которая заключается в том, что в конце каждой из этих строк ставится знак \\ (строка 295). Ячейки таблицы требуют более тонкого подхода, так как надо аккуратно обращаться с литерами-разделителями `&` для столбцов таблицы. В самом деле, надо различать нетерминальные столбцы (строки 298–300), где добавляется разделитель `&`, следующий за контентом (строка 299) и последним столбцом в данной строке таблицы (которая выбирается с помощью предиката XPath, используя функции `posiiton` и `last`; см. строку 301), где надо поместить только контент (строка 302).

Наконец, в завершающей части XSL-файла рассмотрим команду `includegraphics` (строки 325–340). Незначительная сложность в данном случае состоит в том, что надо тщательно проверять, какие из необязательных атрибутов уже заданы. Каждый из этих атрибутов имеет свой собственный структурный элемент `xsl:if` для проверки его наличия (строки 328–337). Для каждого атрибута, который был задан, берется его значение и вводится согласно соответствующей строке выбора для LaTeX'а (например, `width= в` строке 328). В конце рассматриваемого файла выводится обязательное имя (задается в XML-файле с атрибутом `file`), помещаемое в фигурных скобках (строка 338).



The LaTeX DTD and multiple languages

Michel Goossens *

August 4th, 1998

1 The basic principles

This is an example input file. We start in English to show the principle. You should especially pay attention that we have used slightly different notation for some of the common LaTeX constructs, such as the dashes, which come in three sizes: an intra-word dash, a medium dash for number ranges like 1–2, and a punctuation dash—like this. Text can be emphasized as *shown here*. An ellipsis is made with ... Footnotes[1] are tricky constructs, since one must be careful not to nest them.

1.1 Dealing with special characters

XML has a different set of reserved characters than LaTeX, in particular, when you want to use any of the three characters &, <, and >, you should enter them as &, <, and >, respectively.

1.2 LaTeX and mathematical formulae

LaTeX and *a fortiori* TeX are very good at typesetting mathematical formulae, like $x - 3y + z < 7$ or $a_1 > x^{2n} + y^{2n} > x'$ or $(A, B) = \sum_i a_i b_i$. Do not forget that for reasons of consistency, if you want to refer to a variable in one of the formulae, such as the symbol x, you must also use math mode in the text.

2 Beispiel eines Textes in deutscher Sprache

2.1 Eine EPS Abbildung

Dieser Abschnitt zeigt, wie man eine PostScript-Abbildung [1] in ein Dokument einbinden kann. Abbildung 1 wurde mit dem Befehl

```
\includegraphics[width="3cm"]{file="colorcir.eps}
```

in den Text aufgenommen.

*Partly from an example in *The LaTeX Companion*
[1]This is a simple footnote.
It can also contain par elements.

1

Figure 1: Ein EPS Bild

Table 1: Eingabe der deutschen Zusatzzeichen in LaTeX

"a	ä	"A	Ä	"o	ö	"O	Ö	"u	ü	"U	Ü	"s	SS

2.2 Beispiel einer Tabelle

Die Tabelle 1 auf Seite 2 zeigt eine Tabelle.

3 Continuation du texte en français

3.1 Traiter les listes

Les listes sont utilisées fréquemment pour structurer ou mettre en évidence certains éléments d'un document (voir [2]).

- Ceci est le premier élément d'une liste non-ordonnée. Chaque élément de ce type de liste est précédé d'un signe distinctif, comme une puce, un tiret, etc.
- Ce second élément de la même liste contient une liste de *description* imbriquée.

  XML Meta langage pour définir des classes de documents

  XLL Langage pour définir des hyperliens entre différentes parties de documents XML

  Nous continuons notre texte à l'intérieur de la première liste.

References

[1] Adobe Inc. *PostScript Handbuch (2. Auflage)* Addison-Wesley (Deutschland) GmbH, Bonn, 1991.

[2] Michel Goossens. Personnaliser les listes LaTeX. *Cahiers GUTenberg*, 17:32–48, mai 1994.

2

Рис. B.1. PostScript'овский вывод нашего XML-документа, основанного на LaTeX'е.

Если обработать теперь рассматриваемый файл-пример `minilatexexa.xml` вместе с показанным выше XSL-стилем `minilatex.xsl` с помощью программы-процессора `xt`, то получим исходный LaTeX'овский файл `minilatexexa.tex`.

```
xt minilatexexa.xml minilatex.xsl minilatexexa.tex
```

После прогона файла `minilatexexa.tex` через LaTeX'овский процессор получим сформированный PostScript-файл, показанный на рис. B.1.

B.5 Преобразование HTML в XML

В последние годы популярность HTML росла экспоненциально, что связано с возросшей важностью процесса распространения документов через Сеть. HTML используется все большим числом приложений, причем часто в них вводятся новые элементы, так что сейчас HTML превратился в настоящий кошмар в смысле обеспечения совместимости. Все труднее становится гарантировать

межплатформенную мобильность приложений, и это является самым серьезным препятствием для быстрой эволюции HTML в Интернете.

Поэтому решение проблемы распределенной обработки документов на постоянно растущем множестве различных компьютеров и коммуникационных средств настоятельно требует появления у HTML преемника. Участники рабочего совещания *Future of HTML*, проведенного в мае 1998 г., пришли к соглашению, что положение дел можно исправить с помощью нового поколения HTML, которое можно было бы определить как совокупность наборов XML-элементов.

Этот новый HTML включает *ядро набора элементов (core tag set)*, используемых для разметки заголовков, абзацев, списков, гипертекстовых ссылок, изображений и других базовых документных идиом. С другой стороны, разметка форм, таблиц, мультимедийных структурных элементов, графики и т. д. определяется наборами элементов, разными для каждого из этих видов структурных элементов документа. В таких наборах элементов принят синтаксис XML, допускается использование их в произвольных комбинациях. Это дает возможность размечать намного более широкий набор разновидностей документов, чем это возможно при «монолитной» схеме разметки.

Четкое разделение содержания и формы сейчас насаждается все более жестко, так что значимость стилей существенно возрастает при связывании таких характеристик, как цвет, шрифт, макет документа с его структурными элементами. Стили помогают также в описании того, как адаптировать или преобразовать разметку, чтобы оптимизировать ее воспроизведение на устройствах различных видов.

Для того чтобы описать, каким образом HTML-элементы осуществляют поддержку устройств некоторого заданного типа и как реализуются средства стилей, используются *профили согласования (conformance profiles)*. Принцип здесь заключается в том, что два различающихся устройства должны воспроизводить разметку тем же самым способом, как если бы они принадлежали одному и тому же профилю согласования, а рассматриваемый документ использует только средства из этого профиля.

Профиль согласования не только задает, какие элементные наборы HTML поддерживаются и какова функциональность листа стиля, но также содержит информацию относительно семантических ограничений, формата данных, поддержки сценариев и т. д. Некоторое устройство может послать свой профиль на Web-сервер, где установлено программное обеспечение, решающее задачу преобразования разметки в простой и надежный вариант, возвращая документ, подстроенный под конкретное требуемое устройство. Профили документа и устройства будут кодировать информацию в терминах формата описания ресурсов RDF[↪RDF], а также в терминах набора определений DTD.

B.5.1 Использование HTML в XML

Чтобы формализовать вопросы, обсуждавшиеся в предыдущем разделе, Консорциум W3C выпустил недавно предварительный рабочий документ [↪HTMLINXML], который переформулирует HTML 4.0 как одно из приложе-

ний XML и определяет соответствующие пространства имен. Он вводит профили документов в качестве основы для организации взаимодействия между различными множествами средств из HTML в условиях неоднородной среды![1]

B.5.1.1 Добавление модульности в HTML

Было произведено разделение HTML на модули, каждый из которых соответствует четко определенному множеству элементов HTML, которые можно смешивать произвольным образом. Например, «табличный модуль» содержит элементы и атрибуты, необходимые для поддержки работы с таблицами, а «модуль форм» — элементы и атрибуты, требуемые для работы с формами.

Такое введение модульности важно с точки зрения упрощения поддержки, необходимой для работы с информационным содержимым и для доставки его с учетом постоянно растущего числа разнообразных компьютерных платформ — и не только в виде обычных компьютеров, но и различных мобильных устройств (карманных компьютеров, мобильных телефонов), телевизионных устройств (цифровое телевидение, Web-навигаторы на TV-основе), разнообразных устройств другого назначения (с фиксированными функциями). Они могут предъявлять свои собственные специфические требования и ограничения.

Если разделить HTML на различные модули, становится возможным определить для некоторого конкретного заданного устройства, с каким набором элементов следует иметь дело. Эти модули можно использовать как стандартные строительные блоки, которые позволяют осуществлять поддержку работы с тем или иным потребовавшимся устройством.

B.5.1.2 Профили документов

Синтаксис и семантика документов задаются с помощью *профиля документа* (*document profile*). Возможность взаимодействия гарантируется требованием следовать профилю документа, который описывает форматы данных (например, для изображений), уровни поддержки сценариев и стилей (см. нижнюю центральную часть рис. B.2). Профиль документа составляется с использованием формата описания ресурсов RDF.

Определение DTD (или какая-то другая система схематизации документа) задает синтаксис документов, отвечающий профилю документа, в терминах которого используются HTML-модули, а также каким образом можно получить доступ к дополнительным модулям для других наборов элементов XML (химические формулы, математические выражения, музыкальная нотация, векторная графика; см. также верхнюю часть рис. B.2).

Профиль документа, определяющий минимальную поддержку, которую ожидает увидеть агент пользователя, состоит из утверждений, написанных в формате RDF. Это обеспечивает основу, гарантирующую возможность организации взаимодействия, и может использоваться серверами, чтобы установить,

[1] В настоящее время этот документ принят W3C в качестве рекомендации XHTML. — *Прим. ред.*

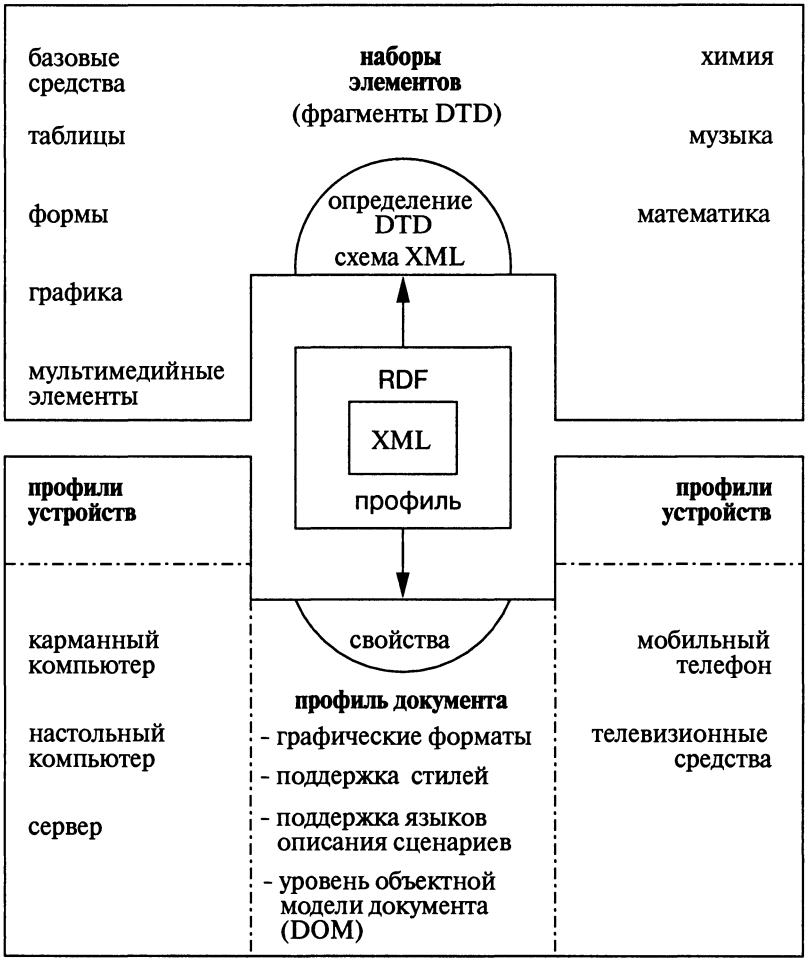


Рис. B.2. Наборы элементов и профили для XHTML.

имеется ли вариант документа, пригодный для передачи пользовательскому агенту с данным профилем устройства. Эта концепция разъясняется в следующем разделе.

B.5.1.3 Профили устройств

Возможности программ-навигаторов, а также предпочтения пользователя задаются с помощью *профилей устройств* (*device profiles*). Это позволяет серверам выбирать подходящий вариант документа для передачи его навигатору или другому средству, возможно, путем преобразования содержимого этого документа, основанного на сопоставлении между профилем документа и исходным документом (см. нижнюю часть рис. B.2).

Описания, доступные в документе, а также профили устройств должны очень существенно упростить оптимизацию разметки, чтобы добиться соответствия потребностей различных устройств внесением небольших изменений в разметку для данного класса устройств простым и надежным способом.

B.5.2 Расширяемый язык гипертекстовой разметки XHTML

Расширяемый язык гипертекстовой разметки XHTML [↪HTMLINXML] представляет собой переформулирование языка HTML 4.0 как одного из приложений языка XML 1.0. В языке XHTML 1.0 заданы три пространства имен XML, соответствующие трем DTD для HTML 4.0: Strict (строгое), Transitional (переходное) и Frameset (фреймовый набор). Каждое из этих пространств имен идентифицируется своим собственным URI.

XHTML 1.0 дает основу для формирования семейства типов документов, которые расширяют XHTML и делят его на подмножества. Это позволяет XHTML поддерживать широкий круг новых устройств, приложений и платформ, как это объясняется в разд. B.5.1. Очевидно, что не все XHTML-элементы потребуются на всех платформах. Поэтому XHTML будет разделен на модули из небольших наборов элементов, которые можно комбинировать друг с другом, чтобы удовлетворить потребностям различных сообществ.

B.5.2.1 Написание XHTML-документов

Какой-либо XHTML-документ можно передать с использованием одного из трех типов интернетовской среды (Internet Media Types). Это дает возможность авторам документа создать мобильное информационное Интернет-содержание, с которым можно затем работать в XML-приложениях (`text/xml`), традиционных пользовательских агентах HTML (`text/html`), а также в новых приложениях XHTML (`text/xhtml`).

Ниже приводится пример простого XHTML-документа:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/Profiles/xhtml1-strict.dtd">
  <head>
    <title>An XHTML document</title>
  </head>
  <body>
    <p>The <a href="http://www.w3c.org/">W3C Home Page</a>.</p>
  </body>
</html>
```

Поскольку XHTML представляет собой XML-приложение, некоторые приемы, которые были вполне допустимы в основанном на SGML языке HTML, придется изменить. Перечень указаний, которым надо следовать при доставке XHTML-документов, включает:

- *XHTML-документы должны представлять собой синтаксически правильно оформленные документы XML.* На практике это означает, что все эле-

менты должны иметь закрывающие метки и для всех этих элементов должны соблюдаться правила вложенности. Если XML-анализатору предъявляется документ, который не будет правильно оформленным, то анализатор выдаст сообщение о серьезной ошибке, причем анализатор при появлении первой же такой ошибки обычно прекращает обработку документа. Для пользователей HTML это довольно неожиданно, поскольку программы-навигаторы обычно пропускают (игнорируют) неверную разметку и продолжают обработку документа.

- *Необходимо использовать атрибут* `xmlns` *для обозначения профиля документа.* Атрибут `xmlns` в элементе `html` должен быть использован для обозначения профиля документа. Если XHTML-документы пересылаются как `text/html`, присутствие атрибута `xmlns` значит, что содержимое рассматриваемого `html`-элемента написано в виде синтаксически корректного XML-текста и *должно* обрабатываться согласно спецификации XML 1.0.
- *Метки и атрибуты должны вводиться строчными (малыми) буквами.* Имена типов элементов и атрибутов в XML (и, следовательно, в XHTML) чувствительны к регистру, в котором вводятся их буквы (строчные или прописные). Поскольку DTD для XHTML определены через строчные буквы, все метки и атрибуты также должны вводиться в формируемом документе строчными буквами.
- *Закрывающие метки требуются всегда.* Все элементы должны иметь закрывающую метку. А именно — пустые метки должны завершаться литерами `/>`, например `<br />` (пробел перед / необходим, чтобы с данной меткой могли работать существующие сейчас HTML-навигаторы).
- *Минимизация атрибутов не разрешается.* Атрибуты должны задаваться как пары имя-значение, со значением, заключенным в кавычки. Тогда компактный список описаний (`dl`) будет иметь такую форму:

```
<dl compact="compact">
  <dt>France</dt><dd>Paris</dd>
  <dt>Spain</dt><dd>Madrid</dd>
</dl>
```

Обратим внимание, что XHTML-строкам в значениях атрибутов должен предшествовать пробел, пробел должен идти и после такой строки, причем последовательности из нескольких пробелов (включая разрывы строк) заменяются единственным междусловным пробелом.

- *Будьте осторожны с элементами* `script` *и* `style`. В языке XHTML элементы `script` и `style` декларируются как имеющие содержимое типа `#PCDATA`, так что объекты, такие как `&lt;` и `&amp;`, будут раскрыты XML-процессором как `<` и `&` соответственно. Если это нежелательно, надо «спрятать» операторы `script` внутрь некоторой помеченной секции `CDATA`, т. е.

```
<script>
  <![CDATA[
     ... unescaped script content ...
  ]]>
</script>
```

B.5.2.2 Пространства имен XHTML

Пространства имен XHTML определяются для трех профилей.

`http://www.w3.org/TR/xhtml1/strict`
: Для документов, конвертированных из HTML 4.0 в строгом соответствии с требованиями DTD `xhtml1-strict` для XHTML.

`http://www.w3.org/TR/xhtml1/transitional`
: Для документов, конвертированных из переходного варианта HTML 4.0, который включает некоторое число презентационных элементов и атрибутов. Они должны быть согласованы с DTD `xhtml1-transitional` XHTML.

`http://www.w3.org/TR/xhtml1/frameset`
: Для документов, конвертированных из фреймового варианта HTML 4.0, работающих как наборы фреймов. Они должны быть согласованы с DTD `xhtml1-frameset` XHTML.

B.5.2.3 Преобразование существующих документов в XHTML

Дейв Раггет написал HTML-программу Tidy [↪HTMLTIDY], позволяющую «приводить в порядок» существующие исходные HTML-файлы и преобразовывать их в XHTML. Программа Tidy может корректно работать с широким кругом HTML-задач. О тех действиях, которые данная программа не может выполнить самостоятельно, выдаются соответствующие сообщения, и пользователь может что-то доделать сам.

Программа Tidy корректирует имеющуюся в документе разметку способом, который согласуется, где это возможно, с характером воспроизведения (отображения) данных, принятым в существующих навигаторах, таких, как Netscape и Microsoft Internet Explorer. Ниже дается список основных проблем, которые программа Tidy может выявлять и устранять:

- обнаруживает и исправляет пропущенные и несовпадающие закрывающие метки;
- исправляет некорректные вложения;
- добавляет пропущенный символ «/» в закрывающих метках;
- завершает списки, добавляя «пропущенные» метки;
- вставляет кавычки вокруг значений атрибутов, если это необходимо;
- сообщает о неизвестных или собственных атрибутах и типах элементов (по отношению к DTD для HTML 4.0);
- сигнализирует о метках с отсутствующим завершающим символом «>».

B.5.2.4 Запуск программы Tidy

Программа Tidy использует ввод с терминала (`stdin`) и вывод на его экран (`stdout`) в качестве значений по умолчанию для потоков ввода/вывода; эти значения будут использоваться, если не задано никакого имени файла. Сооб-

щения об ошибках по умолчанию записываются в `stderr`. Общая форма требуемой командной последовательности имеет вид:

```
tidy [[options] filename]*
```

Возможные значения для *опций* таковы:

`-indent` или `-i` включить режим отступа;
`-omit` или `-o` опустить необязательные закрывающие метки;
`-wrap 72` перейти на новую строку в колонке 72 (по умолчанию 68);
`-upper` или `-u` переводить метки в верхний регистр;
`-clean` или `-c` заменить элементы `font`, `nobr` и `center` с помощью CSS;
`-raw` не выводить объекты для литер от 128 до 255;
`-ascii` использовать ASCII для вывода и Latin 1 для ввода;
`-latin1` использовать Latin 1 для ввода и вывода;
`-utf8` использовать UTF-8 для ввода и вывода;
`-iso2022` использовать ISO2022 для ввода и вывода;
`-numeric` или `-n` выводить числовые, а не именованные объекты;
`-modify` или `-m` модифицировать первоначальные файлы;
`-errors` или `-e` показывать только сообщения об ошибках;
`-f` *file* записывать сообщения об ошибках в файл *file*;
`-xml` задать, если ввод — на языке XML;
`-asxml` преобразовать HTML в XML;
`-slides` разбить на слайды по элементам `h1`;
`-help` перечислить опции командной строки.

Например, следующая команда запустит программу Tidy для обработки файла `index.html`:

```
tidy -f errs.txt -m myfile.html
```

Изменения будут внесены непосредственно в файл `index.html`, а сообщения об ошибках запишутся в файл `errs.txt`. Однобуквенные опции, кроме `-f`, можно комбинировать, в частности,

```
tidy -f errs.txt -imu myfile.html
```

Рассмотрим пример HTML-вывода, подготовленного на с. 357. Сократим этот файл до минимального числа элементов, особенно освобождаясь от всех требуемых закрывающих меток. Файл `tidytest.html` примет следующий вид:

```
<TITLE> Invitation (sgmlpl/CSS formatting) </TITLE>
<LINK href="invit.css" rel="style sheet" type="text/css">
<H1>INVITATION</H1>
<P><TABLE>
<TR><TD class="front">To:
<TD>Anna, Bernard, Didier, Johanna
<TR><TD class="front">When:
<TD>Next Friday Evening at 8 pm
<TR><TD class="front">Venue:
<TD>The Web Cafe
<TR><TD class="front">Occasion:
<TD>My first XML baby
</TABLE>
<P>I would like to invite you all to celebrate
```

```
15  the birth of <EM>Invitation</EM>, my
16  first XML document child.
17  <P>Please do your best to come and join me next Friday
18  evening. And, do not forget to bring your friends.
19  <P>I <EM>really</EM> look forward to see you soon!
20  <P CLASS="signature">Michel
```

Запустим программу Tidy для обработки файла `tidytest.html` с помощью следующей инструкции:

```
tidy -im tidytest.html
Parsing "tidytest.html"
line 4 column 3 - Warning: <TABLE> lacks "summary" attribute
line 4 column 3 - Warning: trimming empty <P>
```

Видно, что элемент <P> в строке 4 исходного HTML-файла будет лишним (избыточным) и что надо задать атрибут `summary` для элемента `table`. Рассмотрим следующий сформированный XML-вывод:

```
1   <?xml version="1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN">
3   <html>
4     <head>
5       <title>
6         Invitation (sgmlpl/CSS formatting)
7       </title>
8       <link href="invit.css" rel="style sheet" type="text/css" />
9     </head>
10    <body>
11      <h1>
12        INVITATION
13      </h1>
14      <table>
15        <tr>
16          <td class="front">
17            To:
18          </td>
19          <td>
20            Anna, Bernard, Didier, Johanna
21          </td>
22        </tr>
23        <tr>
24          <td class="front">
25            When:
26          </td>
27          <td>
28            Next Friday Evening at 8 pm
29          </td>
30        </tr>
31        <tr>
32          <td class="front">
33            Venue:
34          </td>
35          <td>
36            The Web Cafe
37          </td>
38        </tr>
39        <tr>
40          <td class="front">
41            Occasion:
42          </td>
43          <td>
44            My first XML baby
45          </td>
46        </tr>
```

```
47      </table>
48      <p>
49        I would like to invite you all to celebrate the birth of <em>
50        Invitation</em>, my first XML document child.
51      </p>
52      <p>
53        Please do your best to come and join me next Friday evening.
54        And, do not forget to bring your friends.
55      </p>
56      <p>
57        I <em>really</em> look forward to see you soon!
58      </p>
59      <p class="signature">
60        Michel
61      </p>
62    </body>
63  </html>
```

Видно, что все метки записаны строчными буквами, закрывающие метки имеют вывод, записанный для всех элементов, введены также элементы `html`, `head` и `body`. Таким образом, программа Tidy представляет собой очень удобное средство для преобразования HTML-файлов в корректные XML-файлы, отвечающие требованиям XHTML DTD.

B.6 Событийно-управляемый Java-интерфейс

Хотя для использования доступно значительное число XML-анализаторов, перед разработчиками приложений не ставилась задача поддерживать все подходящие интерфейсы API. По этой причине в конце 1997 г. Питер Марри-Раст, автор Jumbo, одного из первых XML-анализаторов, написанных на языке Java [↪JUMBO], предложил на дискуссионном форуме `XML-DEV` XML-разработчиков [↪XMLDEV], чтобы разработчики анализаторов поддерживали общий событийно управляемый API, существующий в языке Java. Эта идея была подхвачена Тимом Бреем, одним из редакторов спецификации XML, а также Дэвидом Меггинсоном, автором программы Ælfred (см. разд. 6.6.5.7) и системы SGMLSpm, описанной в разд. 7.3. Процесс разработки был коллективным, основанным на вкладе большого числа участников форума `XML-DEV`. Итогом его стал единый, стандартный, событийно основанный API для XML-анализаторов, получивший наименование SAX [↪SAX], который был выпущен в мае 1998 г. Реализации его в настоящее время имеются для языков Java и Python.

В следующем разделе вначале будет дан краткий обзор классов, которые имеются в дистрибутиве SAX для языка Java. Затем воспользуемся некоторыми из этих классов, чтобы заново реализовать трансляцию варианта нашего примера `invitation` в виде, пригодном для последующей обработки LaTeX'ом.

B.6.1 Классы языка Java в SAX

Дистрибутив SAX для Java, разработанный Дэвидом Меггинсоном [↪SAXJAVA], содержит одиннадцать базовых классов и интерфейсов вместе с тремя необязательными классами для организации справочной под-

держки, хотя, как это будет видно далее, простое XML-приложение требует использования только двух из них.

Классы и интерфейсы SAX можно разделить на следующие четыре группы:

Интерфейсы для разработчиков анализаторов (пакет `org.xml.sax`)

`Parser` Основной интерфейс к SAX-анализатору. Он регистрирует программы-обработчики событий, устанавливает языковую среду для выдачи сообщений об ошибках, а также запускает процесс грамматического анализа XML-документа.

`AttributeList` Простой интерфейс для итеративной работы со списком атрибутов.

`Locator` Простой интерфейс для локализации текущей точки в обрабатываемом исходном XML-документе.

Интерфейсы для разработчиков приложений (пакет `org.xml.sax`)

`DocumentHandler` Интерфейс, обычно используемый для реализации приложения. Он обеспечивает доступ к основным событиям, относящимся к документу, таким, как начало и завершение работы с элементами.

`ErrorHandler` Интерфейс для реализации специальных средств обработки ошибок.

`DTDHandler` Интерфейс, обеспечивающий работу с обозначениями и (двоичными) объектами (декларациями `NOTATION` и `ENTITY` в DTD), не подвергаемыми синтаксическому анализу.

`EntityResolver` Интерфейс, обеспечивающий подстраиваемую обработку при разрешении ссылок на внешние объекты.

Стандартные классы SAX (пакет `org.xml.sax`)

`InputSource` Класс, содержащий информацию для одного источника ввода.

`SAXException` Общий SAX-класс для описания исключительных состояний.

`SAXParseException` SAX-класс исключительных состояний, связанный с заданной точкой XML-документа.

`HandlerBase` Реализация по умолчанию для `DTDHandler`, `ErrorHandler`, `DocumentHandler` и `EntityResolver`. В нашем примере мы расширим `Handlerbase`, чтобы реализовать требуемый обработчик события.

Классы справочной поддержки, специфические для языка Java (пакет `org.xml.sax.helpers`)

Эти классы не являются составной частью ядра дистрибутива SAX, но включены для удобства Java-программистов.

`ParserFactory` Статические методы в этом классе дают возможность приложению динамически загружать SAX-анализаторы во время исполнения, основываясь на имени класса.

`AttributeListImpl` Делает постоянную копию для списка атрибутов `AttributeList` или дает реализацию по умолчанию `AttributeList` приложению, с которым ведется работа.

`LocatorImpl` Делает постоянную копию состояния для объекта `Locator` в любой точке процесса синтаксического анализа документа.

B.6.2 Исполнение SAX-приложения

Чтобы запустить SAX-приложение, необходим какой-нибудь синтаксический анализатор с SAX-интерфейсом, а также сами библиотеки SAX. Для наших примеров используем компоненты, написанные Дэвидом Меггинсоном, а именно анализатор Ælfred и Java-реализацию SAX.

Чтобы сформировать вывод в формате LaTeX или HTML из XML-документа, потребуется написать обработчик событий для получения информации из анализатора. Как объяснялось в разд. B.6.1, удобным интерфейсом является `DocumentHandler`, который получает события старта и завершения элементов, символьных данных и других основных структур XML. Однако, поскольку нас не интересует реализация интерфейса в целом, можно ограничиться созданием класса, который расширяет `HandlerBase`, где определим только нужные нам методы.

Список методов, доступных в классе `HandlerBase`, включает:

`characters(char[], int, int)` Получает уведомление о символьных данных внутри элемента.

`endDocument()` Получает уведомление о конце документа.

`endElement(String)` Получает уведомление о завершении элемента.

`error(SAXParseException)` Получает уведомление об устранимой ошибке, найденной анализатором.

`fatalError(SAXParseException)` Сообщает о серьезной ошибке в процессе синтаксического анализа XML.

`ignorableWhitespace(char[], int, int)` Получает уведомление о лишнем пробеле в содержимом элемента.

`notationDecl(String, String, String)` Получает уведомление о декларировании нотации.

`processingInstruction(String, String)` Получает уведомление об обрабатывающей инструкции.

`resolveEntity(String, String)` Разрешает ссылку на внешний объект.

`setDocumentLocator(Locator)` Получает объект типа Locator для событий, связанных с документом.

`startDocument()` Получает уведомление о начале документа.

`startElement(String, AttributeList)` Получает уведомление о начале элемента.

`unparsedEntityDecl(String, String, String, String)` Получает уведомление о декларации для объекта, не подвергаемого синтаксическому анализу.

`warning(SAXParseException)` Получает уведомление о сообщении-предупреждении, выданном анализатором.

Нам понадобятся также методы из интерфейсного класса `AttributeList`:

`getLength()` Возвращает число атрибутов в этом списке.

`getName(int)` Возвращает имя атрибута в списке (по его позиции).

`getType(int)`, `getType(String)` Возвращает тип атрибута в списке (по его имени).

`getValue(int)`, `getValue(String)` Возвращает значение атрибута в списке (по его позиции или имени).

Теперь мы готовы написать наш Java-класс `InvitationSAX.java`, который расширяет класс `HandlerBase` и использует интерфейс `AttributeList`. Затем разработаем приложение для форматирования XML-документа `invitation2.xml`, описанного в разд. 7.4.5, в информационном плане эквивалентно документу `invitation.xml`, с которым в разд. 7.3 осуществлялась работа с помощью Perl-интерфейса.

Ниже дается листинг файла `InvitationSAX.java`:

```
1  import org.xml.sax.HandlerBase;
2  import org.xml.sax.AttributeList;
3
4  public class InvitationSAX extends HandlerBase {
5
6    public void startElement (String Ename, AttributeList atts)
7    { if (Ename.equals("invitation"))
8        {System.out.print("\\documentclass[]{article}\n"
9                        + "\\usepackage{invitation}\n"
10                       + "\\begin{document}\n"
11                       + "\\begin{Front}\n");
12       for (int i = 0; i < atts.getLength(); i++) {
13         String Aname = atts.getName(i);
14         String type = atts.getType(i);
15         String value = atts.getValue(i);
16         if (Aname.equals("date"))
17           System.out.print("\\Date{" + value + "}\n");
18         else if (Aname.equals("signature"))
19           System.out.print("\\Signature{" + value + "}\n");
20         else if (Aname.equals("to"))
21           System.out.print("\\To{" + value + "}\n");
22         else if (Aname.equals("where"))
23           System.out.print("\\Where{" + value + "}\n");
24         else if (Aname.equals("why"))
25           System.out.print("\\Why{" + value + "}\n");
26         else System.out.print("INVALID ATTRIBUTE!!! " + value + "\n");
27       } // end attributes of invitation
28       System.out.println("\\end{Front}");
29       System.out.println("\\begin{Body}");
30      } // end element invitation
31    if (Ename.equals("par"))
32      System.out.print("\\par ");
33    if (Ename.equals("emph"))
34      System.out.print("\\emph{");
35   } // End of startElement
36
37   public void endElement (String Ename)
38   { if (Ename.equals("invitation"))
39       System.out.print("\\end{Body}\n"
40                      + "\\begin{Back}\n"
41                      + "\\end{Back}\n"
42                      + "\\end{document}\n");
43     if (Ename.equals("emph"))
44       System.out.print("}");
```

```
45   //  if (Ename.equals("par")) \cdash---> do nothing
46     } // End of endElement
47
48     public void characters(char ch[],int start,int length)
49     { for (int i=start; i<start+length; i++)
50         {System.out.print(ch[i]);}
51     } // End of characters
52
53   } // end of InvitationSAX
```

Здесь можно наблюдать в работе часть методов из класса `HandlerBase`: `startElement` (строки 6–35) для открытия элементов, `endElement` (строки 37–47) для завершения элементов и `characters` (строки 49–53) для работы с символьными данными. В первых двух методах определено действие, которое зависит от того, какой элемент будет обрабатываться. В частности, в начале элемента `invitation` производится инициализация LaTeX'овского документа (строки 9–12). Стандартный вывод записывается с использованием метода `System.out.print` (строки 9, 18, 20 и т. д.) и его разновидности `System.out.println` (строки 28 и 29). Различные атрибуты просматриваются (обрабатываются) циклически с использованием методов из интерфейса `AttributeList`. Число атрибутов дается методом `getLength` (строка 13), который управляет числом итераций цикла `for` (строки 13–27). Внутри цикла метод `getName` (строка 14) осуществляет получение имени атрибута, метод `getType` (строка 15) выдает его тип, а метод `getValue` (строка 16) — его значение. Используя имя атрибута, операторы `if` (строки 17–26) предпринимают действия по установке LaTeX'овских переменных в требуемые для них значения.

Аналогичным образом, начало элемента `par` формирует команду `\par` (строки 31–32), а для стартового дескриптора `emph` запускается LaTeX'овская команда `\emph` (строки 33–34).

Что касается завершения элементов (строки 37–47), для `par` не делается ничего (строка 46 представляет собой Java-комментарий), и мы ставим закрывающую фигурную скобку для завершающего дескриптора `emph` (строки 44–45). Когда будет достигнута метка `</invitation>`, надо завершить LaTeX'овский документ (строки 39–43).

Символьные данные из рассматриваемого исходного XML-документа преобразуются из символьного массива `ch`, первого аргумента метода `characters` (строка 49), в поток вывода с помощью цикла (строки 51–52), границы которого задаются аргументом данного метода.

Последней нерешенной задачей осталось написать основной драйверный класс `MySAXApp.java`, который использует наш обработчик `InvitationSAX`. Это выглядит следующим образом.

```
1   // MySAXApp.java -- Main driver class
2   //  --> calls InvitationSAX which has customized code
3   import org.xml.sax.Parser;
4   import org.xml.sax.DocumentHandler;
5   import org.xml.sax.helpers.ParserFactory;
6
7   public class MySAXApp {
8     static final String parserClass = "com.microstar.xml.SAXDriver";
9     public static void main (String args[])
10      throws Exception
```

```
11      { Parser parser = ParserFactory.makeParser(parserClass);
12        DocumentHandler handler = new InvitationSAX();
13        parser.setDocumentHandler(handler);
14        for (int i = 0; i < args.length; i++) {parser.parse(args[i]);}
15      }
16    }
```

Созданное приложение формирует первым делом объект `Parser` применением имени класса к `ParserFactory` (определяется в строке 8 и используется в строке 11). Затем наш класс для обработчика событий `InvitationSAX` подвергается обработке (строка 12) и регистрируется анализатором (строка 13). Наконец, все XML-документы, которые задаются как аргументы в командной строке, подвергаются синтаксическому анализу (цикл `for` в строках 14). Текущая реализация требует, чтобы рассматриваемые документы задавались через абсолютные URL.

Теперь мы готовы прокомпилировать наше приложение и обработчик событий для документа `invitation2.xml`, после чего запустить его на исполнение (пример выполнен в среде Windows/NT).

```
set classpath=.;d:\aelfred;d:\SAX;d:\jdk1.1.6\src
d:\javac InvitationSAX.java
d:\javac MySAXApp.java
d:\java MySAXApp file:d:/invitation2.xml > inv2.tex
```

Первая строка здесь объявляет, где размещаются библиотеки Java-классов; следующие две строки компилируют два наших Java-класса с помощью Java-компилятора `javac`. Последняя строка запускает приложение `MySAXApp`, которое загружает автоматически остальные классы в файле `invitation2.xml` (заданном как некоторый абсолютный URL) и записывает полученный итоговый LaTeX'овский файл `inv2.tex`. Этот файл выглядит так:

```
1   \documentclass[]{article}
2   \usepackage{invitation}
3   \begin{document}
4   \begin{Front}
5   \To{Anna, Bernard, Didier, Johanna}
6   \Date{Next Friday Evening at 8 pm}
7   \Where{The Web Cafe}
8   \Why{My first XML baby}
9   \Signature{Michel}
10  \end{Front}
11  \begin{Body}
12  \par
13  I would like to invite you all to celebrate
14  the birth of \emph{Invitation}, my
15  first XML document child.
16  \par
17  Please do your best to come and join me next Friday
18  evening. And, do not forget to bring your friends.
19  \par
20  I \emph{really} look forward to see you soon!
21  \end{Body}
22  \begin{Back}
23  \end{Back}
24  \end{document}
```

Этот LaTeX'овский код немного отличается от того, что был показан на с. 343. Тем не менее, можно использовать (строка 3) тот же самый пакет `invitation.sty`, как это было определено на с. 344, поскольку он написан так, что процессом форматирования управляют команды высокого уровня. Следовательно, различные LaTeX'овские исходные файлы будут давать при печати идентичный результат (см. рис. 7.1). Это еще раз демонстрирует преимущество написания высокоуровневого LaTeX'овского кода на уровне приложений и работы с низкоуровневыми наборными средствами и командами отображения в стиле.

Проблемы интернационализации

С.1 Коды языков, стран и систем письма

В таблице С.1 приведены коды для представления названий языков в соответствии со стандартом ISO 639. В первом столбце приведен современный трехбуквенный код (ISO:639-2, 1998). Второй столбец содержит более старый (предыдущий) двухбуквенный код (ISO:639, 1988) (если он существует). И, наконец, в третьем столбце приведены названия языков (или их варианты)[1]. Хотя двухбуквенные коды могут охватить только наиболее распространенные языки, они до сих пор используются во многих компьютерных приложениях; в частности, в языках HTML и XML для идентификации языков обычно используются только двухбуквенные коды.

В таблице С.2 приведены коды для представления названий стран в соответствии со стандартом ISO 3166 (ISO:3166, 1997). В первом столбце содержатся двухбуквенные коды, которые используются для обозначения страны в языках HTML/XML; во втором столбце представлены более легкие для запоминания трехбуквенные коды. В третьем столбце перечислены названия стран.

В таблице С.3 приведены коды для обозначения различных систем письма, предлагаемые в проекте стандарта ISO 15924 (ISO:15924, 1999). В первом столбце содержатся двухбуквенные коды, во втором — более легкие для запоминания трехбуквенные коды, а в третьем — названия систем письма[1].

В таблице С.4 приведен краткий обзор различных кодировок, соответствующих стандарту ISO 8859 (см. (ISO/IEC:8839-1, 1998) и последующие). В частности, недавно принятая кодировка Latin 9 (ISO/IEC:8859-15) в основном совпадает с набором кодов Latin 1 для западноевропейских языков. Однако в кодировку Latin 9 вместо некоторых редко используемых символов включены буквы œ, Œ и ÿ (французского языка), š, Š, ž и Ž (финского языка) и знак денежной единицы Евро: €. Конечно, было бы замечательно как можно скорее перейти от этих стандартов ISO/IEC 8859 к стандарту Unicode (разд. С.2).

---

[1] При переводе названий языков и письменностей использовались следующие источники:
Лингвистический энциклопедический словарь.— М.: Советская энциклопедия, 1990.
БСЭ. — М.: Советская энциклопедия, 1970–1978.
Реформатский А. А. Введение в языкознание. — М.: Аспект Пресс, 1996.
Pocket Oxford Dictionary. — Oxford: Oxford University Press, 1994.
Поскольку не для всех названий удалось найти русские эквиваленты, часть из них приводится только на английском. — *Прим. ред.*

Таблица C.1. Коды и названия языков (ISO 639).

aar	aa	Afar (афар)	abk	ab	Abkhazian (абхазский)
ace		Achinese	ach		Acoli (ачоли)
ada		Adangme (адангме)	afa		Afro-Asiatic (другие афро-азиатские)
afh		Afrihili	afr	af	Afrikaans (африкаанс (бурский))
ajm		Aljamia	aka		Akan (акан)
akk		Akkadian (аккадский)	ale		Aleut (алеутский)
alg		Algonquian (алгонкинские)	amh	am	Amharic (амхарский)
ang		English, Old (древнеанглийский (англосаксонский) (~450–1100 гг.))	apa		Apache languages (языки апачей)
ara	ar	Arabic (арабский)	arc		Aramaic (арамейский)
arn		Araucanian (арауканские)	arp		Arapaho (арапахо)
art		Artificial (другие искусственные)	arw		Arawak (аравакский)
asm	as	Assamese (ассамский)	ath		Athapascan (атапаскские)
aus		Australian (австралийские)	ava		Avaric (аварский)
ave		Avestan (авестийский)	awa		Awadhi
aym	ay	Aymara (аймара)	aze	az	Azerbaijani (азербайджанский)
bad		Banda (банда)	bai		Bamileke (бамилеке)
bak	ba	Bashkir (башкирский)	bal		Baluchi (белуджский (балучи))
bam		Bambara (бамбара)	ban		Balinese (балийский)
bas		Basa	bat		Baltic (другие балтийские)
bej		Beja (бедауйе (беджа))	bel	be	Belarussian (белорусский)
bem		Bemba (бемба)	ben	bn	Bengali (бенгали)
ber		Berber (другие берберские)	bho		Bhojpuri (бходжпури)
bih	bh	Bihari (бихари)	bik		Bikol (бикольский (бикол))
bin		Bini (бини)	bis	bi	Bislama (бислама)
bla		Siksika	bnt		Bantu (другие банту)
bod	bo	Tibetan (тибетский)	bra		Braj (брахуи)
bre	br	Breton (бретонский)	btk		Batak (батак (Индонезия))
bua		Buriat (бурятский)	bug		Buginese (бугийский)
bul	bg	Bulgarian (болгарский)	cad		Caddo (каддо)
cai		Central American Indian (другие индейцев Центральной Америки)	car		Carib (карибские)
cat	ca	Catalan (каталанский)	cau		Caucasian (другие кавказские)
ceb		Cebuano (себуанский)	cel		Celtic (другие кельтские)
ces	cs	Czech (чешский)	cha		Chamorro (чаморро)
chb		Chibcha (чибча)	che		Chechen (чеченский)
chg		Chagatai (чагатайский)	chk		Chuukese (чукотский)
chm		Mari (марийский)	chn		Chinook (чинук (жаргон))
cho		Choctaw (чоктав)	chp		Chipewyan (чипевьян)
chr		Cherokee (чероки)	chu		Church Slavic (церковнославянский)
chv		Chuvash (чувашский)	chy		Cheyenne (чейенн)
cmc		Chamic (чамские)	cop		Coptic (коптский)
cor	kw	Cornish (корнский)	cos	co	Corsican (корсиканский)
cpe		Creoles and pidgins, English-based (другие креольские и пиджины на основе английского)	cpf		Creoles and pidgins, French-based (другие креольские и пиджины на основе французского)

Коды и названия языков (*продолжение*)

cpp	Creoles and pidgins, Portuguese-based (другие креольские и пиджины на основе португальского)	cre	Cree (кри)
crp	Creoles and pidgins (другие креольские и пиджины)	cus	Cushitic (другие кушитские)
cym cy	Welsh (уэльский, валлийский)	dak	Dakota (дакота)
dan da	Danish (датский)	day	Dayak (даякский)
del	Delaware (делаварский)	den	Slave (Athapascan) (слейв)
deu de	German (немецкий)	dgr	Dogrib (догриб)
din	Dinka (динка)	div	Divehi (дивехи, мальдивский)
doi	Dogri	dra	Dravidian (другие дравидийские)
dua	Duala (дуала)	dum	Dutch, Middle (средневековый голландский (~1050–1350 гг.))
dyu	Dyula (диола)	dzo dz	Dzongkha (дунганский)
efi	Efik (эфик)	egy	Egyptian, Ancient (древнеегипетский)
eka	Ekajuk	ell el	Greek, Modern (новогреческий (после 1453 г.))
elx	Elamite (эламский)	eng en	English (английский)
enm	English, Middle (среднеанглийский (1100–1500 гг.))	epo eo	Esperanto (эсперанто)
est et	Estonian (эстонский)	eth	Ethiopic (эфиопский)
eus eu	Basque (баскский)	ewe	Ewe (эве (эвелбе))
ewo	Ewondo (эвенкийский)	fan	Fang (фанджи)
fao fo	Faroese (фарерский)	fas fa	Persian (персидский (фарси))
fat	Fanti (фанти)	fij fj	Fijian (фиджи)
fin fi	Finnish (финский)	fiu	Finno-Ugrian (другие финно-угорские)
fon	Fon (фон)	fra fr	French (французский)
frm	French, Middle (среднефранцузский (~1400–1600 гг.))	fro	French, Old (древнефранцузский (~842–1400 гг.))
fry fy	Frisian (фризский)	ful	Fulah (фула (фульфульде))
fur	Friulian (фриульский)	gaa	Ga (га)
gai ga	Irish (ирландский)	gay	Gayo
gba	Gbaya (гбайя)	gdh gd	Gaelic (галльский)
gem	Germanic (другие германские)	gez	Geez (гээз)
gil	Gilbertese	glg gl	Gallegan (галисийский)
gmh	German, Middle High (средневерхненемецкий (~1050–1500 гг.))	goh	German, Old High (древневерхненемецкий (~750–1050 гг.))
gon	Gondi (гонди)	gor	Gorontalo
got	Gothic (готский)	grb	Grebo (гребо)
grc	Greek, Ancient (древнегреческий (до 1453 г.))	grn gn	Guarani (гуарани)
guj gu	Gujarati (гуджарати)	gwi	Gwich'in (гуичоль)
hai	Haida (хайда)	hau ha	Hausa (хауса)
haw	Hawaiian (гавайский)	heb he/iw	Hebrew (иврит)
her	Herero (гереро)	hil	Hiligaynon (панаянский (хилигайнон))

Коды и названия языков (*продолжение*)

him	Himachali (химачали)	hin hi	Hindi (хинди)
hit	Hittite (древнесирийский)	hmn	Hmong
hmo	Hiri Motu (хири-моту)	hrv hr	Croatian (хорватский)
hun hu	Hungarian (венгерский)	hup	Hupa (хупа)
hye hy	Armenian (армянский)	iba	Iban
ibo	Igbo (игбо)	ijo	Ijo (иджо)
iku iu	Inuktitut	ile ie	Interlingue (международные)
ilo	Iloko (илоканский)	ina ia	Interlingua (интерлингва)
inc	Indic (другие индийские)	ind id/in	Indonesian (индонезийский)
ine	Indo-European (другие индоевропейские)	ipk ik	Inupiak
ira	Iranian (другие иранские)	iro	Iroquoian (ирокезские)
isl is	Icelandic (исландский)	ita it	Italian (итальянский)
jaw jw	Javanese (яванский)	jpn ja	Japanese (японский)
jpr	Judeo-Persian (еврейско-персидский)	jrb	Judeo-Arabic (еврейско-арабский)
kaa	Kara-Kalpak (каракалпакский)	kab	Kabyle (кабильский)
kac	Kachin (качин (цзинпо))	kal kl	Kalaallisut
kam	Kamba (камбатта)	kan kn	Kannada (каннада)
kar	Karen	kas ks	Kashmiri (кашмири)
kat ka	Georgian (грузинский)	kau	Kanuri (канаури)
kaw	Kawi (кави)	kaz kk	Kazakh (казахский)
kha	Khasi (кхаси)	khi	Khoisan (хо)
khm km	Khmer (кхмерский)	kho	Khotanese (хотанский (сакский))
kik	Kikuyu (кикую)	kin rw	Kinyarwanda (киньяруанда)
kir ky	Kirghiz (киргизский)	kmb	Kimbundu
kok	Konkani (конкани)	kom	Komi (коми)
kon	Kongo (конго)	kor ko	Korean (корейский)
kos	Kosraean	kpe	Kpelle (кпелле)
kro	Kru (кру)	kru	Kurukh (курукх)
kua	Kuanyama (куануа)	kum	Kumyk (кумыкский)
kur ku	Kurdish (курдский)	kut	Kutenai (кутенаи)
lad	Ladino (ладино)	lah	Lahnda (лахнда)
lam	Lamba (ламба)	lao lo	Lao (лаосский)
lat la	Latin (латинский (латынь))	lav lv	Latvian (латышский)
lez	Lezghian (лезгинский)	lin ln	Lingala (лингала)
lit lt	Lithuanian (литовский)	lol	Mongo (монгорский)
loz	Lozi (лози)	ltz lb	Lëtzeburgesch (люксембургский)
lua	Luba-Lulua (луба-лулуа)	lub	Luba-Katanga (луба-катанга)
lug	Ganda (ганда)	lui	Luiseno (луисеньо)
lun	Lunda (лунда)	luo	Luo (луо (кения и танзания))
lus	Lushai (лушей)	mad	Madurese (мадурский)
mag	Magahi (магахи)	mah	Marshall (маршалльский)
mai	Maithili (майтхили)	mak	Makasar (макасарский)
mal ml	Malayalam (малаялам)	man	Mandingo (мандинка)
map	Austronesian (другие австронезийские)	mar mr	Marathi (маратхи)
mas	Masai (масаи)	max	Manx (мэнский)
mdr	Mandar (мандарский)	men	Mende (менде)

Коды и названия языков (*продолжение*)

mga	Irish, Middle (средневековый ирландский) (900–1200 гг.)	mic	Micmac (микмак)
min	Minangkabau (миньянка)	mis	Miscellaneous languages (смешанные)
mkd mk	Macedonian (македонский)	mkh	Mon-Khmer (другие мон-кхмерские)
mlg mg	Malagasy (малагасийский)	mlt mt	Maltese (мальтийский)
mni	Manipuri (манипурский)	mno	Manobo (мано)
moh	Mohawk (могавк)	mol mo	Moldavian (молдавский)
mon mn	Mongolian (монгольский)	mos	Mossi (моси (море))
mri mi	Maori (маори)	msa ms	Malay (малайский)
mul	Multiple languages (составные)	mun	Munda languages (языки мунда)
mus	Creek (крик)	mwr	Marwari
mya my	Burmese (бирманский)	myn	Mayan (языки майя)
nah	Aztec (ацтекский)	nai	North American Indian (другие североамериканских индейцев)
nau na	Nauru (науру)	nav	Navajo (навахо)
nbl	Ndebele, South	nde	Ndebele, North
ndo	Ndonga (ндонга)	nep ne	Nepali (непальский)
new	Newari (неварский)	nia	Nias (нийа)
nic	Niger-Kordofanian (другие нигеро-кордофанские)	niu	Niuean (ниуэ)
nld nl	Dutch (нидерландский (голландский))	non	Norse, Old (древнескандинавский)
nor no	Norwegian (норвежский)	nso	Sohto (сото (северный))
nub	Nubian (нубийские)	nya	Nyanja (ньянджа)
nym	Nyamwezi (ньямвези)	nyn	Nyankole
nyo	Nyoro (ньоро)	nzi	Nzima
oci oc	Occitan (окситанский (провансальский)) (после 1500 г.)	oji	Ojibwa (оджибва)
ori	Oriya (ория)	orm om	Oromo (оромо)
osa	Osage (оскский)	oss	Ossetic (осетинский)
ota	Turkish, Ottoman (старотурецкий (староосманский)) (1500–1928)	oto	Otomian (отоми)
paa	Papuan (другие папуасские)	pag	Pangasinan (пангасинанский)
pal	Pahlavi (среднеперсидский (пехлеви))	pam	Pampanga (пампангский)
pan pa	Panjabi (панджаби)	pap	Papiamento (папьяменто)
pau	Palauan (палаунг)	peo	Persian, Old (древнеперсидский (~600–400 гг. до н.э.))
phi	Philippine (другие филиппинские)	phn	Phoenician (финикийский)
pli	Pali (пали)	pol pl	Polish (польский)
pon	Pohnpeian (понапе)	por pt	Portuguese (португальский)
pra	Prakrit languages (пракритские (пракриты))	pro	Provençal, Old (древнепровансальский (до 1500 г.))
pus ps	Pushto (пушту)	que qu	Quechua
raj	Rajasthani (раджастхани)	rap	Rapanui (рапануи)
rar	Rarotongan (раротонга)	roa	Romance (другие романские)
roh rm	Rhaeto-Romance (ретороманский)	rom	Romany (цыганский)
ron ro	Romanian (румынский)	run rn	Rundi (рунди)
rus ru	Russian (русский)	sad	Sandawe (сандаве)

Коды и названия языков (*продолжение*)

sag sg	Sango (санго)	sai	South American Indian (другие индейцев Южной Америки)
sal	Salishan (салишские)	sam	Samaritan Aramaic (самаритянский (арамейский))
san sa	Sanskrit (санскрит)	sas	Sasak
sat	Santali (сантали)	sco	Scots (гэльский (шотландский))
sel	Selkup (селькупский)	sem	Semitic (другие семитские)
sga	Irish, Old (древнеирландский (до 900 г.))	shn	Shan (шанский)
sid	Sidamo (сидамо)	sin si	Sinhalese (сингальский)
sio	Siouan (языки сиу)	sit	Sino-Tibetan (другие китайско-тибетские)
sla	Slavic (другие славянские)	slk sk	Slovak (словацкий)
slv sl	Slovenian (словенский)	smi se	Sámi (саамский (саами))
smo sm	Samoan (самоа)	sna sn	Shona (шона)
snd sd	Sindhi (синдхи)	snk	Soninke (сонинке)
sog	Sogdian (согдийский)	som so	Somali (сомали)
son	Songhai (сонгай)	sot st	Sotho, Southern (сото южный)
spa es	Spanish (испанский)	sqi sq	Albanian (албанский)
srd	Sardinian (сардский (сардинский))	srp sr	Serbian (сербский)
srr	Serer (серер)	ssa	Nilo-Saharan (другие нило-сахарские)
ssw ss	Swati (свази)	suk	Sukuma (сумо)
sun su	Sundanese (сунданский)	sus	Susu (сусу)
sux	Sumerian (шумерский)	swa sw	Swahili (суахили)
swe sv	Swedish (шведский)	syr	Syriac (сирийский)
tah	Tahitian (таити)	tai	Tai (другие тайские)
tam ta	Tamil (тамильский)	tat tt	Tatar (татарский)
tel te	Telugu (телугу)	tem	Timne (темне)
ter	Tereno	tet	Tetum
tgk tg	Tajik (таджикский)	tgl tl	Tagalog (тагальский)
tha th	Thai (тхайский)	tig	Tigre (тигре)
tir	Tigrinya (тиграй)	tiv	Tiv (тив)
tkl	Tokelau (токелау)	tli	Tlingit (тлингит)
tmh	Tamashek (тамашек)	tog	Tonga Nyasa (тонга (ньяса))
ton	Tonga (тонга (о-ва Тонга))	tpi	Tok Pisin (ток-писин)
tsi	Tsimshian	tsn ts	Tswana (тсвана)
tso	Tsonga	tuk tk	Türkmen (туркменский)
tum	Tumbuka (тубу)	tur	Turkish (турецкий)
tut	Altaic (другие алтайские)	tvl	Tuvalu (тувалу)
twi tw	Twi (тви)	tyv	Tuvinian (тувинский)
uga	Ugaritic (угаритский)	uig ug	Uighur (уйгурский)
ukr uk	Ukrainian (украинский)	umb	Umbundu (умбунду)
und	Undetermined (неопределенные)	urd ur	Urdu (урду)
uzb uz	Uzbek (узбекский)	vai	Vai (вайоли (вай))
ven	Venda (вендский)	vie vi	Vietnamese (вьетнамский)
vol vo	Volapük (волапюк)	vot	Votic (вотяцкий (удмуртский))
wak	Wakashan (вакашские)	wal	Walamo
war	Waray (самарский (варай))	was	Washo (вашо)

Коды и названия языков (*продолжение*)

wen	Sorbian (сорабские)	wol wo	Wolof (волоф)
xho xh	Xhosa (коса (кхоса))	yao	Yao (яо)
yap	Yapese	yid yi/ji	Yiddish (идиш)
yor	Yoruba (йоруба)	ypk	Yupik languages
zap	Zapotec	zen	Zenaga (зенага)
zha za	Zhuang (звай)	zho zh	Chinese (китайский)
znd	Zande (занде)	zul zu	Zulu (зулу)
zun	Zuñi (зуньи)		

Таблица C.2. Коды и названия стран (ISO 3166).

AF AFG	Afghanistan (Афганистан)	AL ALB	Albania (Албания)
DZ DZA	Algeria (Алжир)	AS ASM	American Samoa (Восточное Самоа (США))
AD AND	Andorra (Андорра)	AO AGO	Angola (Ангола)
AI AIA	Anguilla (Ангилья)	AQ ATA	Antarctica (Антарктида)
AG ATG	Antigua and Barbuda (Антигуа и Барбуда)	AR ARG	Argentina (Аргентина)
AM ARM	Armenia (Армения)	AW ABW	Aruba (Аруба)
AU AUS	Australia (Австралия)	AT AUT	Austria (Австрия)
AZ AZE	Azerbaijan (Азербайджан)	BS BHS	Bahamas (Багамские о-ва)
BH BHR	Bahrain (Бахрейн)	BD BGD	Bangladesh (Бангладеш)
BB BRB	Barbados (Барбадос)	BY BLR	Belarus (Белоруссия)
BE BEL	Belgium (Бельгия)	BZ BLZ	Belize (Белиз)
BJ BEN	Benin (Бенин)	BM BMU	Bermuda (Бермудские о-ва)
BT BTN	Bhutan (Бутан)	BO BOL	Bolivia (Боливия)
BA BIH	Bosnia and Herzegovina (Босния и Герцеговина)	BW BWA	Botswana (Ботсвана)
BV BVT	Bouvet Island (Остров Буве)	BR BRA	Brazil (Бразилия)
IO IOT	British Indian Ocean Territory (Владения Великобритании в Индийском океане)	BN BRN	Brunei Darussalam (Бруней Дар-эс-салам)
BG BGR	Bulgaria (Болгария)	BF BFA	Burkina Faso (Буркина-Фасо)
BI BDI	Burundi (Бурунди)	KH KHM	Cambodia (Камбоджа)
CM CMR	Cameroon (Камерун)	CA CAN	Canada (Канада)
CV CPV	Cape Verde (Кабо-Верде)	KY CYM	Cayman Islands (Каймановы о-ва)
CF CAF	Central African Republic (Центрально-Африканская Республика)	TD TCD	Chad (Чад)
CL CHL	Chile (Чили)	CN CHN	China (Китай)
CX CXR	Christmas Island (Остров Рождества)	CC CCK	Cocos (Keeling) Islands (Кокосовые о-ва, Килинг)
CO COL	Colombia (Колумбия)	KM COM	Comoros (Коморские о-ва)
CG COG	Congo (Конго)	CD COD	Congo, Democratic Republic (Демократическая Республика Конго)
CK COK	Cook Islands (о-ва Кука)	CR CRI	Costa Rica (Коста-Рика)
CI CIV	Côte d'Ivoire (Кот-д'Ивуар)	HR HRV	Croatia (Хорватия)
CU CUB	Cuba (Куба)	CY CYP	Cyprus (Кипр)

Коды и названия стран (*продолжение*)

CZ CZE Czech Republic (Чешская Республика)	DK DNK Denmark (Дания)
DJ DJI Djibouti (Джибути)	DM DMA Dominica (Доминика)
DO DOM Dominican Republic (Доминиканская Республика)	TP TMP East Timor (provisional) (Восточный Тимор (временно))
EC ECU Ecuador (Эквадор)	EG EGY Egypt (Египет)
SV SLV El Salvador (Сальвадор)	GQ GNQ Equatorial Guinea (Экваториальная Гвинея)
ER ERI Eritrea (Эритрея)	EE EST Estonia (Эстония)
ET ETH Ethiopia (Эфиопия)	FK FLK Falkland Islands (Malvinas) (Фолклендские (Мальвинские) о-ва)
FO FRO Faroe Islands (Фарерские о-ва)	FJ FJI Fiji (Фиджи)
FI FIN Finland (Финляндия)	FR FRA France (Франция)
FX FXX France, Metropolitan (Франция, метрополия)	GF GUF French Guiana (Французская Гвиана)
PF PYF French Polynesia (Французская Полинезия)	TF ATF French Southern Territories (Южные земли Франции)
GA GAB Gabon (Габон)	GM GMB Gambia (Гамбия)
GE GEO Georgia (Грузия)	DE DEU Germany (Германия)
GH GHA Ghana (Гана)	GI GIB Gibraltar (Гибралтар)
GR GRC Greece (Греция)	GL GRL Greenland (Гренландия)
GD GRD Grenada (Гренада)	GP GLP Guadaloupe (Гваделупа)
GU GUM Guam (Гуам)	GT GTM Guatemala (Гватемала)
GN GIN Guinea (Гвинея)	GW GNB Guinea-Bissau (Гвинея-Бисау)
GY GUY Guyana (Гайяна)	HT HTI Haiti (Гаити)
HM HMD Heard Island and McDonald Islands (Херд и о-ва Макдоналд)	HN HND Honduras (Гондурас)
HK HKG Hong Kong (Гонконг)	HU HUN Hungary (Венгрия)
IS ISL Iceland (Исландия)	IN IND India (Индия)
ID IDN Indonesia (Индонезия)	IR IRN Iran (Иран)
IQ IRQ Iraq (Ирак)	IE IRL Ireland (Ирландия)
IL ISR Israel (Израиль)	IT ITA Italy (Италия)
JM JAM Jamaica (Ямайка)	JP JPN Japan (Япония)
JO JOR Jordan (Иордания)	KZ KAZ Kazakhstan (Казахстан)
KE KEN Kenya (Кения)	KI KIR Kiribati (Кирибати)
KP PRK Korea, North (Северная Корея (Корейская Народно-Демократическая Республика))	KR KOR Korea, South (Южная Корея)
KW KWT Kuwait (Кувейт)	KG KGZ Kyrgyzstan (Киргизия)
LA LAO Laos (Лаос)	LV LVA Latvia (Латвия)
LB LBN Lebanon (Ливан)	LS LSO Lesotho (Лесото)
LR LBR Liberia (Либерия)	LY LBY Libya (Ливия)
LI LIE Liechtenstein (Лихтенштейн)	LT LTU Lithuania (Литва)
LU LUX Luxembourg (Люксембург)	MO MAC Macau (Макао (Аомынь))
MK MKD Macedonia (Македония)	MG MDG Madagascar (Мадагаскар)
MW MWI Malawi (Малави)	MY MYS Malaysia (Малайзия)
MV MDV Maldives (Мальдивы)	ML MLI Mali (Мали)
MT MLT Malta (Мальта)	MH MHL Marshall Islands (Маршалловы о-ва)
MQ MTQ Martinique (Мартиника)	MR MRT Mauritania (Мавритания)

Коды и названия стран (*продолжение*)

MU MUS Mauritius (Маврикий)	YT MYT Mayotte (Майотт)
MX MEX Mexico (Мексика)	FM FSM Micronesia (Микронезия)
MD MDA Moldova (Молдова)	MC MCO Monaco (Монако)
MN MNG Mongolia (Монголия)	MS MSR Montserrat (Монтсеррат)
MA MAR Morocco (Марокко)	MZ MOZ Mozambique (Мозамбик)
MM MMR Myanmar (Бирма (Мьянма))	NA NAM Namibia (Намибия)
NR NRU Nauru (Науру)	NP NPL Nepal (Непал)
NL NLD Netherlands (Нидерланды)	AN ANT Netherlands Antilles (Антильские о-ва (Нид.))
NC NCL New Caledonia (Новая Каледония)	NZ NZL New Zealand (Новая Зеландия)
NI NIC Nicaragua (Никарагуа)	NE NER Niger (Нигер)
NG NGA Nigeria (Нигерия)	NU NIU Niue (Ниуэ)
NF NFK Norfolk Island (Норфолк (о-ва))	MP MNP Northern Mariana Islands (Северные Марианские о-ва)
NO NOR Norway (Норвегия)	OM OMN Oman (Оман)
PK PAK Pakistan (Пакистан)	PW PLW Palau (Палау (о-ва))
PA PAN Panama (Панама)	PG PNG Papua New Guinea (Папуа-Новая Гвинея)
PY PRY Paraguay (Парагвай)	PE PER Peru (Перу)
PH PHL Philippines (Филиппины)	PN PCN Pitcairn (Питкэрн)
PL POL Poland (Польша)	PT PRT Portugal (Португалия)
PR PRI Puerto Rico (Пуэрто-Рико)	QA QAT Qatar (Катар)
RE REU Réunion (Реюньон)	RO ROM Romania (Румыния)
RU RUS Russian Federation (Российская Федерация)	RW RWA Rwanda (Руанда)
SH SHN Saint Helena (Остров Святой Елены)	KN KNA Saint Kitts and Nevis (Сент-Китс и Невис)
LC LCA Saint Lucia (Сент-Люсия)	PM SPM Saint Pierre and Miquelon (Сен-Пьер и Микелон)
VC VCT Saint Vincent and the Grenadines (Сент-Винсент и Гренадины)	WS WSM Samoa (Самоа (западное))
SM SMR San Marino (Сан-Марино)	ST STP São Tomé and Príncipe (Сан-Томе и Принсипи)
SA SAU Saudi Arabia (Саудовская Аравия)	SN SEN Senegal (Сенегал)
SC SYC Seychelles (Сейшельские о-ва)	SL SLE Sierra Leone (Сьерра-Леоне)
SG SGP Singapore (Сингапур)	SK SVK Slovakia (Словакия)
SI SVN Slovenia (Словения)	SB SLB Solomon Islands (Соломоновы о-ва)
SO SOM Somalia (Сомали)	ZA ZAF South Africa (Южно-Африканская Республика)
GS SGS South Georgia and the South Sandwich Islands (Южная Георгия и Южные Сандвичевы о-ва)	ES ESP Spain (Испания)
LK LKA Sri Lanka (Шри-Ланка)	SD SDN Sudan (Судан)
SR SUR Suriname (Суринам)	SJ SJM Svalbard and Jan Mayen (Свальбард и Ян-Майен)
SZ SWZ Swaziland (Свазиленд)	SE SWE Sweden (Швеция)
CH CHE Switzerland (Швейцария)	SY SYR Syria (Сирия)
TW TWN Taiwan (Тайвань)	TJ TJK Tajikistan (Таджикистан)
TZ TZA Tanzania (Танзания)	TH THA Thailand (Таиланд)

Коды и названия стран (*продолжение*)

TG	TGO	Togo (Toro)	
TK	TKL	Tokelau (Токелау)	
TO	TON	Tonga (Тонга)	
TT	TTO	Trinidad and Tobago (Тринидад и Тобаго)	
TN	TUN	Tunisia (Тунис)	
TR	TUR	Turkey (Турция)	
TM	TKM	Turkmenistan (Туркменистан)	
TC	TCA	Turks and Caicos Islands (Теркс и Кайкос, о-ва)	
TV	TUV	Tuvalu (Тувалу)	
UG	UGA	Uganda (Уганда)	
UA	UKR	Ukraine (Украина)	
AE	ARE	United Arab Emirates (Объединенные Арабские Эмираты)	
GB	GBR	United Kingdom (Соединенное Королевство Великобритании и Северной Ирландии)	
US	USA	United States (Соединенные Штаты Америки)	
UM	UMI	United States Minor Outlying Islands (мелкие удаленные о-ва США)	
UY	URY	Uruguay (Уругвай)	
UZ	UZB	Uzbekistan (Узбекистан)	
VU	VUT	Vanuatu (Вануату)	
VA	VAT	Vatican City State (Holy See) (город-государство Ватикан (Святейший престол))	
VE	VEN	Venezuela (Венесуэла)	
VN	VNM	Vietnam (Вьетнам)	
VG	VGB	Virgin Islands, British (Виргинские о-ва (Брит.))	
VI	VIR	Virgin Islands, U.S. (Виргинские о-ва (США))	
WF	WLF	Wallis and Fortuna Islands (о-ва Эллис и Фортуны)	
EH	ESH	Western Sahara (provisional) (Западная Сахара (временно))	
YE	YEM	Yemen (Йемен)	
YU	YUG	Yugoslavia (Югославия)	
ZM	ZMB	Zambia (Замбия)	
ZW	ZWE	Zimbabwe (Зимбабве)	

Таблица C.3. Коды и названия систем письма (ISO 15924).

Am	Ama	Aramaic (арамейское) ②	
Ar	Ara	Arabic (арабское) ②	
Av	Ave	Avestan (авестское) ②	
Bn	Ben	Bengali (бенгальское) ④	
Bh	Bhm	Brahmi (Ashoka) (брахми) ④	
Bi	Bid	Buhid ④	
Bo	Bod	Tibetan (тибетское) ④	
Bp	Bpm	Bopomofo ③	
Br	Brl	Braille (брайль) ⑥	
Bt	Btk	Batak ④	
Bu	Bug	Buginese (Makassar) (бугийско-макасарское) ④	
By	Bys	Blissymbols ⑥	
Ca	Cam	Cham ④	
Ch	Chu	Old Church Slavonic (древнецерковнославянское) ③	
Ci	Cir	Cirth ③	
Cm	Cmn	Cypro Minoan (кипро-минойское) ⑤	
Co	Cop	Coptic (коптское) ③	
Cp	Cpr	Cypriote syllabary (кипрская слоговая азбука) ⑤	
Cy	Cyr	Cyrillic (кириллица (кириллическое)) ③	
Ds	Dsr	Deseret (Mormon) ③	
Dv	Dvn	Devanagari (Nagari) (деванагари (нагари)) ④	
Ed	Egd	Egyptian demotic (египетское демотическое) ①	

① Иероглифические и клинописные, ② Алфавитные справа налево, ③ Алфавитные слева направо, ④ На основе письма брахми, ⑤ Силлабические (слоговые), ⑥ Идеографические ⑦ Нерасшифрованные.

Коды и названия систем письма (*продолжение*)

`Eh Egh` Egyptian hieratic (египетское иератическое (жреческое)) ①	`Eg Egy` Egyptian hieroglyphs (египетское иероглифическое) ①
`El Ell` Greek (греческое) ③	`Eo Eos` Etruscan and Oscan (этрусское) ③
`Et Eth` Ethiopic (эфиопское) ⑤	`Gl Glg` Glagolitic (глаголица) ③
`Gm Gmu` Gurmukhi (гурмукхи) ④	`Gt Gth` Gothic (готское) ③
`Gu Guj` Gujarati (гуджарати) ④	`Ha Han` Han ideographs (идеографическое письмо хан) ⑥
`He Heb` Hebrew (древнееврейское) ②	`Hg Hgl` Hangul (хангыль) ⑤
`Hm Hmo` Pahawh Hmong ⑤	`Ho Hoo` Hanunóo ④
`Hr Hrg` Hiragana (хирагана) ⑤	`Hu Hun` Old Hungarian runic (древневенгерское руническое) ②
`Hv Hvn` Kök Turki runic (турецкое руническое) ②	`Hy Hye` Armenian (армянское) ③
`Iv Ivl` Indus Valley ⑦	`Ja Jap` Han + Hiragana + Katakana (хан + хирагана + катакана)
`Jl Jlg` Cherokee syllabary (силлабическое письмо чероки) ⑤	`Jw Jwi` Javanese (яванское) ④
`Ka Kam` Georgian (Mxedruli) (грузинское (мхедрули)) ③	`Kn Kan` Kannada (каннада) ④
`Kx Kax` Georgian (Xucuri) (грузинское (хуцури)) ③	`Km Khm` Khmer (кхмерское) ④
`Kh Khn` Hangul + Han (хангыль + хан)	`Kk Kkn` Katakana (катакана) ⑤
`Kr Krn` Karenni (Kayah Li) ④	`Ks Kst` Kharoshthi (кхароштхи) ④
`Lf Laf` Latin (Fraktur variant) (латинское (готическое)) ③	`Lg Lag` Latin (Gaelic variant) (латинское (галльское)) ③
`Lo Lao` Lao (лаосское) ④	`La Lat` Latin (латинское) ③
`Lp Lpc` Lepcha (Róng) ④	`Mh May` Mayan hieroglyphs (майя иероглифическое) ①
`Md Mda` Mandaean (мандейское) ②	`Me Mer` Meroitic (мероитское) ②
`Ml Mlm` Malayalam (малаялам) ④	`Mn Mon` Mongolian (монгольское) ②
`My Mya` Burmese (бирманское) ④	`Na Naa` Linear A (линейное письмо А) ⑤
`Nb Nbb` Linear B (линейное письмо Б) ⑤	`Og Ogm` Ogham (огамическое) ③
`Or Ory` Oriya (ория) ④	`Os Osm` Osmanya (османское) ③
`Ph Pah` Pahlavi (пехлеви) ②	`Ph Phx` Phoenician (финикийское) ②
`Pl Pld` Pollard Phonetic ③	`Pq Pqd` Klingon pIQaD ③
`Pr Prm` Old Permic (древнепермская письменность) ③	`Ps Pst` Phaistos Disk ⑦
`Rn Rnr` Runic (Germanic) (руническое (германское)) ③	`Rr Rro` Rongo (ронго) ⑦
`Sa Sar` South Arabian (южноарабское) ②	`Si Sin` Sinhala (сингальское) ④
`Sl Slb` Canadian Aboriginal Syllabics (силлабическое письмо канадских аборигенов) ⑤	`Sw Sww` Shavian (Shaw) (шавское) ③
`Sj Syj` Syriac (Jacobite variant) (сирийское (яковитское)) ②	`Sn Syn` Syriac (Nestorian variant) (сирийское (несторианское)) ②
`Sy Syr` Syriac (Estrangelo) (сирийское эстрангела) ②	`Tg Tag` Tagalog (тагальское) ④

① Иероглифические и клинописные, ② Алфавитные справа налево, ③ Алфавитные слева направо, ④ На основе письма брахми, ⑤ Силлабические (слоговые), ⑥ Идеографические ⑦ Нерасшифрованные.

Коды и названия систем письма (*продолжение*)

`Ta Tam`	Tamil (тамильское) ④	`Tb Tbw`	Tagbanwa ④
`Te Tel`	Telugu (телугу) ④	`Tf Tfn`	Tifinagh ②
`Th Tha`	Thai (тайское) ④	`Tn Tna`	Thaana ②
`Tw Twr`	Tengwar ③	`Va Vai`	Vai (ваи) ⑤
`Vs Vsp`	Visible Speech (видимая речь (азбука глухих)) ③	`Xa Xas`	Cuneiform, Sumero Akkadian (шумеро-аккадская клинопись) ①
`Xf Xfa`	Cuneiform, Old Persian (древнеперсидское клинописное) ②	`Xk Xkn`	Hiragana + Katakana (хирагана + катакана) ⑤
`Xu Xug`	Cuneiform, Ugaritic (угаритское клинообразное) ②	`Yi Yii`	Yi ⑤
`Zx Zxx`	Code for unwritten languages (код для бесписьменных языков)	`Zy Zyy`	Code for undetermined (код для невыявленных письменностей)
`Zz Zzz`	Code for uncoded (резервный код)		

① Иероглифические и клинописные, ② Алфавитные справа налево, ③ Алфавитные слева направо, ④ На основе письма брахми, ⑤ Силлабические (слоговые), ⑥ Идеографические ⑦ Нерасшифрованные.

Таблица C.4. Семьи языков, охватываемые стандартами ISO/IEC 8859.

8859-1	Западная Европа, Латинская Америка (Latin 1)
8859-2	Восточноевропейские языки (Latin 2)
8859-3	Остальные языки, использующие латинскую письменность (Latin 3)
8859-4	Североевопейские языки (Latin 4)
8859-5	Латинская письменность и кириллица
8859-6	Латинская и арабская письменность
8859-7	Латинская и греческая письменность
8859-8	Латинская письменность и иврит
8859-9	Вариант кодировки Latin 1 для турецкого языка (Latin 5)
8859-10	Финно-угорские, скандинавские и эскимосский языки (Latin 6)
8859-11	Латинская и тайская письменность
8859-12	Невыявленные
8859-13	Балтийские языки (Baltic Rim) (Latin 7)
8859-14	Кельтские языки (Latin 8)
8859-15	Вариант кодировки Latin 1 для французского и финского языков и символа денежной единицы Euro (Latin 9)

C.2 Стандарт Unicode

Стандарт Unicode Unicode Consortium (1996) — это универсальный стандарт кодировки символов, используемый для воспроизведения многоязычных текстов в электронных средствах информации. Он призван упростить международный обмен компьютерными текстовыми файлами. Он включает в себя не только наиболее употребительные текстовые символы всех основных языков мира, но также и коды различных технических и математических символов, благодаря чему ученые и инженеры могут с пользой для себя принять этот стандарт.

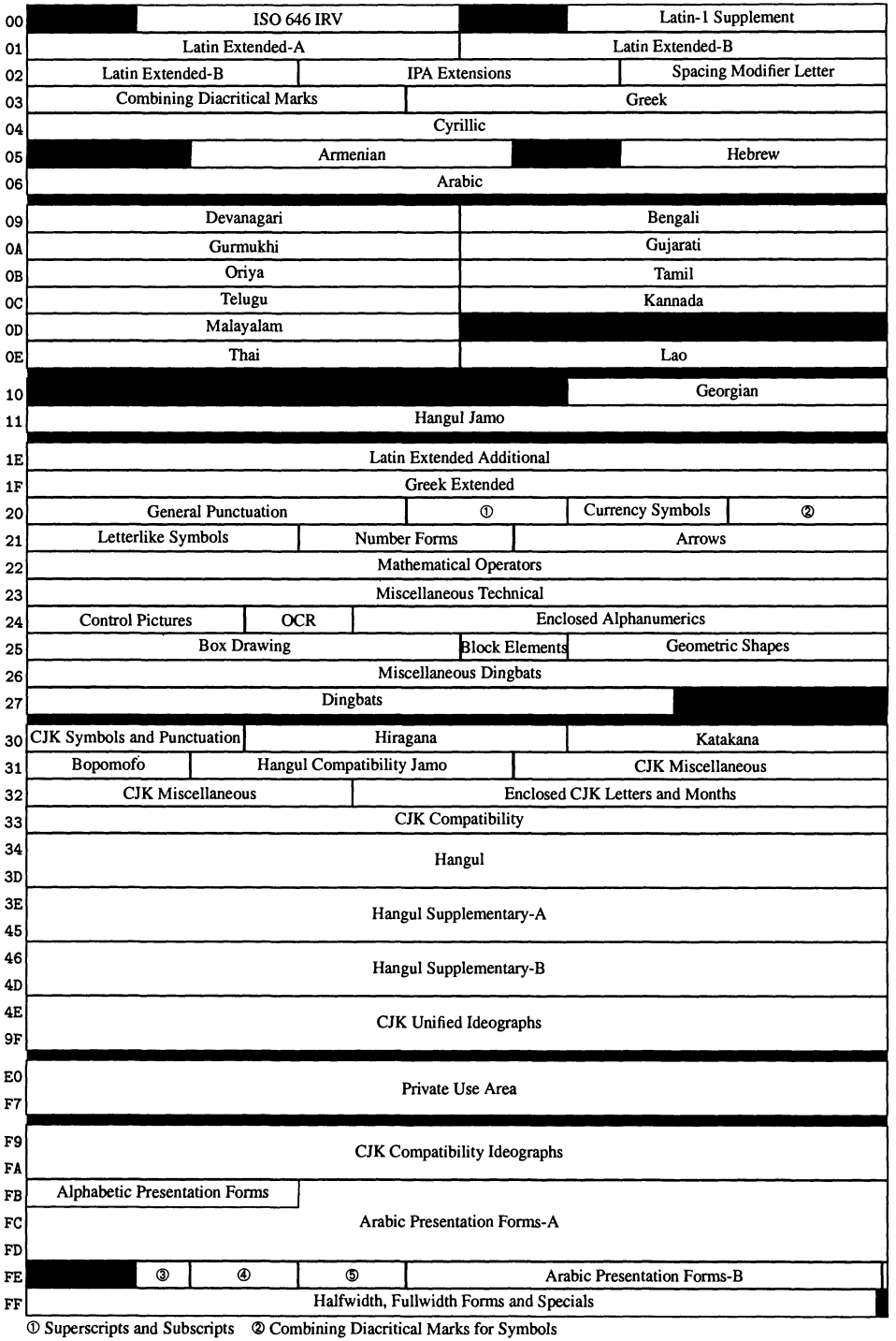


Рис. C.1. Расположение кодов символов в стандарте Unicode (ISO/IEC 10646 BMP).

Стандарт Unicode разрабатывался главным образом в промышленности, и он поддерживается всеми ведущими производителями компьютеров. Он полностью совместим со стандартом ISO/IEC 10646-1 (ISO/IEC:10646-1:1993, 1993). В настоящее время Консорциум Unicode, который осуществляет координацию работ над стандартом Unicode [↪UNICODE], в тесном сотрудничестве с академическими и научно-исследовательскими кругами занят заполнением оставшихся свободными мест из примерно 65000 позиций согласованным набором математических и технических символов.

Стандарт Unicode в сущности является 16-битовым расширением 7-битового стандарта ASCII и 8-битового символьного кода Latin 1 (см. табл. C.4). Однако несмотря на то, что 65000 кодов кажется вполне достаточным для того, чтобы закодировать тысячи общеупотребительных символов всех основных мировых языков, для некоторых приложений требуется большее количество кодов. Поэтому в стандарте Unicode предусмотрен механизм расширения UTF-16, который задействует дополнительные 16-битовые плоскости ISO/IEC 10646-1, позволяя закодировать примерно один миллион символов. Этого более чем достаточно для того, чтобы можно было использовать все известные человечеству символы, включая все исторические письмена в мире.

На рис. C.1 показана структура плоскости Unicode: 256 строк с шестнадцатеричными номерами от `00` до `FF`, каждая из которых содержит 256 символов. Номер строки, показанный с левой стороны рисунка, соответствует старшему байту 16-битового кода символа. По своей структуре Unicode идентичен основной (базовой) плоскости 32-битового стандарта ISO/IEC 10646-1. Приведенный рисунок соответствует первой версии (версии 1) стандарта Unicode. На нем показаны некоторые охватываемые стандартом Unicode системы письма, а именно: латинское, греческое, кириллическое, армянское, иврит, арабское, деванагари, бенгальское, гурмукхи, гуджарати, ория, тамильское, телугу, каннада, малаялам, тайское, лаосское, грузинское, тибетское, японское письмо кана, весь алфавит (полный набор) современного корейского письма хангыль и унифицированный набор китайских, японских и корейских (CJK — Chinese/Japanese/Korean) идеограмм. Продолжается работа над включением в стандарт и других систем письма и (отдельных) символов, в частности эфиопского, канадского силлабического, чероки, сингальского, сирийского, бирманского, кхмерского и Брайля.

В центральной части рисунка C.1 (строки `20`–`27`) видно, что (стандарт) Unicode включает в себя также знаки пунктуации, символы денежных единиц, диакритические знаки, математические и технические символы, стрелки, графические элементы и т. д.

В строках `4E`–`9F` содержатся идеограммы для восточноазиатских языков. Для того чтобы избежать повторного (двойного) кодирования, символы, использующиеся в письменностях разных языков, были *унифицированы*. Это означает, что совпадающим по форме символам соответствует один код. В частности, для объединения китайского, японского и корейского языков (CJK) каждой идеограмме, встречающейся хотя бы в двух из этих языков, присваивался единый код. Этот набор символов известен как «Унифицированный хан»

(«Unified Han»). Однако такая унификация кодов символов не означает, что сами эти символы не могут быть с помощью специально настроенных шрифтов воспроизведены в виде, естественном для читателей, для которых этот язык родной.

С.2.1 Коды и изображения символов

Важно понимать, что в стандарте Unicode числовое выражение получает только код («семантическое значение») символа, но не его воспроизведение в носителе выводимых данных (бумага, экран, звукозапись). Например, «ЛАТИНСКАЯ ПРОПИСНАЯ БУКВА А» (код U-0041), «ГРЕЧЕСКАЯ ПРОПИСНАЯ БУКВА АЛЬФА» (U-0391) и «КИРИЛЛИЧЕСКАЯ ПРОПИСНАЯ БУКВА А» (U-0410) имеют различные семантические значения, хотя видимое воспроизведение у них одно и то же: *глиф* «А».

В стандарте Unicode определяется только интерпретация символов, но не визуальное воспроизведение их графических знаков. Здесь также ничего не говорится о размере, насыщенности, форме, направлении письма и других характеристиках выводимых на экран символов. Эти функции возлагаются на модули воспроизведения символов в программах просмотра или печати.

С.2.2 Unicode и ISO/IEC 10646-1

Стандарт Unicode имеет много общего с международным стандартом ISO/IEC 10646-1, известным также как *Универсальный набор символов* (UCS — *Universal Character Set*). В действительности по своей структуре Unicode совпадает с ISO/IEC 10646-1 и его дополнениями.

Однако благодаря тому, что стандарт ISO/IEC 10646-1 является по существу 32-битовым кодом и поэтому позволяет закодировать более двух (четырех?) миллиардов символов, он имеет гораздо более широкую область применения. В канонической форме ISO/IEC 10646-1 используется четырехмерное пространство кодов, состоящее из 256 трехмерных *групп*. Каждая группа состоит из 256 двумерных *плоскостей*, каждая из которых содержит 256 одномерных *строк*, состоящих из 256 *ячеек* каждая (см. рис С.2). Таким образом, коды символов состоят из четырех байтов, которые, в порядке от младшего к самому старшему, соответствуют номерам ячейки (C-байт), строки (R-байт), плоскости (P-байт) и группы (G-байт).

Первая плоскость (Plane 00) первой группы (Group 00) называется *Базовой многоязычной плоскостью* (BMP). По своей структуре она идентична расположению символов в стандарте Unicode.

Следующие 223 плоскости (от 01 до DF) группы 00 (см. рис. С.3), а также все плоскости (от 00 до FF) в группах 01–5F зарезервированы для дальнейшего расширения стандарта (например, для включения редко используемых символов письма хан или древнейших систем письма). Последние 32 плоскости (от E0 до FF) группы 00, так же, как и все позиции в 32 группах (от

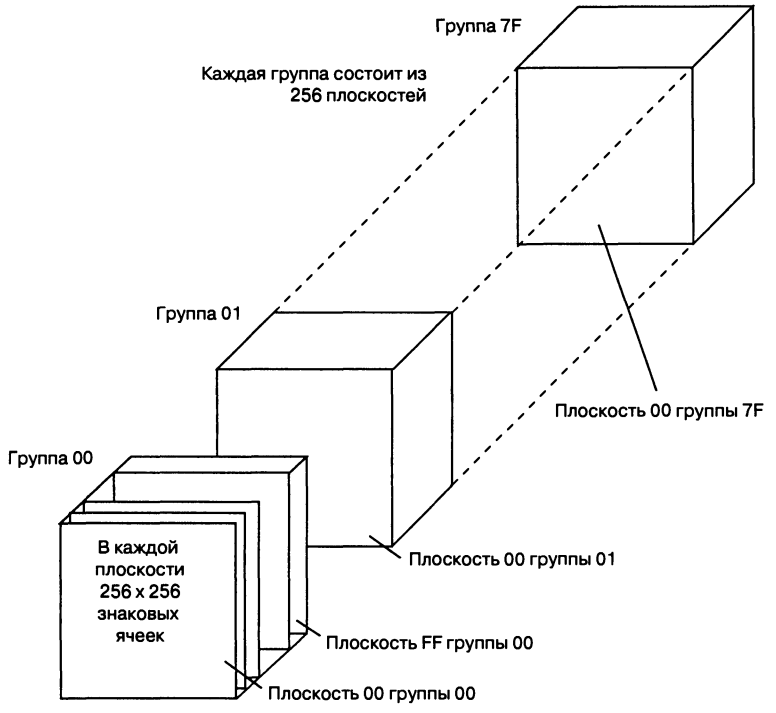


Рис. С.2. Полное пространство четырехбайтовых кодов по стандарту ISO/IEC 10646-1.

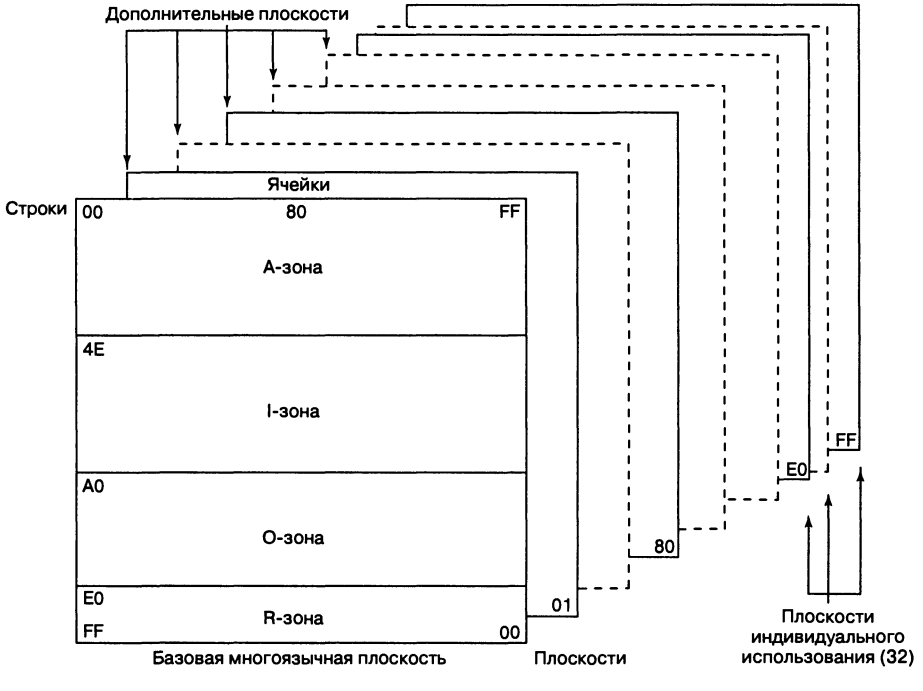


Рис. С.3. Структура группы 00 стандарта ISO/IEC 10646-1.

60 до 7F) зарезервированы для индивидуального использования и стандартом ISO/IEC 10646-1 не определяются.

C.2.3 Кодировки UTF-8 и UTF-16

В стандарте ISO/IEC 10646-1 определены два формата преобразования — UTF-8 и UTF-16, — которые приняты также и в стандарте Unicode (приложение A в Unicode Consortium (1996)). Так же, как и в самом стандарте Unicode, в преобразовании UTF-16 используется 16-битовое кодирование символов, но при этом определенный диапазон кодов может быть использован в качестве механизма расширения для того, чтобы с помощью пар 16-битовых кодов получить доступ к дополнительному миллиону символов.

Возможно, что в скором будущем более полезным окажется формат преобразования UTF-8, в котором все символы Unicode преобразуются в код *переменной длины*, достигающей четырех байт. Первые 128 символов (подмножество ASCII) имеют одинаковые битовые последовательности в кодировках UTF-8 и ASCII, благодаря чему упрощается работа с документами, в которых используется только это подмножество символов. Это означает, что многие существующие программы можно использовать совместно со стандартом Unicode (и UTF-8) без их существенной переработки.

UTF-8 можно рассматривать как метод, позволяющий преобразовывать потоки Unicode или ISO 10646-1 в 8-битовые потоки, при котором символы ASCII остаются неизменными, а остальные расширяются до шести байтов. В действительности для 16-битовых кодов Unicode оказывается достаточно трех байтов, а для кодирования примерно миллиона символов хватит четырех байтов. Последний подход аналогичен UTF-16, в котором 2048 значений кода Unicode (D800–DFFF) резервируются для представления одного миллиона дополнительных символов. Этого миллиона значений кодов должно хватить для всех редких китайских идеограмм и исторических письменностей, которые не помещаются в базовой многоязычной плоскости стандарта ISO 10646.

Ниже показано, как 31 бит кода по стандарту ISO 10646 распределяется по шести байтам кода UTF-8. Необходимо подчеркнуть, что для кодов Unicode никогда не требуется более трех байтов.

Диапазон, охватываемый ISO 10646			Представление UTF-8
Биты	шестнадцатеричный Min	Max	Двоичное представление последовательности байтов
7	00000000	0000007f	0vvvvvvv
11	00000080	000007FF	110vvvvv 10vvvvvv
16	00000800	0000FFFF	1110vvvv 10vvvvvv 10vvvvvv
21	00010000	001FFFFF	11110vvv 10vvvvvv 10vvvvvv 10vvvvvv
26	00200000	03FFFFFF	111110vv 10vvvvvv 10vvvvvv 10vvvvvv 10vvvvvv
31	04000000	7FFFFFFF	1111110v 10vvvvvv 10vvvvvv 10vvvvvv 10vvvvvv 10vvvvvv

Видно, что любой байт UTF-8, начинающийся с двоичного 0, является одинарным (чистый ASCII код). Любой байт, начинающийся с 10, является ведомым байтом в многобайтовой последовательности. Любой другой байт явля-

ется начальным байтом многобайтовой последовательности UTF-8, причем количество двоичных цифр 1 указывает число байтов в этой кодирующей последовательности. Благодаря этому обеспечивается эффективный поиск начала символа, начинающегося с произвольного места в потоке байтов.

Для того чтобы с использованием UTF-16 можно было закодировать миллион символов, из этого процесса преобразований исключается диапазон от 0000D800 до 0000DFFF. В приведенной ниже таблице в шестнадцатеричном представлении показаны различные диапазоны последовательностей UTF-16 и UTF-8, соответствующие полному кодированию UCS-4. Для разделения основных информационных единиц в каждом случае используется точка с запятой.

```
UCS-4           UTF-16        UTF-8

0000 0001;      0001;         01;
0000 007F;      007F;         7F;
0000 0080;      0080;         C2; 80;
0000 07FF;      07FF;         DF; BF;
0000 0800;      0800;         E0; A0; 80;
0000 FFFF;      FFFF;         EF; BF; BF;

0001 0000;      D800; DC00;   F0; 90; 80; 80;
0010 FFFF;      DBFF; DFFF;   F4; 8F; BF; BF;
001F FFFF;                    F7; BF; BF; BF;
0020 0000;                    F8; 88; 80; 80; 80;
03FF FFFF;                    FB; BF; BF; BF; BF;
0400 0000;                    FC; 84; 80; 80; 80; 80;
7FFF FFFF;                    FD; BF; BF; BF; BF; BF;
```

C.3 Иностранные языки в XML

Как выполняется трансляция на язык XML или XSL, если кодировка исходного текста отличается от ASCII? В разделе 6.5.1 было показано, что язык XSL основан на стандарте Unicode. Таким образом, не должно возникать особых проблем при работе с наиболее распространенными мировыми языками. Поэтому рассмотрим два примера, показывающих, как поступать в таких ситуациях.

C.3.1 Кодировки на основе латиницы

В первом случае для воспроизведения написанного по-французски текста используется 8-битовое расширение ASCII, называемое Latin 1 (ISO-8859-1; см. табл. C.4). Исходный файл имеет вид:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE invitation SYSTEM "invitationfr.dtd">
<invitation>
<!-- ++++ Partie entête ++++ -->
<entête>
<à>Anna, Bernard, Didier, Johanna</à>
<date>Vendredi prochain à 20 heures</date>
<où>Le Café du Web</où>
<pourquoi>Mon premier bébé XML</pourquoi>
</entête>
<!-- ++++ Partie corps  ++++ -->
<corps>
```

```
<par>
J'ai le plaisir de vous inviter à la célébration
de la naissance d'<emph>Invitation</emph>, mon
premier enfant document XML.
</par>
<par>
S'il vous plaît, faites tout votre possible pour me rejoindre
vendredi prochain. Et n'oubliez pas d'emmener vos amis.
</par>
<par>
Je me réjouis <emph>vraiment</emph> d'avance de votre présence.
</par>
</corps>
<!-- ++++ Partie finale  ++++ -->
<fin>
<signature>Michel</signature>
</fin>
</invitation>
```

Как видно из приведенного ниже DTD-файла `invitationfr.dtd`, символы со знаками ударения использовались как в тексте, так и для некоторых типов элементов.

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<!-- DTD invitation (version française) -->
<!-- 11 novembre 1998 mg -->
<!ELEMENT invitation (entête, corps, fin) >
<!ELEMENT entête     (à, date, où, pourquoi?) >
<!ELEMENT date       (#PCDATA) >
<!ELEMENT à          (#PCDATA) >
<!ELEMENT où         (#PCDATA) >
<!ELEMENT pourquoi   (#PCDATA) >
<!ELEMENT corps      (par+) >
<!ELEMENT par        (#PCDATA|emph)* >
<!ELEMENT emph       (#PCDATA) >
<!ELEMENT fin        (signature) >
<!ELEMENT signature  (#PCDATA) >
```

Для преобразования этого XML-файла, например в LaTeX, можно использовать программу `xt` с приведенным ниже стилем `invlat1fr.xsl`:

```
<?xml version='1.0' encoding="ISO-8859-1"?>
<!-- minilatex.xsl -->
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="text" indent="no" encoding="ISO-8859-1"/>

<xsl:strip-space elements="*"/>

<xsl:template match="/">
<xsl:text>\documentclass[francais]{article}
\usepackage{invitationfr}
\usepackage[T1]{fontenc}
\begin{document}
</xsl:text>
<xsl:apply-templates/>
<xsl:text>\end{document}
</xsl:text>
</xsl:template>

<xsl:template match="entête">
<xsl:text>\begin{Front}
\To{</xsl:text>
<xsl:value-of select="à"/>
```

```
25  <xsl:text>}
26  \Date{</xsl:text>
27  <xsl:value-of select="date"/>
28  <xsl:text>}
29  \Where{</xsl:text>
30  <xsl:value-of select="où"/>
31  <xsl:text>}
32  \Why{</xsl:text>
33  <xsl:value-of select="pourquoi"/>
34  <xsl:text>}
35  \end{Front}
36  </xsl:text>
37  </xsl:template>
38
39  <xsl:template match="corps">
40  <xsl:text>\begin{Body}
41  </xsl:text>
42    <xsl:apply-templates/>
43  <xsl:text>\end{Body}
44  </xsl:text>
45  </xsl:template>
46
47  <xsl:template match="par">
48  <xsl:text>\par</xsl:text>
49  <xsl:apply-templates/>
50  </xsl:template>
51
52  <xsl:template match="emph">
53  <xsl:text>\emph{</xsl:text>
54  <xsl:apply-templates/>
55  <xsl:text>}</xsl:text>
56  </xsl:template>
57
58  <xsl:template match="fin">
59  <xsl:text>\begin{Back}
60  \Signature{</xsl:text>
61  <xsl:value-of select="signature"/>
62  <xsl:text>}
63  \end{Back}
64  </xsl:text>
65  </xsl:template>
66
67  </xsl:stylesheet>
```

Обратите внимание, что в первой строке этого файла, как и во всех остальных XML-файлах настоящего раздела, присутствует атрибут `encoding` со значением, равным `ISO-8859-1`. Как объяснялось в разд. 6.5.3, кодировку задавать *необходимо*, потому что автоматически распознаются только UTF-8 и Unicode. Аналогично объявляется, что в результирующем текстовом файле используется та же самая кодировка ISO-8859-1 (строка 6). Кроме того, мы определяем несколько большее количество LaTeX'овских команд (строки 11–13), чем в тех случаях, когда мы имеем дело с чисто английскими текстами (см. разд. 7.3.2 на с. 343). К тому же мы внесли некоторые дополнения в пакетный файл `invitation.sty`, чтобы «справиться» с написанным по-французски заголовком.

```
1  % invitation.sty
2  % Package to format invitation.xml
3  \setlength{\textwidth}{22pc}
4  \setlength{\parskip}{1ex}
5  \setlength{\parindent}{0pt}
6  \pagestyle{empty}%% Turn off page numbering
```

```
\RequirePackage{array,calc}
\newcommand{\ToTitle}{To whom}
\newcommand{\WhyTitle}{Occasion}
\newcommand{\WhereTitle}{Venue}
\newcommand{\DateTitle}{When}
\newcommand{\SignatureTitle}{From}
\DeclareOption{francais}{%  French text for fixed texts
  \renewcommand{\ToTitle}{À}
  \renewcommand{\WhyTitle}{À l'occasion de}
  \renewcommand{\WhereTitle}{Où}
  \renewcommand{\DateTitle}{Quand}
  \renewcommand{\SignatureTitle}{De la part de}}
\newenvironment{Front}%
  {\begin{center}
     \Huge\sffamily INVITATION
   \end{center}
  }
  {\begin{flushleft}
   \rule{\linewidth}{1pt}\\[2mm]
   \begin{tabular}{@{}>{\bfseries}ll@{}}
   \ToTitle:    & \@To     \\
   \WhyTitle:   & \@Why    \\
   \WhereTitle: & \@Where  \\
   \DateTitle:  & \@Date
   \end{tabular}\\[2mm]
   \rule{\linewidth}{1pt}
   \end{flushleft}
  }
\newenvironment{Body}{\vspace*{\parskip}}{\vspace*{\parskip}}
\newenvironment{Back}
  {\begin{flushleft}}
  {\hspace*{.5\linewidth}\fbox{\SignatureTitle: \emph{\@Sig}}
   \end{flushleft}
  }
\newcommand{\To}[1]{\gdef\@To{#1}}
\newcommand{\Date}[1]{\gdef\@Date{#1}}
\newcommand{\Where}[1]{\gdef\@Where{#1}}
\newcommand{\Why}[1]{\gdef\@Why{#1}}
\newcommand{\Signature}[1]{\gdef\@Sig{#1}}
\ProcessOptions
```

Как можно видеть, мы параметризовали «фиксированные элементы текста», которые используются в заголовке и подписи. Мы также проверяем наличие опции `francais`, чтобы требуемым образом переопределить «фиксированный текст». Эту более сложную версию целесообразно сравнить с предыдущей, приведенной на с. 344; отпечатанный результат показан на рис. C.4.

C.3.2 Обработка кодировок, отличающихся от латиницы, с помощью UTF-8

Поскольку в языке XML допускается непосредственное использование Unicode, можно ожидать, что не потребуется принимать специальных мер для кодирования языков, алфавиты которых отличаются от латинского, например русского, греческого, китайского, японского, арабского и т. д. В этом разделе показывается, как можно без проблем использовать в одном файле русский и греческий языки, а также несложные математические выражения, однако то же самое относится и к более сложным ситуациям.

Мы использовали редактор для Unicode `Yudit` [↪YUDIT], который поддерживает ввод на нескольких языках и допускает использование многих кодиро-

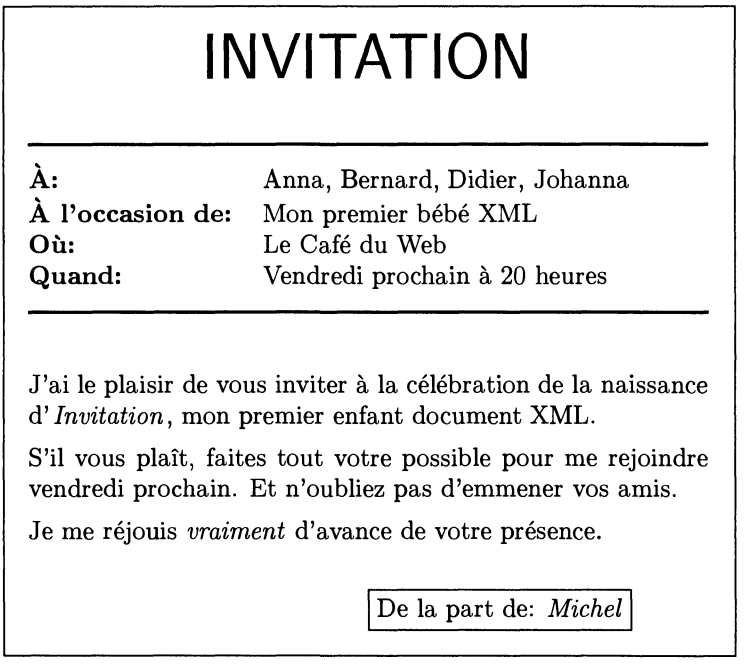


Рис. C.4. Приглашение на французском языке (LaTeX'овская версия).

вок при вводе и выводе. Первоначально он был разработан Гаспаром Синаем и отличается относительно простым и естественным графическим пользовательским интерфейсом.

На рис. C.5 показан файл на языке XML. В его первой части показаны три способа ввода текста на русском языке. Мы (частично) воспроизведем эти строки для более внимательного изучения.

```
 1  <?xml version="1.0"?>
 2  <!DOCTYPE  mydoc [
 3  <!ELEMENT mydoc (#PCDATA)>
 4  <!ENTITY % ISOcyr1 SYSTEM "ISOcyr1.pen">
 5  %ISOcyr1;
 6  ]>
 7  <mydoc>
 8  <par>The word Russian (ÐãÑČÑĄÑĄÐžĐÿĐź) in Cyrillic: <br/>
 9  Using ISO Cyrillic set:
10  &Rcy;&ucy;&scy;&scy;&kcy;&icy;&jcy;  <br/>
11  Using XML Unicode entities:
12  &#x0420;&#x0443;&#x0441;&#x0441;&#x043a;&#x0438;&#x0439;
13  </par>
```

В строках 4–5 определяется внешнее множество объектов `ISOcyr1`, которые в локальной системе будут вводиться из файла `ISOcyr1.pen`. В нем содержатся определения букв кириллицы, такие, как:

```
<!ENTITY rcy    "&#x440;"> <!--small er, Cyrillic -->
<!ENTITY Rcy    "&#x420;"> <!--capital ER, Cyrillic -->
```

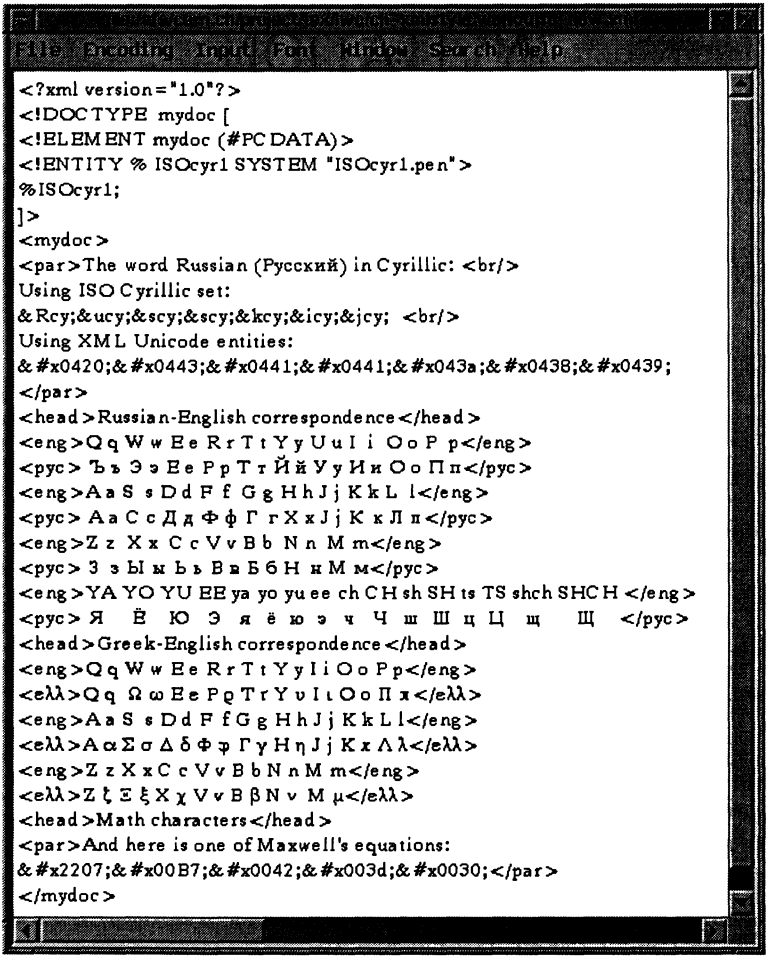


Рис. С.5. XML-файл с русским и греческим языками и математикой в кодировке UTF-8.

```
<!ENTITY scy    "&#x441;"> <!--small es, Cyrillic -->
<!ENTITY Scy    "&#x421;"> <!--capital ES, Cyrillic -->
```

В приведенном выше фрагменте строчные и прописные буквы кириллицы «р» и «с» определяются в зависимости от их номера в Unicode. Именно так получаются буквы кириллицы из символических имен в строке 10 нашего исходного файла. Можно конечно же, как это сделано в строке 12, непосредственно ввести сами определенные стандартом Unicode ссылки на символ (сравните приведенные в строках 10 и 12, соответственно, ссылки на объект: «`&Rcy;`» и «`&#x0420;`» и решите, какая из них легче читается и удобнее в работе). С другой стороны, в строке 8 текст на русском языке непосредственно введен с помощью программы `Yudit`, которая сохранила его в кодировке UTF-8, отво-

дя по два байта на каждый не-ASCII алфавитный символ. Эти байтовые коды (представления) выглядят странно при печати в кодировке T1, которая используется в этом примере, но в кодовой таблице T1 (см., например, табл. 9.1 на с. 261 в книге (Goossens et al., 1994)) можно легко увидеть, что они воспроизводят последовательность байтов

```
D0A0D183D181D181D0BAD0B8D0B9
```

которая, как объяснялось в разд. C.2.3, является представленным в кодировке UTF-8 эквивалентом последовательности кодов Unicode из строки 12.

На рис. C.5 можно также увидеть несколько пар строк, где в первой строке приведены буквы или их комбинации, которые вводятся на стандартной английской клавиатуре, а во второй строке — результат, получаемый при этом с помощью программы `Yudit`. Например, после строки «Russian-English correspondence» в меню «`Input`» был выбран пункт «Russian», а после строки «Greek-English correspondence» в том же меню «`Input`» был выбран пункт «Greek». В соответствии с выбранным в меню «`Input`» языком, программа `Yudit` будет правильно воспроизводить символ на экране и сохранять его в кодировке, которую можно выбрать в меню «`Encoding`». Таким образом, для того чтобы ввести написанные по-русски строки текста, нам пришлось переключить «входной» язык с английского на русский и опять на английский. Аналогично при наборе «греческой части» текста понадобилось переключать язык с английского на греческий и обратно. Видно, что текст на трех языках размещен внутри элементов, у которых имена меток соответствуют названиям языков, написанным на их родном алфавите. Таким образом, наряду с именами элементов XML, задаваемыми в кодировке Latin 1 (см. разд. C.3.1), в языке XML можно определять имена элементов, понятные авторам, живущим в самых разных частях света (см. также разд. 6.5.1), и это — большой шаг вперед по сравнению с языком HTML, в котором набор меток был фиксированным, а их имена имеют смысл только на английском языке.

Заключительная часть XML-файла имеет вид:

```
19  <head>Math characters</head>
20  <par>And here is one of Maxwell's equations:
21  &#x2207;&#x00B7;&#x0042;&#x003d;&#x0030;</par>
22  </mydoc>
```

В строке 21 показаны ссылки Unicode на символы, используемые для записи короткой математической формулы.

Следующая задача состоит в том, чтобы преобразовать этот файл во что-либо такое, что можно посмотреть или напечатать. Приведенная ниже таблица стилей `utf8.xsl` преобразует предшествующий XML-файл `utf8.xml` в формат HTML.

```
1  <?xml version='1.0' encoding="UTF-8"?>
2  <xsl:stylesheet version="1.0"
3       xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4  <xsl:output method="html" encoding="utf-8"/>
5  <xsl:template match="/">
6    <html>
```

```
 7     <head>
 8     <title>UTF8 file</title>
 9     </head>
10     <body>
11     <h1>Handling UTF-8 files</h1>
12     <xsl:apply-templates/>
13     </body>
14     </html>
15   </xsl:template>
16   <xsl:template match="br">
17     <br />
18   </xsl:template>
19   <xsl:template match="par">
20     <p><xsl:apply-templates/></p>
21   </xsl:template>
22   <xsl:template match="head">
23     <h2><xsl:apply-templates/></h2>
24   </xsl:template>
25   <!-- eliminate English keyboard input -->
26   <xsl:template match="eng">
27   </xsl:template>
28   <!-- transmit Russian keyboard input -->
29   <xsl:template match="&#x0440;&#x0443;&#x0441;">
30   <p>&#x25c6; <xsl:apply-templates/></p>
31   </xsl:template>
32   <!-- transmit Greek keyboard input -->
33   <xsl:template match="Îţ Îż Îż">
34   <p>ãÛÑ <xsl:apply-templates/></p>
35   </xsl:template>
36   </xsl:stylesheet>
```

```
<xsl:template match="ελλ">
<p>● <xsl:apply-templates/></p>
```

В этом стиле наибольший интерес представляет его начало. В строке 4 определяется, что результирующий документ создается в формате HTML с использованием кодировки UTF-8. Благодаря этому появляется возможность вывода непосредственно элементов языка HTML. В строках 6–14 целый файл вставляется внутрь корректной конструкции языка HTML. Смысл остальных строк достаточно прост. В частности, строки 26–27 исключают из результирующего документа исходные строки, написанные по-английски. Благодаря тому, что стили XSL сами являются XML-файлами, мы можем использовать полный набор алфавитных и идеографических символов Unicode; в частности, можно делать ссылки на нелатинские имена элементов в XML-документе (в строках 33 и 34 они снова выглядят довольно странно). В действительности, для того чтобы в XSL-файле сделать ссылку на символы Unicode, можно использовать как сами символы, так и коды ссылок. Ссылки на символы используются в строке 29 для того, чтобы получить строку кириллицы «рус», а также в строке 30 — для того чтобы в начале каждого абзаца поместить символ Unicode «25c6» (черный ромб). В разделе, содержащем текст на греческом языке (строки 33–35), мы непосредственно используем символы Unicode (соответствующая часть таблицы стилей показана справа от обсуждаемых строк). В обоих случаях между ромбом или маркером (точкой, кружком) и последующим текстом добавляется неразрывный пробел (` `).

И последнее замечание: имеет смысл обратить внимание на дополнительный пробел, который в пустых элементах вставляется между именем элемента и закрывающей угловой скобкой «/>» (см. строку 17 в таблице стилей). Благодаря этому современные HTML-навигаторы, которые пока еще не понимают

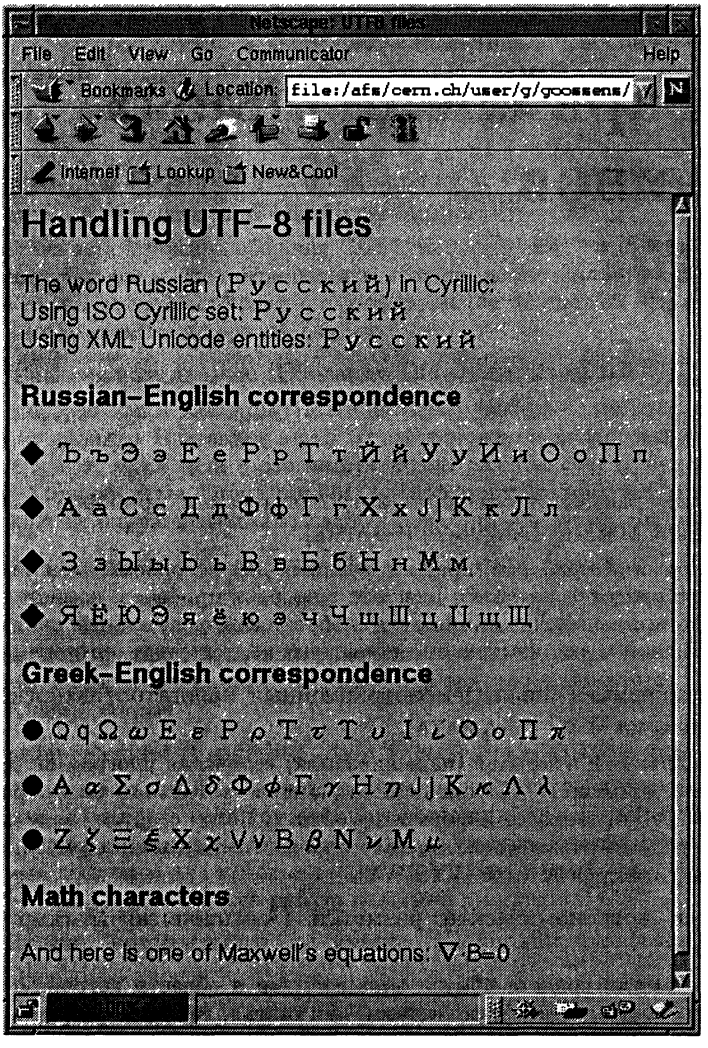


Рис. C.6. Воспроизведение UTF-8-файла в формате HTML с помощью навигатора Netscape.

синтаксис языка XML, могут правильно интерпретировать эти строки (в действительности они будут игнорировать заключительную косую черту «/»; см. также разд. B.5.2.1).

Мы можем обработать оба этих файла процессором XSL, например `xt`, и вывести HTML-файл `utf8.html`:

```
xt utf8.xml utf8.xsl utf8.html
```

На рис. C.6 показано, как будет выглядеть результат в навигаторе Netscape.

Словарь специальных терминов

AIML Язык разметки описаний астрономических приборов (Astronomical Instrument Markup Language) [↪AIML].

Язык описаний приборов, предназначенных для астрономических наблюдений, включает средства разметки для характеристик приборов, команд управления, описаний потоков данных (включая изображения и служебные данные), форматы сообщений, механизмы связи и описания конвейерных алгоритмов.

Amaya Экспериментально-демонстрационный навигатор/авторская система Консорциума W3C [↪AMAYA].

Универсальная и открытая (наращиваемая) сервисная программа, представленная Консорциумом W3C для демонстрации и испытания новых разработок в области Web-протоколов и форматов данных. Помимо других возможностей позволяет редактировать сложные математические выражения в HTML-страницах, используя интерфейс типа WYSIWYG.

AML Язык астрономической разметки (Astronomical Markup Language) [↪AML].

Язык разметки средств обмена метаданными в области астрономии, позволяющий работать с такими объектами, как статья, таблица, рисунок, сведения о работающих в этой области.

Атрибут (Attribute) см. разд. 6.5.4.2.

Позволяет использовать в определениях типа документа DTD языков SGML/XML именованные характеристики для типов элементов.

BIOML Язык разметки текстов о биополимерах (BIOpolymer Markup Language) [↪BIOML].

Дает возможность описывать экспериментальные данные о молекулярных объектах, состоящих из биополимеров, таких, как белки и гены.

BMP Базовая многоязычная плоскость (Basic Multilingual Plane) (см. разд. C.2).

Подмножество набора 31-битовых кодов UCS по стандарту ISO/IEC10646-1 (плоскость 0, группа 0). Структура содержимого идентична Unicode.

BSML Язык разметки биоинформационных последовательностей (Bioinformatic Sequence Markup Language) [↪BSML].

Стандарт кодирования и воспроизведения информации о строении ДНК, РНК и белков.

CALS Постоянные закупки и сопровождение в течение жизненного цикла (Continuous Acquisition and Life-Cycle Support) [↪CALS].

Инициатива министерства обороны США, направленная на совершенствование процессов закупки и сопровождения в течение жизненного цикла для систем оружия путем ускоренного создания и применения данных об объектах и технической информации в цифровой форме.

CDATA Символьные данные (Character Data) (см. разд. 6.5.4.7).

Часть информации в XML-документе, которая вообще не подвергается синтаксическому анализу. Благодаря этому в тексте можно использовать символы разметки `&`, `<` и `>`, даже если в соответствующем разделе отсутствуют какие-либо элементы или объекты. Описания `CDATA` могут присутствовать в атрибутах XML (разд. 6.5.4.2), а фрагменты `CDATA` могут появляться в самих документах.

CML Язык химической разметки (Chemical Markup Language) [↪CML].

Позволяет иметь дело с информацией химического содержания при работе на языке XML. CML и связанные с ним программные средства предоставляют не зависящую от платформы и соглашений спецификацию для обмена информацией в области молекулярных наук. Поставляется вместе с навигатором (Jumbo) для визуализации данных.

CSS Каскадные стили (Cascading Style Sheets) [↪STYLECSS] (см. разд. 7.4).

Простой декларативный язык, который позволяет авторам и пользователям вводить в структурированные документы, написанные на языках HTML или XML, стилистическую информацию (о шрифтах, интервалах, цвете и т. д.). В настоящее время доступны для использования два уровня стилей CSS: CSS1 [↪CSS1] и CSS2 [↪CSS2], продолжается работа над третьим уровнем CSS3.

DCD Описание содержимого документа (Document Content Description) [↪DCD].

Средство формирования структурных схем для описания правил, относящихся к структуре и содержимому XML-документов. DCD представляет собой RDF-словарь и предназначается для определения ограничений, накладываемых на документ синтаксисом языка XML, включая задание основных типов данных.

DDML Язык разметки определений документов (Document Definition Markup Language) [↪DDML].

Простой язык описания схем в XML для кодирования логического содержимого DTD в виде XML-документа. Позволяет анализировать и использовать информацию в виде схем при работе с инструментальными XML-средствами для широкого употребления.

DOM Объектная модель документа (Document Object Model) [↪DOMGEN].

Не зависящий от платформы и языка интерфейс, который позволяет программам и сценариям динамически получать доступ к документам, обновлять их содержимое, структуру и стиль. Допускается возможность дальнейшей обработки документа, причем результаты этой обработки могут быть включены в отобра-

жаемую страницу. Первый уровень был принят как рекомендуемый [↪DOML1], в то же время продолжается работа над спецификацией уровня 2 [↪DOML2], где будут добавлены интерфейсы для модели объекта CSS, модели события и запросов.

DSSSL Язык семантики и описания стиля документа (Document Style Semantics and Specification Language) (см. разд. 7.5).

Международный стандарт ISO 10179 ISO/IEC:10179 (1996), принятый в начале 1995 г. Он представляет собой основу для выражения понятий и действий, необходимых для преобразования документа со структурной разметкой, в его окончательную «физическую» форму. Хотя первоначально этот стандарт предназначался для работы с документами, с его помощью можно определять и другие форматы верстки, например такие, которые требуются при работе с базами данных. Дополнительную информацию о разработанном Джеймсом Кларком языке DSSSL можно получить на сайте [↪DSSSLCLARK].

DTD Определение типа документа (Document Type Definition) (см. разд. 6.5.4).

Набор правил, описывающих, какие типы элементов допустимы в XML-документе, и какова форма их содержимого. DTD также определяет допустимые для каждого типа элемента атрибуты и описывает объекты, на которые делаются ссылки из документа, а также обозначения, которые могут быть использованы.

EAD Кодированное архивное описание (Encoded Archival Description) [↪EAD].

Общедоступный стандарт кодировки для средств машинного поиска, таких, как описи, реестры, каталоги и другие документы, создаваемые архивами, библиотеками, музеями и хранилищами рукописей для обеспечения доступа к их фондам.

GIF Формат графического обмена (Graphics Interchange Format) [Murray and vanRyper (1996)].

Формат, первоначально разработанный службой Compuserve, для того чтобы облегчить передачу изображений между различными платформами путем хранения нескольких растровых изображений в одном и том же файле. В GIF-файлах использовался запатентованный метод сжатия Лемпела-Зева-Уолша (LZW). В случае проблем с лицензированием целесообразно подумать об использовании формата PNG.

HL7 Kona Proposal [↪KONA].

Метод, позволяющий с помощью SGML/XML создавать, изменять и обрабатывать представленные в электронной форме данные из области здравоохранения (EHR).

HTML Язык разметки гипертекста (Hypertext Markup language) (см. разд. 1.1.3).

Важнейший из основанных на SGML языков разметки, используемых в Сети.

HTTP Протокол передачи гипертекста (Hypertext Transport Protocol) (см. разд. 1.1.1).

Метод, с помощью которого WWW-серверы могут обмениваться информацией друг с другом.

ICE Обмен информацией и содержимым (Information and Content Exchange) [↪ICE].

Протокол ICE определяет роли и ответственность распространителей и получателей информации и описывает формат и метод обмена содержимым. В нем имеются средства управления и контроля за отношениями, возникающими в процессе информационного обмена, поэтому ICE будет полезен при автоматическом обмене и повторном использовании содержимого.

Java Объектно-ориентированный язык программирования, разработанный фирмой Sun Microsystems [↪JAVADOC].

Язык Java с самого начала разрабатывался для того, чтобы обеспечить возможность безопасного выполнения машинных программ в Сети даже в тех случаях, когда исходный код ненадежен и, может быть, создан со злым умыслом. Java обеспечивает мобильность программ между платформами не только в исходном виде, но и в двоичном представлении результата компиляции. Для того чтобы обеспечить такую возможность, Java-программа компилируется в промежуточный байт-код, который впоследствии интерпретируется «на ходу» (в фоновом режиме) Java-интерпретатором. Таким образом, переносить с платформы на платформу необходимо только сам интерпретатор и несколько библиотек собственных кодов; сами же Java-программы повсюду выполняются в неизменном виде. Поскольку Java — это весьма компактный язык, характеризующийся строгой проверкой типов и отсутствием ненадежных конструкций, то писать и читать программы на нем легко, и, кроме того, их относительно легко отлаживать. Много полезной информации можно найти на сайте [↪JAVAFAQ].

Javascript Язык подготовки сценариев [↪JAVASC].

Межплатформенный, объектно-ориентированный язык подготовки сценариев, разработанный фирмой Netscape. Javascript — это не Java! С его помощью можно создавать приложения, работающие в Интернете. Клиентские приложения выполняются в навигаторе, тогда как серверные приложения — на сервере. JavaScript позволяет создавать динамические HTML-страницы, которые обрабатывают пользовательский ввод и поддерживают данные постоянного характера, используя специальные объекты, файлы и реляционные базы данных. Впервые язык Javascript был применен фирмой Netscape в ее навигаторе Netscape Navigator 2.0, но в настоящее время существует «стандартная» версия под названием Ecmascript [↪ECMASC].

JPEG Объединенная группа экспертов по машинной обработке фотоизображений (Joint Photographic Experts Group) [↪JPEG].

Стандарт ISO, описывающий механизмы сжатия изображений как для полноцветных, так и полутоновых образов естественных, реальных картин. Он годится для фотографий, произведений искусства и других подобных материалов. Хорошее описание можно найти в работе Murray and vanRyper (1996).

JSML Язык разметки Java Speech (Java Speech Markup Language) [↪JSML].

Язык, позволяющий комментировать текст при работе с синтезатором речи Java Speech API. Элементы JSML предоставляют синтезатору речи детальную информацию о произношении текста. В состав JSML входят элементы, которые описывают структуру документа, задают фонетическую транскрипцию слов и предложений, а также размещают маркеры в тексте. JSML содержит также просодические элементы, которые управляют построением фраз, ударением, высотой тона, скоростью речи и другими важными характеристиками. Соответствующая

разметка текста улучшает качество и естественность синтезированного голоса. В JSML используется набор символов Unicode, благодаря чему его можно использовать для разметки текста на большинстве языков мира.

LinuxML Язык разметки системы Linux (Linux Markup Language) [↪LINUXML].

Проект, направленный на то, чтобы в фактическом стандарте UNIX перейти для межпроцессного взаимодействия и хранения данных от ASCII-строк к языку XML. Идея заключается в том, что команды системы UNIX будут создавать результат в формате XML. Благодаря этому такие обрабатывающие программы, как `sort` или `xterm`, смогут распознавать семантическое значение входных данных и, следовательно, выполнять над ними больше полезных операций.

MathML Язык математической разметки (Mathematical Markup Language) [↪W2CMATH] (см. разд. 8.1).

Область использования языка MathML — это использование математического и научного содержимого Сети, а также другие приложения, например системы компьютерной алгебры, книгоиздание и синтез речи.

MIME Многоцелевые расширения электронной почты в сети Интернет (Multipurpose Internet Mail Extensions) [↪RFC2045].

MIME — это набор спецификаций, обеспечивающих структурирование тела сообщения посредством его расчленения. Отдельные части тела сообщения могут иметь разные типы, например текст, изображение, звукозапись, или полностью инкапсулированное сообщение. В нем также предусмотрена возможность кодирования сообщений с использованием наборов символов, отличных от 7-битового ASCII.

MPEG Экспертная группа по вопросам движущегося изображения (Motion Picture Experts Group) [↪MPEG].

Стандарт ISO, который определяет способ кодирования потоков данных для сжатия аудио- и видеоинформации. См. ISO/IEC:11172 (1993); Murray and vanRyper (1996).

OASIS Организация по развитию стандартов структурированной информации (Organization for the Advancement of Structured Information Standards) [↪OASIS].

Некоммерческий международный консорциум, объединяющий пользователей и поставщиков продуктов и услуг, созданный исключительно с целью обмена не зависящими от продукта документами и данными. Основанный в 1993 г. как SGML Open, впоследствии OASIS был расширен, чтобы охватить весь набор стандартов обработки структурированной информации, включая XML, SGML и HTML.

OFE Открытый обмен финансовой информацией (Open Financial Exchange) [↪OFX].

OFE стандартизирует электронный обмен финансовой информацией между финансовыми учреждениями, фирмами и потребителями через Интернет. Первоначально введенный в начале 1997 г., в настоящее время он поддерживает широкий спектр финансовой деятельности, включая банковское обслуживание потребителей и малого бизнеса, оплату счетов потребителей и малого бизнеса, выставление счетов и инвестиции, включая государственный долг, облигации и совместные фонды. В будущем планируется добавить и включить в спецификацию другие

финансовые услуги, в том числе финансовое планирование и страхование. Разметка, вероятно, будет создаваться и интерпретироваться программно, без вмешательства человека.

OSD Описание формата открытого программного обеспечения (Open Software Format Description).

Распространение и обновление программного обеспечения через Сеть, включая принудительное обновление программ и автоматическую инсталляцию. Скорее всего, разметка должна будет создаваться программами компоновки программного обеспечения и использоваться для установки этого программного обеспечения, оставаясь вне поля зрения человека.

Parser Синтаксический анализатор, программа грамматического разбора (см. разд. 6.6).

Программа, которая преобразует последовательный поток (данных) разметки (например, XML- или SGML-файл) в результирующую (выходную) структуру, доступную другой программе более высокого уровня. Анализаторы XML в процессе обработки разметки могут выполнять аттестацию или проверку, чтобы убедиться в правильности разметки. Некоторые анализаторы XML представлены в разд. 6.6.

PDF Формат мобильных документов (Portable Document Format) (см. гл. 2).

Потомок разработанного фирмой Adobe Systems языка PostScript, оптимизированный для улучшения навигации и доставки по сети Интернет. В частности, в PDF введена независимость страниц, добавлены возможности работы с гипертекстом и защиты, разрешена подстановка шрифта и включены средства сжатия для минимизации размера файла.

Perl [↪PERL]

Язык программирования высокого уровня, первоначально разработанный Ларри Уоллом. Унаследовав множество свойств языка программирования C, он также включает в себя полезные концепции многих других средств системы UNIX. Благодаря возможностям обработки и работы с файлами и с текстом, язык Perl особенно хорошо подходит для задач, включающих быстрое моделирование, для системных утилит, инструментальных программных средств, задач организации работы системы, доступа к базам данных, графического программирования, передачи данных по сети и программирования для WWW.

PGML Язык точной разметки графики (Precision Graphics Markup Language) [↪PGML].

Вариант XML, в котором реализована модель представления изображений PDF/PostScript.

PNG Мобильная сетевая графика (Portable Network Graphics) [↪PNG].

Расширяемый формат файла для полноценного, мобильного, сильно сжатого хранения растровых изображений. Его можно свободно использовать вместо GIF, а также во многих обычных ситуациях — вместо TIFF. Поддерживаются индексированные цвета, полутоновые и полноцветные изображения, а также необязательный альфа-канал. Формат PNG хорош в приложениях, обеспечивающих визуализацию в оперативном режиме (наподобие WWW), поскольку он полностью соответствует современным возможностям дисплея. Формат PNG является робастным, поскольку обеспечивает полную проверку целостности файла, а кроме

того, включает простые средства обнаружения ряда ошибок передачи данных. В данном формате предусмотрено хранение корректировочных данных (яркость, контрастность, цветовая насыщенность), позволяющих сохранять цветопередачу при обмене изображениями между различными компьютерными платформами.

PostScript Язык описания страницы [↪ADOBEPS].

Компьютерный язык, который описывает внешний вид страницы, включая такие элементы, как текст, графика и отсканированные изображения, при выводе на принтер или на другое устройство вывода. Он был предложен фирмой Adobe [↪ADOBE] в 1985 г. и превратился в предпочтительный язык для высококачественной печати на самых разных устройствах вывода, включая черно-белые и цветные принтеры, наборные машины, экранные устройства отображения и системы цифровой полиграфии.

RDF Формат описания ресурса (Resource Description Format) [↪RDF].

Описывает содержимое Web-ресурса для обеспечения его автоматической обработки. Может использоваться для описания содержимого Web-сайта, для задания дополнительной информации машине поиска или интеллектуальному агенту, для декларирования прав собственности и т. д.

SAX Стандарт API для основанного на событиях синтаксического анализа XML [↪SAX].

В разделе B.6 показано, как можно использовать SAX для работы с XML-документами.

SDML Язык разметки подписанных документов (Signed Document Markup Language) [↪SDML].

Язык, разработанный для электронных документов с цифровой подписью. Предусмотрена возможность самоидентификации отдельных фрагментов текста, образующих документ, и объединения их в разделы, которые могут иметь деловое значение и могут подписываться по отдельности или вместе. Части документа можно добавлять или уничтожать, не лишая законной силы ранее поставленные подписи; операции подписания, индоссирования и свидетельствования могут выполняться как над документом в целом, так и по частям.

SGML Стандартный обобщенный язык разметки (Standard Generalized Markup Language) (см. гл. 6).

Международный стандарт (ISO 8879:1986), который описывает обобщенную схему разметки для воспроизведения логической структуры документов способом, не зависящим от системы и платформы.

SMIL Язык синхронизированного объединения мультимедийных объектов (Synchronized Multimedia Integration Language) [↪SMIL].

Приложение XML, которое позволяет объединить набор независимых мультимедийных объектов в синхронизированную мультимедийную презентацию. Функциональные возможности SMIL включают описание протекания презентации по времени, описание ее расположения на экране и связывание гиперссылок с медиаобъектами.

SOX Схема для объектно-ориентированного XML (Schema for Object-oriented XML) [↪SOX].

Средство работы со схемами для определения структуры, содержимого и семантики XML-документов, обеспечивающее проверку правильности XML и более высокий уровень автоматического контроля содержания. Содержит основные внутренние (встроенные) типы данных, расширяемый механизм определения типов данных, модель содержимого, наследование атрибутов интерфейса, мощный механизм работы с пространством имен и встроенную документацию.

SpeechML Язык разметки речи (Speech Markup Language) [↪SPEECHML].

Язык для создания основанных на сетевых технологиях диалоговых приложений, которые взаимодействуют с пользователем посредством речевого ввода и вывода. SpeechML можно использовать для того, чтобы обеспечить диалоговый доступ к источникам информации и приложениям, расположенным в любом месте Интернета, из автомобиля, по телефону, с карманного компьютера, с настольного персонального компьютера и т. д.

SVG Масштабируемая векторная графика (Scalable Vector Graphics) [↪SVGSPEC].

Открытый формат векторной графики, который работает на различных платформах, при различной разрешающей способности устройства вывода, в нескольких видах цветовых пространств и во всей доступной ширине спектра. Цель создания SVG — сделать Web-документы более компактными, быстро загружаемыми, более интерактивными и воспроизводимыми при самом разном разрешении устройств вывода: от маленьких портативных устройств для мониторов офисных компьютеров до принтеров высокого разрешения.

TEI Инициатива в области кодирования текста (Text Encoding Initiative) [↪TEIHOME] (см. разд. B.4.3).

Международный научный проект, направленный на содействие обмену и созданию электронных текстов. Структурные фрагменты текста размечаются в первоисточнике с использованием стандартизованной схемы для упрощения обмена и компьютерной обработки.

TeXML [↪TEXML] см. разд. 8.2.4.

Позволяет использовать TEX'овские средства форматирования при наборе XML-документов.

TIFF Формат файла элемента изображений (Tag Image File Format) [↪TIFF6].

Первоначально являлся методом хранения черно-белых сканированных изображений, хотя впоследствии была добавлена и работа с цветом, благодаря чему TIFF превратился в стандартный формат файла для большинства графических и настольных издательских систем.

UCS Универсальный набор символов (Universal Character Set)

Международный стандарт ISO/IEC10646-1 для кодирования всех мировых систем письма и кодирования символов. Как и Unicode, использует в 31-битовой кодировке только 16-битовое подмножество, соответствующее базовой многоязычной плоскости (см. разд. C.2).

Unicode См. разд. C.2 [↪UNICODE].

Стандарт для международного кодирования символов. В отличие от 8-битовых кодов, которые в настоящее время поддерживает большинство систем, Unicode поддерживает символы, представляемые кодами шириной 2 байта. Благодаря этому можно закодировать 65536 символов, а не только лишь 256, как при однобайтовом кодировании.

URI Универсальный идентификатор ресурса (Universal Resource Identifier) (см. разд. 1.1.2).

Универсальная схема адресации для определения местоположения информации в сети Интернет.

URL Унифицированный указатель местоположения ресурса (Universal Resource Locator) (см. разд. 1.1.2).

Особый вид URI, использовавшийся первоначально, когда Сеть только создавалась.

Valid Корректный документ (см. разд. 6.4.2.1).

Корректный документ — это документ, который правильно оформлен в соответствии с его DTD.

VML Язык векторной разметки (Vector Markup Language) [↪VML].

Приложение XML, которое определяет формат кодирования векторной информации совместно с дополнительной разметкой для описания возможностей воспроизведения и редактирования этой информации.

VRML Язык моделирования виртуальной реальности (Virtual Reality Modeling Language) [↪WEB3D]

Стандарт формата файла для трехмерных мультимедийных и разделяемых виртуальных пространств в Интернете. По сравнению с графическим интерфейсом HTML в VRML добавлены взаимодействие, структурированная графика и дополнительные измерения (глубина и время). Области использования VRML охватывают промышленную, научную и образовательную графику, а также более современные трехмерные разделяемые виртуальные пространства и сообщества.

WAP Протокол для приложений, связанных с радиовещанием (Wireless Application Protocol).

Язык разметки на базе XML для обмена информацией посредством узкополосных устройств.

WebDAV Распределенная по Всемирной паутине авторская система и управление версиями (World Wide Web Distributed Authoring and Versioning) [↪WEBDAV].

Расширения протокола HTTP для обеспечения более полной поддержки распределенной авторской системы.

Well-Formed Правильно оформленный документ (см. разд. 6.4.2.1).

Правильно оформленный документ может иметь, а может и не иметь DTD. Правильно оформленные документы должны начинаться с декларации XML и содержать надлежащим образом вложенные и размеченные элементы.

WIDL Язык описания Web-интерфейса (Web Interface Definition Language) [↪WIDL].

XML-приложение, определяющее метаязык, в котором над ресурсами WWW, организованными по принципу принадлежности к документу, создается архитектура, основанная на выполняемых функциях. В WIDL взаимодействия с Web-серверами могут быть определены как функциональные интерфейсы, доступ к которым возможен с удаленных систем по стандартным Web-протоколам. При этом обеспечивается структура, необходимая для создания клиентского кода на таких языках, как C/C++, COBOL и Visual Basic.

W3C Консорциум производителей программного обеспечения для WWW (World Wide Web Consortium) [↪W3C].

Этот международный промышленный консорциум был образован в октябре 1994 г. для того, чтобы полностью реализовать возможности Всемирной паутины путем разработки общепринятых протоколов, которые способствовали бы ее развитию и обеспечивали бы возможность совместной работы. Консорциум W3C образован совместно Лабораторией вычислительной техники Массачусетского технологического института (США), Национальным институтом исследований в области компьютерной обработки данных и автоматики (Европа) и Keio University Shonan Fujisawa Campus (Япония). В число выполняемых Консорциумом функций входят хранение информации о Всемирной паутине для разработчиков и пользователей, эталонные тексты реализаций для внедрения и продвижения стандартов, а также различные прототипы и пробные (экспериментальные) приложения для демонстрации использования новых технологий.

XCatalog Предложение использовать синтаксис языка XML для каталогов [↪XCATALOG].

Предложенная спецификация основывается на документе Консорциума OASIS [↪ENTIMAN] и предназначена для управления элементами (каталога) путем отображения публичных идентификаторов XML на системные идентификаторы XML с помощью URI.

XHTML Расширяемый язык разметки гипертекста (Extensible HyperText Markup Language) [↪HTMLINXML] (see Section B.5.2).

Спецификация, которая переформулирует HTML 4.0 в виде приложения XML 1.0. XHTML задает три конфигурации документа, каждую со своим собственным URI, в виде пространств имен XML. Семантика и атрибуты элементов определяются в рекомендациях Консорциума W3C для HTML 4.0. Они должны стать основой для последующей расширяемости XHTML.

Xlink Язык компоновки XML [↪XLINKSPEC].

Определяет конструкции, которые можно вставлять в ресурсы XML для описания ссылок между объектами. Ссылка (в контексте Xlink) — это явное взаимоотношение между двумя или более объектами данных или частями объектов данных. Xlink использует синтаксис XML для создания структур, которые могут описывать как присущие современному HTML простые однонаправленные гиперссылки, так и более сложные многосторонние и типизированные ссылки.

XMI Формат обмена метаданными XML (XML Metadata Interchange Format) [↪XML].

Обеспечивает простой обмен метаданными между средствами моделирования, основанными на разработанном группой управления объектами унифициро-

ванном языке моделирования (Object Management Group's Unified Modeling Language OMG (UML)), а также между этими средствами и хранилищами метаданных с помощью средств OMG для работы с метаобъектами (OMG Meta Object Facility (MOF)). Благодаря этой архитектуре, метаданные можно разделять программным способом с использованием интерфейсов CORBA, описанных в стандартах MOF и UML, либо с помощью основанного на XML потока (или файла), содержащего перенастраиваемые спецификации моделирования (спецификации упомянутых стандартов OMG можно найти на сайте [↪OMGTECH]).

XML Расширяемый язык разметки (Extensible Markup Language) [↪XMLSPEC].

Подмножество языка SGML, которое должно позволить использовать, получать и обрабатывать обычный SGML в Сети тем же способом, который годится и для HTML. XML разработан для облегчения реализации и для взаимодействия как с SGML, так и с HTML. Сам язык XML обсуждается в гл. 6, тогда как в гл. 8 речь идет о трансляции LaTeX'а (с математикой или без нее) в XML.

XML-Data XML-данные [↪XMLDATA].

Словарь языка XML, содержащий требующие определения схемы и классы объектов документа.

XML/EDI Обмен электронными данными (Electronic Data Interchange) [↪XMLEDI].

Стандартный комплекс средств для обмена различными типами данных, например счет-фактура, медицинский счет, информация о состоянии проекта. Он позволяет согласованно и корректно осуществлять поиск, декодирование, обработку и отображение данных, обмен которыми осуществляется на уровне транзакций через интерфейс прикладных программ (Application Program Interface — API), Web-средства автоматизации, порталы баз данных, каталоги, потоки документов или сообщений. Эти операции выполняются на основе EDI-словарей; используя хранилища с оперативным (online) доступом, можно расширять состав используемых элементов, включая лексику языка делового общения, правила выполнения операций, объекты операций.

Xpointer Язык указателей XML (XML Pointer Language) [↪XPTR].

Язык, который поддерживает адресацию во внутренних структурах XML-документов. В частности, в нем предусмотрены специальные ссылки на элементы, символьные строки и другие фрагменты XML-документов, независимо от того, обладают ли они явным атрибутом `ID`.

XSL Расширяемый язык стилей (Extensible Stylesheet Language) [↪XSLSPEC] (см. разд. 7.6).

Язык для описания стилей. Он состоит из языка преобразований и словаря форматирующих объектов.

XUL Расширяемый язык пользовательского интерфейса (Extensible User Interface Language) [↪XUL].

Описывает содержимое окон и диалогов. В состав XUL входят конструкции для типовых управляющих воздействий диалога и элементов управления наподобие панелей инструментов, деревьев, индикаторов выполнения и меню.

Каталог WWW-ресурсов

ACROTEX: Основанные на PDF пособия по математике, в которых широко используются формы Acrobat.
`http://www.math.uakron.edu/~dpstory/acrotex.html`

ADOBE: Домашняя страница фирмы Adobe.
`http://www.adobe.com/`

ADOBEPS: Информация фирмы Adobe по языку PostScript.
`http://www.adobe.com/prodindex/postscript/main.html`

AELFRED: Синтаксический анализатор XML Дэвида Меггинсона Ælfred.
`http://www.microstar.com/aelfred.html`

AIML: Язык разметки описаний астрономических приборов.
`http://pioneer.gsfc.nasa.gov/public/aiml/`

AMAYA: Экспериментально-демонстрационный навигатор/авторская система.
`http://www.w3.org/Amaya/`

AML: Язык астрономической разметки.
`http://strule.cs.qub.ac.uk/~damieng/these/`

ASTER: Система Т. В. Рамана для устного чтения математических выражений в TEX'овских документах.
`http://www.cs.cornell.edu/Info/People/raman/aster/demo.html`

AXML: Стандарт XML с комментариями Тима Брея.
`http://www.xml.com/axml/axml.html`

BALISE: Среда программирования SGML для структурированных документов.
`http://www.balise.com`

BIOML: Язык XML для записи информации биополимерных последовательностей.
`http://www.proteometrics.com/BIOML/`

BOSAKXML: Книга Иона Босака *XML, Java, and the Future of the Web.*
`http://metalab.unc.edu/pub/sun-info/standards/xml/why/xmlapps.html`

BSML: Язык разметки биоинформационных последовательностей для графического отображения генома.
`http://visualgenomics.com/sbir/rfc.htm`

CALS: Постоянные закупки и сопровождение в течение жизненного цикла.
`http://navysgml.dt.navy.mil/cals.html`

CERN: Европейская лаборатория по физике элементарных частиц.
`http://www.cern.ch/Public/`

CML: Возможности языка химической разметки.
`http://xml-cml.org`

CONTEXT: Макропакет Ганса Хагена CONTEXt.
`http://www.ntg.nl/context/`

CSS1: Каскадные стили, версия 1, рекомендации Консорциума W3C (декабрь 1996 г.).
`http://www.w3.org/TR/REC-CSS1`

CSS2: Каскадные стили, версия 2, рекомендации Консорциума W3C (май 1998 г.).
`http://www.w3.org/TR/REC-CSS2/Overview.html`

CVSREPOS: (Сайт разработчиков) LaTeX2HTML, репозиторий CVS.
`http://saftsack.fs.uni-bayreuth.de/~latex2ht/`

DARPA: Управление перспективных исследовательских программ.
`http://www.arpa.mil/`

DAVENPORT: Группа Davenport, разработчики Docbook DTD.
`http://www.oasis-open.org/docbook/`

DBDSSSL: Модульные стили DocBook.
`http://nwalsh.com/docbook/dsssl/index.html`

DBVIEW: DocBook 3.0: Алфавитный указатель элементов пользователя.
`http://www.ora.com/homepages/dtdparse/docbook/3.0/elements.htm`

DBXML: Docbook DTD, транслированный в XML.
`http://nwalsh.com/docbook/xml/index.html`

DCD: Описание содержимого документа для языка XML.
`http://www.w3.org/TR/NOTE-dcd`

DDML: Спецификация языка разметки определения документа, версия 1.0.
`http://www.w3.org/TR/NOTE-ddml`

DOCBOOK: Домашняя страница DocBook OASIS.
`http://www.oasis-open.org/docbook/`

DOMGEN: Страница OASIS, посвященная стандарту Document Object Model (DOM) Консорциума W3C.
`http://www.oasis-open.org/cover/dom.html`

DOML1: Document Object Model (DOM), спецификация уровня 1 (версия 1), рекомендация Консорциума W3C от 1 октября 1998 г.
`http://www.w3.org/TR/REC-DOM-Level-1/`

DOML2: Document Object Model (DOM), спецификация уровня 2 (рабочие материалы).
`http://www.w3.org/TR/WD-DOM-Level-2/`

DRAKOSWWW: Работа Никоса Дракоса *From Text to Hypertext: A Post-Hoc Rationalisation of LaTeX2HTML.*
`http://www.cbl.leeds.ac.uk/nikos/doc/www94/www94.html`

DSSSLCLARK: Информация Джеймса Кларка о DSSSL.
`http://www.jclark.com/dsssl/`

DSSSLLIST: Список рассылки DSSSL.
`http://www.mulberrytech.com/dsssl/dssslist/index.html`

DSSSLMML: Стили DSSSL для MathML.
`http://www.nag.co.uk/projects/openmath/mml-files/`

DSSSLONL: DSSSL в режиме онлайн.
`http://metalab.unc.edu/pub/sun-info/standards/dsssl/dssslo/do960816.htm`

DSSSLPDF: Спецификация DSSSL в режиме онлайн в формате PDF.
`ftp://ftp.ornl.gov/pub/sgml/WG8/DSSSL/dsssl96b.pdf`

DSSSLSUM: Краткое изложение синтаксиса DSSSL, сделанное Харви Бингемом.
`http://www.tiac.net/users/bingham/dssslsyn/index.htm`

DSSSLTUTA: Работа Поля Преско *Introduction to DSSSL.*
`http://www.prescod.net/dsssl/`

DSSSLTUTB: Работа Даниель М. Жерман *An Introduction to DSSSL.*
`http://csg.uwaterloo.ca/~dmg/dsssl/tutorial/tutorial.html`

DTDPARSE: Синтаксический анализатор SGML DTD Нормана Уолша.
`http://www.ora.com/homepages/dtdparse/`

DVIOUT: Программа просмотра DVI-файлов для операционной системы Windows 9X/NT, поддерживающая HyperTEX'овские команды `\specials`.
`http://akagi.ms.u-tokyo.ac.jp/dvioute_help.html`

DVIPS: Программа Тома Рокицки для преобразования DVI в PostScript.
`http://www.radicaleye.com/dvips.html`

EAD: Инициатива по кодированию архивных описаний.
`http://www.loc.gov/ead/ead.html`

EC: Web-сервер Европейского союза.
`http://europa.eu.Int/`

ECMASC: Стандарт ECMA-262, спецификация языка ECMAScript.
`http://www.ecma.ch/stand/ecma-262.htm`

ELEMATTR: Использование элементов и атрибутов в языках SGML/XML.
`http://www.oasis-open.org/cover/elementsAndAttrs.html`

ENTIMAN: Управление объектами.
`http://www.oasis-open.org/html/a401.htm`

EPRINT: Лос-Аламосский электронный архив препринтов научных статей.
`http://xxx.lanl.gov/`

EPSIG: Специальная группа, объединяющая заинтересованных в электронных публикациях.
`http://www.oasis-open.org/cover/epsig.html`

EPSTOPDF: Perl-утилита для EPS-файлов, делающая размеры страниц, равными ограничивающему прямоугольнику.
`http://www.tug.org/applications/pdftex/epstopdf`

ESIS: ESIS—ISO 8879 Стандарт структуры информационных документов.
`http://www.oasis-open.org/cover/WG8-n931a.html`

FANCYTREE: Усовершенствованный визуализатор для XML-документов.
`http://www.skew.org/xml/`

FOSI: Стандарт MIL-M-28001C; спецификация форматирования вывода.
`http://www-cals.itsi.disa.mil/core/standards/28001C.PDF`

FPISERVER: Сервер Питера Флинна для разрешения формальных публичных идентификаторов.
`http://www.ucc.ie/cgi-bin/PUBLIC`

FPISYNTAX: Синтаксис формальных публичных идентификаторов.
`http://www.oasis-open.org/cover/tauber-fpi.html`

FRM: Система авторизации и публикации документов (поддерживает SGML).
http://www.adobe.com/prodindex/framemaker/

GROVES: Представление значений характеристик в DSSSL в виде графов.
http://www.oasis-open.org/cover/topics.html#groves

GSHOME: Домашняя страница Ghostscript.
http://www.cs.wisc.edu/~ghost/

HOOD: Определения типов HTML-документов.
http://www.utoronto.ca/webdocs/HTMLdocs/HTML_Spec/html.html

HTML2SPEC: Спецификация HTML 2.0.
http://www.w3.org/MarkUp/html-spec/

HTML4: Спецификация HTML 4.0.
http://www.w3.org/TR/REC-html40/

HTMLENTS: Символьные ссылки на объекты в HTML 4.0.
http://www.w3.org/TR/REC-html40/sgml/entities.html

HTMLINXML: XHTML 1.0: расширяемый язык разметки гипертекста.
http://www.w3.org/TR/xhtml1/

HTMLTIDY: Tidy Дейва Раггета приводит HTML-страницы в порядок и преобразует их в XHTML.
http://www.w3.org/People/Raggett/tidy/

HTTPNG: Консорциум W3C и деятельность проблемной группы проектирования Интернета в области HTTP-NG.
http://www.w3.org/Protocols/HTTP-NG/Activity.html

HTTPRFC: Спецификация HTTP 1.1.
http://www.w3.org/Protocols/rfc2068/rfc2068

HYPERLTX: Hyperlatex Отфрида Чеонга — транслятор LaTeX'а в HTML.
http://www.cs.ust.hk/~otfried/Hyperlatex/

HYPERTEX: Типичные вопросы о HyperTeX'е.
http://xxx.lanl.gov/hypertex/

HYTIME: Стандарт Hytime.
http://www.ornl.gov/sgml/wg8/docs/n1920/html/n1920.html

ICE: Протокол обмена информацией и содержимым.
http://www.w3.org/TR/NOTE-ice

IETF: Проблемная группа проектирования Интернета.
http://www.ietf.org/

IMAGEMAGICK: Утилиты Джона Кристи для обработки изображений.
http://www.wizards.dupont.com/cristy/

INRIA: Национальный институт исследований в области компьютерной обработки данных и автоматики.
http://www.inria.fr/

ISO8879TC2: Адаптация Web SGML.
http://www.ornl.gov/sgml/WG8/document/1955.htm

ITRANS: Пакет для печати текстов, использующих индийские системы письма.
http://www.aczone.com/itrans/

JADE: Джеймс Кларк. Реализация DSSSL.
http://www.jclark.com/jade/

JADETEX: Макропакет JadeTeX.
ftp://ctan.tug.org/tex-archive/macros/jadetex/

JADETEXB: TeXFOTBuilder: генератор TEX-вывода для Jade.
`http://www.jclark.com/jade/TeX.htm`

JAVADOC: Страница фирмы Sun с документацией по языку Java.
`http://java.sun.com/docs/index.html`

JAVAFAQ: Cafe au Lait Java: типичные вопросы, новости и ресурсы.
`http://metalab.unc.edu/javafaq/`

JAVASC: Руководство по языку JavaScript фирмы Netscape.
`http://developer.netscape.com/docs/manuals/communicator/jsguide4/`

JPEG: Типичные вопросы относительно JPEG.
`http://www.faqs.org/faqs/jpeg-faq/`

JSML: Спецификация языка разметки Java Speech.
`http://java.sun.com/products/java-media/speech/forDevelopers/JSML/index.html`

JUMBO: JAVA-XML, навигатор JUMBO.
`http://ala.vsms.nottingham.ac.uk/vsms/java/jumbo/`

KEIO: Университет Киото.
`http://www.keio.ac.jp/`

KONA: Проект по медицине HL7 Kona.
`http://www.mcis.duke.edu:80/standards/HL7/sigs/sgml/WhitePapers/KONA/`

L2HCTAN: Исходные файлы LATEX2HTML в сети CTAN.
`ftp://ctan.tug.org/tex-archive//support/latex2html/sources/`

L2HDOC: Встроенная (оперативно доступная) документация по LATEX2HTML.
`http://www-dsed.llnl.gov/files/programs/unix/latex2html/manual/`

L2HLIST: Список рассылки LATEX2HTML.
`mailto:latex2html@tug.org`

L2HMML: Создание разметки MathML с помощью LATEX2HTML, WebEQ и WebTEX.
`http://www.geom.umn.edu/~ross/webtex/webtex/`

L2HSC: Хранилище исходных файлов LATEX2HTML.
`http://saftsack.fs.uni-bayreuth.de/~latex2ht/`

L2HTUG: Хранилище исходных файлов LATEX2HTML.

LECHE: Новости и ресурсы XML.
`http://metalab.unc.edu/xml/`

LINUXML: Система UNIX в проекте XML.
`http://www.ozemail.com.au/~birchb/linuxml/linuxml.htm`

LOTUSXSL: Процессор XSL, написанный на языке Java.
`http://www.alphaWorks.ibm.com/formula/LotusXSL`

LTXML: Средства LTXML XML.
`http://www.ltg.ed.ac.uk/software/xml/`

MALAYALAM: Малаялам-TEX.
`ftp://ctan.tug.org/tex-archive/languages/malayalam/`

MALEREX: Исключения в SGML и язык XML.
`http://www.arbortext.com/Think_Tank/XML_Resources/SGML_Exceptions_and_XML/body_sgml_exceptions_and_xml.html`

MAPLE: Домашняя страница Waterloo Maple.
`http://www.maplesoft.com/`

MATHEMATICA: Домашняя страница Wolfram Research Mathematica.
`http://www.wolfram.com/`

MATHSYMP: Национальный математический симпозиум.
`http://www-texdev.mpce.mq.edu.au/mathsymp/`

MATHTYPE: Редактор формул, работающий по принципу WYSIWYG и создающий на выходе TEX или MathML.
`http://www.mathtype.com`

MICROPRESS: Домашняя страница Micropress.
`http://www.micropress-inc.com`

MIT: Массачусетский технологический институт.
`http://web.mit.edu/`

MMLGUID: Всеобъемлющее руководство по MathML, поддерживаемое Панкаем Камтханом.
`http://indy.cs.concordia.ca/mathml/`

MMLRES: Каталог ресурсов MathML.
`http://www.webeq.com/webeq/mathml/resources.html`

MMLSPEC: Спецификация MathML.
`http://www.w3.org/TR/WD-math/`

MPEG: Типичные вопросы относительно MPEG.
`http://www.faqs.org/faqs/jpeg-faq/`

NDVI: Модуль расширения фирмы Netscape для просмотра DVI-файлов (поддерживает HyperTEX'овские команды `\specials`).
`http://norma.nikhef.nl/~t16/ndvi_doc.html`

NETPBM: Утилиты `netpbm`.
`ftp://ftp.x.org/contrib/utilities/netpbm-1mar1994.p1.tar.gz`

NIKNAK: Коммерческий преобразователь PostScript в PDF.
`http://www.5-d.com/niknak.htm`

NISTHMF: Цифровая библиотека математических функций.
`http://math.nist.gov/DigitalMathLib/`

OASIS: Организация содействия прогрессу стандартов структурированной информации.
`http://www.oasis-open.org/`

OFX: Свободный обмен финансовыми данными.
`http://www.ofx.net`

OMEGA: 16-битовая реализация TEX'а на основе Unicode.
`http://www.gutenberg.eu.org/omega/`

OMGTECH: Техническая документация группы по управлению объектами.
`http://www.omg.org/techprocess/meetings/schedule/Technology_Adoptions.html`

OMNIMARK: Окружение SGML для управления и распространения индивидуальных материалов в Сети.
`http://www.omnimark.com`

PASSIVETEX: Процессор Себастиана Ратца, переводящий документы из формата TEX в XSL.
`http://users.ox.ac.uk/~rahtz/passivetex/`

PDFMARKD: Документация по Pdfmark.
`http://partners.adobe.com/supportservice/devrelations/technotes.html`

PDFMARKP: Учебник для начинающих по Pdfmark.
`http://www.ifconnection.de/~tm/`

PDFSPEC: Спецификация PDF.
`http://www.adobe.com/supportservice/devrelations/PDFS/TN/PDFSPEC.PDF`

PDFTEXEX: Примеры использования pdfTeX.
`http://www.tug.org/applications/pdftex/`

PDFTEXS: Исходные файлы pdfTeX.
`ftp://ftp.cstug.cz/pub/tex/local/cstug/thanh/`

PDFZONE: PDF Zone — полезная информация о PDF и Acrobat.
`http://www.pdfzone.com/`

PERL: Домашняя страница Perl.
`http://www.perl.com/pace/pub/`

PERLSGML: Perl-программы и библиотеки для обработки описаний типов документов (DTD) на языке SGML.
`http://www.oac.uci.edu/indiv/ehood/perlSGML.html`

PGML: Язык разметки точной графики.
`http://www.w3.org/TR/1998/NOTE-PGML`

PNG: Домашняя страница «мобильной сетевой графики».
`http://www.cdrom.com/pub/png/`

PSGML: Основной режим Emacs для редактирования закодированных SGML-документов.
`http://www.lysator.liu.se/projects/about_psgml.html`

PSGMLXML: «Заплатки» для добавления XML к PSGML.
`http://www.megginson.com/Software/psgmlxml-19980218.zip`

QWERTZ: Конвертер из HTML в LaTeX.
`http://nathan.gmd.de/projects/zeno/qwertz/qwertz.html`

RAGHIST: История HTML (Raggett et al. (1998), гл. 2).
`http://www.w3.org/People/Raggett/book4/ch02.html`

RAGHTML: 10-минутное введение в HTML Раггета.
`http://www.w3.org/MarkUp/Guide/`

RDF: Структура описания ресурсов.
`http://www.w3.org/RDF/`

RFC1630: Универсальные идентификаторы ресурсов во Всемирной паутине.
`http://info.internet.isi.edu:80/in-notes/rfc/files/rfc1630.txt`

RFC1738: Спецификация унифицированных указателей ресурса (URL).
`http://info.internet.isi.edu:80/in-notes/rfc/files/rfc1738.txt`

RFC1866: Язык разметки гипертекста – 2.0.
`http://info.internet.isi.edu:80/in-notes/rfc/files/rfc1866.txt`

RFC2045: Многоцелевые расширения электронной почты в сети Интернет (MIME). Часть 1: формат тела сообщения в Интернете.
`http://info.internet.isi.edu:80/in-notes/rfc/files/rfc2045.txt`

RFC2141: Унифицированное имя ресурса (URN): спецификация синтаксиса.
`http://info.internet.isi.edu:80/in-notes/rfc/files/rfc2141.txt`

RFC2396: Унифицированные идентификаторы ресурса (URI): общий синтаксис.
`http://info.internet.isi.edu:80/in-notes/rfc/files/rfc2396.txt`

SAMANALA: Транслитерация (для транскрипции) Samanala.
`http://www-texdev.mpce.mq.edu.au/l2h/indic/Indica/samanala/`

SAX: Простой программный интерфейс приложения для XML.
`http://www.megginson.com/SAX/index.html`

SAXJAVA: Дистрибутив программного интерфейса SAX на языке Java.
`http://www.megginson.com/SAX/javadoc/packages.html`

SAXON: Набор средств для обработки XML-документов Майкла Кая.
`http://www.jclark.com/xml/xt.html`

SDML: Язык разметки подписанных документов: заметка Консорциума W3C.
`http://www.w3.org/TR/NOTE-SDML/`

SGML2XML: Преобразование SGML в XML.
`http://www.xml.com/xml/pub/98/07/dtd/index.html`

SGMLC: Язык программирования для обработки SGML-документов в операционной системе MS Windows.
`http://www.dircon.co.uk/sgml/`

SGMLNEW: Новости SGML и XML.
`http://www.oasis-open.org/cover/sgmlnew.html`

SGMLSPM: Архив исходных файлов SGMLSpm.
`http://www.megginson.com/Software/SGMLSpm-1.03ii.tar.gz`

SGMLTOOLS: Домашняя страница проекта разработки средств SGML.
`http://www.sgmltools.org/`

SHOWTREE: Стили для воспроизведения дерева XSLT.
`http://www.cranesoftwrights.com/resources/showtree/`

SINDOC: Документация по системе Sinhala-TEX в формате HTML (Sinh-HTMLdocs).
`http://www-texdev.mpce.mq.edu.au/l2h/indic/Sinhala/lreport/`

SINTEX: Sinhala-TEX — система для набора текстов на сингальском языке.
`ftp://ctan.tug.org/tex-archive/language/sinhala/`

SMIL: Язык интеграции мультимедийных объектов в синхронизированную презентацию. Спецификация версии 1.0.
`http://www.w3.org/TR/REC-smil/`

SOTU98: Доклад Президента Клинтона о положении США в 1998 г.
`http://www.whitehouse.gov/WH/SOTU98/address.html`

SOTU99: Доклад Президента Клинтона о положении США в 1999 г.
`http://www.whitehouse.gov/WH/New/html/19990119-2656.html`

SOX: Схема для объектно-ориентированного XML.
`http://www.w3.org/TR/NOTE-SOX/`

SP: `SP` — синтаксический анализатор SGML.
`http://www.jclark.com/sp/`

SPDOC: Документация по синтаксическому анализатору `SP`.
`http://www.jclark.com/sp/nsgmls.htm`

SPEECHML: Язык разметки SpeechML.
`http://www.alphaWorks.ibm.com/formula/speechml`

STYLECSS: Страница Консорциума W3C, посвященная стилям CSS.
`http://www.w3.org/Style/css/`

SVGSPEC: Спецификация масштабируемой векторной графики (рабочие материалы).
`http://www.w3.org/TR/WD-SVG/`

TDTD: `tdtd` — макро Emacs для редактирования DTD.
`ftp://ftp.mulberrytech.com/pub/tdtd/`

TEIGUIDE: TEI Lite: введение в кодирование текста с целью обмена.
`http://www.uic.edu/orgs/tei/intros/teiu5.html`

TEIHOME: Домашняя страница проекта «инициатива в области кодирования текста» (TEI — Text Encoding Initiative).
`http://www.uic.edu/orgs/tei/`

TEILITE: TEI lite DTD.
`http://www-tei.uic.edu/orgs/tei/p3/dtd/teilite.dtd`

TEIXML: XML-версия TEI DTD.
`http://www.loria.fr/~bonhomme/xml.html`

TEX2HTML: Коммерческая версия программы `tth` (обладает дополнительными возможностями).
`http://www.tex2html.com`

TEX4HT: Транслятор TEX4ht.
`http://www.cis.ohio-state.edu/~gurari/TeX4ht/mn.html`

TEXLIVE: Готовый к работе дистрибутив на компакт-диске программного обеспечения для TEX'а (операционные системы UNIX и Windows 9X/NT).
`http://www.tug.org/texlive.html`

TEXML: Система для ввода XML-документов с помощью TEX'а.
`http://www.alphaWorks.ibm.com/formula/texml/`

TEXPIDER: Разработанная фирмой MicroPress версия TEX'а, непосредственно создающая HTML.
`http://www.micropress-inc.com/webb/wbstart.htm`

TIFF6: Спецификация TIFF, версия 6.
`http://www.adobe.com/supportservice/devrelations/PDFS/TN/TIFF6.pdf`

TTH: Транслятор TEX'а в HTML.
`http://hutchinson.belmont.ma.us/tth/`

TUGINDIA: Журнал Группы пользователей TEX'а в Индии (TUGIndia Journal).
`http://ftp.gwdg.de/pub/dante/usergrps/tugindia/tugindia11.pdf`

TXPL: IBM-techexplorer Hypermedia Browser — модуль расширения фирмы IBM.
`http://www.software.ibm.com/enetwork/techexplorer/`

UNICODE: Домашняя страница Консорциума Unicode.
`http://www.unicode.org`

URNIETF: Рабочая группа IETF URN.
`http://www.ietf.org/html.charters/urn-charter.html`

VISXML: Визуальный XML.
`http://www.pierlou.com/visxml/`

VML: Язык векторной разметки.
`http://www.w3.org/TR/NOTE-VML`

W2CMATH: Домашняя страница математической группы Консорциума W3C.
`http://www.w3.org/Math/`

W3C: Домашняя страница Консорциума W3C производителей программного обеспечения для Всемирной паутины.
`http://www.w3.org/`

W3CFUTURE: Планы Консорциума W3C по дальнейшему после HTML развитию языков разметки.
`http://www.w3.org/MarkUp/Activity.html`

W3CGR: Консорциум W3C: работа в области воспроизведения графики.
`http://www.w3.org/Graphics/Activity`

W3CSTYLE: Консорциум W3C. Страница, посвященная стилям.
`http://www.w3.org/Style/`

WAI: Инициатива в области доступности Сети.
`http://www.w3.org/WAI/`

WEB3D: Домашняя страница WEB3D (бывший язык моделирования виртуальной реальности — VRML).
`http://www.web3d.org/home.html`

WEBDAV: Рабочая группа Web Distributed Authoring and Versioning (IETF WEBDAV Working Group).
`http://www.ics.uci.edu/pub/ietf/webdav/`

WEBEQ: Редактор формул WebEQ.
`http://www.webeq.com`

WEBHIST: Краткая история Всемирной паутины.
`http://www.w3.org/History.html`

WIDL: Язык определения Web-интерфейсов.
`http://www.w3.org/TR/NOTE-widl`

XCATALOG: Предложение Джона Коуэна по разработке XML-каталогов (преобразование публичных идентификаторов в системные).
`http://www.ccil.org/~cowan/XML/XCatalog.html`

XDVI: Программа Пауля Войта X Windows для предварительного просмотра TEX'a.
`http://math.berkeley.edu/~vojta/xdvi.html`

XLINKSPEC: Спецификация языка компоновки XML.
`http://www.w3.org/TR/WD-xlink`

XML: Взаимный обмен метаданными XML.
`http://www.oasis-open.org/cover/xmi.html`

XML4J: XML для Java.
`http://www.alphaworks.ibm.com/formula/xml/`

XMLDATA: XML-Data, словарь схем языка XML.
`http://www.w3.org/TR/1998/NOTE-XML-data/`

XMLDEV: Список разработчиков XML.
`http://www.lists.ic.ac.uk/hypermail/xml-dev/`

XMLEDI: Система электронного бизнеса «E-business framework». Благодаря сочетанию языка XML со средствами обмена электронными данными (EDI), обеспечивает возможность обмена различными типами данных.
`http://www.geocities.com/WallStreet/Floor/5815/`

XMLERRATA: Спецификация XML 1.0. Список опечаток.
`http://www.w3.org/XML/xml-19980210-errata`

XMLFAQ: Питер Флинн. Часто задаваемые вопросы о языке XML.
`http://www.ucc.ie/xml/`

XMLINTRO: Введение в язык XML.
`http://www.oasis-open.org/cover/xmlIntro.html`

XMLNS: Пространства имен в языке XML.
`http://www.w3.org/TR/REC-xml-names/`

XMLPAGE: OASIS: страница, посвященная языку XML.
`http://www.oasis-open.org/cover/xml.html`

XMLPARS: Перечень открытого программного обеспечения для XML.
`http://www.stud.ifi.uio.no/~lmariusg/linker/XMLtools.html`

XMLRES: Ресурсы XML (дискуссионный клуб).
`http://capita.wustl.edu/XMLRes/`

XMLSPEC: Спецификация XML (версия 1).
`http://www.w3.org/TR/REC-xml`

XMLSTYLE; Стили XML.
`http://www.w3.org/TR/xml-stylesheet/`

XPATH: Язык адресации XML-документов (XPath), версия 1.0 (рекомендация Консорциума W3C).
`http://www.w3.org/TR/xpath`

XPDF: `Xpdf` — независимая программа просмотра PDF-файлов.
`http://www.aimnet.com/~derekn/xpdf/`

XPPARS: `xp` — программа Джеймса Кларка для синтаксического анализа XML.
`http://www.jclark.com/xml/xp/index.html`

XPTR: Язык указателей XML (XPointer).
`http://www.w3.org/TR/WD-xptr`

XSL97: XSL (первоначальное предложение 1997 г.).
`http://www.w3.org/TR/NOTE-XSL.html`

XSLCSS: Совместное использование XSL и CSS.
`http://www.w3.org/TR/NOTE-XSL-and-CSS`

XSLMAIL: Список рассылки XSL.
`http://www.mulberrytech.com/xsl/xsl-list/`

XSLOASIS: Web-страница SGML/XML, поддержка программного обеспечения для XSL/XSLT.
`http://www.oasis-open.org/cover/xslSoftware.html`

XSLREQ: XSL: сводка требований.
`http://www.w3.org/TR/WD-XSLReq`

XSLSPEC: Расширяемый язык стилей XSL, версия 1.0 (рабочие материалы Консорциума W3C).
`http://www.w3.org/TR/WD-xsl/`

XSLT: Преобразования XSL (XSLT), версия 1.0 (перспективная рекомендация Консорциума W3C).
`http://www.w3.org/TR/xslt`

XTPROC: `xt` Джеймса Кларка — средство преобразования XSL.
`http://www.jclark.com/xml/xt.html`

XUL: Расширяемый язык пользовательских интерфейсов.
`http://www.oasis-open.org/cover/xul.html`

YANDY: Y&Y Inc.
`http://www.yandy.com/`

YUDIT: Yudit — редактор Unicode.
`http://czyborra.com/yudit/`

Литература

Abramowitz M., Stegun I. A. 1972. *Handbook of Mathematical Functions.* New York: Dover Publications.

Bienz T., Cohn R., Meehan J. R. 1996. *Portable Document Format Reference Manual Version 1.2.* San Jose, Calif.: Adobe Systems Incorporated. Доступно в режиме онлайн на [↪PDFSPEC].

Boumphrey F. 1998. *Professional Style Sheets for HTML and XML.* Chicago: Wrox Press, Inc.

Bradley N. 1998. *The XML Companion.* Reading, Mass.: Addison Wesley Longman.

Carr L., Rahtz S., Hall W. 1991. Experiments with TEX and hyperactivity. *TUGboat*, **12** (1), 13–20.

Drakos N., Moore R. 1998. *The LaTeX2HTML Translator, User's Guide and Manual.* Accompanies the software, 1998. Доступно в режиме онлайн на [↪L2HDOC].

Flynn P. 1998. *Understanding SGML and XML Tools.* Norwell, Mass.: Kluwer Academic Publishers.

Foster K. R. 1999. Math on the Internet. *IEEE Spectrum*, **36** (4), 36–40.

Goldfarb C. F., Prescod, P. 1998. *The XML Handbook.* Englewood Cliffs, N.J.: Prentice Hall.

Goossens M., Mittelbach F., Samarin A. 1994. *The LaTeX Companion.* Reading, Mass.: Addison-Wesley. [Имеется русский перевод: Гуссенс М., Миттельбах Ф., Самарин А. Путеводитель по пакету LaTeX и его расширению LaTeX 2_ε. — М.: Мир, 1999].

Goossens M., Rahtz S., Mittelbach F. 1997. *The LaTeX Graphics Companion: Illustrating Documents with TEX and PostScript.* Tools and Techniques for Computer Typesetting. Reading, Mass.: Addison-Wesley. [Готовится русский перевод: Гуссенс М., Ратц С., Миттельбах Ф. Путеводитель по пакету LaTeX и его графическим расширениям. — М.: Мир, 2001].

Harold E. 1998. *XML Extensible Markup Language.* Foster City, Calif.: IDG Books Worldwide, Inc.

ISO:15924 1999. *Codes for the Representation of Names of Scripts.* International Organization for Standardization, Geneva, Switzerland. International Standard ISO 15924:1999.

ISO:3166 1997. *Codes for the Representation of Names of Countries and Their Subdivisions—Part 1: Country Codes.* International Organization for Standardization, Geneva, Switzerland. International Standard ISO 3166-1:1997.

ISO:639 1988. *Code for the Representation of Names of Languages.* International Organization for Standardization, Geneva, Switzerland. International Standard ISO 639:1988.

ISO:639-2 1998. *Code for the Representation of Names of Languages—Part 2: Alpha-3 Code.* International Organization for Standardization, Geneva, Switzerland. International Standard ISO 639:1998.

ISO:8879 1986. *Information Processing—Text and Office Systems—Standard Generalized Markup Language (SGML). First edition, 1986-10-15.* International Organization for Standardization, Geneva, Switzerland. International Standard ISO 8879:1986.

ISO/IEC:10179 1996. *Information Technology—Processing Languages—Document Style Semantics and Specification Language (DSSSL). First edition, 1996.* International Organization for Standardization, Geneva, Switzerland. International Standard ISO/IEC 10179:1996. Для личного использования доступна PDF-версия в режиме онлайн; см. [↪DSSSLPDF].

ISO/IEC:10646-1:1993 1993. *Information Technology—Universal Multiple-Octet Coded Character Set (UCS)—Part 1: Architecture and Basic Multilingual Plane, (with amendments).* International Organization for Standardization, Geneva, Switzerland. International Standard ISO/IEC:10646-1:1993.

ISO/IEC:11172 1993. *Information Technology—Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1,5 Mbit/s—Parts 1 to 4.* International Organization for Standardization, Geneva, Switzerland. International Standard ISO/IEC 9070:11172.

ISO/IEC:14772-1 1998. *Information Technology—Computer Graphics and Image Processing—The Virtual Reality Modeling Language—Part 1: Functional Specification and UTF-8 Encoding.* International Organization for Standardization, Geneva, Switzerland. International Standard ISO/IEC 14772-1:1998.

ISO/IEC:8839-1 1998. *Information Technology—8-Bit Single-Byte Coded Graphic Character Sets—Part 1: Latin Alphabet No. 1.* International Organization for Standardization, Geneva, Switzerland. International Standard ISO/IEC 8859-1:1998.

ISO/IEC:9070 1991. *Information Processing—SGML Support Facilities—Registration Procedures for Public Text Owner Identifiers. Second edition, 15 April 1991.* International Organization for Standardization, Geneva, Switzerland. International Standard ISO/IEC 9070:1991.

Jelliffe R. 1998. *The XML and SGML Cookbook: Recipes for Structured Information.* Englewood Cliffs, N.J.: Prentice Hall.

Knuth D. E. 1986. *The TEXbook*, volume A of *Computers and Typesetting.* Reading, Mass.: Addison-Wesley. [Имеется русский перевод: Кнут Д. Все про TEX. — Протвино: RDTEX, 1993].

Lamport L. 1994. *LATEX: A Document Preparation System: User's Guide and Reference Manual.* 2nd edition. Reading, Mass.: Addison-Wesley.

Leventhal M., Lewis D., Fuchs M. 1998. *Designing XML Internet Applications.* Englewood Cliffs, N.J.: Prentice Hall.

Lie H. W., Bos, B. 1997. *Cascading Style Sheets: Designing for the Web.* Reading, Mass.: Addison-Wesley.

Maler E., Andaloussi J. E. 1996. *Developing SGML DTDs: From Text to Model to Markup.* Englewood Cliffs, N.J.: Prentice Hall.

McGrath S. 1998. *XML by Example: Building E-Commerce Applications.* Englewood Cliffs, N.J.: Prentice Hall.

Megginson D. 1998. *Structuring XML Documents.* The Charles F. Goldfarb series on open information management. Englewood Cliffs, N.J.: Prentice Hall.

Merz T. 1998. *Web Publishing with Acrobat/PDF.* Berlin, Germany: Springer-Verlag.

Murray J., vanRyper W. 1996. *Graphics File Formats.* 2nd edition. Sebastopol, Calif.: O'Reilly & Assoc. Inc.

Raggett D., Lam J., Alexander I., Kmiec M. 1998. *Raggett on HTML 4.* Reading, Mass.: Addison-Wesley.

Rahtz S. 1995. Another look at LaTeX to SGML. *TUGboat*, **16** (3).

Smith N. E. 1998. *SGML/XML Filters.* Plano, Tex.: Wordware Publishing, Inc.

St. Laurent S. 1997. *XML: A Primer.* Portland, Ore.: MIS Press.

Tanenbaum A. S. 1996. *Computer Networks.* 3rd edition. Englewood Cliffs, N.J.: Prentice Hall.

Unicode Consortium 1996. *The Unicode Standard, Version 2.0.* Reading, Mass.: Addison-Wesley.

Wall L. Christiansen T., Schwartz R. L. 1996. *Programming Perl.* 2nd edition. Sebastopol, Calif.: O'Reilly & Assoc. Inc.

Именной указатель

Предметный указатель

На страницах, выделенных полужирным шрифтом, имеется важная информация, например точное определение команды или подробное объяснение; светлым шрифтом набраны номера страниц, соответствующие обычным ссылкам.

Все команды

Список иллюстраций

Список таблиц

Оглавление

Справочное издание

Мишель Гуссенс, Себастиан Ратц
Путеводитель по пакету LaTeX и его Web-приложениям

Зав. ред. И. А. Маховая
Художник И. И. Куликова
Художественный редактор Н. В. Зотова
Технический редактор О. Г. Лапко
Корректор Э. В. Назидзе

Оригинал-макет подготовлен в пакете LaTeX 2_ε
с использованием семейства шрифтов Computer Modern
с кириллическим расширением LH

Лицензия ЛР № 010174 от 20.05.97 г.

Подписано к печати 14.09.2001 г. Формат 70 × 100/16.
Печать офсетная.
Объем 19,00 бум. л. Усл.-печ. л. 49,40.
Уч.-изд. л. 46,91. Изд. № 20/9741.
Тираж 3000 экз. Заказ 4603

Издательство «Мир»
Министерства РФ по делам печати,
телерадиовещания и средств массовых коммуникаций
107996, ГСП-6, Москва, 1-й Рижский пер., 2.

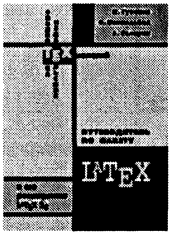
Диапозитивы изготовлены в издательстве «Мир»

Отпечатано с готовых диапозитивов в
ППП «Типография Наука»
121099, Москва, Шубинский пер., 6.

ЛУЧШИЕ КНИГИ МИРА — В ИЗДАТЕЛЬСТВЕ «МИР»

Имеются в продаже:

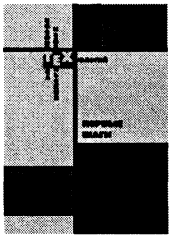
Серия: Библиотека издательских TEXнологий



Гуссенс М., Миттельбах Ф., Самарин А. **ПУТЕВОДИТЕЛЬ ПО ПАКЕТУ LATEX И ЕГО РАСШИРЕНИЮ LATEX2ε**: Пер. с англ. — 606 с., ил.

Исчерпывающее справочное руководство по пакету LATEX2ε, широко применяемому при подготовке изданий как по точным, так и по гуманитарным наукам и принятому в качестве международного стандарта во всех ведущих научно-технических издательствах. В книге в полном объеме собран мощный инструментарий LATEX'а: NFSS2, AMS-LATEX, epic, eepic, MakeIndex и BIBTEX, а также описывается язык PostScript и пакет многоязыковой поддержки Babel.

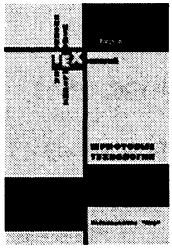
Книга предназначена для тех, кто хочет знать, как расширить возможности LATEX'а, чтобы уметь красиво оформлять издания: для профессиональных полиграфистов, авторов, разработчиков и программистов.



Грэтцер Г. **ПЕРВЫЕ ШАГИ В LATEX'е:** Пер. с англ. — 172 с., ил.

Компактное пособие по работе в LATEX2ε для новичка, написанное канадским математиком Г. Грэтцером, который известен отечественному читателю по книге «Общая теория решеток» (М.: Мир, 1981). Основной упор делается на описание AMS-LATEX'а для набора математических формул. На конкретных образцах журнальных статей объясняются основные команды и окружения пакетов LATEX и AMS, приводится много полезных адресов в Internet'е, даются подробные указатели. Изложение построено так, что читатель в короткий срок успевает не только ознакомиться с предметом, но и приобрести первые навыки.

Книга рассчитана на ученого — физика, химика, математика, еще не умеющего работать в пакете LATEX2ε, но желающего быстро его освоить, чтобы готовить свои работы к публикации. Много полезного найдет в ней и профессиональный наборщик, впервые приступающий к работе в LATEX'е.



Каров П. **ШРИФТОВЫЕ ТЕХНОЛОГИИ. ОПИСАНИЕ И ИНСТРУМЕНТАРИЙ:** Пер. с англ. — 454 с., ил.

Книга написана немецким специалистом с мировым именем. Освещаются такие темы, как история и технология производства шрифтов, шрифтовой рынок, методы определения качества шрифта, удобочитаемость, классификация шрифтовой формы и проблемы защиты авторского права. Издание дополнено материалом, отражающим историю кириллического шрифта, написанным В. В. Ефимовым.

Представляет интерес для полиграфистов, дизайнеров-шрифтовиков, программистов, связанных с разработкой и использованием цифровых шрифтов, а также студентов и аспирантов соответствующих специальностей.

Адрес: 107996, ГСП-6, Москва, 1-й Рижский пер., 2
Контактные телефоны: (095)286-83-88, 286-82-33, 286-25-50
Факс: (095)286-84-55 ✧ **E-mail:** victor@mir.msk.su ✧ **http://www.mir-pubs.dol.ru**