А. В. СТОЛЯРОВ

СВЕРСТАЙ ДИПЛОМ КРАСИВО: LaTeX ЗА ТРИ ДНЯ

$\forall$	\forall	$\exists$	\exists	∞	\infty	$\mapsto$	\mapsto
$\emptyset$	\emptyset	∇	\nabla	$\in$	\in	$\ni$	\ni
$\sim$	\sim	$\subset$	\subset	$\supset$	\supset	$\setminus$	\setminus
$\subseteq$	\subseteq	$\supseteq$	\supseteq	$\equiv$	\equiv	$\leq$	\leq
$\geq$	\geq	$\neq$	\neq	$\cdot$	\cdot	$\approx$	\approx
$\cong$	\cong	$\ll$	\ll	$\gg$	\gg	$\mid$	\mid
$\vee$	\vee	$\wedge$	\wedge	$\triangle$	\triangle	$\perp$	\perp
$\parallel$	\parallel	$\angle$	\angle	$\lfloor$	\lfloor	$\rfloor$	\rfloor
∂	\partial	$\lceil$	\lceil	$\rceil$	\rceil	$\neg$	\neg
$\pm$	\pm	$\mp$	\mp	$\times$	\times	$\cap$	\cap
$\cup$	\cup	$\div$	\div	$\oplus$	\oplus	$\ominus$	\ominus
$\otimes$	\otimes	$\oslash$	\oslash	$\odot$	\odot	$\circ$	\circ

ПУБЛИЧНАЯ ЛИЦЕНЗИЯ

Пособие Андрея Викторовича Столярова ≪Сверстай диплом красиво: LaTeX за три дня≫, впервые опубликованное в издательстве МАКС Пресс в 2010 году, называемое далее ≪Произведением≫, защищено действующим авторско-правовым законодательством. Все права на Произведение, предусмотренные действующим законодательством, как имущественные, так и неимущественные, принадлежат его автору.

Настоящая Лицензия устанавливает способы использования электронной версии Произведения, право на которые предоставлено автором и правообладателем неограниченному кругу лиц, при условии безоговорочного принятия этими лицами всех условий данной Лицензии. Любое использование Произведения, не соответствующее условиям данной Лицензии, а равно и использование Произведения лицами, не согласными с условиями Лицензии, возможно только при наличии письменного разрешения автора и правообладателя, а при отсутствии такого разрешения является противозаконным и преследуется в рамках гражданского, административного и уголовного права.

Автор и правообладатель настоящим **разрешает** следующие виды использования данного файла, являющегося электронным представлением Произведения, без уведомления правообладателя и без выплаты авторского вознаграждения:

1. Воспроизведение Произведения (полностью или частично) на бумаге путем распечатки с помощью принтера в одном экземпляре для удовлетворения личных бытовых или учебных потребностей, без права передачи воспроизведенного экземпляра другим лицам;

2. Копирование и распространение данного файла в электронном виде, в том числе путем записи на физические носители и путем передачи по компьютерным сетям, с соблюдением следующих условий: (1) **все воспроизведенные и передаваемые любым лицам экземпляры файла являются точными копиями исходного файла** в формате PDF, при копировании не производится никаких изъятий, сокращений, дополнений, искажений и любых других изменений, включая и изменение формата представления файла; (2) **распространение и передача копий другим лицам производится исключительно *бесплатно*, то есть при передаче не взимается никакое вознаграждение ни в какой форме**, в том числе в форме просмотре рекламы, в форме платы за носитель или за сам акт копирования и передачи, даже если такая плата оказывается значительно меньше фактической стоимости или себестоимости носителя, акта копирования и т. п.

Любые другие способы распространения данного файла при отсутствии письменного разрешения автора запрещены. В частности, **запрещается**: внесение каких-либо изменений в данный файл, создание и распространение искаженных экземпляров, в том числе экземпляров, содержащих какую-либо часть произведения; распространение данного файла в Сети Интернет через веб-сайты, оказывающие платные услуги, через сайты коммерческих компаний, а также **через сайты, содержащие рекламу любого рода**; продажа и обмен физических носителей, содержащих данный файл, даже если вознаграждение значительно меньше себестоимости носителя; включение данного файла в состав каких-либо информационных и иных продуктов; распространение данного файла в составе какой-либо платной услуги или в дополнение к такой услуге. С другой стороны, **разрешается** дарение (бесплатная передача) носителей, содержащих данный файл, запись данного файла на носители, принадлежащие другим пользователям, распространение данного файла через бесплатные файлообменные сети и т. п. Ссылки на экземпляр файла, расположенный на официальном сайте автора, разрешены без ограничений.

А. В. Столяров *запрещает* Российскому авторскому обществу и любым другим организациям производить любого рода лицензирование любых его произведений и осуществлять в интересах автора какую бы то ни было иную связанную с авторскими правами деятельность без его письменного разрешения.

А. В. СТОЛЯРОВ

СВЕРСТАЙ ДИПЛОМ КРАСИВО: LaTeX ЗА ТРИ ДНЯ

Москва
2010

УДК 681.322
ББК 32.973-018.2
С81

Столяров А. В.
С81 **Сверстай диплом красиво: LaTeX за три дня.** — М.: МАКС Пресс, 2010. — 100 с.
ISBN 978-5-317-03440-5

Краткое пособие по ускоренному практическому освоению системы компьютерной верстки LaTeX, ориентированное на оформление курсовых и дипломных работ. Содержит ответы на наиболее типичные вопросы, возникающие у начинающих пользователей LaTeX, и возможные способы решения часто встречающихся проблем. Для студентов математических, естественнонаучных и технических специальностей, преподавателей и всех желающих освоить систему LaTeX.

УДК 681.322
ББК 32.973-018.2

Корректор Е. Ясиницкая

ISBN 978-5-317-03440-5

От автора

Книжка, которую Вы держите в руках, представляет собой пособие по очень быстрому освоению системы компьютерной верстки LaTeX[1]. По этой теме написаны и изданы десятки, если не сотни книг; к сожалению, несмотря на все это изобилие, люди продолжают при верстке сложных документов (к которым, без сомнения, относятся и курсовые и дипломные работы) пользоваться инструментами, заведомо не предназначенными и непригодными для таких целей.

Существует весьма, увы, популярный миф, утверждающий, что некоторые широко известные коммерческие программные инструменты, построенные по принципу WYSIWYG[2], якобы являются *дружественными пользователю и интуитивно понятными*[3], тогда как LaTeX то ли чрезмерно сложен, то ли годится только математикам, то ли требует программистского образования, то ли все это вместе.

В действительности все обстоит несколько иначе. С одной стороны, LaTeX, вопреки всем расхожим мифам, оказывается неожиданно прост в освоении. Отметим, что **три дня, упомянутые в названии данного пособия, вовсе не являются преувеличением или фигурой речи: автор в свое время с нуля и полностью самостоятельно научился верстать с помощью LaTeX'а до уровня, позволившего оформить научную работу для международной конференции, и заняло это как раз три дня, притом неполных** (больше просто не было времени, поджимали сроки). Разумеется, навыки, полученные за эти три дня, не позволяли использовать и сотой доли возможностей системы, но для оформления статьи этого оказалось достаточно.

С другой стороны, существует очень и очень немного людей, реально умеющих пользоваться пресловутыми средствами на основе WYSIWYG, во всяком случае, если речь идет о серьезной верстке; в большинстве случаев работа, набранная в WYSIWYG-редакторе, отличается от сверстанной в LaTeX'е даже с самого первого беглого взгляда — по общей неряшливости, по постоянно сползающим размерам отступов и полосы текста, по наличию совершенно неожиданных фрагментов с рваным правым краем, по отсутствию нумерации рисунков, таблиц, а иногда и заголовков глав и параграфов; наконец, просто по совершенно случайным образом «гуляющим» размерам и начертанию шрифтов. Умение красиво и аккуратно верстать с помощью WYSIWYG-редакторов — это высокая квалификация, за которую многие люди получают серьезные деньги. Не

[1] Название произносится как «латéх», с ударением на второй слог и звуком «ха» (а не «кс», как могут предположить читатели, знающие английский); Дональд Кнут, придумавший изначальную систему TeX, объясняет такое произношение тем, что буква «X» в названии — это греческая хи, а вовсе не латинский икс.

[2] «What you see is what you get», или «Что видите — то и получите».

[3] Не буду называть здесь эти «продукты», все и так знают, о чем идет речь.

может быть и речи о том, чтобы навыки такой верстки сами собой появлялись при использовании «интуитивно понятного интерфейса»: эти навыки — всегда результат долгого и кропотливого обучения (обычно на своих ошибках), и на получение такой квалификации у пользователя может уйти несколько лет. В отличие от WYSIWYG-редакторов, LaTeX изначально ориентирован на автоматическое соблюдение типографских норм и создание эстетично выглядящих документов. Верстка с его помощью отнимает иногда в несколько раз меньше времени и нервов, чем та же работа, выполненная в WYSIWYG-редакторах, при этом давая существенно лучшие (в плане аккуратности, читаемости, соответствия издательским традициям и т. п.) результаты. Как правило, документ, созданный с помощью LaTeX'а, просто приятно взять в руки.

К сожалению, большинство доступных пособий по LaTeX'у одной своей толщиной невольно поддерживает миф о недоступности и кошмарной сложности этого на самом деле простого и эффективного инструмента. Странно, что при этом толщина пособий по все тому же неназываемому коммерческому программному продукту обычно в расчет не принимается. Между тем, несмотря на безусловную полезность всевозможных сведений о системе LaTeX, практически любая книга по нему содержит в десятки, если не в сотни раз больше информации, чем необходимо, чтобы просто начать и срочно сверстать, скажем, ту же квалификационную (дипломную или курсовую) работу. То, что нужно пользователю здесь и сейчас, приходится выискивать в толстых томах по крупицам, продираясь через дебри незнакомых понятий. Пособие, которое у вас в руках, призвано исправить эту, прямо скажем, странную ситуацию.

Отмечу, что в большинстве случаев вам не потребуется полностью изучать даже содержимое этой сравнительно короткой книжки, чтобы сверстать свою работу. Разумеется, никто не мешает затем повышать свою квалификацию, узнавая все новые и новые возможности LaTeX'а, но *для начала* будет достаточно набора сведений, которые вполне можно уложить в своей голове за один вечер. Не верите? Попробуйте! В конце концов, мне это когда-то удалось, причем конспективного пособия у меня не было.

Считаю своим приятным долгом поблагодарить Екатерину Ясиницкую, чья помощь в подготовке этого пособия к публикации вышла далеко за рамки обычной корректуры.

Введение

О чем написано в этой книжке и как ею пользоваться

Основной принцип, в соответствии с которым написано данное пособие, состоит в соблюдении сугубо прагматичного стиля изложения (как сделать то и как сделать это) в сочетании с разумным ограничением рассматриваемых возможностей. Разумеется, в небольшой книжке невозможно (и не нужно!) пытаться рассказать обо всех существующих возможностях LaTeX'а и даже только о возможностях, известных автору: пособий, претендующих на полноту, хватает и так, а вот конспективные введения автору пока не попадались.

В связи с этим, например, в пособии рассматривается только один класс документов, `extreport`, как наиболее удобный для верстки курсовых и дипломных работ[4], и все команды секционирования, опции и т. п. даны в расчете на этот класс. Кроме того, в тексте пособия предполагается, что ваш документ пишется на русском языке со всеми вытекающими последствиями в виде специфических настроек[5].

Многие книги по LaTeX'у построены в соответствии с логикой технической документации: сначала излагаются основы функционирования TeX'а с использованием таких терминов, как «боксы», «горизонтальная и вертикальная мода», «клей» и т. п., а остальной материал подается с активным использованием этих терминов в предположении, что читатель уже понял всю низкоуровневую «кухню». В то же время автор этих строк на основе собственного опыта готов ответственно заявить, что вникать в эти тонкости пользователю совершенно ни к чему, особенно на начальном этапе освоения LaTeX'а: если необходимо в сжатые

[4] Отметим, что само пособие набрано с помощью класса `extbook`, а еще существуют классы `letter, slides, article, book, report, extarticle` и т. д. При желании читатель может освоить их самостоятельно.

[5] Для заинтересованных читателей отметим, что это означает использование пакета `babel`; подробности можно узнать, обратившись к соответствующей документации.

сроки сверстать текст работы, пользователя интересует ответ на вопрос «как этим воспользоваться», а не «как это работает», и LaTeX вполне позволяет такое «потребительское» отношение к себе. Возможно, позже пользователю понадобится более глубокое понимание происходящего, но когда сроки поджимают, изучение «кухни» можно оставить на потом.

В пособии не рассматриваются многочисленные макропакеты, позволяющие добиваться интересных спецэффектов и предоставляющие разнообразные полезные возможности. Мы ограничиваемся абсолютным минимумом, без которого ваша курсовая работа не будет соответствовать требованиям на оформление таких текстов; вместе с тем в пособие по возможности включены ответы на наиболее часто возникающие у студентов (а ранее возникавшие у автора) вопросы, поиск ответа на которые зачастую съедает очень много времени.

Даже этот минимум вам может потребоваться не весь. Например, если ваша работа посвящена программированию, вам вряд ли потребуется бóльшая часть главы 10, в которой рассматриваются средства для набора математических формул. **Рекомендуется в любом случае сначала прочитать главы 1–3**; остальные главы можно читать в любой последовательности.

Наиболее естественно LaTeX выглядит в операционных средах с развитым интерфейсом командной строки, таких, как многочисленные системы семейства Unix (включая, разумеется, и GNU/Linux). Поэтому изложение в основном ведется в расчете на пользователей Unix; тем не менее, некоторые полезные комментарии даются специально для пользователей Windows, поскольку большинство конечных пользователей пока к переходу на GNU/Linux не готовы[6]. Примерно 98% текста данного пособия актуально для всех пользователей LaTeX вне зависимости от того, к какой они привыкли операционной системе, поскольку входной язык LaTeX, которому и посвящена бóльшая часть пособия, никак от операционной системы не зависит.

Все же отметим, что в большинство дистрибутивов GNU/Linux входит набор пакетов, содержащих сам LaTeX и все необходимое для работы с ним, тогда как под Windows придется, скорее всего, доставать и устанавливать соответствующее программное обеспечение. Впрочем, по наблюдениям автора, большинство пользователей Windows это не останавливает.

[6] Автор не может удержаться от комментария по этому поводу: на самом деле создавшееся положение обусловлено не мифической «сложностью» GNU/Linux и других Unix-подобных операционных систем, но исключительно **вредными привычками**, возникшими у большинства пользователей под давлением пропаганды со стороны соответствующих корпораций; впрочем, читателю, безусловно, виднее, а использовать LaTeX под Windows ничуть не сложнее, чем под GNU/Linux и другими Unix-системами.

В данной книге фрагменты исходных текстов (то есть фрагменты того текста, который вы должны набрать в редакторе, чтобы получить нужный результат) набраны `моноширинным шрифтом`. На некоторых важных моментах внимание читателя акцентируется **использованием ожирнения**. Наконец, некоторые примечания набраны <small>уменьшенным шрифтом без засечек</small>; эти фрагменты можно при желании пропустить.

Что такое LaTeX и как с ним работать

Прежде всего необходимо уяснить, чем LaTeX **не является**. Как можно было догадаться из текста предисловия, LaTeX не является WYSIWYG-редактором текстов; добавим, что LaTeX **вообще не является редактором текстов, текстовым процессором и т. п.**, как не является он и законченной системой верстки. Сам по себе LaTeX — это не более чем интерпретатор специального формального языка, формирующий на выходе файл определенного формата, который можно просматривать и печатать. Сразу хотелось бы попросить читателей, не имеющих отношения к программированию, не паниковать при виде слова «интерпретатор»: вышеприведенная фраза очень легко может быть переведена на понятный язык без использования специальной терминологии. В двух словах, LaTeX — это такая программа, для которой пользователь с помощью обычного текстового редактора подготавливает в виде простого текста некий набор указаний, как должен выглядеть итоговый документ, а LaTeX с использованием этих указаний уже создает сам документ.

Входной язык LaTeX довольно развит, но, опять же, не следует думать, что для создания документов с его помощью необходимы программистские навыки. Входной язык LaTeX — это *язык разметки*. Это значит, что текст на входном языке содержит, во-первых, собственно текст документа, и, во-вторых, еще специальные пометки, указывающие, как те или иные фрагменты текста документа должны быть оформлены. Например, если мы наберем фразу

```
У попа была собака
```

то именно такая фраза и будет воспроизведена при печати итогового документа, причем будет использован обычный (стандартный, не наклонный, без ожирнения и т. п.) шрифт, тогда как если мы наберем

```
У попа {\it была} {\bf собака}
```

то напечатана будет та же фраза, но слово «была» будет набрано «италиком» (курсивом), а слово «собака» — ожирненным шрифтом. Все вместе будет выглядеть примерно так:

У попа *была* **собака**

Возможно, читателю приходилось создавать страницы www-сайтов на языке HTML; этот язык также представляет собой язык разметки. Основные возможности LaTeX'а оказываются и проще, и не столь трудоемки в наборе, так что, если у вас есть опыт работы с HTML, то с LaTeX'ом у вас все и подавно получится; впрочем, у вас все получится в любом случае: подчеркнем еще раз, что LaTeX в освоении гораздо проще «привычных» инструментов подготовки текста.

Итак, общая схема работы с LaTeX'ом выглядит примерно следующим образом. Пользователь набирает свой документ (*исходный текст документа*) в любом удобном ему текстовом редакторе. Этот текст затем подается на вход программе `latex`, а результатом работы этой программы оказывается файл с расширением `.dvi`, содержащий готовый документ, пригодный к просмотру и распечатке. Просмотр документа на экране осуществляется еще одной программой (в ОС Unix она называется `xdvi`); затем документ для распечатки переводят в более удобный для принтера формат (обычно это PostScript, но бывают и другие варианты) и распечатывают.

Очень важно, чтобы ваш редактор текстов позволял сохранять файлы в формате «просто текст» (ASCII text); разумеется, если редактор имеет встроенные средства ввода сложного текста, такие, как переключение шрифтов, печать жирным или наклонным шрифтом и т. п., эти возможности ни в коем случае не следует использовать: если, например, вам надо набрать слово «жирный» жирным шрифтом, вместо того, чтобы тянуться мышиным курсором к нарисованной кнопке с буквой Ж, следует набрать (обычным шрифтом!) примерно такой текст: `\textbf{жирный}`. Помните, что обычный текст (ASCII text) не имеет средств переключения шрифтов и прочего «сложного» форматирования, так что при сохранении в формате «просто текст» результаты ваших нажатий на всякие хитрые кнопки будут потеряны, а ничего кроме «просто текста» LaTeX воспринимать не способен.

Пользователи Unix для набора текстов могут использовать любой традиционный редактор текстов, такой, как `vim`, `emacs`, `joe`, встроенный редактор из Midnight Commander и т. п. (но ни в коем случае не OpenOffice.org). Пользователям Windows можно порекомендовать редактор «блокнот» (NotePad). Возможно также, что программное обеспечение, которое вам дадут вместе LaTeX'ом, будет содержать и редактор текстов, специально предназначенный для работы с ним.

После того, как исходный текст документа в более-менее законченном виде (то есть содержащий все минимально необходимое для его восприятия LaTeX'ом) набран, следует сохранить текст в файле с расширением `.tex` или `.ltx`, после чего запустить LaTeX (то есть программу с именем `latex` или `latex.exe`), указав ему параметром командной строки имя вашего исходного файла (то есть файла, содержащего набранный текст).

Если ваш исходный текст не содержит ошибок, в том же каталоге[7] появится файл, имеющий то же самое имя, что и исходный файл, но с расширением `.dvi`. Это и есть полученный документ, который можно просмотреть на экране. Пользователи Unix для просмотра могут воспользоваться командой `xdvi`; под Windows, скорее всего, в среде, в которой вы будете работать, найдется соответствующая кнопка или пункт меню, которые сами запустят нужную программу.

Учтите, что для корректного построения перекрестных ссылок внутри вашего документа вам, возможно, придется запустить LaTeX дважды.

Полученный файл в формате `.dvi` обычно следует перевести в формат PostScript, чтобы его можно было напечатать. Пользователям Unix можно порекомендовать программу `dvips`; что касается Windows, то ваше программное обеспечение, скорее всего, будет иметь встроенную функцию печати, которая все необходимые переводы произведет самостоятельно.

Если ваш документ предполагается передавать кому-то в электронном виде, для этого может оказаться удобнее формат Adobe PDF. Чтобы получить документ в этом формате, можно воспользоваться командой `pdflatex` вместо просто `latex`.

Просмотрев документ, вы можете пожелать что-то в нем изменить; это делается редактированием исходного текста (и никак иначе!), после которого понадобится снова запустить LaTeX и программу просмотра.

[7] В среде Windows сейчас в ходу термин ≪папка≫; этот термин некорректен и нарушает традиции компьютерной терминологии. В данном пособии используется традиционный термин ≪каталог≫.

Глава 1

Главное — начать

Известное изречение, что любая дорога начинается с первого шага, особенно важно принимать во внимание при освоении новых рабочих инструментов. В этой главе приводится минимум начальных сведений, которые позволят вам набрать с помощью LaTeX'а простой пример, оттранслировать его и увидеть результат. Обязательно сделайте это как можно раньше. Момент, когда на экране появится сверстанный текст, можно считать переломным: после этого никакие трудности вас уже не остановят, а процесс изучения LaTeX'а станет существенно комфортнее.

Если предыдущий абзац вызвал у вас скептическую усмешку или любую другую реакцию, отличную от намерения включить компьютер и попробовать приведенный ниже пример, специально для вас подчеркнем, что предложение высказано совершенно серьезно. Если угодно, считайте это личным одолжением автору книжки — только, пожалуйста, все-таки включите компьютер и попробуйте.

Чтобы эксперимент удался, совершенно не обязательно вчитываться в текст всех параграфов главы. Можно просто просмотреть ее по диагонали, а затем по мере необходимости возвращаться и прорабатывать отдельные фрагменты более внимательно. Если по каким-то причинам вы до сих пор не прочитали введение, особенно его вторую часть (см. стр. 7), то текст этой главы может оказаться совершенно непонятен (вплоть до вопроса «к чему это все»). Проще всего вернуться назад и прочесть введение, благо оно достаточно короткое.

1.1. Что надо знать с самого начала

Исходный файл для LaTeX'а состоит из *преамбулы* и *основного текста*. Преамбула, располагающаяся в начале файла, задает класс документа, используемые пакеты и прочие технические параметры; ниже приводится пример преамбулы, которая точно заработает и в большинстве

случаев вас устроит, так что если вас беспокоят перспективы изучения чего-то «сложного», о преамбуле можете не волноваться.

После преамбулы идет строка `\begin{document}`, с которой и начинается основной текст. В самом конце файла необходимо поместить строку `\end{document}`.

1.1.1. Символы обычные и специальные

Большинство символов в основном тексте обозначают сами себя, так что вы вполне можете просто набирать ваш текст между строками `\begin{document}` и `\end{document}`; если вам повезет и текст не будет содержать ни одного из символов, имеющих специальное значение, то результат будет более-менее похож на введенный вами текст.

Некоторые символы, однако, воспринимаются LaTeX'ом специфически. Так, с символа «\» (обратная косая черта) начинаются *команды* LaTeX'а, указывающие на необходимость чего-то более сложного, чем просто вставка еще одной буквы. После символа обратной косой черты должно идти *имя команды*, которое может состоять из одного символа, не являющегося буквой или цифрой, либо из нескольких латинских букв. С командами из букв мы уже встречались в примере про попа и собаку на стр. 7: это команды `\bf` и `\it`; примером команды с небуквенным именем может служить `\-`, позволяющая разделить слово для переноса.

Символ «%» обозначает начало комментария[1], то есть все, что в вашем исходном тексте находится после символа «%» и до конца строки, будет попросту проигнорировано; так, если вы наберете

```
имеется 78% от необходимой суммы
```

то при просмотре результата вы обнаружите, что напечатано только «имеется 78», и все. Чтобы символ процента появился в итоговом тексте, необходимо набрать двухсимвольную комбинацию «\%»:

```
имеется 78\% от необходимой суммы
```

Полный список специальных символов таков:

```
\   {   }   $   &   #   ^   _   %   ~   "
```

Если в тексте потребуются символы «{», «}», «$», «_», «&» и «#», набрать их можно аналогично символу «%», поставив перед ними обратную косую черту: «\{», «\}», «\$», «_», «\&» и «\#». Символ «\» набирается сложнее: чтобы он появился в тексте, необходимо воспользоваться командой `\textbackslash`. Аналогично можно применить

[1] Под комментарием понимается такой фрагмент исходного текста, который предназначен для людей, работающих с этим исходным текстом, и должен игнорироваться программой-интерпретатором (в данном случае программой `latex`).

команду `\textasciitilde` для набора символа «~», команду `\dq` — для символа «"», команду `\textasciicircum` — для символа «^»,

Особо остановимся на символе «"» (двойные кавычки). Надо сказать, что этот символ имеет специальное значение исключительно благодаря пакету `babel`, то есть является одной из упоминавшихся выше особенностей верстки русскоязычных документов, и используется в основном для генерации традиционных русских типографских кавычек — «елочек» и «лапок», причем комбинация «"<» обозначает левую «елочку», комбинация «">» — правую, а «лапки» набираются с помощью комбинаций «"`» и «"'» (сначала символ кавычки, потом обратный апостроф для открывающей «лапки», прямой апостроф — для закрывающей). Таким образом, чтобы напечатать

«Прочитай свежие „Ведомости“» — сказал папа

надо ввести

```
"<Прочитай свежие "`Ведомости"'">~--- сказал папа
```

При необходимости можно использовать и обычные двойные кавычки; для их набора используются два обратных апострофа (открывающая кавычка) и два прямых апострофа (закрывающая кавычка):

```
прочитай свежие ``Ведомости''
```

даст на выходе

прочитай свежие “Ведомости”

Приведем еще несколько полезных комбинаций. Знак параграфа «§» можно получить, набрав «`\S`»; знак номера «№» набирается как комбинация «`\No`»; чтобы поставить над буквой знак ударения, используйте комбинацию «`\'`» (например, чтобы напечатать «бóльший», нужно набрать «`б\'ольший`»); для набора других часто встречающихся диакритических знаков можно использовать «`\"`», «`\~`», «`\^`», например, исходная строка «`\"a\~a\^a`» превратится в «ä$\tilde{a}$â».

Знак тильды «~», встреченный в тексте, воспринимается LaTeX'ом как «неразрываемый пробел», т. е. такой пробельный символ, который недопустимо растягивать, сжимать и разрывать концом строки. Часто используется также комбинация «`\,`», обозначающая неразрываемый пробел уменьшенного размера.

Коль скоро речь зашла о символах, отметим еще один важный момент. В типографском тексте обычно символ дефиса (то есть черточки внутри слов) весьма существенно отличается от символа тире (то есть знака препинания). LaTeX позволяет включать в документ целых четыре вида «черточек», а именно дефис, встречающийся в словах типа «кусочно-непрерывный», короткое тире (его рекомендуется использовать в значении «от–до», например, если нужно набрать что-то вроде

«дорога отнимает тридцать–сорок минут»), длинное тире — знак препинания, который ставят, например, перед союзом «это», и, наконец, математический символ минуса, используемый в формулах (что-то вроде $25 - 16 = 9$). Набирать эти символы очень просто: один символ «-» воспринимается как дефис, два символа подряд — как короткое тире, три подряд — как длинное тире. Например, для набора фразы

> в кофе можно добавить десять–пятнадцать мл. виски и сливки — это называется «кофе по-ирландски»

необходимо ввести примерно такой текст:

```
в кофе можно добавить десять--пятнадцать мл. виски
и сливки~--- это называется "<кофе по-ирландски">
```

Обратите внимание на знак тильды перед длинным тире. Использование этого знака вместо обычного пробела гарантирует, что тире не будет перенесено первым символом на следующую строку.

О знаке минуса мы узна́ем из главы, посвященной формулам.

1.1.2. Еще о командах

Некоторые команды LaTeX'а требуют указания *параметров*. Обычно параметр заключают в фигурные скобки, например:

```
\subsection{Еще о командах}
```

Здесь `\subsection` — это команда, начинающая очередной подпараграф («подсекцию») текста, а «`Еще о командах`» — ее параметр. Существуют команды, требующие двух, трех и более параметров, причем иногда некоторые параметры являются необязательными (такие параметры заключаются в квадратные скобки вместо фигурных). В некоторых случаях (например, если параметр состоит из одного символа) без фигурных скобок можно обойтись. Именно так мы в предыдущем параграфе поступали с командами для отображения диакритических знаков: мы писали «`\"a\~a\^a`», хотя можно было бы написать и «`\"{a}\~{a}\^{a}`».

Обычно фигурные скобки используются, чтобы объединить несколько символов в единое целое, называемое *блоком*, хотя иногда приходится использовать их и для обрамления параметра, состоящего из одного символа. Так, например, команды для некоторых диакритических знаков имеют буквенные имена, и мы вынуждены заключать их аргументы в скобки, чтобы LaTeX не воспринял их как часть имени команды: `\u{a}\r{a}\d{a}\v{a}\b{a}\c{a}\H{a}` (напечатано будет «ăḁ̊ǎa̲a̧a̋»).

При использовании команд, имена которых состоят из букв, LaTeX воспринимает в качестве имени команды все буквы, стоящие после обратной косой черты до первого небуквенного символа. Если команда имеет параметры и мы заключаем их в фигурные скобки, то все выглядит

вполне естественно, как в примерах выше. Иное дело, если аргументов у команды не предусмотрено. Если сразу после имени команды будет стоять небуквенный символ, то никаких проблем не возникнет, но вот от букв имя команды необходимо отделить. Обычно это делают, вставляя символ пробела: например, `\bf bold` переключит шрифт на ожирненный и напечатает слово «**bold**». При этом сам символ пробела не печатается. Более того, LaTeX в такой ситуации игнорирует любое количество идущих подряд пробелов, что не всегда соответствует нашим потребностям. Пусть, например, нам нужно набрать фразу «палата № 6». Первое, что приходит в голову — это ввести примерно такой код: `палата \No 6`. Результат получится не совсем такой, как мы хотели, потому что `\No` — это команда с буквенным именем, так что пробел после нее напечатан не будет, символ номера «склеится» с цифрой 6 и итог будет смотреться несколько неказисто, примерно так: «палата №6».

В такой ситуации можно поступить двумя способами. Первый — это применить специальную команду «защищенный пробел», состоящую из обратной косой черты и пробела. В отличие от простого пробела, такой пробел LaTeX не проигнорирует: код `палата \No\ 6` напечатает именно то, что нам нужно. Второй способ состоит в использовании «пустого блока»: `палата \No{} 6` тоже напечатает то, что мы хотели.

Кроме команд, в языке разметки LaTeX часто используются *окружения* — фрагменты текста, заключенные между командами `\begin{}` и `\end{}`; обе команды в качестве параметра принимают *имя окружения*. Окружение позволяет распространить то или иное свойство на весь заключенный в окружение текст. Например, окружение `large` позволяет напечатать фрагмент текста увеличенным шрифтом; для этого фрагмент заключают между командами `\begin{large}` и `\end{large}`.

1.1.3. Пробелы, строки и абзацы

Любое количество идущих подряд *пробельных символов*, то есть пробелов, табуляций и переводов строки, LaTeX воспринимает как один пробел; таким образом, даже если между какими-то двумя словами вы вставите сто пробелов вместо одного, на результате это никак не скажется. Размер пробела между словами LaTeX выбирает сам, причем так, чтобы все пробелы в одной строке были одинаковы. Не отразится на результате и вставка в исходный текст переводов строки: даже если вы введете каждое слово на отдельной строке, результат все равно будет отформатирован в обычный абзац.

Из этого правила есть одно важное исключение: пустая строка (то есть два подряд символа перевода строки) обозначает *разделитель абзацев*, то есть **чтобы разбить текст на абзацы, достаточно оставить между ними пустые строки**.

Абзац форматируется в соответствии с размером отступа и полосы набора, заданных стилем и преамбулой документа, что полностью исключает ошибки типа «случайно сдвинулся ползунок», «нечаянно нажалась кнопочка» и тому подобных. Разумеется, можно заставить LaTeX сделать абзац любой формы, размера, с любыми отступами и прочими свойствами, есть даже средства для формирования абзацев в форме сердечка, кружочка и т. п., однако задействовать все эти возможности можно только преднамеренно, точно зная при этом, чего мы хотим добиться.

На форматирование абзаца можно повлиять множеством различных способов; как обычно, мы не станем пытаться перечислить их все, а ограничимся наиболее употребительными. Команда `\noindent`, вставленная непосредственно перед абзацем, отменяет отступ (красную строку) для этого абзаца. Комбинация из двух символов «\\» принудительно завершает строку в месте своего появления, но не завершает абзац. Команда `\par` означает то же самое, что и пустая строка, то есть отделяет один абзац от другого.

Вставить между абзацами вертикальный интервал желаемого размера можно, например, командами `\bigskip`, `\medskip` и `\smallskip`.

1.2. Верстаем документ на русском языке

В этом параграфе мы приведем простой пример исходного текста для русскоязычного документа, чтобы дать представление о том, на что это похоже. Несмотря на то, что документ сам по себе достаточно короткий, его исходный текст может показаться громоздким. Это, однако, лишь первое впечатление; дело в том, что размер преамбулы от размеров документа никак не зависит, то есть одна и та же преамбула потребуется нам и для пробного документа в одну страницу, и для книги в тысячу страниц. В нашем простом примере преамбула занимает заметную часть, и именно поэтому текст примера может показаться громоздким.

1.2.1. Набираем преамбулу

На рис. 1.1 приведен пример преамбулы, которая обладает одним очень важным свойством: она работает. Поначалу многое будет непонятно, но мы постараемся все разъяснить. Впрочем, если разъяснения вас не особенно интересуют, содержание этого параграфа можно просмотреть по диагонали. Если вы используете Windows, все-таки обратите внимание на задание исходной кодировки; если используете Unix, можете просто набрать приведенный тут текст и перейти к следующему параграфу.

Как можно заметить, каждая строка в нашей преамбуле начинается с символа «\», то есть представляет собой команду. Самая первая строка

```
\documentclass[oneside,final,14pt]{extreport}
\usepackage[koi8-r]{inputenc}
\usepackage[russianb]{babel}
\usepackage{vmargin}
\setpapersize{A4}
\setmarginsrb{2cm}{1.5cm}{1cm}{1.5cm}{0pt}{0mm}{0pt}{13mm}
\usepackage{indentfirst}
\sloppy
\begin{document}
```

Рис. 1.1: Пример преамбулы

задает *класс документа*; как уже говорилось, мы будем рассматривать только один класс, который называется `extreport`. Слова `oneside`, `final` и `14pt` — это *опции класса документа*. Опция `oneside` задает одностороннюю печать; вместо этого можно задать `twoside`, тогда на страницах с четными номерами левое и правое поле будут меняться местами. Слово `final` означает чистовую версию; вместо него можно указать `draft`, тогда LaTeX будет помечать жирной вертикальной чертой случайно вылезшие на поля рисунки и строчки, плюс к тому некоторые рисунки отображаться вообще не будут. Последняя опция задает размер шрифта: `14pt` означает 14-й кегль, наиболее популярный среди студентов при написании всяческих рефератов, курсовых и дипломов. Если хотите сэкономить бумагу, перейдите на 12-й кегль, указав `12pt`, или на 10-й (`10pt`).

Вторая строка подключает пакет `inputenc`, чтобы обучить LaTeX понимать буквы кириллицы. Опция `koi8-r` означает используемую кодировку. Кодировка koi8-r принята в системах семейства Unix; **пользователям Windows следует вместо** `koi8-r` **написать** `cp1251`**:**

```
\usepackage[cp1251]{inputenc}
```

Многие современные дистрибутивы ОС GNU/Linux по умолчанию используют «кодировку» utf-8, в которой один символ может задаваться последовательностью из двух, трех или четырех байтов; в частности, символы кириллицы кодируются двухбайтными комбинациями. Благодаря универсальной системе локализации unix-подобных операционных сред вы всегда можете изменить системную локаль, заставив приложения использовать koi8-r. Если же такое решение вам не подходит, скорее всего, LaTeX корректно воспримет исходный текст в utf-8, если вместо `koi8-r` в директиве подключения пакета `inputenc` вы подставите строку `utf8` или `utf8x`: `\usepackage[utf8x]{inputenc}`

Третья строка нашей преамбулы подключает пакет `babel`, который адаптирует LaTeX к работе с русским языком. Забегая вперед, отметим, что именно благодаря этому пакету, например, команда `\chapter{}` выдаст не английское «**Chapter**», а русское «**Глава**». Этот же пакет дает

возможность использования русских типографских кавычек и многого другого.

Следующие три строки преамбулы настраивают размер полосы набора (и, соответственно, размер оставляемых полей). Это делается с помощью пакета `vmargin`; команда `\setpapersize` устанавливает формат бумаги (A4), команда `\setmarginsrb` — размеры полей. В нашем примере левое поле будет 2 см, верхнее — 1.5 см, правое — 1 см, нижнее — 1.5 см. Следующие три параметра команды, в нашем примере оставленные нулевыми, предназначены для управления верхним и нижним колонтитулами, но мы обойдемся без них. Последний параметр (в нашем примере равный 13 мм) имеет отношение к расположению (по вертикали) номера страницы; точнее говоря, он задает расстояние между нижним краем нижней строки и нижним краем номера страницы.

После этого в преамбуле идет команда `\usepackage{indentfirst}`; эта команда нужна, чтобы первый абзац главы или параграфа начинался с отступа (красной строки), как и любой другой. Дело в том, что в соответствии с западными типографскими нормами первый абзац после заголовка не имеет отступа (красной строки). Поскольку традиции оформления русскоязычных текстов требуют отступа перед каждым абзацем, необходимо для этого принять специальные меры, что мы и делаем включением пакета `indentfirst`.

Последняя команда нашей преамбулы, `\sloppy`, указывает LaTeX'у, что с залезанием строк на поля следует бороться, даже если для этого требуется заполнить строку недопустимо длинными пробелами. Во время подготовки документа вы можете заменить эту команду на `\fussy`, в результате чего некоторые строки залезут за правую границу полосы набора; в этих местах следует по возможности проинструктировать LaTeX, как переносить те или иные слова. Перед печатью окончательного варианта документа обязательно замените `\fussy` обратно на `\sloppy`.

Если содержание параграфа показалось вам чрезмерно сложным, не паникуйте! Просто наберите точно такую преамбулу, как в нашем примере (разве что заменив во второй строке `koi8-r` на `cp1251`, если вы используете Windows, или на `utf8` или `utf8x`, если в вашей операционной среде используется кодировка utf-8). Скорее всего, результаты вас устроят.

1.2.2. Набираем текст

Прежде чем приступить к набору текста, добавим в конец нашего файла строку, завершающую документ (`\end{document}`). Теперь между преамбулой и этой строкой можно набрать обычный русский текст с учетом того, что говорилось в § 1.1 про специальные символы. Для начала будет вполне достаточно нескольких абзацев. На рис. 1.2 приведен

```
\documentclass[oneside,final,14pt]{extreport}
\usepackage[koi8-r]{inputenc}
\usepackage[russianb]{babel}
\usepackage{vmargin}
\setpapersize{A4}
\setmarginsrb{2.5cm}{2cm}{1.5cm}{2cm}{0pt}{0mm}{0pt}{13mm}
\usepackage{indentfirst}
\sloppy
\begin{document}
Фольклор~--- явление очень интересное и многоплановое, не
перестающее занимать умы исследователей.  Каких только
способов не изобретают люди, чтобы повеселиться. Вот,
например, неизвестный автор взял два серьезных стихотворения
двух поэтов-классиков и сделал из них своеобразный винегрет:

\bigskip
\noindent Однажды, в студеную зимнюю пору, \\
Сижу за решеткой в темнице сырой.\\
Гляжу, поднимается медленно в гору\\
Вскормленный в неволе орел молодой,\\
И, шествуя важно, в спокойствии чинном\\
Мой грустный товарищ, махая крылом,\\
В больших сапогах, в полушубке овчинном\\
Кровавую пищу клюет под окном.
\bigskip

Вот такое вот "<народное творчество">, извольте видеть.
\end{document}
```

Рис. 1.2: Простой пример исходного файла

пример законченного исходного файла для LaTeX'а; результат его трансляции будет выглядеть примерно так, как показано на рис. 1.3.

Читатель может обратить внимание на то, что каждая строка стихотворения в нашем примере завершается командой принудительного разрыва строки; при этом, хотя ничего похожего на команду `\noindent` перед каждой строкой не наблюдается, строки все же не подвергаются отступу. Дело в том, что команда `\\` только обрывает текущую строку, **не завершая абзац**. Тем не менее, перед первой строкой стихотворения поставить `\noindent` все же пришлось, поскольку предыдущий абзац завершен, так что, не будь этой команды, первая строка стихотворения оказалась бы сдвинута, а остальные — нет (смотрится это довольно некрасиво). Само стихотворение обрамлено командами `\bigskip`, чтобы отделить его от остального текста. На самом деле для вставки в текст

Фольклор — явление очень интересное и многоплановое, не перестающее занимать умы исследователей. Каких только способов не изобретают люди, чтобы повеселиться. Вот, например, неизвестный автор взял два серьезных стихотворения двух поэтов-классиков и сделал из них своеобразный винегрет:

Однажды, в студеную зимнюю пору,
Сижу за решеткой в темнице сырой.
Гляжу, поднимается медленно в гору
Вскормленный в неволе орел молодой,
И, шествуя важно, в спокойствии чинном
Мой грустный товарищ, махая крылом,
В больших сапогах, в полушубке овчинном
Кровавую пищу клюет под окном.

Вот такое вот «народное творчество», извольте видеть.

Рис. 1.3: Результат трансляции

подобных объектов существуют гораздо более совершенные средства, но здесь нашей задачей было привести простой пример, содержащий некоторые характерные для LaTeX'а команды, что мы и сделали.

1.2.3. Запускаем LaTeX и просматриваем результаты

Итак, файл (назовем его, к примеру, `sample.tex`) набран и готов к использованию. Запустите LaTeX (в ОС Unix это делается командой `latex sample.tex`). Если все пройдет нормально, в текущем каталоге появится файл `sample.dvi`. Просмотреть этот файл можно с помощью команды `xdvi sample.dvi`. Программа `xdvi` интерактивна. Перелистывание страниц вперед осуществляется клавишей «пробел», назад — клавишей Backspace. Выйти из программы можно кнопкой «`Q`».

На всякий случай расскажем, что делать, если при трансляции возникли ошибки. Для иллюстрации вставим в наш файл после слов «шествуя важно» какую-нибудь несуществующую команду, например, `\abcd`, и посмотрим, что получится. LaTeX выдаст примерно такое сообщение:

```
! Undefined control sequence.
l.21 И, шествуя важно\abcd
                       , в спокойствии чинном,\\
?
```

Текст в строке, начинающейся с восклицательного знака, содержит сообщение об ошибке. В большинстве случаев понять из него все равно ничего не удается, так что долго ломать над ним голову не стоит. Правда, в данном конкретном случае для знающих английский все как раз понятно: «Undefined control sequence» переводится буквально как «Неопределенная управляющая последовательность». Но даже если вы не знаете английского, это не страшно: точно зная, где произошла ошибка, обычно можно легко понять, в чем она заключается. И в этом нам поможет информация из следующей строки. Надпись `l.21` означает, что ошибка произошла в 21-й строке исходного файла; дальше идет сама строка, в месте ошибки разорванная для наглядности переводом строки. Здесь уже нет никаких проблем заметить, что LaTeX'у не понравилась именно «команда» `\abcd`.

Как правило, после первой ошибки дальнейшая работа программы `latex` бессмысленна, однако программа работает так, как будто после ошибки что-то еще можно сделать, причем, что самое неприятное, `latex` даже нельзя прервать нажатием `Ctrl-C` (зачем так сделано — вопрос, который давным-давно мучает многих пользователей LaTeX'а). Секрет мгновенного выхода из программы в том, что надо сразу, получив первое же сообщение, немедленно нажать комбинацию «конец файла» (в Unix'е это обычно `Ctrl-D`, в Windows — `Ctrl-Z`). Если не сделать этого сразу, то после выдачи второго сообщения никакие комбинации уже не помогают и приходится ждать, пока LaTeX не выдаст все сообщения, которые ему хочется, и не успокоится.

Глава 2

Жирный, курсив и все-все-все

Выше уже говорилось, что в LaTeX'е все управление внешним видом текста, включая и виды выделения, осуществляется с помощью команд. Попробуем научиться этим пользоваться.

2.1. Управление формой шрифта

2.1.1. Чем тут можно управлять?

Для начала разберемся, какие вообще у нас имеются возможности по части шрифтов. LaTeX предоставляет три *гарнитуры*[1]: обычная (Roman), гарнитура без засечек (Sans Serif) и моноширинная (`Typewriter`). Последняя обычно используется для набора текстов программ и вообще текстов на формальных языках; так, в данном пособии моноширинным шрифтом набраны фрагменты исходных текстов на входном языке LaTeX'а.

Кроме гарнитуры, шрифт обладает *жирностью*, то есть может быть либо нормальным, либо **жирным**, и *формой*[2], которая бывает нормальной, *курсивной*, *наклонной* и МЕЛКОПРОПИСНОЙ (КАПИТЕЛЬ). Гарнитура, жирность и форма изменяются независимо друг от друга, хотя некоторые комбинации могут и не работать: например, в гарнитуре без засечек нет курсива, а в моноширинной гарнитуре невозможно выделение жирным [2].

[1] В англоязычной документации используется термин *font family*.

[2] Нашим терминам *жирность* и *форма* в англоязычной документации соответствуют *series* и *shape*.

`\sffamily`	гарнитура без засечек (Sans Serif)
`\ttfamily`	моноширинная гарнитура (Typewriter)
`\rmfamily`	обычная гарнитура (Roman)
`\bfseries`	включение ожирнения (bold face)
`\mdseries`	отмена ожирнения (medium density)
`\itshape`	курсив (italics)
`\slshape`	наклонный шрифт (slanted)
`\scshape`	капитель (small caps)
`\upshape`	обычное начертание (upright)
`\normalfont`	основной шрифт документа (устанавливает обычную гарнитуру, обычное начертание и выключает ожирнение)

Таблица 2.1: Команды переключения формы шрифта

2.1.2. Декларирующие команды

Существует три основных способа задать нужное начертание шрифта. Самый простой из них — *декларирующие команды*, заставляющие LaTeX переключиться на требуемую гарнитуру, жирность или форму. Эффект от такой команды распространяется до следующей команды той же категории. Например, если вы установили курсивную форму, то именно она и будет использоваться, пока вы не установите другую; при этом вы можете, если нужно, переключать гарнитуры и жирность. Рассмотрим пример:

```
В этой \itshape фразе почти все \bfseries
набрано курсивом, \ttfamily но кое-что
\upshape еще и жирное, \rmfamily а где-то
и вовсе \mdseries моноширинное.
```

Результат будет примерно такой:

> В этой *фразе почти все* ***набрано курсивом,*** *но* ***кое-что*** еще и **жирное, а где-то и вовсе** моноширинное.

Здесь мы с помощью декларирующих команд сначала задали курсивную форму, потом установили жирное выделение (в результате получился жирный курсив), затем переключились на моноширинную гарнитуру (жирность при этом перестала работать, хотя и не пропала), потом мы отменили курсив, еще через три слова вернулись к обычной гарнитуре и наконец перед последним словом сняли жирность.

Все команды этой категории перечислены в табл. 2.1. Обратите внимание на команду `\normalfont`: в сложной ситуации она поможет вам не запутаться.

Отметим еще одну особенность декларирующих команд. Их область действия не может распространяться дальше «блока», в котором они появились; в качестве такого блока, например, может выступать текст, заключенный в фигурные скобки. Поэтому в некоторых случаях оказывается удобнее вместо двух команд (включения выделения и отмены выделения) просто заключить выделяемый текст в фигурные скобки и в начале поставить включающую команду. Поясним сказанное на примере:

```
Сначала {\bfseries жирное, потом {\itshape курсив, потом}
                  без курсива, потом} без жирного.
```

В результате получится:

> Сначала **жирное, потом *курсив, потом* без курсива, потом** без жирного.

2.1.3. Команды с параметром

Кроме декларирующих команд, шрифтами можно управлять с помощью *команд с параметром*. Этим командам в качестве параметра в явном виде указывается текст, на который необходимо распространить их действие. Такие команды имеют вид `\textXXX{`<текст>`}`, где вместо `XXX` подставляется обозначение требуемого изменения: `bf` для выделения жирным, `it` для курсива, `tt` для моноширинного шрифта и т. д. Последний пример предыдущего параграфа можно с помощью этих команд набрать так:

```
Сначала \textbf{жирное, потом \textit{курсив, потом}
                  без курсива, потом} без жирного.
```

Повторим еще раз, что категория включает в себя команды `\textsf{}`, `\texttt{}`, `\textrm{}`, `\textbf{}`, `\textmd{}`, `\textit{}`, `\textsl{}`, `\textsc{}`, `\textup{}` и `\textnormal{}`, причем каждой из них соответствует команда в декларирующей форме (команде `\textnormal{}` соответствует команда `\normalfont`). К рассматриваемой категории относится и еще одна команда, `\emph{}` (от слова *emphasis*, то есть *выделение*). Она осуществляет переключение между обычным шрифтом и курсивом. Декларирующей формы у этой команды нет.

2.1.4. Команды в форме окружения

Еще один способ задать свойства шрифта — это заключить набираемый текст в *окружение*, то есть обрамить его командами `\begin{`<имя окружения>`}` и `\end{`<имя окружения>`}`, причем в качестве имени окружения использовать слово, являющееся именем соответствующей декларирующей команды. Уже знакомый нам пример можно набрать еще и так:

```
Сначала \begin{bfseries} жирное, потом
          \begin{itshape} курсив, потом \end{itshape}
          без курсива, потом
        \end{bfseries} без жирного.
```

2.1.5. Устаревшая форма декларирующих команд

Для совместимости со старыми версиями TEX'а и LATEX'а поддерживаются также короткие декларирующие команды `\bf`, `\it`, `\sl`, `\sc`, `\sf` `\tt` и `\rm`. Иногда их удобно использовать благодаря их лаконичности. К сожалению, у этих команд есть серьезный недостаток: их эффект не может быть наложен, то есть с их помощью нельзя, например, задать жирный курсив или наклонный моноширинный шрифт. Внимательный читатель, возможно, уже заметил отсутствие команд `\md` и `\up`. Этих команд действительно нет, а отменить любое выделение (ведь действовать здесь может только какое-то одно выделение!) можно командой `\rm`, которая устанавливает гарнитуру Roman в обычной форме и без ожирнения. Несмотря на весьма ограниченные возможности этих команд, многие пользователи LATEX'а все-таки продолжают с ними работать (кто-то по привычке, кому-то нравится лаконичность). Обычно при использовании этих команд выделяемый текст берется в фигурные скобки, в начале которых ставится команда, например, так: `{\it тут курсив}`.

2.2. Управление размером шрифта

2.2.1. Размеры шрифта и команды

Как уже говорилось в § 1.2.1, основной размер шрифта вашего документа устанавливается в первой строке преамбулы, в качестве опции для класса документа. При необходимости можно изменить размер шрифта для отдельных фрагментов документа. Команды в декларирующей форме (то есть команды, эффект от которых действует либо до конца блока, либо до следующей команды той же категории), предназначенные для управления размером шрифта, перечислены в табл. 2.2.

`\tiny`	самый мелкий возможный размер, обычно совершенно нечитаемый
`\scriptsize`	чуть больше, но все равно очень мелкий
`\footnotesize`	размер, которым набираются сноски
`\small`	шрифт чуть мельче обычного
`\normalsize`	обычный размер шрифта (тот, что задан в преамбуле)
`\large`	слегка укрупненный
`\Large`	еще крупнее
`\LARGE`	совсем крупный
`\huge`	гигантский
`\Huge`	супергигантский

Таблица 2.2: Команды, задающие размер шрифта

Отметим одну очень важную особенность LaTeX'а: абсолютный размер шрифта (номер кегля) обычно фигурирует **только в одном месте** — а именно в преамбуле. Команды локальной смены шрифта работают **относительно** основного размера, т. е. команда `\normalsize` устанавливает размер шрифта, равный указанному в преамбуле, команда `\large` устанавливает размер, чуть бóльший заданного в преамбуле, и т. д. Таким образом, если вам захочется напечатать весь документ крупнее или мельче, достаточно будет изменить одно слово в преамбуле. При этом изменятся соответствующим образом **все** размеры шрифтов вашего документа, так что целостность верстки нарушена не будет (то есть везде, где вы использовали выделение бóльшим или меньшим шрифтом, такое выделение сохранится[3]).

Как и для команд, декларирующих начертание шрифта (см. § 2.1.4), для всех команд, устанавливающих размеры, имеются аналогичные окружения. Например, если некий текст поместить между командами `\begin{large}` и `\end{large}`, весь этот текст будет набран увеличенным шрифтом.

2.2.2. Абзацы и интерлиньяж

У алгоритма, используемого LaTeX'ом для размещения информации на странице, есть одна особенность, о которой полезно знать, прежде чем начинать использовать команды смены шрифта. При смене размера шрифта (непосредственно в том месте, где вставлена соответствующая

[3] На самом деле не все размеры шрифтов реально существуют, так что, например, при основном кегле №14 размеры `Huge`, `huge` и `LARGE` совпадают, представляя собой наибольший возможный кегль (№25).

команда) определяется только размер символов, которые с этого момента будут использоваться при верстке. Что же касается, например, расстояния между строками (говоря более строго, расстояния между базовыми линиями строк, или интерлиньяжа), то оно определяется в тот момент, когда очередная строка завершается (либо по принудительной команде, либо, при верстке целого абзаца, по его окончании). Так, если вы начали писать абзац крупным шрифтом, а закончили мелким, то начальные строки (которые набраны крупными символами) окажутся друг от друга на неожиданно малом расстоянии. Друг на дружку строки не налезут, за этим LaTeX следит, но смотреться это все будет довольно-таки неказисто:

Вороне где-то бог послал кусочек сыру; На ель Ворона взгромоздясь, Позавтракать было совсем уж собралась, Да позадумалась, а сыр во рту держала. На ту беду, Лиса близехонько бежала; Вдруг сырный дух Лису остановил. Лисица видит сыр,— Лисицу сыр пленил.

Если, наоборот, начать абзац с мелкого шрифта, а закончить крупным, получим обратный эффект:

Плутовка к дереву на цыпочках подходит; Вертит хвостом, с Вороны глаз не сводит И говорит так сладко, чуть дыша: ≪Голубушка! Как хороша! Ну что за шейка! Что за глазки! Рассказывать, так, право, сказки! Какие перышки! Какой носок!≫

Даже если набрать весь абзац одним шрифтом, можно сделать одну достаточно характерную ошибку, из-за которой у вас «съедет» интерлиньяж. Если, скажем, в начале абзаца переключиться на крупный шрифт (например, командой `\large`), набрать весь абзац, после абзаца вставить команду перехода обратно на стандартный (`\normalsize`) и только после этого оставить пустую строку, то LaTeX **сначала** перейдет на стандартный размер (и характерный для него интерлиньяж), и только **потом** «увидит», что пора заканчивать абзац и формировать строки.

Простое и надежное решение этой проблемы состоит в соблюдении двух правил: **всегда набирайте абзац шрифтом одного размера** и **всегда оставляйте пустую строку перед командой смены шрифта**. Это касается и использования окружения: пустую строку для безопасности следует оставить и перед командой начала окружения, и перед командой его конца, то есть если, например, вы решили использовать окружение `small`, то пустые строки следует оставить и перед `\begin{small}`, и перед `\end{small}`. В этом случае неприятности с некрасивым интерлиньяжем вам не грозят.

Глава 3

Как сделать рубрикацию и оглавление

Любой более-менее сложный документ имеет разбивку на части, главы, параграфы и т. п. Такая разбивка называется *рубрикацией*. Следует отметить, что требования к оформлению курсовых и дипломных работ обычно в явном виде виде упоминают необходимые элементы рубрикации, такие как, например, ≪введение≫ и ≪заключение≫. Кроме того, любая квалификационная работа обязана, разумеется, иметь оглавление.

WYSIWYG-редакторы (будем верны себе и не станем их называть) имеют средства корректной организации рубрикации и даже автоматического создания оглавлений, однако этими средствами мало кто из пользователей (surprize!) владеет. Автору неоднократно приходилось видеть дипломные работы, в которых номера страниц в оглавлении либо не совпадали с реальностью, либо были вписаны от руки (!), либо вообще отсутствовали. Удивительно, сколь часто люди пытаются создать заголовок очередного пункта путем ручного центрирования соответствующей строки, ручной же смены размера шрифта для нее и отбивки сверху и снизу пустыми абзацами. Оглавление при этом тоже формируется вручную. При таком подходе, понятное дело, к концу работы над документом ручное оглавление и настоящее содержимое заголовков оказываются весьма друг от друга далеки[1]. Ну а рекомендация написать фразу типа ≪это утверждение мы уже доказывали в § 2.15, см. стр. 39≫ попросту повергает автора работы в шок — *ведь и номер параграфа, и номер страницы еще не раз изменятся!*

Разумеется, LaTeX, изначально предназначенный для верстки научных трудов, включает и средства рубрикации (с автоматической нуме-

[1] Это уже не говоря о том, что сами заголовки могут в итоге оказаться оформленными в разном стиле.

рацией разделов), и возможность автоматического создания оглавления и отслеживания перекрестных ссылок; в освоении всех этих средств ничего сложного нет.

Помните одно правило: **ни в коем случае не надо делать вручную то, что LaTeX может сделать автоматически**.

3.1. Команды рубрикации для класса extreport

В зависимости от класса документа LaTeX поддерживает разные наборы *уровней рубрикации*. Для класса `extreport` это ≪часть≫ (`\part`), ≪глава≫ (`\chapter`), ≪секция≫[2] (`\section`), ≪подсекция≫ (`\subsection`), ≪подподсекция≫ (`\subsubsection`), ≪пункт≫ (`\paragraph`) и ≪подпункт≫ (`\subparagraph`). Все эти команды имеют один параметр — заголовок соответствующего раздела, например:

```
\chapter{Введение}
\section{Постановка задачи}
```

Из перечисленных семи уровней в большинстве документов используется лишь два-три. До пунктов и подпунктов дело обычно не доходит, а ≪часть≫ оказывается слишком ≪тяжелым≫ средством оформления. В курсовых и дипломных работах команду `\part` обычно не применяют; делить текст на части с помощью этой команды имеет смысл, если общий объем текста перевалил за две-три сотни страниц (например, при подготовке монографии), и при этом имеющиеся главы можно логически объединить в группы (эти группы и станут частями).

На случай, если в ваши планы входит как раз верстка монографии, учебника или другой книги большого объема, отметим, что команда `\part` использует отдельную страницу, по центру которой выводится надпись ≪Часть≫ с номером части (нумерация частей начинается с единицы, для нее используются римские цифры), потом на отдельной строке (или строках) располагается взятый из аргумента команды *заголовок части*. При верстке в режиме `twoside` (для двухсторонней печати) заголовок части всегда размещается на нечетной странице, а следующая за ней страница оставляется пустой (то есть для обозначения начала новой части используется целый лист). Еще раз отметим, что при оформлении курсовых и дипломных работ использование этого уровня рубрикации не рекомендуется; более того, не используется этот уровень и в диссертациях, т. к. диссертации традиционно делятся именно на главы, а не на части.

Следующий уровень рубрикации (главы), напротив, используется обязательно. Дело в том, что, начиная с уровня глав (команда `\chapter`),

[2] На самом деле слово section следовало бы переводить как ≪параграф≫, но как раз этого мы во избежание лишней путаницы делать не будем, поскольку английское paragraph, означающее ≪абзац≫, в рубрикации тоже используется.

для нумерации разделов документа используется система *подчиненных счетчиков*. Это означает, что сами главы имеют номера 1, 2, 3 и т. д., секции первой главы нумеруются как 1.1, 1.2, 1.3, ... , подсекции, соответственно, имеют номера 1.1.1, 1.1.2, ... , 1.2.1, ... , 2.3.5 и т. д., поэтому ясно, что начав рубрикацию с какого-то более низкого уровня, вы получите очень странную нумерацию разделов.

Итак, команда `\chapter{`<заголовок>`}` начинает новую главу. Это означает, что LaTeX переходит к началу новой страницы (глава всегда начинается с новой страницы), печатает слово «Глава» и номер главы, потом выводит заголовок главы. При двухсторонней печати главы обычно начинают с нечетных страниц; можно, однако, попросить LaTeX не делать этого, добавив в первую строку преамбулы опцию `openany` (она вставляется в список в квадратных скобках в команде `\documentclass`). Впрочем, это вам вряд ли понадобится, т. к. курсовые и дипломные работы обычно верстают в режиме односторонней печати.

При желании главу можно разделить на *секции* командой `\section`. Эта команда уже никаких лишних слов не печатает: выводится только номер секции из двух чисел, разделенных точкой, и заголовок, который набирается сравнительно крупным шрифтом (тем же, который используется командой `\Large`) с ожирнением.

Секция, в свою очередь, может быть разделена на подсекции командой `\subsection`; эта команда также печатает номер раздела (для подсекций он состоит уже из трех чисел) и заголовок, причем шрифт пока еще крупнее основного — здесь он тот же, что породила бы команда `\large`.

Если вам не хватило этих уровней рубрикации (что само по себе очень странно), вы можете воспользоваться командой `\subsubsection` для деления подсекций на еще меньшие части, подподсекции. Заголовок подподсекции печатается уже шрифтом нормального размера, только жирным. В классе `extreport` подподсекции по умолчанию не имеют номеров, однако вы можете это изменить, дав такую команду:

```
\setcounter{secnumdepth}{3}
```

Эта команда задает максимальный уровень рубрикации, подлежащий нумерации. В классе `extreport` части имеют уровень -1, главы — 0, секции — 1, подсекции — 2 и т. д. По умолчанию, как вы, возможно, догадались, значение `secnumdepth` равно 2.

Использование пунктов и подпунктов для отечественной печати нехарактерно, но если вам не хватило деления на подподсекции, можно воспользоваться и ими[3]. Заголовки пунктов и подпунктов печатаются жир-

[3] Хотя лучше, пожалуй, подумать, не реорганизовать ли тем или иным способом рубрикацию вашего документа; например, можно посоветовать разбить одну главу на несколько.

ным шрифтом обычного размера, причем текст самого пункта или подпункта начинается на той же строке, где располагается заголовок[4]. Заголовок подпункта отличается тем, что печатается с красной строки.

По умолчанию, разумеется, пункты и подпункты не нумеруются, но вы можете исправить и это, установив `secnumdepth` в значение 4 (нумеруем пункты) или 5 (нумеруем и пункты, и подпункты). Прежде чем это делать, подумайте о том, что номер подпункта будет состоять из **шести** разделенных точками чисел, что-то вроде 2.12.7.32.19.4. Настоятельно рекомендуется еще раз рассмотреть вопрос об отказе от пунктов и подпунктов! Уровней рубрикации вам должно хватить и без них. Обратите внимание, что в книге, которую вы читаете, ни разу не используется даже деление на подподсекции.

3.2. Оглавление

Сформировать оглавление в LaTeX'е очень просто: для этого достаточно в том месте, где вы желаете видеть оглавление, вставить команду

```
\tableofcontents
```

Учтите, что теперь вам может понадобиться каждый раз прогонять программу `latex` дважды. Дело в том, что для формирования оглавления LaTeX использует дополнительный файл (с расширением `.toc`), в который во время трансляции вашего исходного текста заносится информация для включения в оглавление. Само оглавление команда `\tableofcontents` формирует, включая в трансляцию этот файл. Таким образом, на первом проходе LaTeX может сформировать некорректное оглавление (особенно если оглавление вставляется в начало документа), поскольку файл `.toc` может отсутствовать либо содержать устаревшую информацию. На втором проходе этой проблемы уже не будет. Если тут что-то непонятно, не пугайтесь: просто дайте команду `latex` два раза, и все.

Если результаты формирования оглавления вас устроили, можете остаток этого параграфа пропустить.

По умолчанию в документах класса `extreport` в оглавление включаются названия частей (если они есть), глав, секций и подсекций. Более низкие уровни рубрикации в оглавлении не отражаются. Вы можете изменить это, установив значение `tocdepth`, например

```
\setcounter{tocdepth}{3}
```

[4] Буквально команды `paragraph` и `subparagraph` переводятся с английского как ≪абзац≫ и ≪подабзац≫, так что их заголовки — это заголовки абзацев.

заставит LaTeX включать в оглавление информацию о подподсекциях, ну а если вы установите значение 5, то в оглавление попадут и пункты с подпунктами. Еще раз настоятельно не советуем этого делать.

Значение `tocdepth` можно и уменьшить. Если, например, упоминание секций и подсекций в оглавлении представляется вам ненужным, установите `tocdepth` в значение 0; тогда в оглавление пойдут только главы и части (конечно, если они есть).

3.3. Перекрестные ссылки

Часто (особенно в научных работах) возникает потребность написать что-то вроде «это мы обсуждали в §2.5» или «см. рассуждение на стр. 37». Проблема тут в том, что в процессе редактирования нашей рукописи номер нужного параграфа может измениться, а рассуждение — ненавязчиво «уползти» на другую страницу. Отслеживать эти эффекты вручную — задача неблагодарная. Естественно, LaTeX позволяет автоматизировать формирование таких ссылок.

Чтобы сослаться на какой-либо раздел текста или на номер страницы, где некий текст расположен, этот текст следует *пометить*, вставив в него команду `\label{`<метка>`}`. В качестве имени метки можно использовать любую последовательность латинских букв, цифр и некоторых знаков препинания (особенно часто используются двоеточие, подчеркивание и тире), например:

```
\label{main:theorem:prove}
```

Сама команда `\label` ничего не печатает; в принципе, о ней можно просто забыть, ничего страшного от этого не случится.

Метка оказывается связана одновременно с двумя параметрами: самым длинным из номеров разделов, в которых оказалась метка (так, метка может оказаться в подсекции, которая находится в секции, которая находится в главе, но с меткой будет связан номер подсекции, а не секции и не главы)[5], и с номером страницы.

Чтобы сослаться на помеченный параграф, необходимо использовать команду `\ref{`<метка>`}`:

```
доказательство см.~в~\S\,\ref{main:theorem:prove}
```

Если же нужна ссылка на страницу, следует пользоваться командой `\pageref{`<метка>`}`:

```
приведено на стр.\,\pageref{main:theorem:prove}
```

[5]Забегая вперед, отметим, что меткой можно помечать не только обычный текст в рамках разделов, но и пункты нумерованных перечислений, а также нумеруемые объекты, такие как таблицы и рисунки.

3.4. Оформление приложений

Если в вашей работе предусмотрены приложения, поставьте перед ними команду `\appendix`. Дать эту команду следует только один раз, чтобы отделить текст всех приложений от основного текста документа. После команды `\appendix` каждое приложение оформляется уже знакомой нам командой `\chapter{}`, но вместо слова «Глава» и номера арабскими цифрами эта команда теперь печатает слово «Приложение» и вместо номера — заглавную латинскую букву (первое приложение будет обозначено «Приложение A», второе — «Приложение B» и так далее).

При желании каждое приложение, как и обычная глава, может быть разбито на секции, подсекции и т. д., при этом, например, вторая подсекция третьей секции приложения B будет иметь номер B.3.2.

3.5. Низкоуровневое управление

Иногда возникает потребность вставить в документ заголовок, не имеющий номера и не влияющий на нумерацию остальных пунктов. Это делается «звездочными» версиями команд рубрикации: `\part*`, `\chapter*`, `\section*` и т. д. Например, именно так оформлено предисловие в нашем пособии — как «секция», не относящаяся при этом ни к какой главе, а затем и введение — как глава без номера. Собственно говоря, все, что делает «звездочная» версия команды — это печать заголовка жирным шрифтом соответствующего размера со всеми подобающими отступами (для `\part*`, как и для `\part`, выделяется отдельная страница, для `\chapter*` осуществляется принудительный переход к следующей странице). Команды не влияют ни на счетчики разделов, ни на оглавление.

При необходимости можно принудительно добавить пункт оглавления. Это делается командой `\addcontentsline`. Например, в этой книжке перед предисловием вставлены следующие две команды:

```
\section*{От автора}
\addcontentsline{toc}{chapter}{От автора}
```

Слово `chapter` означает, что вставляемый пункт должен выглядеть так же, как пункты для глав (то есть разделов, оформленных командой `\chapter`); «От автора» — это вставляемый текст. Что обозначает первый параметр команды (`toc`), читатель может узнать самостоятельно; в большинстве случаев следует писать именно это.

Отметим, что директивой `\addcontentsline` можно (и нужно) воспользоваться также для того, чтобы внести в оглавление пункт «Литература», который сам LaTeX в оглавление не добавляет.

Глава 4

Список литературы и ссылки на него

Обязательной частью любого научного текста является список литературы, на элементы которого в тексте имеются ссылки. Такие ссылки обычно представляют собой заключенный в квадратные скобки номер, под которым нужная публикация фигурирует в списке литературы[1].

Ясно, что практика расстановки ссылок вручную чревата ошибками, особенно в случае, если по тем или иным причинам придется вносить изменения в список источников; при этом некоторые источники могут сменить номера, а внести соответствующие изменения в текст мы, как водится, забудем.

LaTeX может взять на себя заботу о расстановке номеров ссылок. От нас для этого требуется оформить список литературы с помощью специально предназначенного окружения `thebibliography`, а ссылки вставлять в текст командой `\cite`.

Окружение `thebibliography` начинается командой

```
\begin{thebibliography}{00}
```

Параметр (в нашем примере `00`) используется для указания *максимальной ширины* номера ссылки и используется LaTeX'ом для горизонтального выравнивания элементов списка. Если ваш список литературы состоит не более чем из девяти пунктов, можно вместо `00` указать просто `0`, если же количество источников составляет сто и больше, стоит указать `000`.

Внутри окружения располагаются описания библиографических ссылок, каждое из которых начинается командой

[1] Бывают и другие стили оформления ссылок, например, по фамилии автора и году издания, но в современной русскоязычной печати такое встречается редко.

`\bibitem[`<*номер*>`]{`<*метка*>`}`. Необязательный параметр *номер* обычно опускают, доверяя LaTeX'у самостоятельно расставить номера источников в списке. Параметр *метка* задает метку, по которой мы будем ссылаться на данный литературный источник в команде `\cite`; как и для обычной метки, здесь можно применять любую последовательность из латинских букв, цифр и некоторых знаков препинания (обычно используют двоеточие и знак подчеркивания).

Приведем пример окружения `thebibliography`:

```
\begin{thebibliography}{00}

\bibitem{knuth:tex} Дональд Е.~Кнут.
\emph{Все про \TeX}. Изд-во АО RDTeX,
Протвино, 1993.

\bibitem{kuhn:revolutions} T.~S.~Kuhn.
\emph{The structure of scientific revolutions}.
University of Chicago Press,
Chicago, second, enlarged edition, 1970.

\bibitem{stolyarov:oorefal} А.~В.~Столяров.
\emph{Расширенный функциональный аналог языка
Рефал для мультипарадигмального программирования.}
// Л.~Н.~Королев, ред., \emph{Программные
системы и инструменты}. Тематический сборник,
том~2, стр.~184--195. Издательский отдел ВМиК МГУ,
Москва, 2001.

\end{thebibliography}
```

В этом примере первые два пункта — книги, а третий — статья в сборнике. Как можно заметить, заглавия (книг, статьи, сборника) мы выделяем курсивом, используя команду `\emph` (см. стр. 23).

Существуют и другие стили оформления списка источников. Например, можно выделять имя автора ожирнением, заглавие набирать нормальным шрифтом, а выходные данные — курсивом и т. п. Требования к оформлению могут зависеть от традиций, принятых в вашем ВУЗе, так что рекомендуем уточнить их заблаговременно. Часто требуется оформить библиографию в соответствии с действующим ГОСТ; как это делается, можно узнать из книги [2].

Как уже говорилось, в текст работы ссылки на источники вставляются командой `\cite[`<*текст*>`]{`<*метка*>`}`. Например, мы можем сослаться на книгу Куна из нашего примера командой `\cite{kuhn:revolutions}`; поскольку в списке она на втором месте,

LaTeX преобразует эту команду в строку «[2]». Необязательный параметр *текст* можно использовать, чтобы вставить в текст ссылки дополнительный комментарий, например, указать номер страницы: `\cite[стр.~27]{kuhn:revolutions}`. В тексте в этом случае появится строка «[2, стр. 27]».

В некоторых случаях от вас могут потребовать, чтобы номера источников заключались в квадратные скобки только в ссылках в тексте, но не в самом списке литературы. На наглядности это сказывается не лучшим образом, но требование есть требование; чтобы исполнить его, добавьте в преамбулу вашего документа следующие строки:

```
\makeatletter
\renewcommand*{\@biblabel}[1]{\hfill#1.}
\makeatother
```

Поясним, что здесь мы меняем определение команды `\@biblabel`, которую LaTeX использует для формирования меток в списке литературы. Эта команда относится к категории *внутренних*, то есть предназначена для использования внутри пакетов, и поэтому содержит в имени символ «@»; обычно при обработке файлов пакетов LaTeX считает этот символ обычной буквой, а при обработке файлов документа — специальным символом, так что, не предприми мы особых мер, последовательность «`\@biblabel`» он бы за единую команду не воспринял. Именно поэтому мы сначала заставляем LaTeX считать «@» обычной буквой (команда `\makeatletter`), а потом отменяем это (команда `\makeatother`).

Отметим, что часто окружение `thebibliography` генерируют автоматически, используя программу `bibtex`, входящую в комплект поставки LaTeX'а. Эта программа использует библиографическую базу данных, находящуюся в отдельном файле, выбирая из нее только те источники, на которые в вашем документе имеются ссылки. Можно, например, составить один большой библиографический каталог, который вы будете использовать при формировании списков литературы во всех своих работах. Описание программы `bibtex` достаточно велико по объему, так что мы не будем его приводить в этом пособии. Читатель может освоить программу `bibtex`, обратившись, например, к книге [4].

Глава 5

Описание новых команд и окружений

При первом прочтении эту главу можно спокойно пропустить. В следующих главах нам понадобятся средства описания новых (пользовательских) команд и окружений, так что мы вынуждены рассказать, как это делается; однако материал следующих глав будет вполне понятен и без этого, а потребность в собственных командах у вас, скорее всего, возникнет еще не скоро. Поэтому, если не хочется забивать голову лишней информацией, вы можете эту главу не читать: вернуться к ней вы всегда успеете.

5.1. Введение новых команд

Для создания новой команды применяют директиву `\newcommand`:

`\newcommand`<*имя*>`[`<*арг*>`][ `<*умолч*>`]{`<*содержание*>`}`

Сразу за словом `\newcommand` должно идти имя определяемой команды, включая символ «\». Если мы хотим, чтобы команда имела параметры, следует в квадратных скобках указать их количество (параметр *арг*). Если параметров не предполагается, *арг* вместе с квадратными скобками можно опустить. Отметим, что количество параметров не может быть больше девяти.

Если параметры есть, то первый из них можно сделать допускающим значение по умолчанию; для этого применяется аргумент *умолч* (также в квадратных скобках). В этом случае и первый параметр нашей команды тоже будет записываться в квадратных скобках, если же квадратных

скобок нет, LaTeX будет считать, что необязательный параметр опущен, и использует то значение, которое мы при описании команды задали в качестве аргумента *умолч.*

Наконец, в фигурных скобках записывается *содержание* нашей команды. Если у команды есть параметры, на них можно сослаться как `#1`, `#2`, ..., `#9`.

Рассмотрим примеры. Допустим, каждый раз при введении нового понятия нам хочется это понятие выделять из текста, но мы пока не знаем, захотим мы это делать курсивом, ожирнением или как-то еще. Поэтому мы опишем команду `\notion` с одним аргументом и для выделения понятий будем везде применять эту команду. Тогда окажется достаточно поменять содержание команды (то есть изменить наш исходный текст в одном месте), чтобы стиль выделения изменился во всем документе. Команду мы опишем так:

```
\newcommand\notion[1]{\textit{#1}}
```

Таким образом, мы ввели команду с именем `\notion`, имеющую всегда один параметр, которая воспроизводит свой параметр (текст) курсивным шрифтом. Если теперь мы вводим новое понятие, для выделения его мы будем пользоваться командой `\notion`, например:

```
Если операция, заданная в группе G, коммутативна,
G называется \notion{абелевой группой}.
```

Рассмотрим еще один пример. Допустим, нам хочется время от времени вставлять в текст «отбивку» — строку, состоящую из нескольких одинаковых декоративных символов, чтобы, например, в конце каждой главы помещать краткое резюме, отделенное от основного текста. Введем для этого следующую команду:

```
\newcommand\Brief[1][*]{
   \bigskip
   \centerline{#1~~#1~~#1~~#1~~#1}
   \bigskip
}
```

Теперь команда `\Brief` без параметров выдаст «отбивку» из пяти звездочек. Если же мы захотим воспользоваться для отбивки другим символом (например, плюсом), достаточно будет дать команду `\Brief[+]`.

5.2. Описание новых окружений

Новое окружение вводится с помощью команды

`\newenvironment{`<*имя*>`}[`<*арг*>`] [`<*умолч*>`]{`<*нач*>`}{`<*кон*>`}`

В отличие от `\newcommand`, имя окружения берется в фигурные скобки; количество аргументов и умолчание задается точно так же, а содержательная часть состоит из двух частей: *нач* задает действия при открытии окружения (то есть по команде `\begin`), *кон* — при закрытии (по команде `\end`).

Рассмотрим следующий пример. Допустим, в нашем тексте встречаются пространные примечания в виде отдельных абзацев, которые желательно набирать шрифтом меньшего размера и, возможно, другой гарнитуры и начертания. При этом мы еще не знаем, какой именно размер, начертание и гарнитуру будем применять в финальной версии документа. Чтобы оставить себе возможность изменить решение в любой момент, опишем новое окружение, назвав его, например, `remarks`:

```
\newenvironment{remarks}{
  \begin{sffamily}
  \small
}{
  \par
  \normalsize
  \end{sffamily}
}
```

Теперь мы можем заключать наши примечания между командами `\begin{remarks}` и `\end{remarks}`; перед их отображением LaTeX перейдет на гарнитуру Sans Serif и установит уменьшенный шрифт, по окончании — восстановит нормальный размер шрифта и использовавшуюся до примечаний гарнитуру. Если же нам захочется сменить шрифт, которым набираются такие примечания (например, набирать их обычным курсивом), достаточно будет изменить описание окружения `remarks` (то есть внести изменения в исходный текст только в одном месте).

Глава 6

Списки из нескольких пунктов

Часто в рукописи требуется построить перечисление, в котором каждый пункт начинается с новой строки. Например, мы могли бы отметить, что при создании текста необходимо обращать внимание на

- логику изложения;
- полноту охвата материала;
- адекватность используемых терминов;
- орфографическую грамотность текста.

Вышеприведенный текст и есть пример *ненумерованного перечисления*, которое содержит четыре пункта.

6.1. Простые перечисления

Для включения в текст простого (ненумерованного) перечисления необходимо воспользоваться окружением `itemize`; иначе говоря, начало перечисления обозначается командой `\begin{itemize}`, а конец — командой `\end{itemize}`. Начало каждого пункта (в том числе первого) отмечается командой `\item`. Например, перечисление, приведенное выше, набрано так:

```
необходимо обращать внимание на
\begin{itemize}
  \item логику изложения;
```

```
    \item полноту охвата материала;
    \item адекватность используемых терминов;
    \item орфографическую грамотность текста.
  \end{itemize}
```

В любом перечислении обязана присутствовать хотя бы одна команда `\item`, иначе произойдет ошибка.

Перечисления могут быть вложены одно в другое, то есть можно вставить внутрь окружения `itemize` еще одно такое окружение. Для примера заметим, что среди живых организмов выделяют

- растения, в том числе
  - одноклеточные растения,
  - травы,
  - деревья, среди которых можно назвать
    - сосны,
    - березы,
    - клены и т. п.,
  - кустарники;
- животные, включая
  - простейших,
  - рыб,
  - земноводных,
  - птиц,
  - млекопитающих;
- грибы, которые не относятся ни к растениям, ни к животным.

При наборе этих вложенных перечислений никаких новых команд мы не использовали. LaTeX автоматически сдвигает содержимое вложенного перечисления относительно предыдущего и меняет символ, вставляемый перед каждым пунктом. На всякий случай отметим, что использовать вложенность более чем на четыре уровня глубины все же не стоит.

6.2. Перечисления с нумерацией

Можно заставить LaTeX нумеровать пункты перечисления; для этого достаточно заменить слово `itemize` на слово `enumerate`. Вспомним, например, что множество называется кольцом, если на его элементах определены операции сложения и умножения, причем

1. сложение коммутативно: $a + b = b + a$;
2. сложение ассоциативно: $a + (b + c) = (a + b) + c$;
3. умножение коммутативно: $ab = ba$;
4. умножение ассоциативно: $a(bc) = (ab)c$;
5. выполняется дистрибутивность: $(a + b)c = ac + bc$;
6. сложение имеет обратную операцию (вычитание), не выводящую за пределы рассматриваемого множества.

Это перечисление набрано следующим образом[1]:

```
\begin{enumerate}
  \item сложение коммутативно: $a+b = b+a$;
  \item сложение ассоциативно: $a+(b+c) = (a+b)+c$;
  \item умножение коммутативно: $ab = ba$;
  \item умножение ассоциативно: $a(bc) = (ab)c$;
  \item выполняется дистрибутивность: $(a+b)c = ac+bc$;
  \item сложение имеет обратную операцию (вычитание), не
        выводящую за пределы рассматриваемого множества.
\end{enumerate}
```

Как и в случае обычных перечислений, нумерованные перечисления могут быть вложены друг в друга.

Отметим еще один немаловажный факт: пункты нумерованных перечислений могут быть помечены командой `\label`, что дает возможность ссылаться на них с помощью команды `\ref` точно так же, как мы ранее ссылались на номера частей, глав и секций.

6.3. Как убрать интервалы между пунктами перечисления

Как видно из примеров в двух предыдущих параграфах, LaTeX оставляет между пунктами перечислений достаточно большие интервалы. Чаще всего это нас устраивает, но бывает и так, что промежутки нежелательны; особенно это справедливо для обычных перечислений. Полностью искоренить интервалы, сохранив всю функциональность вышеописанных средств, оказывается сравнительно сложно; в этом параграфе мы

[1] Знак «`$`» означает переход в математический режим; подробнее об этом — в главе 10.

расскажем, как сделать свое собственное простейшее окружение, позволяющее создавать списки без нумерации, в которых интервалы между пунктами точно такие же, как и между обычными строками.

Итак, вставьте в преамбулу вашего документа следующий текст:

```
\newenvironment{compactlist}{
  \begin{list}{{$\bullet$}}{
    \setlength\partopsep{0pt}
    \setlength\parskip{0pt}
    \setlength\parsep{0pt}
    \setlength\topsep{0pt}
    \setlength\itemsep{0pt}
  }
}{
  \end{list}
}
```

Теперь вы можете использовать в документе слово `compactlist` вместо `itemize`. Никакой пустоты между пунктами LaTeX в таком перечислении не вставит, поскольку все составляющие этих отступов установлены равными нулю. Основным недостатком такого окружения является его неспособность к изменению символа, вставляемого перед пунктами перечисления, в зависимости от уровня вложенности; этот недостаток можно исправить, но описание окружения станет при этом гораздо сложнее.

Если заменить в одном из вышеприведенных примеров `itemize` на `compactlist`, результат будет выглядеть так:

- растения, в том числе
  - одноклеточные растения,
  - травы,
  - деревья, среди которых можно назвать
    - сосны,
    - березы,
    - клены и т. п.,
  - кустарники;
- животные, включая
  - простейших,
  - рыб,
  - земноводных,
  - птиц,
  - млекопитающих;
- грибы, которые не относятся ни к растениям, ни к животным.

Глава 7

Таблицы и рисунки

7.1. Плавающие объекты

7.1.1. Общие сведения о плавающих объектах

Когда в документ требуется вставить рисунок или таблицу, обычно в тексте ставится ссылка (что-то вроде «см. рис. 2» или «см. табл. 8»), а сами рисунки и таблицы размещаются вверху или внизу текущей страницы или даже на другой странице, в зависимости от того, как удобнее набирать рукопись. Зачем это делается, понять легко: достаточно представить себе, что вам потребовалось вставить рисунок высотой в полстраницы, а страница уже на две трети заполнена (то есть места для рисунка не хватит). Если попытаться сохранить жесткую привязку расположения рисунка к тексту, остаток страницы придется оставить пустым, что нежелательно.

Кроме того, сверстанный документ лучше смотрится, если таблицы и рисунки размещаются всегда в верхней части страницы и занимают не более двух третей ее высоты; рисунки и таблицы, имеющие бóльшую высоту, принято выносить на отдельные страницы, не содержащие текста.

LaTeX позволяет описывать объекты, которые следует разместить поблизости от места в тексте, куда вставлено их описание, но конкретное размещение можно выбрать, руководствуясь соображениями удобства и эстетичности. Такие объекты называются *плавающими*; мы будем рассматривать только два их вида — таблицы (tables) и рисунки (figures). LaTeX позволяет вводить и новые виды плавающих объектов, причем некоторые дополнительные пакеты именно это и делают; подробности, однако, выходят за рамки конспективного введения.

Очень важно понять, что сам по себе плавающий объект — это нечто, имеющее заголовок и номер; конкретное содержимое объекта, то есть то,

что должно быть напечатано, можно сделать какое угодно (например, можно в качестве рисунка или таблицы оформить обычный абзац текста, или даже разместить таблицу в объекте рисунка и наоборот, хотя делать так, конечно же, не следует). Деление на таблицы и рисунки обусловлено прежде всего тем, что нумерация у каждого типа плавающих объектов своя.

7.1.2. Описание плавающих объектов

Для описания плавающих объектов используются окружения `table` (для таблиц) и `figure` (для рисунков). Плавающие объекты обоих видов следует снабжать названиями, для чего используется команда `\caption`, например:

```
\begin{table}
    % ... содержание таблицы ...
    \caption{Удельный вес некоторых веществ}
\end{table}
```

или

```
\begin{figure}
    % ... содержание рисунка ...
    \caption{Электрическая схема усилителя}
\end{figure}
```

Как задавать содержание таблиц и рисунков, мы обсудим в следующих параграфах.

Отметим, что команду `\caption` можно дать как до, так и после описания содержимого плавающего объекта. От этого зависит, будет ли заголовок таблицы или подпись к рисунку располагаться сверху или снизу от самого объекта. Обычно подписи размещают снизу (то есть команду `\caption` вставляют после описания содержимого). Если вы примете решение делать подписи сверху, рекомендуем хотя бы придерживаться этого решения во всем документе.

7.1.3. Ссылки на плавающие объекты

Задав окружение `figure` или `table`, мы получим плавающий объект, снабженный подписью и номером; например, приведенный в предыдущем параграфе код плавающей таблицы сгенерирует примерно такой заголовок:

Таблица 2.7: Удельный вес некоторых веществ

Номер присваивается плавающему объекту автоматически. В рассматриваемом классе `extreport` номер таблицы или рисунка состоит из двух частей: номера главы и порядкового номера плавающего объекта данного типа в рамках этой главы. Таким образом, в нашем примере 2.7 означает седьмую таблицу во второй главе.

Разумеется, упоминать полученный номер в исходном тексте в явном виде было бы крайне неудобно. Например, при редактировании рукописи мы можем вставить во вторую главу еще одну таблицу, в результате чего номера таблиц изменятся, и если бы мы ссылались на них явно (набрали бы в тексте что-то вроде `см. табл. 2.7`), пришлось бы просматривать исходный текст и исправлять все такие номера. Поэтому для формирования ссылки на плавающий объект следует воспользоваться командами `\label` и `\ref`, уже знакомыми нам по § 3.3. Если команду `\label` поместить в описание плавающего объекта (причем обязательно **после** команды `\caption`), введенная метка окажется связана с номером данного объекта и страницей, на которой объект в итоге окажется размещен. Итак, для описания таблицы используем примерно такой фрагмент исходного текста:

```
\begin{table}
    % ... содержание таблицы ...
    \caption{Удельный вес некоторых веществ}
    \label{some_densities}
\end{table}
```

Теперь сослаться на таблицу можно будет, например, так:

```
см.~табл.\,\ref{some_densities} на
стр.\,\pageref{some_densities}
```

7.1.4. Управление размещением объектов

В начале окружения, задающего плавающий объект, можно указать дополнительный параметр, информирующий LaTeX о наших пожеланиях по поводу размещения данного объекта. Для этого к команде `\begin{figure}` или `\begin{table}` добавляются квадратные скобки, содержащие одну или несколько из следующих букв: `t` означает размещение в верхней части страницы (от слова top), `b` — в нижней части страницы (bottom), `p` — на отдельной странице, специально выделенной под плавающие объекты (page), и `h` — сразу же после появления в исходном тексте (here). Пожелание можно превратить в требование, добавив восклицательный знак; LaTeX сделает в этом случае все, что может, но, к сожалению, нужного размещения все равно не гарантирует. Можно указать несколько возможностей в порядке убывания их желательности. Например, команда

`\begin{figure}[hp]`

означает, что данный рисунок хотелось бы увидеть размещенным непосредственно в том месте текста, где он описан, если же это не получится — то вынести его на отдельную страницу.

Учтите, что LaTeX предпочитает размещать объекты в верхней части страницы (как мы уже говорили, это смотрится более эстетично), а размещать объект в верхней части *текущей* страницы может оказаться уже поздно, если к моменту появления вашего объекта на текущей странице осталось мало места. Так что в большинстве случаев ваш плавающий объект появится только на следующей странице, что не всегда хорошо: поставьте себя на место читателя, который вынужден переворачивать страницу туда и обратно, чтобы сверять текст с иллюстрацией. С этим можно справиться, перенеся описание плавающего объекта на несколько абзацев выше того места, где он нам потребуется, то есть «скормив» объект LaTeX'у несколько заранее. Именно так следует поступить, если хочется видеть рисунок или таблицу в верхней части той страницы, на которой данный объект упоминается.

7.2. Таблицы

В предыдущем параграфе мы рассматривали только оформление таблиц и рисунков в качестве плавающих объектов, не обсуждая при этом, как задать сами рисунки и таблицы. Рассмотрим теперь верстку таблиц как таковых.

Таблицы создаются с помощью окружения `tabular`. В начале окружения (сразу после команды `\begin{tabular}`) необходимо поместить описание столбцов таблицы. Описание представляет собой несколько символов, заключенных в фигурные скобки; символ «`l`» соответствует столбцу, в котором текст выравнивается по левому краю, символ «`r`» — столбцу с выравниванием по правому краю, символ «`c`» — столбцу, содержимое которого центрируется. Столбцы этих трех видов будут иметь такую ширину, чтобы содержимое каждой из строк таблицы умещалось в соответствующих столбцах в одну строчку. Символ «`|`» (вертикальная черта) обозначает вертикальную линию между столбцами. Наконец, в качестве описания столбца можно задать комбинацию `p{`<*ширина*>`}`, которая будет обозначать столбец заданной ширины. В таком столбце текст форматируется в виде абзаца (возможно, в несколько строк). Ширину можно задать в сантиметрах (`cm`), миллиметрах (`mm`) или пунктах (`pt`); LaTeX понимает и другие единицы измерения, но перечисленных обычно хватает. Например, `p{11.5cm}` означает столбец шириной 11,5 см. Часто бывает удобно привязать ширину столбца к общей ширине полосы набора; это

```
\begin{table}[t]
\begin{tabular}{|r|c|p{0.15\textwidth}|
                p{0.25\textwidth}|c|r|}
  \hline \No & Тип & Автор & Заглавие & Год & Тир. \\ \hline
  1 & книга & Артур Конан Дойл & Собака Баскервилей
    & 1975 & 10\,000 \\ \hline
  2 & книга & Жюль Верн & Пять недель на воздушном шаре
    & 1981 & 7000 \\ \hline
  3 & журнал & \multicolumn{2}{c|}{Вокруг света (\No 5)}
    & 1995 & 5000 \\ \hline
\end{tabular}
\caption{Пример таблицы}
\label{sample_table}
\end{table}
```

Рис. 7.1: Пример кода таблицы

можно сделать, например, так: `p{0.7\textwidth}` (здесь описан столбец, занимающий 70% от общей ширины текста на странице).

Приведем пример заголовка окружения `tabular`:

```
\begin{tabular}{|r|l|c|p{4cm}|}
   % ... строки таблицы ...
```

Здесь мы описали таблицу из четырех столбцов, разделенных вертикальными линиями, причем в первом из столбцов производится выравнивание по правому краю (это удобно, например, для порядковых номеров и других чисел), во втором — выравнивание по левому краю, в третьем — центрирование, четвертый же представляет собой столбец фиксированной ширины (4 см), в котором возможно многострочное содержание в отдельно взятой клетке.

После заголовка помещаем содержание таблицы. Каждая строка таблицы задается обычным текстом, в котором символ «&» (амперсанд) играет роль разделителя полей, а знак принудительного разрыва строки («\\») обозначает конец строки таблицы.

Чтобы разделить строки таблицы горизонтальной чертой, следует воспользоваться командой `\hline`. При необходимости можно провести горизонтальную черту не через все столбцы, а только через некоторое их подмножество. Это делается командой `\cline`: например, `\cline{2-4}` проведет горизонтальную черту через столбцы со второго по четвертый.

Иногда бывает полезна команда `\multicolumn`, позволяющая в отдельно взятой строке объединить несколько столбцов в один. Эта команда имеет три параметра: первый из них — количество столбцов, второй

№	Тип	Автор	Заглавие	Год	Тир.
1	книга	Артур Конан Дойл	Собака Баскервилей	1975	10 000
2	книга	Жюль Верн	Пять недель на воздушном шаре	1981	7000
3	журнал	Вокруг света (№5)		1995	5000

Таблица 7.1: Пример таблицы

задает расположение текста аналогично тому, как это делается в заголовке окружения `tabular`, третий — собственно текст, который следует поместить в получившуюся клетку таблицы.

На рис. 7.1 приведен пример исходного текста таблицы вместе с описанием плавающего объекта[1]. Результат интерпретации этого кода — таблица 7.1.

7.3. Рисунки

Существует два основных способа создания рисунков: встроенная графика LaTeX (при этом содержимое рисунка описывается командами) и включение внешних графических файлов, подготовленных внешними средствами (например, с помощью графического редактора). Первый способ позволяет рисовать только примитивные диаграммы; второй таких ограничений не имеет, но требует сравнительно сложного взаимодействия с программами обработки изображений.

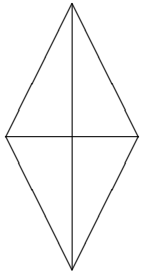
7.3.1. LaTeX-графика

Задействовать встроенные в LaTeX графические возможности позволяет окружение `picture`:

```
\begin{picture}(100,200)(15,-20)
   % ... содержимое рисунка
\end{picture}
```

Первая пара координат (в этом примере `(100,200)`) задает размер картинки, вторая — сдвиг начала координат (точнее, этот параметр задает значения координат, которые будут приписаны левому нижнему углу картинки). Изначально сдвиг рекомендуется указать нулевой или вообще не указывать, и воспользоваться им, если в процессе редактирования

[1] В принципе, можно и не объявлять таблицу плавающим объектом, тогда она будет вставлена в документ непосредственно в месте своего появления и не будет иметь ни номера, ни названия; именно поэтому делать так не рекомендуется.



```
\begin{picture}(60,100)(0,0)
   \put( 5,50){\line( 1, 2){25}}
   \put( 5,50){\line( 1,-2){25}}
   \put( 5,50){\line( 1, 0){50}}
   \put(55,50){\line(-1, 2){25}}
   \put(55,50){\line(-1,-2){25}}
   \put(30, 0){\line( 0, 1){100}}
\end{picture}
```

Рис. 7.2: Рисунок из отрезков в LaTeX-графике

картинки возникнет потребность сдвинуть весь рисунок в ту или иную сторону.

Описание картинки производится командами, которым в качестве параметра указываются координаты. Первой всегда указывается координата x, отсчитываемая по горизонтали слева направо; второй указывается координата y, отсчитываемая по вертикали снизу вверх.

Рисунок состоит из элементов, каждый из которых задается командой `\put(`<x>`,`<y>`){`<*описание*>`}`. Координаты в этой команде задают *точку привязки* для объекта, задаваемого *описанием*; в частности, для текста точка привязки используется как координата левого нижнего угла первой буквы текста, для линии — как начальная точка и т. д.

Простейший случай элемента рисунка — это обычный текст, набираемый по обычным правилам: например, команда

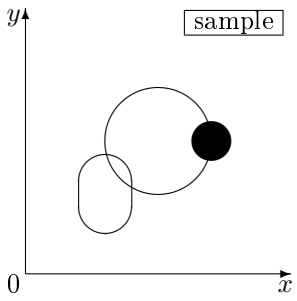
```
\put(40,100){just a text}
```

разместит на рисунке строку «just a text», начиная с точки `(40,100)`.

Следующий вид элемента — прямой отрезок, задаваемый командой `\line(`<dx>`,`<dy>`){`<*длина*>`}`. Координаты dx и dy задают наклон отрезка (выражение dy/dx принимается за тангенс угла наклона), причем оба числа должны быть целыми от -6 до 6. В частности, `\line(0,1){15}` нарисует отрезок длиной 15, направленный горизонтально слева направо от точки привязки; `\line(-1,0){25}` даст отрезок длиной 25, направленный вертикально вниз; `\line(1,1){30}` — отрезок длиной 30 под 45° к горизонтали вправо вверх и т. д.

Отметим, что понятие «длина» воспринимается LaTeX'ом весьма своеобразно: под длиной понимается на самом деле проекция на ось x, если только отрезок не вертикальный; для вертикальных отрезков (как, впрочем, и для горизонтальных) длина — это действительно длина.

С помощью команды `\linethickness` (например, `\linethickness{2mm}`) можно изменить толщину рисуемых отрез-




```
\begin{picture}(110,110)(-10,-10)
  \put(0,0){\vector(0,1){100}}
  \put(0,0){\vector(1,0){100}}
  \put(-7,-7){0}
  \put(95,-7){$x$}
  \put(-7,95){$y$}
  \put(30,30){\oval(20,30)}
  \put(50,50){\circle{50}}
  \put(70,50){\circle*{15}}
  \put(60,90){\frame{~sample~}}
\end{picture}
```

Рис. 7.3: Пример использования других элементов LaTeX-графики

ков, но повлияет это только на горизонтальные и вертикальные отрезки; толщина наклонных отрезков останется прежней.

Пример рисунка, составленного из элементов `\line`, и его исходный код приведены на рис. 7.2. Здесь мы рисуем картинку размером 60x100, оставляя слева и справа по 5 единиц свободного пространства. Первые три элемента рисуются из точки `(5,50)` (это левая вершина ромба): два наклонных отрезка длиной (точнее, проекцией на x) 25 представляют левую верхнюю и левую нижнюю стороны ромба, горизонтальный отрезок длиной 50 — короткую диагональ ромба. Следующие два элемента рисуем из точки `(55,50)` (правая вершина ромба): это правая верхняя и правая нижняя стороны. Наконец, длинную диагональ ромба рисуем из нижней его вершины (точка с координатами `(30,0)`).

Команда `\vector` аналогична команде `\line` с той разницей, что на конце отрезка будет нарисована стрелка.

Команды `\circle{`<*диаметр*>`}` и `\circle*{`<*диаметр*>`}` рисуют соответственно окружность и круг (то есть закрашенную окружность) с центром в точке привязки и заданным диаметром. Команда `\oval(`x,y`)` рисует овал (то есть две полуокружности, соединенные прямыми), причем x и y задают горизонтальный и вертикальный диаметры. Здесь необходимо отметить, что максимальный радиус окружностей в LaTeX-графике составляет 20 pt, так что окружности бóльшего радиуса нарисовать не получится, а овалы с бóльшими диаметрами будут выглядеть как прямоугольники со скругленными углами.

Также может быть полезна команда `\frame{`<*аргумент*>`}`, заключающая свой аргумент в прямоугольную рамку. Обычно эту команду используют для обрамления надписей.

На рис. 7.3 приведен еще один пример рисунка в LaTeX-графике. Здесь мы сразу сдвинули координатную сетку так, чтобы нижний левый угол

имел координаты `(-10,-10)`; это дало нам возможность нарисовать координатные оси из точки `(0,0)`, которая на нашем рисунке отстоит на 10 единиц вправо и вверх от нижнего левого угла. Первые две команды рисуют координатные оси; следующие три выводят обозначения начала координат (0) и осей (x, y). Затем мы рисуем овал, окружность и закрашенный круг, а последняя команда выдает надпись «sample», обведенную рамкой. Символы неразрываемых пробелов с обеих сторон слова «sample» понадобились, чтобы оставить немного места между текстом и рамкой слева и справа; без этого пример смотрелся достаточно некрасиво.

Отметим, что в обоих примерах мы опустили окружение `figure`, необходимое для объявления рисунка плавающим объектом. Это окружение строится точно так же, как было построено окружение `table` для таблицы в примере на стр. 47.

Как можно заметить, LaTeX-графика представляет собой примитивный инструмент, имеющий весьма неудобные ограничения. Использовать его имеет смысл только для создания простейших диаграмм; в более сложных случаях следует подготовить иллюстрацию с помощью того или иного графического редактора и вставить в документ с помощью средств, с которыми мы познакомимся в следующем параграфе.

7.3.2. Включение рисунков из внешних файлов

Существует несколько способов загрузки рисунков из внешних файлов. Мы рассмотрим только один из них, осуществляемый с помощью пакета `graphicx`.

Использование пакета `graphicx`

Сразу отметим, что файл, содержащий иллюстрацию, для обработки обычным интерпретатором LaTeX с использованием пакета `graphicx` должен быть представлен в формате Encapsulated PostScript и иметь расширение `.eps`. О том, как получить файл в таком формате, речь пойдет в следующем параграфе.

В случае, если вам захочется воспользоваться программой `pdflatex` для генерации выходного файла в формате PDF, внешние файлы иллюстраций следует подготовить либо в формате PDF, либо в одном из поддерживаемых графических форматов (например в JPG или PNG). Отметим, что использовать растровые форматы (JPG, PNG и др.) следует исключительно для фотографических изображений, а все изображения искусственного происхождения (рисунки, чертежи, диаграммы и прочее) необходимо изначально готовить с использованием векторной графики и никогда не переводить в растровые форматы.

Чтобы воспользоваться возможностями пакета `graphicx`, необходимо добавить в преамбулу вашего документа директиву подключения этого пакета:

```
\usepackage{graphicx}
```

Для включения в документ рисунка из внешнего файла воспользуйтесь командой `\includegraphics[`<*опции*>`]{`<*имя файла*>`}`. Имя файла указывается **без расширения**, что позволяет использовать один и тот же исходный текст без изменений как с обычным LaTeX'ом, так и с программой `pdflatex`; в первом случае система попытается найти файл с расширением `.eps` или `.ps`, во втором — с расширением `.png`, `.pdf`, `.jpg`, `.mps` или `.tif`. Команда поддерживает достаточно большое количество опций, среди которых одна из наиболее важных — опция `width`, задающая ширину, которую будет иметь вставляемый рисунок в вашем документе. В отсутствие других опций рисунок будет соответствующим образом (без изменения пропорций) масштабирован.

Допустим, вам нужно включить в документ изображение (например, чертеж) грузовика, и вы подготовили файлы с этим изображением в форматах EPS (`truck.eps`) и PDF (`truck.pdf`). Из эстетических соображений вы решили, что в ширину рисунок должен занимать 90% полосы набора. Тогда соответствующий код, включая обрамление плавающего объекта, будет выглядеть следующим образом:

```
\begin{figure}[t]
  \centering
  \includegraphics[width=0.9\textwidth]{truck}
  \caption{Грузовик}
  \label{truck_figure}
\end{figure}
```

Подготовка файлов иллюстраций в ОС Unix

Рисовать схемы и диаграммы для включения их в документ LaTeX рекомендуется с помощью графического редактора `XFig`. Программа `XFig` входит в поставку большинства дистрибутивов GNU/Linux либо может быть получена в сети Internet по адресу `ftp://www-epb.lbl.gov/xfig`.

Интерфейс `XFig` достаточно понятен на интуитивном уровне. При необходимости можно обратиться к документации, которая всегда поставляется вместе с программой и может быть просмотрена через ее меню.

Если при использовании `XFig` вы не видите или не можете набирать русские буквы, это означает, что у вас неправильно

выставлены переменные окружения, отвечающие за локализацию. Попробуйте запустить `XFig` следующей командой[2]:

```
LANG=ru_RU.KOI8-R xfig myfile.fig
```

Программа `XFig` сохраняет результаты в файле своего собственного формата с расширением `.fig`; сконвертировать такой файл в формат `.eps` можно с помощью программы `fig2dev`, которая обычно поставляется вместе с `XFig`. Командная строка, которая, скорее всего, сделает все, что нужно, выглядит примерно так:

```
LANG=ru_RU.KOI8-R fig2dev -j -L eps\
                myfile.fig myfile.eps
```

Подробности читатель может узнать самостоятельно из документации к программе `fig2dev`.

Если необходим также файл в формате PDF, его обычно получают уже из файла `.eps` с помощью программы `epstopdf`:

```
epstopdf myfile.eps
```

Наконец, если ваша иллюстрация изначально представлена в формате PNG или JPG (например, если это фотография), для `pdflatex` оставьте ее как есть[3], а для обычного LaTeX'а создайте версию в формате EPS с помощью программы `convert`:

```
convert myphoto.jpg myphoto.eps
convert mypicture.png mypicture.eps
```

Подготовка файлов иллюстраций в ОС Windows

Для Windows существует широкий спектр программ, работающих с графическими файлами, однако порекомендовать какие-либо конкретные программы оказывается несколько затруднительно. В первую очередь это обусловлено традициями мира Windows, где за использование большинства программ требуется так или иначе заплатить деньги, иногда весьма значительные.

Для рисования диаграмм можно посоветовать аналог `XFig`, который называется `WinFIG` и распространяется бесплатно. Программа `WinFIG` может быть загружена с сайта `http://www.schmidt-web-berlin.de/WinFIG.htm`.

[2]Здесь и далее предполагается, что в вашей операционной среде используется кодировка koi8-r и что вы используете командный интерпретатор `sh` или `bash`. Если вы используете кодировку utf-8, то переменной окружения `LANG` следует задать значение `ru_RU.UTF-8`. В командных интерпретаторах, отличных от классического Bourne Shell, задание значения переменной окружения может выглядеть не так, как это показано здесь; обратитесь к документации на ваш командный интерпретатор.

[3]Если ваше изображение имеет другой формат (например, GIF), обязательно сконвертируйте его в PNG или JPG.

Глава 8

Оформляем листинги программ

Если в вашей работе фигурируют фрагменты компьютерных программ, конфигурационных файлов к ним или вообще текстов на формальных языках (включая, например, HTML или тот же LaTeX), к их оформлению необходим особый подход.

В отличие от обычных текстов, тексты на формальных языках принято набирать моноширинным шрифтом (то есть шрифтом, в котором все символы имеют одинаковую ширину). Тексты на формальных языках обычно имеют специфический стиль форматирования, основанный на количестве пробелов в начале каждой строки; сами строки тоже требуют внимания, поскольку произвольное изменение разбивки на строки может исказить смысл такого текста или сделать его нечитаемым. Наконец, в текстах на формальных языках часто встречаются символы, которые LaTeX воспринимает особым образом. Конечно, средствами LaTeX'а любой из этих символов так или иначе можно воспроизвести, но когда, например, в листингах постоянно встречается знак «$» (а это именно так, скажем, для программы на языке Perl, где с «доллара» начинаются имена переменных), каждый раз экранировать этот знак обратной косой чертой быстро надоедает, не говоря уж о том, что во многих языках программирования фигурные скобки (имеющие, как мы помним, специальное значение в LaTeX'е) используются едва ли не в каждой строке.

8.1. Окружение `verbatim` и команда `verb`

Простейший способ справиться с этими проблемами — это воспользоваться возможностью «воспроизведения как есть». Набрать много-

строчный фрагмент компьютерной программы или другого текста на формальном языке можно с помощью окружения `verbatim`. Начиная с команды `\begin{verbatim}` и до тех пор, пока не встретится строка `\end{verbatim}` (важно помнить, что в ней не допускаются пробелы), LaTeX все встреченные символы будет выдавать моноширинным шрифтом (то есть шрифтом из гарнитуры Typewriter, см. стр. 21), причем никакие символы (кроме комбинации `\end{verbatim}`, если она встретится целиком) не будут интерпретироваться никаким специальным образом. Каждый встреченный пробел, в том числе и в начале строки, будет восприниматься именно как пустое место шириной как у обычного символа; символ конца строки будет вызывать перевод строки, как при печати обыкновенного текстового файла. К примеру, простейшую программу на Паскале можно было бы оформить так:

```
\begin{verbatim}
   program example;
   begin
     { This program just prints a message }
     writeln('Hello world');
   end.
\end{verbatim}
```

Часто бывает нужен не многострочный фрагмент, а, напротив, фрагмент, существенно меньший одной строки; например, при ссылках на упоминаемые в листингах идентификаторы и другие слова принято эти слова для наглядности набирать все тем же моноширинным шрифтом. Это можно сделать с помощью команды установки моноширинного шрифта `\texttt`, например, так:

```
Процедура \texttt{writeln} осуществляет вывод
```

Может, однако, возникнуть необходимость напечатать небольшой фрагмент кода на формальном языке, содержащий специальные символы. Как, например, набрать что-то вроде «выражение `(a[~'$']&b)^c` является в языке Си вполне корректным»? Громоздкое и непонятное

```
\texttt{(a[\textasciitilde{}'\$']\&b)\textasciicircum{}c}
```

выглядит не слишком привлекательной альтернативой.

Предназначенная для таких ситуаций команда `\verb` несколько отличается от всех, которые мы обсуждали раньше. Дело в том, что у обычных команд, как правило, имеются заранее заданные символы для ограничения аргумента (в большинстве случаев это символ закрывающей фигурной скобки), но для команды `\verb` такой способ не годится: она должна любой символ, включая и закрывающую скобку, уметь

воспринимать как обычный, а не специальный символ. Поэтому символ-ограничитель мы задаем сами для каждой конкретной команды `\verb`; можно использовать любой символ, кроме латинских букв и «звездочки» (`*`). Команда считывает символ, стоящий непосредственно после нее, и именно этот символ использует как признак конца аргумента. Так, при воспроизведении вышеприведенного выражения из языка Си мы заметили, что в это выражение не входит символ двоеточия, что позволило нам применить следующую команду: `\verb:(a[~'$']&b)^c:`.

К сожалению, команду `\verb` нельзя использовать в аргументах других команд, в частности, в заголовках рубрик, в сносках и т. п. Кроме того, следует помнить, что содержимое команды `\verb` не может быть частично перенесено на следующую строку, так что употребление этой команды с длинными аргументами заставляет LaTeX выдавать абзацы весьма уродливого вида. Можно рекомендовать, во-первых, по возможности использовать `\texttt`, и, во-вторых, более-менее длинные фрагменты выносить в отдельные строки, используя окружение `verbatim`.

Окружение `verbatim` и команда `\verb` имеют альтернативную «звездочную» форму, при использовании которой символ пробела превращается в знак «␣». Например, если в вышеприведенном примере добавить в имя окружения символ «*», то есть использовать команды `\begin{verbatim*}` и `\end{verbatim*}`, то напечатано будет следующее:

```
␣␣␣program␣example;
␣␣␣begin
␣␣␣␣␣{␣This␣program␣just␣prints␣a␣message␣}
␣␣␣␣␣writeln('Hello␣world');
␣␣␣end.
```

8.2. Средства из дополнительных пакетов

Прежде всего упомянем пакет, который так и называется `verbatim`. Его основное достоинство — команда `\verbatiminput{`<*имя файла*>`}`, позволяющая прочитать текст будущего листинга прямо из файла (например, из исходного файла вашей программы).

Если подключить пакет `moreverb` (то есть вставить в преамбулу директиву `\usepackage{moreverb}`), в ваше распоряжение поступят еще несколько окружений, предназначенных для работы с листингами программ. Одно из них так и называется `listing`; оно автоматически нумерует строки в листинге. Заголовок окружения выглядит так: `\begin{listing}[`<*шаг*>`]{`<*старт*>`}`. Параметр *старт* задает номер первой строки, а необязательный параметр *шаг* позволяет показывать номер не перед каждой строкой, а, например, перед каждой пятой или десятой.

В этом же пакете описана удобная команда `\listinginput[`<*шаг*>`]{`<*старт*>`}{`<*имя файла*>`}`, позволяющая напечатать с нумерацией строк текст из файла.

И команда `\listinginput`, и окружение `\listing` имеют «звездочные» формы, при использовании которых пробел показывается как знак «␣».

Наконец, нельзя не упомянуть пакет `listings`, специально предназначенный именно для оформления листингов компьютерных программ. Возможности этого пакета весьма обширны; в частности, пакет содержит сведения о лексическом и синтаксическом составе нескольких десятков языков программирования и способен формировать листинги с выделением ключевых слов и других синтаксических элементов. Учитывая конспективный характер пособия, подробное описание возможностей пакета `listings` мы приводить не будем; при необходимости читатель может освоить этот пакет самостоятельно, обратившись к соответствующей документации.

Глава 9

Сноски и заметки на полях

9.1. Сноски

Сноски оформляются с помощью команды `\footnote{`<*текст*>`}`. В результате выполнения этой команды в том месте, где она встретилась, появляется номер сноски, набранный цифрами уменьшенного размера, сдвинутыми вверх; текст сноски, снабженный тем же номером, обычно размещается в нижней части страницы. От основного текста сноски отделяются горизонтальной чертой.

В рамках каждой главы сноски имеют сквозную нумерацию; при смене главы (то есть по команде `\chapter`) счетчик сносок сбрасывается, так что первая сноска каждой главы имеет номер 1.

Сноску можно пометить командой `\label` (например, `\label{my_footnote}`), вставив ее в текст сноски. Это позволит где-то в другом месте написать что-то вроде

```
См.\,сноску\,\ref{my_footnote}
на стр.\,\pageref{my_footnote}
```

Бывает так, что текст сноски не помещается целиком на текущей странице. Это может произойти, например, если вы потребуете от LaTeX'а сгенерировать сноску, когда до конца страницы осталась одна строка, а текст сноски при этом в одну строку не поместится. В этом случае часть текста сноски будет перенесена в нижнюю часть следующей страницы.

Если команда `\footnote` появится в процессе верстки последней строки на странице (то есть когда на текущей странице невозможно разместить даже одну строку сноски), на следующую страницу будет перенесена вся эта строка вместе со сноской, то есть текущая страница окажется недозаполненной. Обычно это невооруженным глазом не заметно, так

что в большинстве случаев можно ставить сноски, не задумываясь об их размещении.

Иногда возникает желание сослаться на одну и ту же сноску несколько раз (обычно на одной странице). Для этого первое вхождение сноски следует оформить как обычно и снабдить меткой с помощью команды `\label{my_footnote}`, а в остальных местах применить команду `\footnotemark[\ref{my_footnote}]`.

9.2. Текст и знаки на полях

С помощью команды

`\marginpar[`<*левая версия*>`]{`<*текст*>`}`

можно вынести текст на поля документа, то есть за пределы полосы набора. При односторонней печати используется поле справа, при двухсторонней — справа на нечетных страницах, слева на четных.

Параметр *левая версия* можно опустить вместе с квадратными скобками; в этом случае на поля будет выноситься текст, заданный параметром *текст*, вне зависимости от того, располагаются эти поля в текущей ситуации слева или справа от текста. Если параметр *левая версия* все же задать, то для выноса вправо (то есть если команда встретилась на нечетной странице или при односторонней печати) будет использоваться *текст*, а для выноса влево (то есть если команда встретилась при двухсторонней печати на четной странице) — *левая версия*.

Чтобы комментарии на полях выглядели сколько-нибудь уместно, сами поля должны быть достаточно широкими. В западной полиграфической традиции часто специально оставляют широкие поля, чтобы выносить на них комментарии. В отечественной полиграфической традиции это не принято.

Кроме того, при попытке применения команды `\marginpar` вы можете с удивлением обнаружить, что LaTeX почему-то считает вашу бумагу несколько шире, чем она есть, в результате чего изрядная часть комментария на полях оказывается за пределами физического листа бумаги (то есть, попросту говоря, теряется). Справиться с этим, разумеется, можно, однако результат обычно не стоит потраченных усилий.

Единственное разумное применение команды `\marginpar` состоит в размещении на полях небольших пиктограмм, привлекающих внимание к особо важным местам текста, подобно той, что стоит на полях возле следующего абзаца (треугольник с восклицательным знаком).

Эту пиктограмму мы сгенерировали командой `\attention`, которая описывается следующим кодом:

```
\newcommand\attentionpicture{
   \begin{picture}(20,20)(0,0)
      \put(0,0){\line(1,0){20}}
      \put(0,0){\line(2,3){10}}
      \put(20,0){\line(-2,3){10}}
      \put(8.3,3){\bf !}
   \end{picture}
}
\newcommand\attention{
   \marginpar[\hfill\attentionpicture]
             {\attentionpicture}
}
```

Напомним, что команда `\newcommand` задает новые команды (см. § 5.1). Сначала мы ввели команду `\attentionpicture`, которая рисует пиктограмму (картинка 20x20, треугольник из трех отрезков и восклицательный знак в середине; подробности см. в § 7.3.1 на стр. 48). На основе этой команды мы сделали саму команду `\attention`, которая использует `\marginpar` для вынесения пиктограммы на поля. Отметим, что при выносе пиктограммы вправо никаких особых проблем не возникает, тогда как при выносе влево она оказывается далеко от текста, будучи прижата к воображаемой левой границе физического листа бумаги; поскольку, как мы уже говорили, эта граница оказывается дальше границы реальной, часть пиктограммы оказывается за пределами листа. Чтобы этого избежать, мы указали параметр *левая версия*, который использует «горизонтальную пружину» (команду `\hfill`), чтобы «прижать» пиктограмму максимально вправо.

Глава 10

Как набирать математические формулы

10.1. Основы математического режима

Для набора математических формул LaTeX имеет специальный *математический режим*, называемый также *математической модой*. Этот режим существенно отличается от обычного. Обычные буквы в математическом режиме LaTeX воспринимает как имена математических переменных и набирает их шрифтом, специально предназначенным для формул; пробелы в этом режиме нужны разве что в качестве ограничителей имен команд, поскольку расстояния между элементами формул LaTeX выбирает в соответствии со своими правилами, полностью игнорируя пробельные символы (то есть, например, набрав в математическом режиме «`a b c`», мы получим абсолютно тот же результат, как если бы мы набрали «`abc`»). Большинство команд обычного режима в математическом недоступно, зато появляется достаточно широкий набор команд, специально предназначенных для верстки формул, с помощью которых можно задать практически любое математическое выражение, в том числе содержащее простые дроби, верхние и нижние индексы, знаки интегрирования, суммирования, произведения, матрицы, системы и совокупности уравнений и неравенств, радикалы и другие традиционные математические символы. Команды математического режима настолько просты, что многие математики пользуются ими вместо традиционных формул, общаясь на форумах в Интернете, где, как правило, набирать формулы в традиционном виде затруднительно. Например, в текстах на математических интернет-форумах можно встретить что-то

вроде `\int_0^{2\pi}{\frac{\sin x}{x}dx}`, что означает

$$\int_0^{2\pi} \frac{\sin x}{x} dx$$

Поясним, что команда `\int` в математическом режиме дает знак интеграла, символы «`_`» (подчеркивание) и «`^`» (крышка) обозначают нижний и верхний индексы (если индексное выражение состоит более чем из одного символа, его следует взять в фигурные скобки), `\frac{`<*числитель*>`}{`<*знаменатель*>`}` служит для набора дробей, а команда `\pi` выдает, как нетрудно догадаться, греческую букву π.

Поскольку пробелы в математическом режиме игнорируются, для набора любых многобуквенных обозначений приходится применять команды. Так, для обозначения функции $\sin x$ применяется команда `\sin`, а если написать просто `sin x`, LaTeX воспримет это как произведение четырех переменных: $sinx$.

Команды математического режима LaTeX, составляющие формулу, необходимо заключать в одно из трех окружений `math`, `displaymath` или `equation`; первое из них вставляет формулу непосредственно в текст, второе располагает формулу на отдельной строке (как это сделано в вышеприведенном примере с интегралом), а третье также располагает формулу в отдельной строке, но, кроме того, еще и снабжает ее номером, на который можно сослаться. Формулы, вынесенные в отдельные строки, печатаются по центру текста, а номера — вдоль правого края.

Поскольку окружения `math` и `displaymath` используются достаточно часто, для их обозначения есть сокращенные формы. Вместо

```
\begin{math} ... \end{math}
```

можно написать `\( ... \)` или даже `$ ... $`, а вместо

```
\begin{displaymath} ... \end{displaymath}
```

можно использовать `\[ ... \]` или `$$ ... $$`. Отметим, что вариант со знаками «доллара» является самым старым и имеет один серьезный недостаток: начало конструкции ничем не отличается от ее конца. Если сделать ошибку и пропустить один из символов, открывающий либо закрывающий, то выданная LaTeX'ом диагностика может оказаться совершенно непонятной. Поэтому рекомендуется использовать `\( ... \)` для строчных формул и `\[ ... \]` для выносных.

Пример, иллюстрирующий использование строчных и выносных формул, приведен на рис. 10.1.

Обратите внимание в этом примере на символы «`\,`» в исходном тексте выносной формулы перед dx. Команда `\,` вставляет в формулу «тонкий пробел», благодаря чему dx оказывается зрительно отделено от интегрируемого выражения, как это обычно и делается; если убрать эту команду, формула будет выглядеть несколько непривычно.

```
Если функция \( F(x) \)
является одной из
первообразных функции
\( f(x) \) на интервале
\( (a, b) \), то
\[\int f(x)\,dx=F(x)+C,\]
где \(C\)~---
произвольная постоянная.
```

Если функция $F(x)$ является одной из первообразных функции $f(x)$ на интервале (a, b), то

$$\int f(x)\,dx = F(x) + C,$$

где C — произвольная постоянная.

Рис. 10.1: Пример использования строчных и выносных формул

Для расстановки пробелов можно также использовать команды `\:`, `\;`, `\quad` и `\qquad`. Кроме того, существуют команды «отрицательных пробелов»: `\negthinspace`, `\negmedspace` и `\negthickspace`; первая из них имеет синоним `\!`:

```
x\negthickspace x   xx
x\negmedspace x     xx
x\!x                xx
xx                  xx
x\,x                x x
x\:x                x x
x\;x                x x
x\quad x            x  x
x\qquad x           x    x
```

Важно понимать, что при появлении в тексте любых переменных, функций и других сущностей, упоминаемых в формулах, они обязательно должны быть сами оформлены как формулы. Даже если вам надо всего лишь упомянуть переменную x, необходимо написать `\(x\)` или `$x$`, но не просто `x`. Дело тут в том, что шрифты, используемые в математическом режиме, существенно отличаются от обычных шрифтов, и ваши символы будут в обычном тексте выглядеть совсем не так, как в формулах — если, конечно, их не оформить как формулы.

Если в вашем документе используются формулы, рекомендуется подключить пакет `amsmath`, то есть вставить в преамбулу команду `\usepackage{amsmath}`. Этот пакет содержит много удобных средств для верстки формул.

В качестве примера такого средства приведем команду `\text`, с помощью которой можно при необходимости вставить в формулу фрагмент обычного текста. Чаще всего это используется при формировании индексов переменных. Например, формула

$$V_{\text{сближения}} = V_{\text{автомобиля}} + V_{\text{велосипедиста}}$$

может быть получена с помощью

```
\[ V_\text{сближения} = V_\text{автомобиля} +
                        V_\text{велосипедиста} \]
```

Далее в этой главе мы предполагаем, что пакет `amsmath` подключен, и никак не оговариваем необходимость его подключения для использования некоторых команд.

10.2. Дроби и корни

В те времена, когда рукописи печатались на пишущей машинке, для математических формул приходилось оставлять место, а потом вписывать их от руки. Не в последнюю очередь причиной этого были всевозможные конструкции типа простых дробей, радикалов, интегралов и т. п., объединенные одним свойством: в них используются символы, которые никак не укладываются в скучную машинопись постоянного размера, а вокруг этих символов в самых неожиданных местах располагается текст.

Дроби типа $\frac{x}{2}$ иногда еще пытались набирать в три строчки (сверху x, в самой строке тире, внизу двойка), хоть и выглядело это, прямо скажем, отвратительно. Но вот если попадалось какое-нибудь $\frac{x+\frac{1}{y}}{\frac{z+1}{3}-15}$ — тут уж ничего не оставалось, кроме как пропустить нужное количество места и потом, допечатав очередной лист, вооружаться пером или рейсфедером и вписывать «проклятую» формулу от руки. Набирать на пишущей машинке что-нибудь вроде $\sqrt[3]{a+b}$ никто и не пытался: в самом деле, а как?

Как мы уже упоминали в примерах, для набора дробей служит команда `\frac{`*числитель*`}{`*знаменатель*`}`. В частности, дробь из предыдущего абзаца набрана так:

```
\( \frac{x+\frac{1}{y}}{\frac{z+1}{3}-15} \)
```

Обратите внимание, что LaTeX меняет размеры символов по мере углубления внутрь дроби. Это можно рассмотреть еще нагляднее: в формуле $12x+\frac{\frac{2x}{y}+20x}{\frac{2y}{x}+30y}$ хорошо видно, что использовано три размера символов. Интересно, что если ту же самую формулу оформить как выносную, размеры будут иными:

$$12x+\frac{\frac{2x}{y}+20x}{\frac{2y}{x}+30y}$$

При необходимости этими размерами можно управлять; к этому вопросу мы вернемся в § 10.7.

Для набора знака радикала (корня) используется команда `\sqrt[`<*степень*>`]{`<*выражение*>`}`[1]. Параметр *степень* необязательный, он позволяет указать степень извлекаемого корня; если предполагается квадратный корень, параметр можно опустить вместе с квадратными скобками. Параметр *выражение* задает подкоренное выражение. Например, $\sqrt[3]{a+b}$ набирается как `\sqrt[3]{a+b}`, а $\sqrt{1-\sin^2 x}$ — как `\sqrt{1 - \sin^2 x}`.

Команды `\sqrt` могут быть вложенными, могут сочетаться с дробями и другими элементами формул; размер «галочки» слева и «планочки» сверху LaTeX определит самостоятельно. В частности, если мы наберем

```
\[ \sqrt{ x_1 + \sqrt{ x_2 + \sqrt{ x_3 }}} \],
```

у нас получится

$$\sqrt{x_1 + \sqrt{x_2 + \sqrt{x_3}}}$$

В этой формуле хорошо видно, как меняются размер и начертание знака корня.

10.3. Сумма, произведение и интеграл

10.3.1. Список «больших операторов» и примеры

$\int$	`\int`	$\oint$	`\oint`		
$\sum$	`\sum`	$\prod$	`\prod`	$\coprod$	`\coprod`
$\bigcap$	`\bigcap`	$\bigcup$	`\bigcup`	$\bigsqcup$	`\bigsqcup`
$\bigoplus$	`\bigoplus`	$\bigotimes$	`\bigotimes`	$\bigodot$	`\bigodot`
$\biguplus$	`\biguplus`	$\bigvee$	`\bigvee`	$\bigwedge$	`\bigwedge`

Таблица 10.1: Символы «больших операторов»

Математические знаки интеграла, а также суммирования, произведения, пересечения и объединения последовательности множеств и т. п. с точки зрения компьютерного набора оказываются очень схожи: каждый из них представляет собой «большой оператор», у которого есть необязательные нижние и верхние пределы (обозначающие «от» и «до»), а также собственно выражение, подлежащее интегрированию, суммированию или еще какой-то операции. Полный список «больших операторов» и соответствующих им команд приведен в таблице 10.1. Отметим, что для

[1] Название `sqrt` происходит от слов *square root*, то есть «квадратный корень».

задания пределов интегрирования, суммирования и т. п. используются те же знаки «_» и «^», что и для обозначения верхних и нижних индексов в обычных формулах. Приведем несколько примеров:

```
\int_a^b f(x) \, dx
```
$$\int_a^b f(x)\,dx$$

```
\oint_C a \; dr
```
$$\oint_C a\;dr$$

```
\sum_{i=0}^{n} q_i
```
$$\sum_{i=0}^{n} q_i$$

```
\bigcup_{-n \le k \le n} W_k
```
$$\bigcup_{-n\le k\le n} W_k$$

```
\prod_{f \in F} f(x)
```
$$\prod_{f\in F} f(x)$$

10.3.2. Расположение пределов суммирования и интегрирования

Как можно заметить из примеров предыдущего параграфа, LaTeX располагает пределы интегрирования справа от знака интеграла, а не над ним и под ним, как это привычно для отечественного читателя. То же самое происходит и с пределами суммирования, если знак суммы поместить в строчную формулу: $\sum_{i=0}^{n} q_i$ или же, например, в числитель или знаменатель дроби, пусть даже и в выносной формуле:

$$\frac{\sum_{i=0}^{n} q_i}{n}$$

Если выбранный LaTeX'ом стиль размещения пределов нас не удовлетворяет, можно взять контроль в свои руки, поставив сразу после команды «большого оператора» команду `\limits` (располагаем пределы под и над знаком оператора) или команду `\nolimits` (располагаем пределы справа от знака оператора). Например:

```
\[\int\limits_a^b f(x)\,dx\]
```
$$\int\limits_a^b f(x)\,dx,$$

```
\[\sum\nolimits_{i=0}^{n}q_i\]
```
$$\sum\nolimits_{i=0}^{n} q_i$$

В строчных формулах `\(\sum_{i=0}^{n}q_i\)` даст $\sum_{i=0}^{n} q_i$, тогда как `\(\sum\limits_{i=0}^{n}q_i\)` даст $\sum\limits_{i=0}^{n} q_i$. Наконец, пример с суммой в числителе дроби можно привести в порядок так:

```
\[\frac{\sum\limits_{i=0}^{n} q_i}{n}\]
```

что дает

$$\frac{\sum\limits_{i=0}^{n} q_i}{n}$$

10.3.3. Кратные интегралы

Если поставить несколько знаков интеграла подряд, результат получится довольно неказистый:

$$\int\int\int F(x,y,z)\,dx\,dy\,dz$$

Кроме того, часто нужно указать область интегрирования сразу для всех знаков интеграла (например, тройной интеграл по объему шара). Поэтому обычно кратные интегралы набирают с помощью специальных команд `\iint` ($\iint$), `\iiint` ($\iiint$) и `\iiiint` ($\iiiint$), например `\[\iiint\limits_V f(v)\,dv\]` даст

$$\iiint\limits_V f(v)\,dv$$

.

Также может пригодиться команда `\idotsint`, дающая два знака интеграла с многоточием между ними:

$$\int\limits_V \cdots \int f(v)\,dv$$

.

10.4. Часто используемые математические символы

10.4.1. Символы операций и отношений

В таблице 10.2 приведены команды математического режима, печатающие наиболее употребительные символы, используемые в формулах. В отдельную таблицу (10.3) мы вынесли символы всевозможных «стрелочек», среди которых стоит выделить, видимо, символы $\Longrightarrow$

$\forall$	\forall	$\exists$	\exists	∞	\infty	$\mapsto$	\mapsto
$\in$	\in	$\ni$	\ni	$\emptyset$	\emptyset	$\setminus$	\setminus
$\subset$	\subset	$\supset$	\supset	$\subseteq$	\subseteq	$\supseteq$	\supseteq
$\leq$	\leq	$\geq$	\geq	$\equiv$	\equiv	$\neq$	\neq
$\cdot$	\cdot	$\approx$	\approx	$\sim$	\sim	$\cong$	\cong
$\ll$	\ll	$\gg$	\gg	$\mid$	\mid	$\neg$	\neg
$\vee$	\vee	$\wedge$	\wedge	$\lceil$	\lceil	$\rceil$	\rceil
$\perp$	\perp	$\parallel$	\parallel	$\angle$	\angle	$\triangle$	\triangle
$\lfloor$	\lfloor	$\rfloor$	\rfloor	∂	\partial	∇	\nabla
$\pm$	\pm	$\mp$	\mp	$\div$	\div	$\times$	\times
$\oplus$	\oplus	$\ominus$	\ominus	$\otimes$	\otimes	$\oslash$	\oslash
$\odot$	\odot	$\cap$	\cap	$\cup$	\cup	$\circ$	\circ

Таблица 10.2: Наиболее употребительные математические символы

$\leftarrow$	\leftarrow	$\rightarrow$	\rightarrow
$\Leftarrow$	\Leftarrow	$\Rightarrow$	\Rightarrow
$\leftrightarrow$	\leftrightarrow	$\Leftrightarrow$	\Leftrightarrow
$\longleftarrow$	\longleftarrow	$\longrightarrow$	\longrightarrow
$\Longleftarrow$	\Longleftarrow	$\longleftrightarrow$	\longleftrightarrow
$\Longrightarrow$	\Longrightarrow	$\Longleftrightarrow$	\Longleftrightarrow

Таблица 10.3: Разные стрелочки

(\Longrightarrow) и $\Longleftrightarrow$ (\Longleftrightarrow), часто используемые в значении «следовательно» и «тогда и только тогда». Для примера приведем текст определения предела по Коши с использованием соответствующих символов:

$$\forall \varepsilon > 0 \quad \exists \delta(\varepsilon) > 0 : \forall x \; 0 < |x - a| < \delta \Rightarrow |f(x) - b| < \varepsilon$$

Набрано это так:

```
\[ \forall \varepsilon>0 \quad \exists\delta(\varepsilon)>0
\; \colon \forall x \; 0<|x-a|<\delta \Rightarrow
|f(x)-b|<\varepsilon \]
```

Отрицание любого логического символа можно получить, поставив перед ним команду \not; так, например, $x \notin T$ набирается с помощью `x \not\in T`, а $x \not< y$ — как `x \not< y`.

Символами, приведенными в таблице 10.2, возможности LaTeX'а не исчерпываются; более полные таблицы поддерживаемых математических символов можно найти в книге [4].

α	\alpha	β	\beta	γ	\gamma	δ	\delta
ϵ	\epsilon	ε	\varepsilon	ζ	\zeta	η	\eta
θ	\theta	ϑ	\vartheta	ι	\iota	κ	\kappa
λ	\lambda	μ	\mu	ν	\nu	ξ	\xi
π	\pi	ϖ	\varpi	ρ	\rho	ϱ	\varrho
σ	\sigma	ς	\varsigma	τ	\tau	υ	\upsilon
ϕ	\phi	φ	\varphi	χ	\chi	ψ	\psi
ω	\omega	Γ	\Gamma	Δ	\Delta	Θ	\Theta
Λ	\Lambda	Ξ	\Xi	Π	\Pi	Σ	\Sigma
Υ	\Upsilon	Φ	\Phi	Ψ	\Psi	Ω	\Omega

Таблица 10.4: Греческие буквы

10.4.2. Греческие буквы

Поскольку в математических формулах часто используются буквы греческого алфавита, LaTeX поддерживает специальный набор команд для их воспроизведения. Эти команды приведены в таблице 10.4. Обратите внимание, что для некоторых букв имеются различные варианты начертания (сравните `\phi` и `\varphi` или `\epsilon` и `\varepsilon`, дающие соответственно ϕ, φ, ϵ и ε). Как можно заметить, заглавные греческие буквы представлены не все. В формулах не используются те из них, что совпадают по начертанию с буквами латинского алфавита (например, греческая альфа в заглавном варианте выглядит точно так же, как латинская «A»). То же самое можно сказать о строчной букве омикрон, начертание которой совпадает с курсивным латинским «o».

10.4.3. Дополнительные шрифты для формул

Если загрузить пакет `amsfonts`, вам станут доступны еще несколько команд, в том числе `\mathcal`, `\mathfrak` и `\mathbb`, включающие соответственно каллиграфический, готический и «ажурный»[2] шрифты:

```
\mathcal{ABCDEFGHIJKLM}
\mathcal{NOPQRSTUVWXUZ}
\mathfrak{ABCDEFGHIJKLM}
\mathfrak{NOPQRSTUVWXUZ}
\mathfrak{abcdefghijklm}
\mathfrak{nopqrstuvwxuz}
\mathbb{ABCDEFGHIJKLM}
\mathbb{NOPQRSTUVWXUZ}
```

$\mathcal{ABCDEFGHIJKLM}$
$\mathcal{NOPQRSTUVWXUZ}$
$\mathfrak{ABCDEFGHIJKLM}$
$\mathfrak{NOPQRSTUVWXUZ}$
$\mathfrak{abcdefghijklm}$
$\mathfrak{nopqrstuvwxuz}$
$\mathbb{ABCDEFGHIJKLM}$
$\mathbb{NOPQRSTUVWXUZ}$

[2] В английском оригинале этот шрифт называется «Blackboard bold», то есть «жирный шрифт для грифельной доски».

$\hat{x}$	`\hat x`	$\check{x}$	`\check x`	$\acute{x}$	`\acute x`	$\grave{x}$	`\grave x`
$\dot{x}$	`\dot x`	$\ddot{x}$	`\ddot x`	$\dddot{x}$	`\dddot x`	$\ddddot{x}$	`\ddddot x`
$\bar{x}$	`\bar x`	$\vec{x}$	`\vec x`	$\breve{x}$	`\breve x`	$\tilde{x}$	`\tilde x`
$\mathring{x}$	`\mathring x`						

Таблица 10.5: Акценты в формулах

Строчные буквы в «ажурном» и каллиграфическом шрифтах не предусмотрены.

Помимо перечисленных команд, `amsfonts` делает доступными многие другие символы, полезные при наборе формул; подробности читатель найдет в книге [4].

10.4.4. Акценты в формулах

При необходимости можно любые буквы, используемые в математическом режиме, снабдить акцентами, аналогичными диакритическим знакам. Соответствующие команды показаны в таблице 10.5.

10.4.5. Многоточия

Если в формуле необходимо многоточие, для его изображения следует воспользоваться командами `\ldots` (точки вдоль низа строки) и `\cdots` (точки, центрированные по вертикали, то есть расположенные вдоль середины строки). Обычно `\ldots` используют при перечислении через запятую или в ряду элементов, между которыми вообще нет знаков (например, в произведении сомножителей), а `\cdots` — в суммах и других перечислениях, разделенных знаками операций или отношений:

```
$i = 1, 2, \ldots, n$                 i = 1, 2, \ldots, n
$\sum_{i=1}^{n} a_i = a_1 + a_2 + \cdots + a_n$   \sum_{i=1}^n a_i
                                      =a_1+a_2+\cdots+a_n
$x_1 + x_1x_2 + \cdots + x_1 \ldots x_n$   x_1+x_1x_2+ \cdots+x_1\ldots x_n
```

При верстке матриц используются также команды `\vdots` (вертикальное многоточие) и `\ddots` (диагональное многоточие). К этим командам мы вернемся в § 10.8.

$(x+y)$	`(x+y)`	$\langle x+y \rangle$	`\langle x+y \rangle`
$[x+y]$	`[x+y]`	$\lfloor x+y \rfloor$	`\lfloor x+y \rfloor`
$\{x+y\}$	`\{x+y\}`	$\lceil x+y \rceil$	`\lceil x+y \rceil`

Таблица 10.6: Математические скобки

10.5. Скобки разных размеров

LaTeX поддерживает несколько видов скобок: обычные круглые, квадратные, фигурные, угловые, а также скобки, обозначающие ближайшее целое снизу и сверху (см. таблицу 10.6).

В §10.2 мы отмечали, что LaTeX способен самостоятельно увеличивать при необходимости размер знака радикала. Символы скобок также могут изменять свой размер (в том числе и автоматически). Кроме них такой способностью обладают еще и символы-*ограничители*, к которым относятся «/» (/), «\» (`\backslash`), «|» (|), «‖» (`\|`), «↑» (`\uparrow`), «⇑» (`\Uparrow`), «↓» (`\downarrow`), «⇓» (`\Downarrow`), «↕» (`\updownarrow`) и «⇕» (`\Updownarrow`).

Чтобы получить слегка увеличенную версию какой-либо из открывающих скобок (или другого ограничителя, выступающего в роли открывающей скобки, как, например, при обозначении модуля), следует перед ограничителем вставить команду `\bigl` (big left). Перед закрывающей скобкой или ограничителем используют команду `\bigr` (big right). Если ограничитель не является открывающим или закрывающим, как, например, вертикальная черта в формуле

$$\bigl(x \in A(n) \bigm| x \in B(n)\bigr),$$

для его увеличения необходимо использовать команду `\bigm` (big middle). Все эти команды, кроме собственно увеличения символа, влияют еще и на выбор расстояния между символами, а `\bigl` и `\bigm` выполняют заодно функцию группировки на случай автоматической расстановки размеров (об этом позже). Если нужно просто увеличить символ, можно использовать команду `\big`.

Если нужно сделать скобки еще больше, можно воспользоваться командами `\Big`, `\Bigl`, `\Bigm` и `\Bigr`. Обычно такие команды применяют только в выносных формулах, поскольку для строчных получаемые символы слишком велики (в полтора раза выше, чем для `big`-команд). Имеются, кроме этого, команды `\bigg`, `\biggl`, `\biggm` и `\biggr` (вдвое больше, чем для `big`-команд), а также `\Bigg`, `\Biggl`, `\Biggm` и `\Biggr` (больше в 2,5 раза).

Можно также поручить LaTeX'у самостоятельно выбирать размеры скобок. Для этого вместо вышеперечисленных команд используются команды `\left` и `\right`. Если пометить ограничители этими командами,

LaTeX вставит символы ровно такого размера, чтобы целиком охватить находящуюся между ограничителями подформулу. Например:

$$e_n = \left(1+\frac{1}{n}\right)^n$$

```
\[
e_n=\left(1+\frac{1}{n}\right)^n
\]
```

Отметим, что размеры, выбираемые автоматически — это не всегда то, что нужно. Например, в формуле $\big||x|+|y|\big|$ для увеличения ограничителей внешнего модуля мы применили команды `\bigl` и `\bigr`. Дело в том, что команды `\left` и `\right` в этой ситуации ничего не увеличат, поскольку заключенная в ограничители формула «помещается» в ограничители обычной высоты. С другой стороны, одним из достоинств команд `\left` и `\right` является их способность создавать ограничители произвольно большого размера.

Отметим еще один момент. Иногда бывает нужен только один из парных ограничителей — например, для обозначения системы уравнений нужна открывающая фигурная скобка и не нужна закрывающая. В этом случае следует в качестве парного ограничителя использовать *нулевой ограничитель*, который набирается в виде символа точки непосредственно после команды `\right` или `\left`, например `\left\{...\right.`. К системам уравнений мы вернемся в § 10.9.

10.6. Дополнительные математические символы

Шрифты, входящие в комплект LaTeX'а, содержат множество весьма экзотических символов, предназначенных для применения в математических формулах. Некоторые из них показаны в табл. 10.7; на самом деле символов гораздо больше, и полный список мы здесь не приводим. В случае, если в ваших формулах требуются как раз такие символы, вам следует подключить пакет `amssymb`, в котором описаны показанные в таблице команды.

$\leqslant$	`\leqslant`	$\geqslant$	`\geqslant`
$\lll$	`\lll`	$\ggg$	`\ggg`
$\lessdot$	`\lessdot`	$\gtrdot$	`\gtrdot`
$\fallingdotseq$	`\fallingdotseq`	$\risingdotseq$	`\risingdotseq`
$\doteqdot$	`\doteqdot`	$\vDash$	`\vDash`
$\circlearrowleft$	`\circlearrowleft`	$\curvearrowleft$	`\curvearrowleft`

Таблица 10.7: Некоторые математические символы из пакета `amssymb`

Отметим, что именно в пакете `amssymb` определены команды для значков «больше или равно» и «меньше или равно» ($\leqslant$ и $\geqslant$, обозначаются `\leqslant` и `\geqslant`), имеющих вид более привычный для российского читателя, нежели знаки, порождаемые командами `\leq` и `\geq`:

$a \leqslant b$ `a \leqslant b` $a \geqslant b$ `a \geqslant b`
$a \leq b$ `a \leq b` $a \geq b$ `a \geq b`

Попадание столь привычных нам символов в категорию «экзотики» обусловлено различиями в западных и отечественных типографских традициях: на Западе в математических текстах обычно используются как раз знаки $\leq$ и $\geq$. Если вам требуются только эти два символа, включать весь пакет `amssymb` не обязательно. Проще будет добавить в преамбулу документа следующие две строки:

```
\DeclareMathSymbol{\leqslant}{\mathrel}{AMSa}{"36}
\DeclareMathSymbol{\geqslant}{\mathrel}{AMSa}{"3E}
```

Вы можете дать этим символам и более удобные имена, чтобы не писать каждый раз сравнительно длинные команды `\leqslant` и `\geqslant`.

10.7. Еще о размерах

Как видно из обсуждавшихся выше примеров, LaTeX автоматически выбирает различные размеры математических символов в формулах в зависимости от ситуации. В некоторых случаях контроль за этим необходимо взять в свои руки; для этого рассмотрим механизм выбора размеров в формулах подробнее. LaTeX имеет восемь *стилей*, в которых может обрабатываться формульный текст. Стили делятся на четыре *основных* и соответствующие им четыре *сжатых*. Основные стили можно при необходимости устанавливать вручную, на сжатые TeX переходит автоматически. Команда `\displaystyle` устанавливает стиль D, применяющийся в выносных формулах; команда `\textstyle` — стиль T, применяемый в строчных формулах; `\scriptstyle` — стиль S, применяемый для верхних и нижних индексов, а `\scriptscriptstyle` — стиль SS индексов к индексам.

Если не применять эти команды, стиль выбирается автоматически по принципу, описанному в таблице 10.8 (D', T', S' и SS' обозначают соответствующие сжатые версии). Иногда автоматический выбор стиля приводит к довольно странным результатам: например, код

```
\[ 1+\frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{x}}}} \]
```

исходный стиль	верхний индекс	нижний индекс	числитель	знаменатель
D	S	S’	T	T’
D’	S’	S’	T’	T’
T	S	S’	S	S’
T’	S’	S’	S’	S’
S, SS	SS	SS’	SS	SS’
S’, SS’	SS’	SS’	SS’	SS’

Таблица 10.8: Автоматический выбор стилей в формулах

даст

$$1+\frac{1}{1+\frac{1}{1+\frac{1}{1+\frac{1}{x}}}}$$

Существенно более эстетичных результатов можно добиться с помощью команд явного переключения стилей:

$$1+\frac{1}{\displaystyle 1+\frac{1}{\displaystyle 1+\frac{1}{\displaystyle 1+\frac{1}{x}}}}$$

```
\[
 1+\frac{1}{\displaystyle 1+
   \frac{1}{\displaystyle 1+
   \frac{1}{\displaystyle 1+
   \frac{1}{x}}}}          \]
```

10.8. Матрицы

Матрицы набираются способом, похожим на применяемый для таблиц (см. § 7.2): используется специальное окружение, внутри которого набираются строки матрицы, разделяемые командой \\, а столбцы в строке разделяются символом «&»:

$$\begin{pmatrix} 1 & 2 & 3 \\ x & y & z \\ a & b & c \end{pmatrix}$$

```
\[ \left( \begin{matrix}
1 & 2 & 3 \\ x & y & z \\ a & b & c \\
          \end{matrix} \right) \]
```

В этом примере мы в явном виде поставили ограничивающие скобки, которые благодаря командам `\left` и `\right` оказались нужного размера (см. § 10.5); само окружение `matrix` дает только содержание матрицы без ограничителей. С помощью окружений `pmatrix`, `bmatrix`, `vmatrix` и

`Vmatrix` можно получить матрицу вместе с ограничителями. Например, с помощью кода

```
\[
\begin{pmatrix} a_1^1 & a_1^2 \\ a_2^1 & a_2^2 \end{pmatrix}
\begin{bmatrix} a_1^1 & a_1^2 \\ a_2^1 & a_2^2 \end{bmatrix}
\begin{vmatrix} a_1^1 & a_1^2 \\ a_2^1 & a_2^2 \end{vmatrix}
\begin{Vmatrix} a_1^1 & a_1^2 \\ a_2^1 & a_2^2 \end{Vmatrix}
\]
```

мы получим

$$\begin{pmatrix} a_1^1 & a_1^2 \\ a_2^1 & a_2^2 \end{pmatrix} \quad \begin{bmatrix} a_1^1 & a_1^2 \\ a_2^1 & a_2^2 \end{bmatrix} \quad \begin{vmatrix} a_1^1 & a_1^2 \\ a_2^1 & a_2^2 \end{vmatrix} \quad \begin{Vmatrix} a_1^1 & a_1^2 \\ a_2^1 & a_2^2 \end{Vmatrix}$$

Максимальная ширина матрицы — 10 столбцов. Если необходима матрица с бóльшим количеством столбцов, следует временно увеличить этот максимум, который хранится в счетчике `MaxMatrixCols`:

```
\setcounter{MaxMatrixCols}{20}
```

После набора матрицы рекомендуется вернуть `MaxMatrixCols` значение 10, поскольку LaTeX тратит на верстку матриц тем больше времени, чем больше текущее значение `MaxMatrixCols`.

С помощью окружения `smallmatrix` можно получить матрицу, пригодную для размещения в качестве строчной формулы. Например, матрица $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ получена вставкой в этот абзац следующего кода:

```
\(\left( \begin{smallmatrix} a & b \\ c & d \end{smallmatrix}
  \right)\)
```

10.9. Системы уравнений и другие многострочные конструкции

10.9.1. Группа формул с выравниванием

С помощью окружения `aligned` можно вывести группу формул, в каждой из которых задается *точка выравнивания*, с тем чтобы эти точки располагались вертикально одна над другой. Точка выравнивания задается символом «`&`». Например, код

```
\[ \begin{aligned}
\sin x\pm\sin y& = 2\sin\frac{x\pm y}{2}\cos\frac{x\mp y}{2}\\
\sin x\sin y   & = \frac{1}{2}(\cos(x-y)-\cos(x+y))\\
\end{aligned} \]
```

выдаст следующий результат:

$$\sin x \pm \sin y = 2 \sin \frac{x \pm y}{2} \cos \frac{x \mp y}{2}$$
$$\sin x \sin y = \frac{1}{2}(\cos(x - y) - \cos(x + y))$$

Точку выравнивания мы в обеих формулах поставили непосредственно перед знаком равенства, по этим знакам и произошло выравнивание (они оказались точно один над другим).

10.9.2. Системы уравнений и неравенств

Общий принцип набора систем (или совокупностей) уравнений и неравенств таков: сами уравнения и неравенства набираются в окружении `aligned` (см. § 10.9.1), причем точка выравнивания ставится перед началом каждого уравнения (неравенства); знак системы (фигурная скобка) или совокупности (квадратная скобка или просто вертикальная черта) набираются как «большой ограничитель» с помощью команды `\left` (см. § 10.5) перед содержимым системы или совокупности, так что LaTeX автоматически выбирает нужный размер скобки. Для парной команды `\right` применяют «нулевой ограничитель», то есть ставят точку сразу после команды (см. стр. 72). Приведем пример:

$$\begin{cases} x + y = 5 \\ x - 2y = 2 \end{cases}$$

```
\[ \left\{\begin{aligned}
  & x+y=5 \\ & x-2y=2
  \end{aligned}\right. \]
```

Рассмотрим более сложный пример. Формула

$$\sqrt{f(x)} > g(x) \Longleftrightarrow \left[\begin{aligned} & \begin{cases} f(x) > (g(x))^2 \\ g(x) \ge 0 \end{cases} \\ & \begin{cases} f(x) \ge 0 \\ g(x) < 0 \end{cases} \end{aligned} \right.$$

набрана так:

```
\[ \sqrt{f(x)}>g(x) \Longleftrightarrow
     \left[ \begin{aligned}
       & \left\{ \begin{aligned}
          & f(x) > (g(x))^2 \\
          & g(x) \ge 0
         \end{aligned} \right. \\
```

```
    & \left\{ \begin{aligned}
       & f(x) \ge 0 \\
       & g(x) < 0
      \end{aligned} \right. \\
\end{aligned} \right.    \]
```

Еще раз обратите внимание на то, что система заключается между командами `\left\{ ... \right.`, а совокупность — между командами `\left[ ... \right.`. Для обозначения совокупности можно использовать и простую вертикальную черту «`|`»:

$$\left| \begin{aligned} x &< 5 \\ x &> 15 \end{aligned} \right.$$

```
\[ \left|\begin{aligned}
   & x < 5 \\ & x > 15
   \end{aligned}\right. \]
```

10.9.3. Перечисление случаев

Для набора часто встречающегося в математических текстах «перечисления случаев» (такое-то значение при таких-то условиях, такое-то при таких-то) удобно применять специальное окружение `cases`:

$$|x| = \begin{cases} x, & x \ge 0, \\ -x, & x < 0 \end{cases}$$

```
\[ |x| = \begin{cases}
         x,  & x \ge 0, \\
         -x, & x < 0
         \end{cases}  \]
```

В примере видно, что строки (случаи) перечисления разделяются командой `\\`, а условие отделяется от значения символом «`&`».

10.9.4. Подписанные знаки отношений

Часто возникает необходимость снабдить знак равенства, стрелку или другой символ отношения тем или иным текстом. Это можно сделать с помощью команды `\stackrel`:

$\ln x \stackrel{\text{Df}}{=} y,\ e^y = x$ — `\ln x\stackrel{\text{Df}}{=}y,\, e^y=x`

$\mathbb{R}\times\mathbb{R} \stackrel{f(x,y)}{\longmapsto} \mathbb{R}$ —
```
\mathbb{R\times R}
\stackrel{f(x,y)}{\longmapsto}
\mathbb{R}
```

$X \stackrel{\alpha}{\longrightarrow} Y$ — `X \stackrel{\alpha}{\longrightarrow} Y`

Стрелки вправо и влево можно растянуть так, чтобы подпись над ними уместилась целиком. Для этого используются команды `\xleftarrow` и `\xrightarrow`; обе команды имеют обязательный параметр, задающий текст подписи сверху (над стрелкой), и необязательный — текст подписи снизу под стрелкой. Например, формула

$$f(x,y) \xrightarrow[D\to D_0]{(x,y)\in D} 0$$

получена с помощью

```
\[ f(x,y) \xrightarrow[D\rightarrow D_0] {(x,y)\in D} 0 \]
```

Если подпись снизу не нужна, первый (необязательный) аргумент можно опустить вместе с квадратными скобками.

10.9.5. Многострочные индексы

Иногда требуется набрать индекс из нескольких строк. Это можно сделать с помощью специальной команды `\substack`:

$$\lim_{\substack{x\to 0\\ y\to 0\\ z\to 0}} F(x,y,z)$$

```
\lim_{\substack{
   x\rightarrow 0 \\
   y\rightarrow 0 \\
   z\rightarrow 0
     }} F(x,y,z)
```

$$\int\cdots\int_{\substack{x\in V\\ \tau(x)>0}} \varphi\,ds$$

```
\idotsint\limits_{
    \substack{x \in V\\
              \tau(x)>0}}
  \varphi\,ds
```

При использовании команды `\substack` индекс (включая саму команду `\substack`) необходимо взять в фигурные скобки. Строки индекса в аргументе команды `\substack` отделяются друг от друга с помощью команды `\\`. Команда `\substack` центрирует все строки, составляющие многострочный индекс. В качестве альтернативы можно воспользоваться окружением `subarray`, выравнивающим строки по левому краю:

$$\int\cdots\int_{\begin{subarray}{l} x\in V\\ \tau(x)>0 \end{subarray}} \varphi\,ds$$

```
\idotsint\limits_{
    \begin{subarray}{l}
        x \in V \\ \tau(x)>0
    \end{subarray}} \varphi\,ds
```

Глава 11

Когда основной текст готов

Итак, ваш документ полностью набран, осталось придать ему законченный вид: снабдить титульной страницей, убедиться, что оформление полностью соответствует требованиям и, возможно, в самом конце озаботиться «тонкой настройкой» вашей верстки, чтобы результат лучше выглядел. Содержание этой (последней) главы книги потребуется вам именно теперь — когда набор основного текста уже завершен.

11.1. Как сверстать титульную страницу

Титульная страница отличается от всего остального текста тем, что при ее создании нам необходимо явно задать расположение текста — верхней «шапки», имени и фамилии автора, заголовка работы, всевозможных подзаголовков, сведений о научном руководителе, и, наконец, традиционной информации о времени и месте (город/год) создания работы.

До сих пор мы намеренно не касались методов влияния на физическое размещение текста: в большинстве случаев будет лучше, если мы не будем мешать LaTeX'у верстать документ в соответствии с типографскими нормами. Это, однако, не относится к титульной странице.

11.1.1. Центрирование текста

Начнем с команды `\centerline` и окружения `centering`. Легко догадаться, что они позволяют выводить строки текста, центрируя их на листе. С командой `\centerline{`<*строка*>`}` все более-менее понятно: она принимает на вход аргумент, обычно заключаемый в фигурные скобки, и выводит его всегда в одну строку (даже если эта строка не влезет на лист), причем середина строки располагается в середине полосы набора.

Речь идет именно о полосе набора, а не о физическом листе, так что если в нашем документе заданы разные величины для левого и правого полей (см. § 1.2.1, стр. 17), при центрировании это окажется учтено.

Окружение `centering` позволяет центрировать текст, состоящий из нескольких строк. Если при этом потребовать набрать с центрированием целый абзац, он будет автоматически разбит по имеющимся пробелам на строки такой длины, чтобы каждая из них уместилась в полосе набора.

Здесь необходимо учитывать эффект, с которым мы уже сталкивались при обсуждении управления размером шрифта и интерлиньяжа (см. § 2.2.2). LaTeX форматирует абзац только тогда, когда этот абзац закончился, то есть встречена пустая строка или команда `\par`. Таким образом, если вы заключите некий текст между командами `\begin{centering}` и `\end{centering}`, не отделив последнюю от текста пустой строкой, эффект может вас несколько удивить: ведь LaTeX «увидит» конец абзаца уже **после** того, как окружение закончилось, а значит, и форматировать абзац будет как обычно, то есть без всякого центрирования. Простейший вариант решения — **всегда** отделять начало и конец окружения от текста пустой строкой.

Если автоматически выбранная LaTeX'ом разбивка на строки вас не устроила, можно расставить переводы строк вручную, используя команду `\\`.

11.1.2. Пропуск места по вертикали и по горизонтали

Между элементами оформления титульной страницы обычно вставляются свободные участки. Есть два основных способа задать такой промежуток. Первый способ предполагает явное указание величины промежутка (например, в сантиметрах). Второй позволяет равномерно распределить «вакантное» вертикальное пространство между несколькими указанными местами в тексте.

Сделать промежуток фиксированного размера можно с помощью команды `\vskip`, указав (без пробела) размер в сантиметрах (cm), миллиметрах (mm) или пунктах (pt). Например, `\vskip2cm` создаст пустое пространство в 2 см, `\vskip15mm` или `\vskip1.5cm` — в 15 мм и т. п.

Для равномерного распределения вертикального пространства используется команда `\vfill`. По окончании генерации содержания страницы все оставшееся свободное место вдоль вертикальной оси распределяется поровну между встреченными на странице командами `\vfill`. Таким образом, если вставить `\vfill` в одном месте, то все свободное вертикальное пространство будет вставлено именно в это место. Если использовать две команды, на месте каждой будет ровно половина свободного места, если три — соответственно, треть, и т. д.

Если нужно сделать вертикальные отступы разного размера, можно в одном месте использовать несколько команд `\vfill`. Например, если в одном месте вставить две команды, а в другом три, то и свободное место распределится в отношении 2 : 3.

Аналогичные возможности предусмотрены и для горизонтальных отступов в строках. Команда `\hspace{`<*длина*>`}` вставляет в текст горизонтальный отступ, заданный параметром *длина* (следует учесть, что эта команда не сработает в конце строки; если вы с этим столкнулись, попробуйте «звездочную» версию `\hspace*`). Команда `\hfill` задает точку, для которой производится распределение свободного горизонтального пространства; иначе говоря, оставшееся свободным горизонтальное пространство в строке распределяется поровну между командами `\hfill`, аналогично тому, как вертикальное пространство распределяется между командами `\vfill`.

11.1.3. Горизонтальное отчеркивание

Чтобы провести горизонтальную черту (в том числе в местах, предполагающих вписывание от руки), используется команда `\rule[`<*высота*>`]{`<*длина*>`}{`<*толщина*>`}`. Здесь обязательные параметры *длина* и *толщина* задают длину и толщину проводимой линии, а необязательный параметр *высота* указывает, насколько линию следует поднять вверх относительно нижнего края букв в строке. Если это не нужно, параметр следует опустить вместе с квадратными скобками.

Полезной может оказаться и команда `\hrulefill`, по сути аналогичная команде `\hfill` из предыдущего параграфа, с той только разницей, что свободное место в строке она заполняет отчеркиванием.

11.1.4. Выравнивание по правому краю

Часто на титульной странице требуется прижать определенные фрагменты текста к правому краю полосы набора. В диссертации, например, именно так размещается код УДК и стандартная надпись «на правах рукописи»; на титульной странице дипломной или курсовой работы обычно по правому краю выравнивается информация о научном руководителе, а иногда и об исполнителе работы.

Прижать вправо отдельную строку можно командой `\rightline{`<*строка*>`}`. Если же прижать вправо необходимо несколько строк подряд, можно воспользоваться окружением `flushright`: все строки, заключенные между командами `\begin{flushright}` и `\end{flushright}`, будут сдвинуты к правой границе полосы набора. Рекомендуется внутри окружения `flushright` разделить строки явным образом с помощью команды `\\`.

Наконец, можно сформировать сдвинутый вправо абзац или несколько абзацев, воспользовавшись окружением `minipage`, которое будет описано в следующем параграфе.

11.1.5. Министраницы

При верстке титульной страницы и других сложных элементов текста могут пригодиться окружения `minipage` и `boxedminipage`. Эти окружения позволяют сверстать *министраницу*, то есть прямоугольную область текста, внутри которой текст верстается по тем же правилам, что и для обычной страницы (даже, при необходимости, со сносками), а во внешнем тексте министраница обрабатывается как обычный символ, только очень большой.

Окружение имеет обязательный параметр, задающий ширину министраницы. Параметр указывается в фигурных скобках сразу после команды `\begin{minipage}`. Так, например, если нужно сверстать абзац шириной в 30% полосы набора и прижать его вправо, можно воспользоваться следующим кодом:

```
\null\hfill
\begin{minipage}{0.3\textwidth}
  Весь этот текст сформатирован в отдельный
  абзац и сдвинут вправо к краю страницы.
\end{minipage}\\
```

Результат будет примерно таким:

Весь этот текст сформатирован в отдельный абзац и сдвинут вправо к краю страницы.

Если подключить в преамбуле пакет `boxedminipage`, вам станет доступно окружение `boxedminipage`, работающее точно так же, как и `minipage`, только сверстанная страница будет обведена рамкой. Например, следующий код:

```
\null\hfill
\begin{boxedminipage}{0.3\textwidth}
   Это пример министраницы в рамке
\end{boxedminipage}\\
```

даст примерно такой результат:

Это пример министраницы в рамке

11.1.6. Как закончить титульную страницу

Если никак не сообщить LaTeX'у об окончании титульной страницы, то текст, следующий непосредственно за ней, будет выведен на этой же странице, что вряд ли соответствует вашим желаниям.

Заявить об окончании титульной страницы можно двумя способами. Более простой из них — команда `\clearpage`. Встретив эту команду, LaTeX немедленно выведет все накопленные абзацы, распределив оставшееся вертикальное место между командами `\vfill`, и приступит к верстке следующей страницы. При этом, однако, будет выведен на печать номер страницы, что нежелательно. Избежать этого можно, поместив в код титульной страницы в произвольном месте команду `\thispagestyle{empty}`.

Другой способ — заключить содержимое титульной страницы в окружение `titlepage`. В некотором смысле это более правильно. Следует, однако, учесть, что дальнейшую нумерацию страниц в этом случае LaTeX продолжит с номера 1, тогда как в соответствии с отечественными стандартами первой страницей считается сама титульная страница (хотя номер на ней, естественно, не ставится). Исправить ситуацию поможет команда `\setcounter{page}{2}`, вставленная сразу после окончания титульной страницы (то есть, в данном случае, после команды `\end{titlepage}`). Эта команда принудительно установит номер текущей страницы в значение 2, что, собственно, и требуется.

11.1.7. Пример

На рис. 11.1 показан пример[1] титульной страницы. Чтобы сверстать этот пример, мы воспользовались следующим кодом:

```
\centerline{ИНСТИТУТ БРЕВЕН И СУЧКОВ РАН}
\centerline{\hfill\hrulefill\hrulefill\hfill}
\vfill
\rightline{на правах рукописи}
\vfill
\vfill
\large
\centerline{{\bf Вумников} Виталий Александрович}
\vfill
\Large
\begin{centering}
{\bf Качение бревна\\ по наклонной плоскости\\
```

[1] Название института, тема диссертации и имена действующих лиц взяты из замечательной сатирической пьесы В. Ф. Турчина ≪Защита≫. Читатель без труда найдет текст пьесы в сети Internet, задав в поисковой машине слово ≪бревнология≫.

ИНСТИТУТ БРЕВЕН И СУЧКОВ РАН

на правах рукописи

Вумников Виталий Александрович

Качение бревна по наклонной плоскости с учетом сучковатости

Специальность 05.88.99 —
механические и кинематические свойства
сучковатых бревен

Диссертация на соискание ученой степени
кандидата бревнологических наук

Научный руководитель:
д. бр. н. Персикович И. И.

Москва — 2010

Рис. 11.1: Пример титульной страницы

```
с учетом сучковатости\\}
\end{centering}
\normalsize
\vfill
\centerline{Специальность 05.88.99~--- }
\centerline{механические и кинематические свойства}
\centerline{сучковатых бревен}
\vfill
\centerline{Диссертация на соискание ученой степени}
\centerline{кандидата бревнологических наук}
\vfill
\vfill
\begin{flushright}
Научный руководитель:\\
д.\,бр.\,н.~Персикович~И.\,И.
\end{flushright}
\vfill
\vfill
\centerline{Москва~--- 2010}
```

11.2. Неожиданные требования к оформлению

Когда ваша рукопись окажется, по крайней мере, на ваш взгляд, полностью готова, вам могут предъявить требования, которые кому-то покажутся простыми, но начинающего верстальщика ставят в тупик. Это особенно касается случаев, когда рукопись предназначена для типографского размножения, но может случиться и при оформлении квалификационной работы. Как, например, убедить LaTeX ставить точки после номеров секций? Как изменить размер шрифта, которым LaTeX набирает слово «глава» в заголовках глав? Как заставить LaTeX центрировать заголовки, а то и вовсе прижимать их к правому краю полосы набора, а не к левому? Как в подписях к таблицам и рисункам заменить двоеточие после номера на точку?

Мы попытаемся ответить на эти вопросы, но прежде хотелось бы дать читателю один небольшой совет: не прибегайте к таким (практически крайним) мерам, пока вас к этому не вынудят. Как показывает опыт, принудительные изменения стиля оформления не идут на пользу общему виду документа. Впрочем, как можно заметить, в самой книжке, которую вы читаете, точки после номеров секций все же расставлены. Увы, таковы традиции отечественного книгопечатания; хорошо это или плохо — вопрос отдельный.

11.2.1. Меняем вид заголовков

Чаще всего нам приходится задуматься о внешнем виде заголовков глав и секций, когда кто-то требует от нас поставить точки после номеров секций и подсекций. В разных источниках, посвященных LaTeX'у, можно встретить различные рекомендации, как этого достичь. Чаще всего рекомендуют переопределить команды `\thesection` и `\thesubsection`, однако результат может вас обескуражить: например, при этом «сломается» внешний вид ссылок, заданных командой `\ref`.

Вторая часто возникающая проблема — как «отучить» LaTeX от привычки переносить слова в длинных заголовках. Добиться этого можно было бы явным запретом переноса соответствующих слов с помощью команды `\mbox`, но при этом мы рискуем увидеть в заголовках чрезмерно растянутые пробелы. Есть и другой способ: воспользовавшись пакетом `sectsty` и командой `\allsectionsfont{\raggedright}`, заставить LaTeX все заголовки считать «текстом с рваным правым краем», в котором переносы не предусмотрены. К сожалению, пакет `sectsty` не работает для многих классов документов, в том числе и для класса `extreport`, который мы рассматриваем и который нам наиболее удобен, так что и это решение нам не совсем подходит.

Справиться с обеими проблемами (и не только с ними) позволяет подход, который требует несколько бóльшего количества кода, но при этом более универсален и не имеет неожиданных негативных побочных эффектов. Присутствующие в LaTeX'е команды рубрикации создают заголовки вполне определенного вида, и, к сожалению, совершенно не приспособлены к настройке вида заголовков в соответствии с пожеланиями пользователя. Все это, однако, никак не может помешать нам ввести **свои** команды рубрикации.

Мы воспользуемся при этом «звездочной» формой стандартных команд рубрикации (см. § 3.5), позволив им сделать за нас бóльшую часть работы. Свои команды мы назовем так же, как стандартные, только заменим первую букву в их названиях на заглавную: в качестве замены команде `\chapter` мы опишем команду `\Chapter`, а вместо `\section` и `\subsection` будем использовать `\Section` и `\Subsection`. Как вводить новые команды, мы рассказали в § 5.1. При описании своих версий команд рубрикации нам придется самим позаботиться о счетчиках, в которых хранятся текущие номера главы, секции и подсекции. Кроме того, добавление строк в оглавление тоже придется выполнить вручную. Как добавлять информацию в оглавление, мы уже обсуждали в § 3.5; о счетчиках придется сказать несколько слов, прежде чем приступать к делу.

Счетчик в LaTeX'е — это некое имя, с которым связано целое число[2]. Номера текущей главы, секции и подсекции LaTeX хранит в счетчиках, которые называются соответственно `chapter`, `section` и `subsection`. Чтобы выдать на печать текущее значение счетчика, применяют команду `\arabic`[3]: например, `\arabic{chapter}` напечатает номер текущей главы. Кроме того, нам понадобится команда `\refstepcounter`, которая, во-первых, увеличивает на единицу значение заданного счетчика, и, во-вторых, обнуляет значения *подчиненных* счетчиков (например, `\refstepcounter{chapter}` увеличивает номер главы и сбрасывает номера секций, подсекций и т. д.).

Применив все сказанное, мы можем легко написать замену команде `\section`, которую, как и обещали, назовем `\Section`:

```
\newcommand\Section[1]{
 \refstepcounter{section}
 \section*{\raggedright
  \arabic{chapter}.\arabic{section}. #1}
 \addcontentsline{toc}{section}{%
  \arabic{chapter}.\arabic{section}. #1}
}
```

Как видим, наша команда принимает на вход один параметр (текст заголовка секции), увеличивает номер секции (это должно быть сделано **до** выдачи его на печать, потому что на момент начала выполнения команды в счетчике еще хранится номер предыдущей секции, а если мы только что начали главу — то и вовсе ноль), выдает заголовок секции, используя «звездочную» команду, при этом номер секции мы как бы включаем в ее заголовок, применяя явные обращения к соответствующим счетчикам (номер главы и номер секции, через точку). Использование команды `\raggedright` гарантирует нам отсутствие переносов слов в заголовке. После этого мы добавляем соответствующую строку в оглавление с помощью команды `\addcontentsline`.

Совершенно аналогично мы можем описать и команду `Subsection`:

```
\newcommand\Subsection[1]{
  \refstepcounter{subsection}
  \subsection*{\raggedright
    \arabic{chapter}.\arabic{section}.\arabic{subsection}. #1
  }
```

[2] На самом деле мы уже встречались с понятием счетчика, например, при описании титульной страницы документа и при верстке матриц в математических формулах.

[3] Название связано с тем, что число выдается на печать в арабских цифрах. Предусмотрены аналогичные команды `roman` (выдает римские цифры) и `alph` (выдает вместо номеров латинские буквы «a», «b», «c» и т. д.); версии этих команд `Alph` и `Roman` используют заглавные буквы вместо строчных.

```
  \addcontentsline{toc}{subsection}{%
    \arabic{chapter}.\arabic{section}.\arabic{subsection}.
    #1
  }
}
```

Вставив эти два описания в преамбулу документа, воспользуйтесь в вашем редакторе текстов режимом поиска с заменой и поменяйте все вхождения команд `\section` и `\subsection` на новые команды. Можно надеяться, что буквоеды, требовавшие расстановки точек после номеров секций, после этого от вас отстанут — однако они могут и не отстать, поскольку мы устранили не все дефекты. Дело в том, что команду `\chapter` мы пока еще не заменили. В самом заголовке главы нам менять нечего, поскольку, в отличие от секций и подсекций, номер главы печатается на отдельной строке и точки не требует. Остается, однако, еще и номер главы в соответствующей строке оглавления — и вот там, формально говоря, точка все-таки нужна, а ее пока нет.

Особенно обидно, что заголовок главы по своей структуре гораздо сложнее, чем заголовки секций и подсекций, и все это придется генерировать вручную, а ведь менять мы в этом заголовке ничего не хотим. Но если кто-то заметил отсутствие точек после номеров глав в оглавлении и нам на это указал, нам ничего другого, увы, не остается. Итак, пишем:

```
\newcommand\Chapter[1]{
 \refstepcounter{chapter}
 \chapter*{
  \begin{huge}
   \textbf{\chaptername\ \arabic{chapter}\\}
  \end{huge}%
  \bigskip \bigskip
  \raggedright #1
 }
 \addcontentsline{toc}{chapter}{\arabic{chapter}. #1}
}
```

Поясним, что `\chaptername` — это команда, которая печатает слово «Глава» (кстати, в некоторых случаях оказывается полезно переопределить эту команду, чтобы печаталось какое-то другое слово). Можно, в принципе, обойтись без этой команды и вставить слово «Глава» в нашу команду в явном виде; единственный недостаток такого решения в том, что мы не сможем использовать нашу команду в документах, написанных на других языках (на английском, например).

Мы не стали приводить здесь описания команд более низких уровней рубрикации, таких как подподсекции и абзацы; если вы почему-либо не

можете отказаться от столь глубокой рубрикации, опишите соответствующие команды по аналогии с приведенными в нашем примере.

Единожды написав свои команды рубрикации и перейдя к их использованию, вы получаете абсолютную власть над внешним видом заголовков, причем для любых изменений вам достаточно лишь поменять текст команд, не затрагивая собственно текст документа. Например, если от вас потребуют вставить в заголовки символ параграфа «§» (напомним, что этот символ набирается комбинацией «\S»), а сами заголовки центрировать относительно текста, просто измените команду \Section следующим образом:

```
\newcommand\Section[1]{
 \refstepcounter{section}
 \section*{\centering
  \S~\arabic{chapter}.\arabic{section}. #1}
 \addcontentsline{toc}{section}{%
  \arabic{chapter}.\arabic{section}. #1}
}
```

и аналогично поступите с командой \Subsection.

Более того, если по каким-то причинам вас не устраивает использование «звездочных» команд \section* и \subsection* (например, если от вас потребовали чего-то совсем экзотического), можно отказаться от них и оформить заголовки в полном соответствии даже с самыми сложными и неожиданными требованиями. Поступать так, однако, следует с известной осторожностью. Дело в том, что оригинальные версии команд рубрикации реализованы гораздо сложнее, чем в нашем примере. В частности, вы можете с неприятным удивлением обнаружить, что LaTeX поставил заголовок вашей секции последней строкой на странице, а текст секции начинается уже на следующей; настоящая команда \section так не сделает никогда. Не сделает так и ее «звездочная» форма, так что, пока мы используем ее, бояться нечего. Но как только мы окончательно забираем на себя управление внешним видом заголовка, LaTeX умывает руки и оставляет нас одних с нашими проблемами.

Чтобы решить проблему «повисших» заголовков, придется воспользоваться внутренней (то есть изначально предназначенной только для реализации пакетов LaTeX'а) командой \@afterheading, которую нужно вставить в конец описания ваших команд секционирования. Трудность тут в том, что в обычном режиме LaTeX воспринимает символ «@» как небуквенный и, соответственно, команду, в имени которой этот символ используется, мы просто так дать не можем. Придется временно (например, перед началом описания ваших команд секционирования) объявить символ «@» обычной буквой с помощью команды \makeatletter, а потом (когда все команды секционирования уже описаны) вернуть ему статус спецсимвола командой \makeatother.

11.2.2. Подписи к рисункам и таблицам

Для начала не лишним будет помнить, что положение подписи к таблице или рисунку зависит от того, где вы поставите команду `\caption`. Если от вас требуют, чтобы подписи к рисункам располагались снизу от рисунков, а заголовки таблиц — сверху от них,— не паникуйте, просто разместите команду `\caption` сразу после начала окружения таблицы (то есть на следующей строчке после `\begin{table}`).

К сожалению, на этом ваши приключения, скорее всего, не кончатся. Первое, чего от вас обязательно потребует консервативно настроенный научный руководитель (или, что хуже, издатель) — это заменить двоеточие после номеров таблиц и рисунков на точку. К счастью, сделать это очень просто, достаточно добавить в преамбулу такую строку:

```
\usepackage[labelsep=period]{caption}
```

Пакет `caption`, который мы только что подключили, позволяет менять внешний вид подписей к таблицам и рисункам, в частности, выбирать шрифт, которым они набираются, расположение (центрирование, прижатие к левому или правому краю), вид заголовков, состоящих из нескольких строк, максимальную ширину заголовков и т. д. За подробностями следует обращаться к соответствующей технической документации, мы же ограничимся одним (к сожалению, тоже часто требующимся) примером.

Вполне возможно, что от вас потребуется разделить подпись к рисунку или таблице на две строки, причем в первой расположить слово «Таблица» или «Рисунок», снабженное номером, а собственно название рисунка или таблицы разместить отдельной строкой, примерно так:

Таблица 11.1
Пример таблицы со странным заголовком

white	yellow	red	purple	green	blue
black	gray	magenta	pink	violet	orange

Если это нужно сделать и для таблиц, и для рисунков, то достаточно будет вставить в преамбулу после подключения пакета `caption` следующие две строки:

```
\DeclareCaptionLabelSeparator{par}{\par}
\captionsetup{labelsep=par,justification=centering}
```

Скорее всего, однако, так изменить вид подписей потребуется только для таблиц, а рисунки вас попросят оставить как есть. В этом случае следует использовать особенность команды `\captionsetup`: она, к счастью,

действует только на текущее окружение. Чтобы достичь требуемого эффекта, рекомендуется описать в преамбуле новую команду (пусть она называется `\tablecaption`):

```
\newcommand\tablecaption[1]{
  \captionsetup{labelsep=par,justification=centering}
  \caption{#1}
}
```

и поменять во всех описаниях таблиц (то есть в окружениях `table`) команду `\caption` на `\tablecaption`.

11.2.3. Заменяем «стандартные» слова

Следующая проблема, с которой вы можете столкнуться в процессе окончательного оформления работы — это слова, которые LaTeX печатает «сам». Если быть точным, эти слова задаются в пакете `babel`. Благодаря ему LaTeX выводит перед номером главы слово «Глава», а не какое-то другое; так же обстоят дела со словом «Оглавление»[4] перед оглавлением, со словом «Литература» перед библиографическим списком, со словами «Таблица» и «Рис.» в соответствующих подписях и т. п.

Вполне возможно, что какая-либо из выводимых LaTeX'ом надписей не понравится вашему научному руководителю (или редактору, если речь идет о публикации). Наиболее часто вызывает возражения слово «Литература»; вместо него требуют писать «Список литературы», «Библиография» или вовсе «Публикации». Эту ситуацию мы и рассмотрим в качестве примера.

Когда LaTeX выполняет команду `\thebibliography` (см. § 4), он первым делом печатает слово «Литература», но делает он это, используя, в свою очередь, команду `\bibname`, которая как раз и выдает то самое слово. Изначально класс документа определяет команду `\bibname` как «Bibliography», но пакет `babel` переопределяет ее, благодаря чему мы и видим слово «Литература». Однако же никто не мешает нам и самим переопределить команду `\bibname`, чтобы она выдавала то, что нужно нам. Итак, непосредственно перед окружением `thebibliography`[5] вставляем, например, следующую строку:

```
\renewcommand\bibname{Библиография}
```

В некоторых классах документов (например, в классе `article`) вместо команды `\bibname` используется команда `\refname`; по умолчанию она выдает надпись

[4] Для документов, в которых отсутствуют главы (например, для набранных с использованием класса `article`), выводится слово «Содержание».

[5] На самом деле, сделать это можно где угодно в тексте документа, но **не в преамбуле**. Объяснение, почему это так, оставляем за рамками пособия.

≪Список литературы≫. Если переопределение команды \bibname не дает эффекта, попробуйте переопределить \refname.

Аналогичным образом обстоят дела и с другими фрагментами текста, которые генерирует сам LaTeX. Команда \contentsname отвечает за заголовок перед оглавлением, команда \chaptername — за слово ≪Глава≫ в заголовках глав, команды \figurename и \tablename — за слова ≪Рис.≫ и ≪Таблица≫ в подписях к плавающим объектам. При переопределении всех этих команд **необходимо помнить, что такие переопределения должны быть помещены *после* строки** \begin{document}, а при размещении в преамбуле от них не будет никакого эффекта.

11.3. Тонкая настройка

В этой главе мы рассмотрим несколько приемов, позволяющих устранить некоторые неприятные особенности готового сверстанного текста. Подчеркнем, что к описанным здесь приемам следует прибегать **в самый последний момент**, после того, как рукопись полностью набрана. Любые правки, внесенные в текст после тонкой настройки, могут сделать команды тонкой настройки бессмысленными и даже вредными.

11.3.1. Управление страницами

Перечислим для начала ситуации, которых нам хотелось бы избежать:

- страницу нежелательно начинать с неполной строки, то есть с последней строки абзаца, которая короче других строк;
- нежелательно заканчивать страницу первой строкой абзаца, имеющей отступ (красную строку);
- если в вашем документе имеются листинги программ, описанные в § 8.1, нежелательно, чтобы переход со страницы на страницу разбил ваш листинг, отделив одну-две строчки. Небольшие (до четырехпяти строк) листинги вообще нежелательно разрывать, а листинги бóльшего объема при возникновении такой необходимости лучше разрывать ближе к середине (во всяком случае, на расстоянии не менее трех-четырех строк от начала и конца);
- если в вашем тексте имеются перечисления (см. § 6), нежелательно разносить на соседние страницы строки одного пункта. Особенно неказисто смотрятся одна или две последние строки пункта перечисления, оказавшиеся в начале страницы.

Проще всего справиться с «висящими» строками, то есть с разрывом абзаца после первой строки или перед последней. LaTeX имеет для этого готовые средства; достаточно вставить в преамбулу строки

```
\clubpenalty=9999
\widowpenalty=9999
```

и практически все случаи «висящих» строк исчезнут. Первая команда отвечает за разрыв абзаца после первой строки, то есть когда первая строка абзаца оказывается последней строкой страницы, а вторая — за разрыв перед концом абзаца, то есть когда страница начинается с неполной строки. Число 9999 в данном случае означает степень серьезности запрета. Максимальное значение — 10000, так что мы такими командами запрещаем LaTeX'у неудачно разрывать абзацы, кроме совсем уж крайних случаев. Можно использовать и значение 10000, то есть запретить соответствующие разрывы совсем, категорически и безоговорочно, а можно при желании и снизить строгость запрета.

С некрасиво разорванным листингом проще всего справиться, объявив его плавающим объектом типа «рисунок» (см. § 7.1), однако это не всегда вписывается в общую концепцию вашей верстки. Остальные ситуации требуют использования команд *явного управления страницами*.

Принудительно завершить страницу в любом удобном месте можно с помощью команд `\newpage` и `\clearpage`. Разница между ними в том, что `\newpage` просто обрывает страницу и начинает новую, тогда как `\clearpage` перед этим выводит все накопленные плавающие объекты.

Обе команды `\newpage` и `\clearpage` оставляют внизу страницы пустое место. В качестве альтернативы можно попробовать применить команду `\pagebreak`, которая растянет по вертикали весь имеющийся на странице материал, заполнив все вертикальное пространство равномерно. В некоторых случаях может оказаться удобной команда `\nopagebreak`, запрещающая разрыв страницы в данном месте.

При необходимости можно увеличить размер отдельной страницы, например, чтобы уместить на ней последнюю строку абзаца, не перенося ее на следующую страницу. Это делается командой `\enlargethispage{`<*длина*>`}`. Длину лучше задать с привязкой к высоте строки; например, `\enlargethispage{\baselineskip}` позволит разместить на странице ровно на одну строку больше. Использовать эту команду следует с осторожностью, поскольку физически лист, конечно, не увеличится, так что вертикальный размер полосы набора будет увеличен за счет полей.

Отметим, что `\enlargethispage` можно использовать и для уменьшения высоты страницы, если задать отрицательную *длину*. Во многих случаях оказывается удобнее не увеличивать количество строк на странице, стараясь впихнуть туда неудачно выпавшую строку, а, наоборот,

уменьшить страницу, чтобы, скажем, вместе с неудачной строкой на следующую страницу ушла и предыдущая строка. Такой прием вполне способен исправить ситуацию и не сопряжен с лишним риском.

11.3.2. Расстановка переносов

Обычно LaTeX самостоятельно расставляет в словах переносы, однако в некоторых случаях он с этой задачей может и не справиться. Например, слово, содержащее дефис, LaTeX переносит только по дефису (заметим, что по правилам это допустимо, но нежелательно), а некоторых сложных слов он попросту не знает.

Чтобы найти места, где необходимо ваше вмешательство в механизм переноса, следует установить режимы `draft` и `\fussy`, как это описано в § 1.2.1, оттранслировать ваш документ и внимательно просмотреть его в поисках слов, «заехавших» на правое поле текста. Именно в этих словах и следует расставить точки переноса вручную. Делается это комбинацией «`\-`», например: `квад\-рат\-но-гнез\-до\-вой`.

Не забудьте по окончании работы с переносами вернуть режимы `final` (вместо `draft`) и `\sloppy` (вместо `\fussy`).

Литература

[1] Дональд Е. Кнут. *Все про TEX.* Изд-во АО RDTeX, Протвино, 1993.

[2] А. И. Роженко. *Искусство верстки в LATEX'е.* Изд-во ИВМиМГ СО РАН, Новосибирск, 2005.

[3] С. М. Львовский. *Набор и верстка в системе LATEX.* М.: МЦНМО, 2006

[4] М. Гуссенс, Ф. Миттельбах, А. Самарин. *Путеводитель по пакету LATEX и его расширению LATEX2e.* М.: МИР, 1999.

Домашняя страница этой книги в сети Интернет расположена по адресу `http://www.stolyarov.info/books/latex3days`. Здесь вы можете получить тексты примеров, приведенных в этой книге, а также электронную версию самой книги.

Оглавление

СТОЛЯРОВ Андрей Викторович

СВЕРСТАЙ ДИПЛОМ КРАСИВО: LaTeX ЗА ТРИ ДНЯ

Корректор Е. Ясиницкая

Напечатано с готового оригинал-макета

Подписано в печать 22.10.2010 г.
Формат 60x90 1/16. Усл.печ.л. 6.25. Тираж 200 экз. Заказ 468.

Издательство ООО "МАКС Пресс"
Лицензия ИД № 00510 от 01.12.99 г.

11992 ГСП-2, Москва, Ленинские горы,
МГУ им. М.В.Ломоносова, 2-й учебный корпус, 627 к.
Тел. 939-3890, 939-3891. Тел./Факс 939-3891