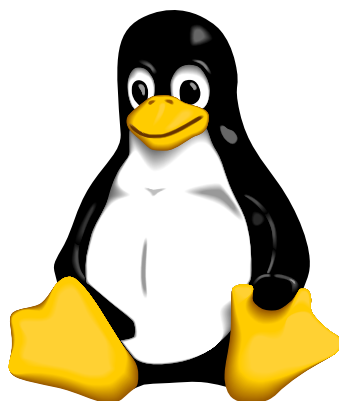


# L<sup>A</sup>T<sub>E</sub>X, GNU/Linux и русский стиль.

© Е. М. Балдин\*



## L<sup>A</sup>T<sub>E</sub>X в России



Данный текст распространяется под лицензией Creative Commons Attribution-Share Alike 3.0 License (CC-BY-SA-3.0). Если будет необходимость перелицензировать его под другой свободной лицензией, то свяжитесь со мной по электронной почте. Замечания и предложения принимаются с благодарностью.

---

\*e-mail: [E.M.Baldin@inp.nsk.su](mailto:E.M.Baldin@inp.nsk.su)

Эмблемы T<sub>E</sub>X и METAFont, созданные Дуайном Библи, взяты со странички Д.Э. Кнута.

# Оглавление

<b>1. Л<sup>A</sup>T<sub>E</sub>X— компьютерная типография</b>	<b>1</b>
1.1. Немного истории . . . . .	1
1.1.1. Доисторический период . . . . .	1
1.1.2. Роль Человека в истории . . . . .	2
1.1.3. Техническое отступление . . . . .	2
1.1.4. Дистрибутивы Л <sup>A</sup> T <sub>E</sub> X . . . . .	3
1.2. Запускаем Л <sup>A</sup> T <sub>E</sub> X . . . . .	5
1.3. Проблемы с компиляцией . . . . .	7
1.4. Л <sup>A</sup> T <sub>E</sub> X-конвейер . . . . .	8
1.5. Литература . . . . .	10
1.5.1. Классика . . . . .	11
1.5.2. Учебники и справочники . . . . .	11
1.5.3. L <sup>A</sup> T <sub>E</sub> X в России . . . . .	12
1.6. Список рассылки . . . . .	12
1.7. Благодарности . . . . .	12
<b>2. Базовые элементы</b>	<b>13</b>
2.1. «Командная логика» . . . . .	13
2.2. Логика документа . . . . .	15
2.2.1. Структура Л <sup>A</sup> T <sub>E</sub> X-файла . . . . .	15
2.2.2. Класс документа . . . . .	16
2.2.3. Стили . . . . .	17
2.2.4. Тело документа . . . . .	18
2.3. Логика набора . . . . .	18
2.3.1. Печатаем текст . . . . .	18
2.3.2. Пунктуация . . . . .	19
2.4. Структурная логика . . . . .	21
2.4.1. Титульный лист . . . . .	21
2.4.2. Секционирование . . . . .	22
2.4.3. Перекрёстные ссылки . . . . .	23
2.4.4. Сложные документы . . . . .	23

<b>3. Набор математики</b>	<b>27</b>
3.1. Набор формул . . . . .	28
3.2. Кириллица в формулах . . . . .	29
3.3. Школьная математика . . . . .	30
3.3.1. Индексы . . . . .	30
3.3.2. Математические символы . . . . .	31
3.3.3. Дроби . . . . .	32
3.3.4. Корни . . . . .	32
3.3.5. Квадратное уравнение . . . . .	33
3.3.6. Скобки . . . . .	33
3.3.7. Функции . . . . .	34
3.3.8. Производная и интеграл . . . . .	35
3.4. Перенос формул . . . . .	36
3.5. Заключение . . . . .	37
<b>4. Графика</b>	<b>39</b>
4.1. Encapsulated PostScript . . . . .	39
4.2. Как из растра сделать EPS . . . . .	41
4.3. graphicx . . . . .	42
4.4. Плавающие объекты . . . . .	44
4.5. Заключение . . . . .	49
<b>5. Документация и программный код</b>	<b>51</b>
5.1. Спецсредства . . . . .	51
5.1.1. keystroke . . . . .	51
5.1.2. LCD-дисплей . . . . .	52
5.1.3. Битовые поля . . . . .	53
5.2. Форматирование кода . . . . .	54
5.2.1. verbatim . . . . .	55
5.2.2. listings . . . . .	55
5.3. Представление алгоритмов . . . . .	58
5.3.1. algorithms . . . . .	58
5.3.2. Клоны algorithm . . . . .	59
5.3.3. clrscod . . . . .	60
5.3.4. pseudocode . . . . .	60
5.4. Заключение . . . . .	60
<b>6. Вёрстка I</b>	<b>63</b>
6.1. Определённые «размеры» и переменные «длины» . . . . .	63
6.2. Скелет страницы . . . . .	66
6.3. Меняем макет . . . . .	69
6.3.1. Двигаем размеры . . . . .	69
6.3.2. Стили страницы . . . . .	71
6.4. Причёсываем текст . . . . .	72

6.5. Послесловие . . . . .	75
<b>7. Путеводитель по классам <math>\LaTeX</math></b>	<b>76</b>
7.1. Зачем нужны эти классы? . . . . .	76
7.2. Классовая база . . . . .	77
7.3. Классификация . . . . .	78
7.3.1. Модификации и улучшения базы . . . . .	78
7.3.2. Поддерживаем стандарты . . . . .	79
7.3.3. Пишем письма . . . . .	80
7.3.4. Верстаем книги . . . . .	80
7.3.5. Создаём отчёты . . . . .	81
7.3.6. Делаем презентации . . . . .	81
7.3.7. Защищаем диссертации . . . . .	82
7.3.8. Организуем резюме . . . . .	83
7.3.9. Журнальные и конференционные классы . . . . .	83
7.3.10. Всякая всячина . . . . .	84
<b>8. Делаем презентации I</b>	<b>86</b>
8.1. slides . . . . .	86
8.2. Немного о PDF . . . . .	87
8.2.1. Простота создания . . . . .	87
8.2.2. Переносимость . . . . .	88
8.2.3. Интерактивность . . . . .	89
8.3. beamer . . . . .	89
8.4. Правила хорошей презентации . . . . .	96
<b>9. Справочно-поисковый аппарат издания</b>	<b>97</b>
9.1. Рубрикация и оглавление . . . . .	98
9.2. Ссылки и примечания . . . . .	100
9.2.1. Механизм ссылок . . . . .	100
9.2.2. Подстрочные примечания . . . . .	101
9.3. Колонтитулы . . . . .	102
9.4. Библиография . . . . .	102
9.4.1. В $\LaTeX$ . . . . .	103
9.5. Алфавитный указатель . . . . .	107
<b>10. Всё о таблицах</b>	<b>111</b>
10.1. Немного теории . . . . .	111
10.2. tabbing . . . . .	112
10.3. tabular и array . . . . .	113
10.3.1. К вопросу о разделительных линиях . . . . .	115
10.3.2. Клетки . . . . .	115
10.3.3. Выравнивание чисел . . . . .	117
10.3.4. Доступ к данным . . . . .	117

## Оглавление

10.3.5. Клоны <code>tabular</code> . . . . .	117
10.4. Многополосные таблицы . . . . .	118
10.5. Вывод . . . . .	119
10.6. И это тоже таблицы? . . . . .	120
10.7. В заключение о таблицах . . . . .	120
<b>11. Начала программирования</b>	<b>121</b>
11.1. Создаём свои . . . . .	121
11.2. Счётчики и другие переменные . . . . .	123
11.3. Создаём свой пакет . . . . .	126
11.3.1. Установочный <code>ins</code> -файл . . . . .	126
11.3.2. Пакетный <code>dtx</code> -файл . . . . .	127
11.3.3. Пакетирование . . . . .	129
11.4. Напутствие . . . . .	130

# L<sup>A</sup>T<sub>E</sub>X — компьютерная типография

L<sup>A</sup>T<sub>E</sub>X — это истинная T<sub>E</sub>Xнология.  
Никогда ещё создание книг не было таким интересным.

---

Я действительно так думаю.

Человеческая цивилизация зависит от книг. Передача знаний от поколения к поколению — это то, что делает человека разумным. Написание книги всегда было/есть/будет одним из самых сложных видов деятельности. L<sup>A</sup>T<sub>E</sub>X берёт на себя техническую часть по подготовке рукописи, оставляя человеку больше времени на творчество, и, в тоже время, позволяя ему контролировать весь процесс создания от начала и до конца.

«Если кто-то другой набирает ваше произведение, то у вас нет возможности контролировать появление ошибок; если же вы выполняете эту работу самостоятельно, то можете винить только себя» — Дональд Э. Кнут.

## 1.1. Немного истории

Есть популярная идея по поводу того, что «история учит тому, что ничему не учит». Возможно, это так. Но чтобы понять логику текущих событий и явлений, всё-таки необходимо знать как «оно» зарождалось и почему «оно» до сих пор существует.

### 1.1.1. Доисторический период

Сначала вообще не было компьютеров, и люди всё писали вручную. Но прогресс неумолим, и вслед за печатной машиной появилась коммерческая выгода от создания книг. Время шло, процесс печати удешевлялся — все были довольны, пока не случилось ...

### 1.1.2. Роль Человека в истории

Дональд Эрвин Кнут (Donald Ervin Knuth) является одним из немногих людей, благодаря которым информатика заслуженно носит звание научной дисциплины. Произведение, которое принесло ему широкую известность — это «пятитомник» «Искусство программирования»<sup>1</sup>. После того как в 1975 году был издан третий том «пятитомника», издатель окончательно избавился от печатной машины с металлическим набором типа «монотип» и заменил его на фотонаборное устройство. Результат превзошёл все ожидания: получив оттиски, сделанные по новой технологии, Д. Э. Кнут, который как раз подготовил второе издание второго тома, сильно загрустил. Сама мысль, что книги, написание которых он потратил свыше пятнадцати лет, будут так плохо выглядеть, не давала Кнуту покоя.

Новые машины были не аналоговыми, а дискретными — буквы составлялись из точек. «Это объект для компьютерной науки,» — подумал Кнут и решил научить компьютер делать буквы из точек такими, как надо, то есть красивыми. Поначалу задача казалась несложной. Кнут потом признавал, что это был его личный рекорд по недооценке сложности проекта.

«Лучший способ разобраться до конца — это попробовать научить этому компьютер.» — Д. Э. Кнут.

Примерно через десять лет после начала работы над проектом системы METAFONT (создание шрифтов) и  $\TeX$  (лучшая программа разбиения абзацев на строки) были стабилизированы (версия 2.7 для METAFONT и 3.1 для  $\TeX$ ). Кнут отошёл от активной разработки. В дальнейшем METAFONT и  $\TeX$  модифицировались только с целью исправления ошибок. На текущий момент номер версии METAFONT равен 2.71828, а  $\TeX$  — 3.141592. Кнут завещал, что после его смерти номера версий будут заморожены и равны числу  $e$  и числу  $\pi$  соответственно, а все оставшиеся неисправленные ошибки будут считаться особенностями реализации.

На текущий момент  $\TeX$ , скорее всего, самая свободная от ошибок программа. Код  $\TeX$  выпускался отдельной книгой «TeX: The Program» (ISBN: 0201134373), за обнаружение ошибки в своей программе Кнут выплачивает вознаграждение.  $\TeX$  является примером свободной программы, которая возникла в академической среде задолго до наступления эпохи GPL.

«Математическая формула не может быть чьей-то „собственностью“! Она принадлежит Богу.» — Д. Э. Кнут.

Сегодня мастер на пенсии и всё своё время посвящает написанию «пятитомника». На его домашней страничке можно заметить, что дело явно движется. Ждём результата с нетерпением.

### 1.1.3. Техническое отступление

В основу  $\TeX$  была заложена относительно простая идея.  $\TeX$  работает только с боксами (box) и клеем (glue). Элементарные боксы — это буквы, которые объ-

---

<sup>1</sup>На сайте мастера <http://www-cs-faculty.stanford.edu/~knuth/> можно найти предварительные версии пока неопубликованных глав четвёртого тома.

единяются в боксы-слова, которые, в свою очередь, сливаются в боксы-строчки, боксы-абзацы и так далее. Между боксами «разлит» клей, который имеет ширину по умолчанию и степени увеличения/уменьшения этой ширины. Объединяясь в бокс более высокого порядка, элементарные боксы могут шевелиться, но после того как найдено оптимальное решение, это состояние замораживается и полученный бокс выступает как единое целое. Оптимальное решение находится с помощью системы штрафов за то, что клея больше или меньше чем оптимальное значение, а также за разрывы абзаца в неподходящем месте. Чем меньше штрафа было получено, тем размещение «красивее». В зависимости от системы штрафов меняется форматирование.

Первоначально Кнут предполагал, что у T<sub>E</sub>X будет множество модификаций, то есть каждая типография будет держать мастера-T<sub>E</sub>Xника для создания своей версии T<sub>E</sub>X под свои нужды. В начале T<sub>E</sub>X не являлся в полном смысле языком программирования. Управляющие конструкции были добавлены в него позже, когда стало понятно, что развитие T<sub>E</sub>Xнологии пошло совсем по другому пути.

А началось всё с Лесли Лэмпорта, который в начале 80-х годов начал разработку издательской системы Л<sup>A</sup>T<sub>E</sub>X, в основе которой лежал T<sub>E</sub>X. Л<sup>A</sup>T<sub>E</sub>X представляет собой набор макросов на языке T<sub>E</sub>X, позволяющих решить ту или иную задачу. Иными словами, это сборник рецептов. Чтобы выбрать сценарий стирки на автоматической стиральной машине, нет необходимости думать в терминах скорости вращения, уровня воды и количества порошка — достаточно просто выбирать готовое решение. Чтобы пользоваться системой Л<sup>A</sup>T<sub>E</sub>X, не надо быть T<sub>E</sub>Xником — достаточно выбрать готовый стиль и использовать несколько простых команд в зависимости от того, что нужно сделать.

#### 1.1.4. Дистрибутивы Л<sup>A</sup>T<sub>E</sub>X

Л<sup>A</sup>T<sub>E</sub>X так же, собственно говоря, как и Linux, не является монолитной программой. Л<sup>A</sup>T<sub>E</sub>X состоит из набора пакетов/программ, причём набор пакетов не фиксирован, что позволяет создавать дистрибутивы, преследующие ту или иную цель.

На сегодня все дистрибутивы Л<sup>A</sup>T<sub>E</sub>X имеют общий корень, и этот корень носит название CTAN или The Comprehensive TeX Archive Network (<http://www.ctan.org>). CTAN — это репозиторий, в который стекаются все сколько-нибудь стоящие разработки в области T<sub>E</sub>Xстроения. CTAN — это множество зеркалируемых серверов по всему миру. Модель была настолько успешна, что её на вооружение взяло perl-сообщество, организовав CPAN — The Comprehensive Perl Archive Network.

Наиболее известным в среде GNU/Linux является дистрибутив teTeX<sup>2</sup>. Если Вы не знаете, что за дистрибутив стоит на вашем компьютере, то это, скорее всего, teTeX. Этот дистрибутив был создан Томасом Эшером (Thomas Esser). Первая публичная версия в 1994 умещалась на три дискетки. В мае этого (2006) года Томас принял решение о прекращении поддержки своего детища в пользу настоящего

---

<sup>2</sup><http://www.tug.org/teTeX/>



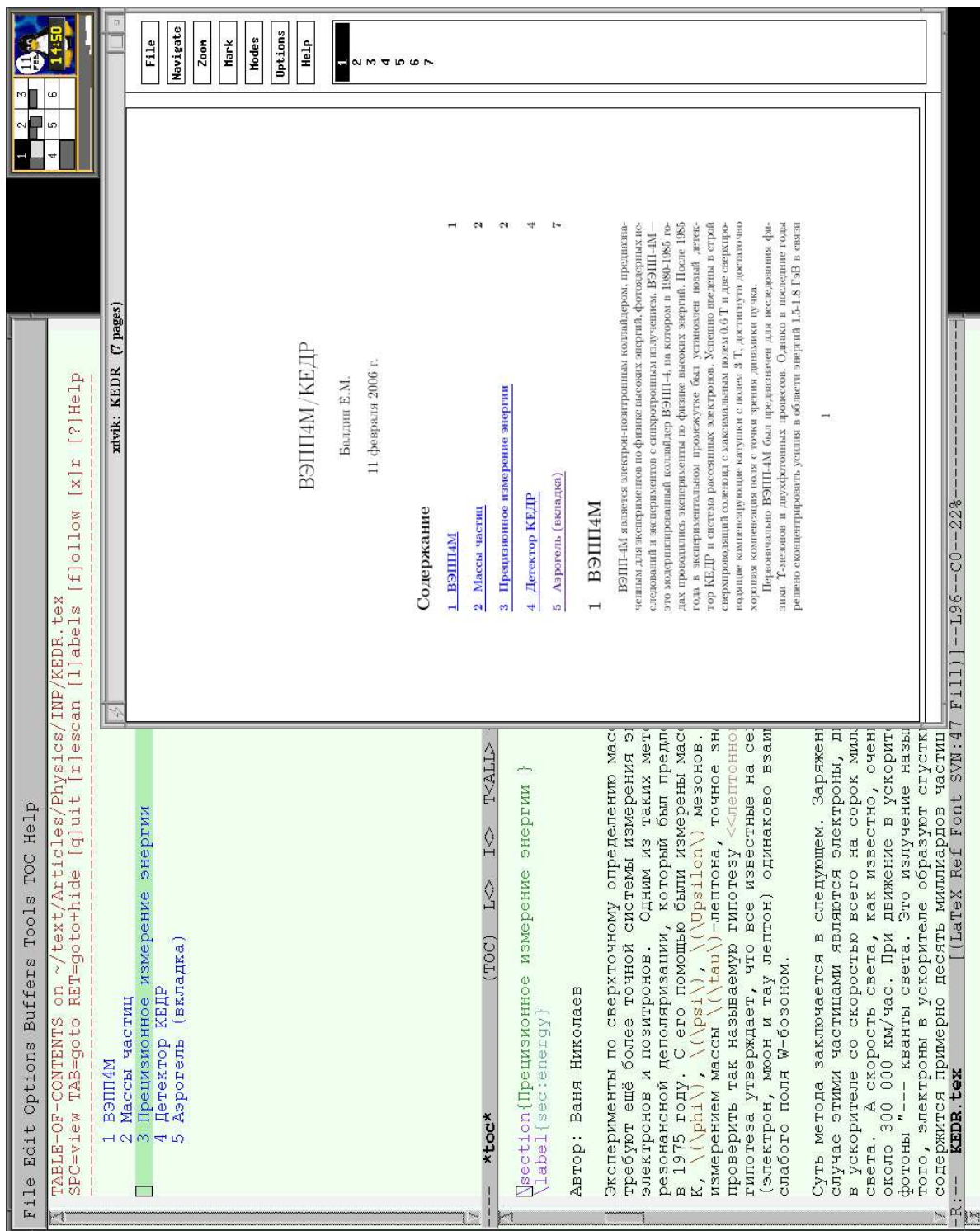


Рис. 1.1. Пример набора текста в текстовом редакторе слева. Просмотр результата набора справа.

флагмана T<sub>E</sub>X-сообщества (TUG <http://www.tug.org/> — T<sub>E</sub>X User Groups) — дистрибутива T<sub>E</sub>X Live<sup>3</sup>.

T<sub>E</sub>X Live создавался как дистрибутив, который можно было запускать прямо с CD. Базой для дистрибутива стал teTeX. Первая версия T<sub>E</sub>X Live была выпущена в 1996 году. Со временем дистрибутив рос и к 2003 году T<sub>E</sub>X Live стал умещаться только на DVD. Каждый год выпускается новая версия дистрибутива. Текущая нумерация идёт по номеру года. T<sub>E</sub>X Live поддерживает все сколько-нибудь распространённые платформы. T<sub>E</sub>X вообще отличается исключительной переносимостью.

T<sub>E</sub>X Live — это свободный софт. Мантейнеры дистрибутива используют определение понятия свободы, принятое Фондом открытого программного обеспечения (<http://www.gnu.org/philosophy/free-sw.html> — Free Software Foundation’s definition) или сообществом Debian ([http://www.debian.org/social\\_contract](http://www.debian.org/social_contract) — Debian Free Software Guidelines). В случае конфликтов этих определений обычно принимается сторона FSF.

На DVD, который шёл с майским номером журнала Linux Format (Номер 9 (79) Май 2006), был размещён дистрибутив T<sub>E</sub>X Live. Так что этот дистрибутив вполне можно «потрогать руками» уже сейчас.

## 1.2. Запускаем Л<sup>A</sup>T<sub>E</sub>X

Предполагается, что дистрибутив Л<sup>A</sup>T<sub>E</sub>X на вашем компьютере уже установлен и настроен. Если это не так, то потревожьте суперпользователя по этому поводу. Обычно проблем с установкой не возникает. Возможные шероховатости с русским языком в современных дистрибутивах возникают по недоразумению. Основной дистрибутив GNU/Linux на сегодня — это **tetex**. В будущем по возможности рекомендуется переходить на T<sub>E</sub>X Live.

По традиции для начала скажем «Здравствуй, мир!!!». Для этого в любом удобном для вас текстовом редакторе<sup>4</sup> создайте файл `helloworld.tex` следующего содержания:

```
%helloworld.tex
% Выбор класса документа
\documentclass{article}
% Чтобы можно было использовать русские буквы в формулах
%но в случае использования предупреждать об этом
\usepackage[warn]{mathtext}
% Выбор внутренней TEX-кодировки (можно опустить)
\usepackage[T2A]{fontenc}
% Выбор кодовой страницы документа.
%Так же можно выбрать cp1251 или koi8-r.
\usepackage[utf8]{inputenc}
```

<sup>3</sup><http://www.tug.org/texlive/>

<sup>4</sup>Было бы лучше, если бы этот редактор оказался **emacs**’ом ☺.

```
% Выбор языка документа.
\usepackage[english,russian]{babel}
% Начинать первый параграф с красной строки.
\usepackage{indentfirst}
% Конец преамбулы и начало текста.
\begin{document}
% Поздравляем мир.
\LARGE Здравствуй, мир!!!
% Конец текста.
\end{document}
```

Комментарии, которые начинаются со знака % можно опускать. Всё, что идёт до `\begin{document}`, называется преамбулой или «шапкой». Преамбула определяет вид итогового документа. Нет необходимости каждый раз набивать эти строчки с нуля. Для этого достаточно обучить текстовый редактор вставлять их автоматически при создании нового `tex`-файла. После создания текста его необходимо откомпилировать:

```
> latex helloworld.tex
This is pdfTeX, Version 3.141592-1.30.3-2.2 (Web2C 7.5.5)
%&-line parsing enabled.
entering extended mode
(./helloworld.tex
LaTeX2e <2003/12/01>
```

...

```
(./helloworld.aux) [1] (./helloworld.aux) )
Output written on helloworld.dvi (1 page, 240 bytes).
Transcript written on helloworld.log.
>
```

В качестве результата *Л<sup>A</sup>T<sub>E</sub>X* выдаёт файл `helloworld.dvi`. Далее есть выбор:

- посмотреть результат с помощью **xdvi**:
 

```
> xdvi helloworld.dvi
```
- преобразовать `dvi` в PostScript<sup>5</sup> и посмотреть его с помощью **gv**, а потом распечатать на PostScript-принтере (если он есть, естественно):
 

```
> dvips helloworld
> gv helloworld.ps
> lpr helloworld.ps
```

---

<sup>5</sup>PostScript — язык описания страниц, разработан Джоном Уорноком и Чаком Гешке из Adobe Systems. Интерпретаторы PostScript, аппаратные или программные (`ghostscript`), широко используются при печати документов.

- сделать PDF<sup>6</sup>, ну и, естественно, посмотреть его с помощью Acrobat Reader:
  - > `dvips helloworld`
  - > `ps2pdf helloworld.ps helloworld.pdf`
  - > `acroread helloworld.pdf`

Во всех случаях на экране будет отображено одно и то же:

Здравствуй, мир!!!

Рис. 1.2. «Здравствуй, мир» от Л<sup>A</sup>T<sub>E</sub>X.

### 1.3. Проблемы с компиляцией

Случается, что при наборе делается ошибка, и тогда при компиляции исходника Л<sup>A</sup>T<sub>E</sub>X может затребовать дополнительную информацию.

Если просто запустить `latex` без каких либо инструкций, то на экране появится приглашение:

```
> latex
This is pdfTeX, Version 3.141592-1.30.3-2.2 (Web2C 7.5.5)
%&-line parsing enabled.
**
```

Л<sup>A</sup>T<sub>E</sub>X ждёт ввода имени текстового файла, чтобы начать его обработку. Можно прервать ожидание по `^C`. Если же правильно задать файл при запуске `latex`, но при этом ошибиться в коде, то Л<sup>A</sup>T<sub>E</sub>X выдаст сообщение об ошибке с номером строчки, где возникла проблема и предложит сделать выбор:

```
! Undefined control sequence.
l.11 \errorinbody
```

```
? h
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., '\hobx'), type 'I' and the correct
spelling (e.g., 'I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.
```

```
? x
```

---

<sup>6</sup>PDF—Portable Document Format. Этот формат, как и PostScript, создан фирмой Adobe Systems. Является стандартом для электронной документации.

На запрос (?) можно ввести `h`, тогда будет выдана догадка о том, с чем может быть связана ошибка, или `x`, для того чтобы прервать выполнение компиляции. В случае обычного перевода строки компиляция продолжится до следующей ошибки или до самого конца. Краткую информацию об управляющих командах можно получить, введя ?.

*Добрый совет:* увидел ошибку — сразу исправил. Следующие предупреждения могут быть следствием предыдущей ошибки.

Интерактивный режим для работы с ошибками ЛАТЭХ — довольно мощный инструмент для их разбора, но на первых порах лучше следовать «Доброму совету». Текстовые редакторы, в которых предусмотрена поддержка редактирования исходников ЛАТЭХ, обычно на основании сообщения об ошибке позволяют её локализовать.

## 1.4. ЛАТЭХ-конвейер

В процессе работы ЛАТЭХ читает и записывает несколько файлов. Полезно знать, что это за файлы и зачем они нужны.

На вход подаётся текстовый файл с ЛАТЭХ-разметкой. Традиционно файл имеет расширение `tex`.

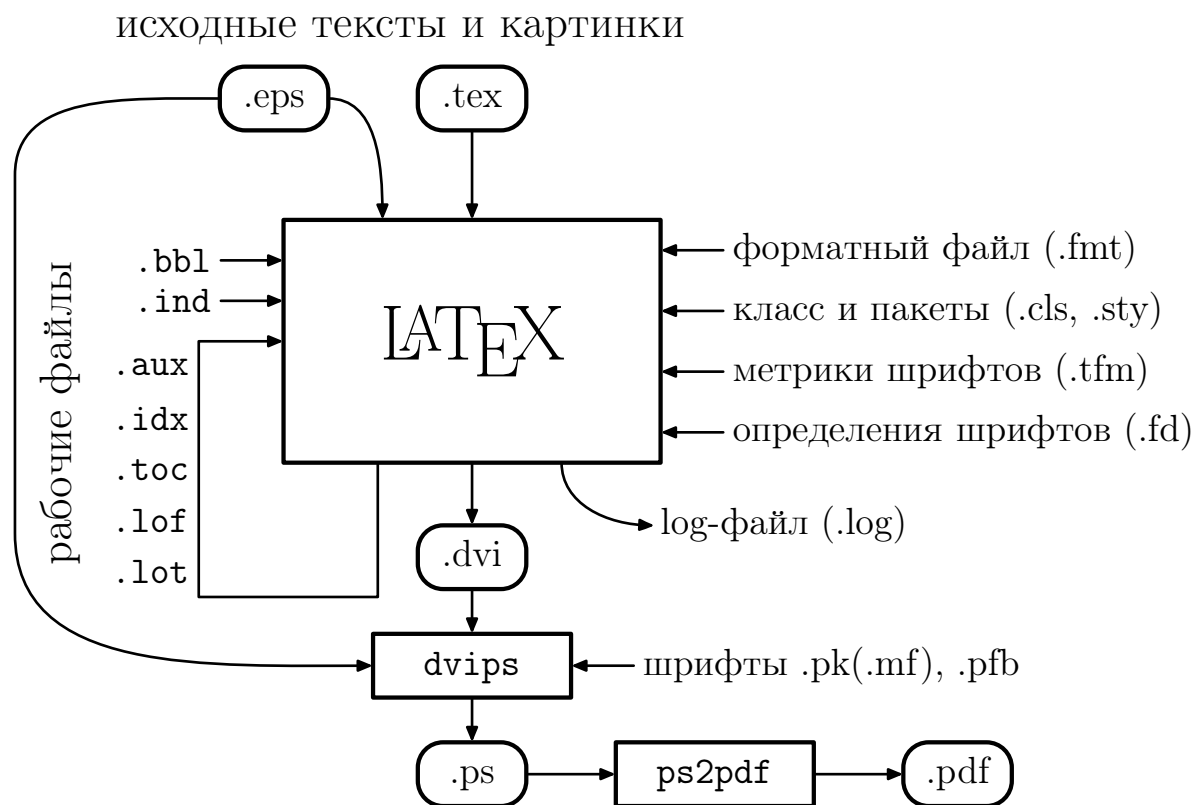
В качестве результата на выходе получается файл с тем же самым именем, что и на входе, но с расширением `dvi`. `dvi` — device independent (не зависящий от устройства) формат, который хранит информацию о форматировании текста и размещении всех его элементов на странице, но без самих букв и картинок. Программы преобразующие `dvi`-файл в другое представление называются `dvi`-драйверами.

Программа `xdvi` преобразует `dvi`-файл в картинку на экране монитора. Это очень продвинутый драйвер. Он реалистично представляет вид напечатанной страницы, поддерживает гиперссылки и позволяет организовать обратную связь с текстом. Ещё одним популярным `dvi`-драйвером является **dvips**. **dvips** производит качественный PostScript, который уже можно распечатать на принтере либо напрямую (если принтер поддерживает PostScript аппаратно), либо через программный интерпретатор `ghostscript`. Существуют и другие `dvi`-драйверы, например, **dvi2tty** пытается перевести `dvi` обратно в текст, **dvilj** переводит `dvi` в инструкции РСЛ для принтеров HP, **dvipdf** транслирует `dvi` в `pdf`. Обилие `dvi` драйверов позволяет рассмотреть/напечатать `dvi` файлы практически в любой ситуации.

Свободный программный интерпретатор Ghostscript (**gs**), в свою очередь, позволяет преобразовывать PostScript файлы (`.ps`) в другие форматы. Обычно PDF получают именно из PostScript с помощью скрипта **ps2pdf**.

Графика в ЛАТЭХ добавляется через `eps`-файлы. EPS или Encapsulated PostScript — это векторный графический формат, который представляет собой инструкции на языке PostScript с некоторыми ограничениями. Одно из основных требований заключается в том, что в заголовке `eps`-файла обязательно должны быть указаны его размеры (BoundingBox). Пример заголовка `eps`-файла, созданного на основе примера «Здравствуй, мир!!!»:

```
%!PS-Adobe-2.0 EPSF-2.0
```

Рис. 1.3.  $\LaTeX$ -конвейер.

```

%%Creator: dvips(k) 5.95b Copyright 2005 Radical Eye Software
%%Title: helloworld.dvi
%%BoundingBox: 148 651 288 668
%%DocumentFonts: SFRM1728
%%EndComments
  
```

Перечислим теперь остальные файлы, которые используются в процессе работы:

#### 1) Внешние файлы

**fmt** форматный файл. Содержит, главным образом, все команды  $\LaTeX$ 'а в предварительно откомпилированной форме. Также содержит информацию о переносах. При изменении значений глобальных переменных  $\TeX$  требуется пересборка форматного файла: `texconfig init`.

**cls, sty** определение макета и структуры документа. Класс документа (`.cls`) выбирается с помощью инструкции `\documentclass`. Дополнительные возможности и изменение поведения класса по умолчанию достигаются с помощью пакетов (`.sty`), выбираемых посредством инструкции `\usepackage`.

**tfm** метрики шрифтов. Размеры и правила взаимодействия литер друг с другом.

**fd** приведение внешних названий шрифтов к виду, принятому в  $\LaTeX$ .



`ps` векторные Type1-шрифты.

`pk` (`mf`) растровые `pk`-шрифты по мере необходимости создаются из векторных `mf`-шрифтов (`METAFONT`) с необходимым разрешением. При печати качество не страдает, но при просмотре из-за низкого разрешения дисплея предпочтительнее векторные шрифты.

2) Файлы, создаваемые в процессе Л<sup>A</sup>T<sub>E</sub>X-конвейера. Обычно для получения итоговой копии документа требуется несколько раз запускать `latex`. При первом проходе записывается некоторое число информационных файлов, которые при последующих проходах используются для нумерации ссылок, составления оглавления и тому подобного.

`log` файл протокола. В него выводится вся информация, имеющая отношение к компиляции. Фактически дублирует стандартный вывод на экран.

`aux` информация о перекрёстных ссылках.

`toc` файл оглавления (table of contents)

`lof` список иллюстраций (list of figures)

`lot` список таблиц (list of tables)

`bb1` список литературы сформированный с помощью программы `ВibTEX`. На первых порах можно обойтись без этой T<sub>E</sub>X-технологии, но для серьёзных проектов управление списком литературы без автоматизации становится не простой задачей.

`ind` предметный указатель, сформированный программой `MakeIndex`. В каждой уважающей себя несущей полезную информацию книге есть такой. Для составления используются `idx`-файлы. Организация указателя — это отдельная задача.

Выше описан «классический» Л<sup>A</sup>T<sub>E</sub>X-конвейер. Программа `latex` может быть заменена на `pdflatex`, тогда на выходе сразу будет получаться pdf, а графическая информация должна быть представлена в форматах `png` или `pdf`. Возможны и другие вариации, но в целом структура остаётся той же.

## 1.5. Литература

Л<sup>A</sup>T<sub>E</sub>X'у уже свыше двадцати лет. За время своего существования многие из его частей существенно переделывались и совершенствовались. База же, в виде T<sub>E</sub>X, до сих пор остаётся стабильной основой. Видимо, поэтому документация к Л<sup>A</sup>T<sub>E</sub>X устаревает чрезвычайно медленно.

Доступных в России книг по Л<sup>A</sup>T<sub>E</sub>X относительно немного. С другой стороны, если удастся добыть хоть одну из перечисленных в этом разделе, то для обычного набора её, скорее всего, будет достаточно. Логичная организация позволяет Л<sup>A</sup>T<sub>E</sub>X расширяться, не сильно ломая совместимость.

Для более подробного ознакомления с конкретными пакетами следует обратиться к документации идущей с пакетом. Чего-чего, а описаний в дистрибутиве Л<sup>A</sup>T<sub>E</sub>X хватает. Также для поиска того или иного решения можно обратиться к сайту CTAN (<http://www.ctan.org>) или TUG (<http://www.tug.org>).

### 1.5.1. Классика

Д. Э. Кнут «Всё про T<sub>E</sub>X» [1]. Библия T<sub>E</sub>X. Для обычного набора текстов в Л<sup>A</sup>T<sub>E</sub>X информация, собранная в этой книге, не обязательна. T<sub>E</sub>Xпертам и тем, кто такими себя считает, читать по несколько раз. Книга содержит сквозной разноуровневый по сложности материал. При прочтении можно выбирать свой уровень.

Д. Э. Кнут «Всё про МЕТАФОНТ» [2]. Всё, что сказано про библию T<sub>E</sub>X, относится и к этой библии МЕТАФОНТ. Если вы создаёте иллюстрации с использованием MetaPost, то эту книгу следует прочитать хотя бы для общего развития.

Д. Э. Кнут «Компьютерная типография» [3]. Сборник статей Д. Кнута, написанных в процессе создания T<sub>E</sub>X и МЕТАФОНТ. В книге подробно разобраны алгоритмы, которые легли в основу T<sub>E</sub>X и перечислены проблемы, которые необходимо решить в процессе создания «компьютерной типографии». Книга интересна и в историческом плане — становление одного из самых успешных компьютерных проектов.

### 1.5.2. Учебники и справочники

Г. Грэтцер «Первые шаги в Л<sup>A</sup>T<sub>E</sub>X» [4]. Предназначено для новичков. Компактный учебник, позволяющий быстро освоить базовые приёмы. Упор на математику.

М. Гуссенс, Ф. Миттельбах и А. Самарин «Путеводитель по пакету Л<sup>A</sup>T<sub>E</sub>X и его расширению Л<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>» [5]. Очень полное и исчерпывающее справочное руководство по основным пакетам и приёмам Л<sup>A</sup>T<sub>E</sub>X. На английском недавно вышло второе издание этой книги. Возможно, со временем она доберётся и до нас. Это моя основная «рабочая» книга по Л<sup>A</sup>T<sub>E</sub>X.

М. Гуссенс, С. Ратц и Ф. Миттельбах. «Путеводитель по пакету Л<sup>A</sup>T<sub>E</sub>X и его графическим расширениям. Иллюстрирование документов при помощи T<sub>E</sub>X'а и PostScript'а» [6]. При подготовке основного путеводителя авторы обнаружили, что описание систем для создания рисунков по объёму начинает превосходить базовый текст. Пришлось выделить его в отдельную книгу. Здесь есть всё: от шахмат, нот и электронных схем до трюков с PostScript и MetaPost. Очень полезный учебник-справочник для тех, кто самостоятельно делает иллюстрации.

М. Гуссенс, С. Ратц. «Путеводитель по пакету Л<sup>A</sup>T<sub>E</sub>X и его Web-приложениям» [7]. Лучше бы авторы описали бы ещё несколько пакетов Л<sup>A</sup>T<sub>E</sub>X. ИМНО книга — дань моде. С другой стороны, расписано всё, что связано с PDF и что с ним можно сделать. Интересно будет любителям XML.

«Не очень краткое введение Л<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>или Л<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>за 94 минуты» в переводе Бориса Тоботраса от 1999 г. Электронная версия и исходники доступны на домашней страничке переводчика: <http://xtalk.msk.su/tex/>. В названии всё сказано. Самый доступный и маленький из имеющихся на сегодня учебников на русском языке.



Из недостатков: отсутствует информация о кириллизации. Полезно для начального изучения.

### 1.5.3. *LaTeX* в России

Е. М. Балдин. «Компьютерная типография *Л<sup>A</sup>T<sub>E</sub>X*» [8] В основу этой книги которой лёг этот текст. Между ними есть множество пересечений, но есть и значительные отличия, например, в объёме.

С. М. Львовский. «Набор и вёрстка в системе *Л<sup>A</sup>T<sub>E</sub>X*.» [9]. Хороший переплёт. Основной упор на математику. Автор ориентируется на нестандартную русификацию, которая имеет свои преимущества, хотя и спорные. Есть свободная электронная версия, которую можно взять здесь: <http://www.mccme.ru/free-books/>.

И. А. Котельников, П. З. Чеботаев. «*Л<sup>A</sup>T<sub>E</sub>X* по-русски.» [10]. Очень качественный учебник. К сожалению, качество переплёта не очень высокое, что не позволяет активно работать с этой книгой. Электронная версия книги доступна для скачивания по адресу: <http://www.tutor.nsu.ru/books/tex/>.

А. И. Рожено «Искусство вёрстки в *Л<sup>A</sup>T<sub>E</sub>X*'е» [11]. Краткий и достаточно исчерпывающий справочник с упором на разработанные автором макросы. Обсуждаются особенности русского стиля. Из недостатков: есть привязка к альтернативной операционной системе.

Кроме перечисленного, может показаться интересной документация, созданная В. Сюткиным: [http://www-sbras.nsc.ru/win/docs/TeX/LaTeX2e/docs\\_koi.html](http://www-sbras.nsc.ru/win/docs/TeX/LaTeX2e/docs_koi.html).

## 1.6. Список рассылки

На базе Воронежского государственного университета действует неофициальный список рассылки на русском языке по вопросам *Л<sup>A</sup>T<sub>E</sub>X* *CyrTeX-ru*. Сообщения в списке можно почитать по адресу: <https://info.vsu.ru/Lists/CyrTeX-ru/>.

Для того чтобы подписаться на этот список рассылки нужно по электронному адресу [CyrTeX-ru-feed@vsu.ru](mailto:CyrTeX-ru-feed@vsu.ru) послать письмо с темой «Subscribe me!» (без кавычек). В ответ будет отослано письмо с подтверждением рассылки. Для того чтобы подписаться, надо ответить, используя соответствующую функцию почтового клиента, либо создать новое письмо и в качестве темы указать номер, упомянутый в теле письма (имеет смысл прочитать, что там написано).

## 1.7. Благодарности

Эта глава предназначена для благодарностей.

Я благодарен своим родителям Балдиным Наталье Павловне и Михаилу Николаевичу за то, что я есть. Благодарен Кириллу и Мефодию за то, что есть кириллица. Благодарен Дональду Эрвину Кнуту за то, что есть *TeX*. Благодарен Ольге Лапко за то, что она сделала кириллические шрифты для *Л<sup>A</sup>T<sub>E</sub>X*. Благодарен Егору Ежову за то, что он вычитал первую главу этого текста.

## Базовые элементы

Мы говорим на русском языке и пользуемся операционной системой GNU/Linux — подобное на нашей планете не так уж и часто встретишь. Хотите стать ещё более элитарным подмножеством? Используйте ЛАТЭХ ☺!

### 2.1. «Командная логика»

Щенок должен понять, что быстрое и четкое выполнение команд хозяина всегда вознаграждается лакомством или лаской.

Для набора кода в ЛАТЭХ знание английского языка приходится очень кстати. Основных команд немного и их можно запомнить и так, но для совершенствования английский необходим, хотя бы для чтения документации к пакетам. Названия у команд, как правило, вполне осмыслены, что очень помогает при поиске чего-нибудь необходимого в алфавитном указателе. «Правильный» текстовый редактор тоже не является лишним.

**Спецсимволы** Не все символы одинаково равноправны. За частью символов в ЛАТЭХ зарезервированы специальные значения.

Это: «\», «\$», «%», «\_», «{», «}», «&», «#», «^» и «~». В процессе изложения их роль будет со временем раскрыта.

Чтобы отобразить эти символы при печати необходимы дополнительные усилия. Предыдущий параграф в текстовом редакторе выглядел бы так:

Это: <<\textbackslash>>, <<\\$>>, <<\%>>, <<\\_>><sup>1</sup>, <<\{>>, <<\}>>, <<\&>>, <<\#>>, <<\^{}>> и <<\~{}>>. В процессе изложения их роль будет раскрыта.

<sup>1</sup>Можно воспользоваться пакетом **underscore** — в этом случае необходимость экранировать знак «\_» в текстовой моде отпадает.

**Группировка** Группировка осуществляется с помощью фигурных скобок: { группа }. Фигурные скобки при печати не отображаются.

Сложные конструкции, которые имеют открывающую и закрывающую команды (например, окружения) тоже группируют текст.

**Построение команды** Команды в ЛАТЭХ начинаются с символа «\» (backslash или обратная косая черта) и продолжаются комбинацией, состоящей только из стандартных латинских букв<sup>2</sup>. Команды завершаются пробелом, цифрой или не латинской буквой. Все пробельные символы после команды игнорируются. Для того чтобы пробел после команды не игнорировался, достаточно вставить «пустую группу»: \command{}. Например, чтобы лого ЛАТЭХ (команда \LaTeX) не слилось со следующим за ним словом следует написать \LaTeX{ }.

**Аргументы** Командам ЛАТЭХ могут передаваться внешние аргументы:

```
\command [param 1] [param 2] {param3} {param4}
```

В квадратные скобки заключаются не обязательные параметры (param1 и param2), а в фигурные — обязательные (param3 и param4).

Некоторые из команд ЛАТЭХ влияют только на свои аргументы. Например, команда \textbf{текст} печатает «текст» жирным шрифтом.

**Декларативные команды** Часть команд ЛАТЭХ являются своеобразными переключателями режимов.

Область действия декларативной команды может ограничиваться логической группой или единицей структуры печатного документа, например, страницей.

```
%дальнейший текст будет печататься жирным шрифтом
\bfseries
%убрать заголовки и нумерацию для текущей страницы
\thispagestyle{empty}
```

**Окружения** Сложные конструкции, которые имеют открывающую и закрывающую команды вида \begin{имя} и \end{имя} называют окружениями. Вместо слова «имя» подставляется название соответствующего окружения

```
\begin{center}
Это строка будет центрирована
\end{center}
```

Окружения могут вкладываться друг в друга как матрёшки, но их область действия не может перекрываться.

---

<sup>2</sup>Это верно для стандартной кириллизации. Существуют варианты русификации, например, русификация Шеня, где русский буквы так же могут входить в имена команд.

## 2.2. Логика документа

Вид документа определяется выбором класса и стилей. Хорошим приёмом является наличие личных готовых шаблонов с уже предопределёнными предпочтениями, которые могут меняться по мере развития документа.

### 2.2.1. Структура ЛАТЭХ-файла

Текстовый `tex`-файл состоит из двух частей: заголовка или преамбулы и, собственно, самого текста, и выглядит примерно следующим образом:

```
%—начало заголовка—
%выбор класса документа, например article или book
\documentclass{...}
%минимальная кириллизация
\usepackage[koi8-r]{inputenc}
\usepackage[english,russian]{babel}
\usepackage[indentfirst]
%загрузка пакетов по выбору
\usepackage{...}
...
%определение своих команд или переопределение уже существующих
\newcommand{\myscommand}{...}
\renewcommand{\oldcommand}{...}
...
%локальные настройки
...
%—конец заголовка—
\begin{document}

%тело документа

\end{document}
```

Первым делом с помощью инструкции `\documentclass` осуществляется выбор класса документа. Далее загружаются стилевые пакеты.

Для того чтобы можно было набирать русский текст, необходимо с помощью пакета `inputenc` указать кодировку текстового файла, например, `koi8-r`. Далее нужно подключить пакет `babel`, который отвечает за локализацию, в частности за настройку переносов и «национальные особенности» набора. Например, при включении русского языка доопределяется символ номера № (`\No`), символ параграфа § (`\S`) и многое другое. Для формирования отступа или красной строки у первого параграфа, как это принято в России, необходимо загрузить пакет `indentfirst`. По идее это должно относиться к «национальным особенностям», но в `babel` по умолчанию не подгружается.

**К вопросу у переносах** Пожалуй единственное, что возможно потребует настройки в свежееустановленном дистрибутиве ЛАТ<sub>E</sub>X, это включение переносов. Этой проблемы заведомо не возникает при установке Т<sub>E</sub>X Live и в большинстве современных дистрибутивах GNU/Linux, но всякое бывает.

Для установки переносов можно воспользоваться утилитой **texconfig**<sup>3</sup>. После запуска программы следует выбрать меню HYPHENATION, а затем меню **latex**. Далее будет предложено отредактировать<sup>4</sup> файл переносов `language.dat`. Обычно достаточно таких настроек:

```
english hyphen.tex
russian ruhyphen.tex
```

Всё остальное по желанию можно закомментировать. За переносы отвечает пакет **babel**. По умолчанию, когда включаются переносы для определённого языка, все остальные правила переносов отключаются. Но в случае английского и русского языков это можно обойти, воспользовавшись русско-английской таблицей переносов:

```
ruseng ruenhyph.tex
=russian
=english
```

Следует понимать, что подобная настройка с точки зрения философии ЛАТ<sub>E</sub>X не совсем корректна. Для гарантированно одинакового результата компиляции не зависимо от платформы лучше поступить с некоторыми удобствами.

### 2.2.2. Класс документа

С помощью обязательной инструкции `\documentclass [опции] {класс}` можно задать к какому классу будет относиться рабочий текст.

Класс документа следует выбирать в зависимости от того, что должен из себя представлять текст. Для начала можно остановиться на стандартном классе **article**. Этот класс разработан специально для статей и небольших отчётов. Для отчётов побольше можно использовать класс **report**, а для книг класс **book**.

Перечисленные стандартные классы сложились очень давно и многие производные классы документов основаны на них. Как следствие стандартные базовые классы абсолютно статичны. Поэтому для специализированных вещей используются свои классы. Например, для научных статей популярны различные модификации **revtex4**. Вячеслав Фёдоров разработал класс **eskd** (стандарт ЕСКД), который

<sup>3</sup>**texconfig** — это простенькое dialog-подобное консольное приложение. Некоторые настройки могут потребовать привилегии суперпользователя. После изменения настроек через **texconfig** автоматически регенерируются необходимые форматные файлы. В противном случае может потребоваться сделать это в ручную, например, с помощью инструкции вида: `texconfig init`.

<sup>4</sup>Редактор можно определить с помощью переменной окружения `$EDITOR`. Если переменная не определена, то вызывается редактор **vi**. В случае отсутствия опыта работы с **vi** следует выйти из него с помощью последовательности `:q` и настроить переменную окружения.

можно найти на CTAN или в стандартной поставке T<sub>E</sub>X Live. Тех, кого «напрягают» большие поля в стандартных L<sup>A</sup>T<sub>E</sub>X-классах, могут обратить внимание на набор классов КОМА-script (**scrartcl**, **scrrreprt** и **scrbook** вместо **article**, **report** и **book**, соответственно).

По началу в выбранном классе ничего менять не следует. То, что кажется с непривычки не удобным, на самом деле может улучшать восприятие от печатной копии. Например, относительно узкая ширина текста в стандартных классах (следствие больших полей) позволяет при прочтении охватывать взглядом всю строку целиком, что увеличивает скорость чтения.

Обычно, разумные модификации можно выбирать с помощью передачи параметров при выборе класса, например, так:

```
\documentclass[a4paper,12pt,oneside]{scrbook}
```

**a4paper** — размер листа бумаги (можно выбрать другой стандарт, например, **a5paper**), **12pt** — базовый размер шрифта (в стандартных классах доступны размеры в **10pt** и **11pt**), а **oneside** — односторонняя печать (удобнее при просмотре электронной версии).

В заключении хотелось бы отметить набор классов *NC* активно разрабатываемых А. И. Рожено. Класс **ncc** автором позиционируется как «русскоязычная статья». Класс можно взять на CTAN и он присутствует в стандартной поставке T<sub>E</sub>X Live.

### 2.2.3. Стили

Стилевой файл (**.sty**) или пакет представляет собой набор макросов и определений, созданных для решения какой-то определённой задачи. Для подключения стилового файла используется команда `\usepackage[опции]{стиль}`.

Основное отличие классов от пакетов, что на документ может быть ровно один класс и сколько угодно стиливых пакетов. Фактически на любую задачу в L<sup>A</sup>T<sub>E</sub>X находится ответ в виде соответствующего пакета. В стандартной поставке T<sub>E</sub>X Live присутствует свыше двух тысяч **.sty**-файлов, кроме того ничего не мешает создать свой, заточенный под свои локальные проблемы.

**К вопросу о кириллизации** Чтобы кириллизировать L<sup>A</sup>T<sub>E</sub>X необходимы шрифты. Благодаря Ольге Лапко на свете есть шрифты семейства **lh**, которые отлично согласуются с базовыми шрифтами Computer Modern. Мало иметь просто кириллические буквы — надо чтобы их начертания соответствовали и другим шрифтам, в том числе и математическим. В 2001 году Владимир Волович проделал огромную работу по переводу METAFONT-шрифтов в формат Type1, что теперь позволяет создавать не только хорошие печатные копии, но и вполне качественные электронные pdf-версии документов.

За перевод из кодировки во внутреннюю кодировку L<sup>A</sup>T<sub>E</sub>X отвечает пакет **inputenc**. В качестве опции при загрузке с ним передаётся текущая восьмибитная кодовая

страница документа. Для кириллицы могут оказаться интересны следующие варианты: `ko18-r`, `ko18-u`, `cp866`, `cp1251` и `8859-5`. Собственно говоря, всё. Единственное неудобство, которое возникает из-за этого, заключается в том, что сообщения об ошибке  $\LaTeX$  выдаёт в своей внутренней T2A кодировке<sup>5</sup>. Для исправления этого неудобства можно воспользоваться простейшим фильтром. Для начало его надо собрать:

```
> locate t2filter.c
{TEXMF}/texmf-dist/doc/generic/t2/etc/t2filter.c
> cd {TEXMF}/texmf-dist/doc/generic/t2/etc/
> gcc -Wall -O2 -s -o ~/bin/t2filter t2filter.c
> latex {файл}.tex | t2filter
```

### 2.2.4. Тело документа

Всё, что заключено внутри окружения `document`, является телом документа. Если у вас есть какие-то куски текста, которые печатать не хочется, а выкинуть жалко, то их достаточно вынести в конец за инструкцию `\end{document}`.

## 2.3. Логика набора

Объявление в газете: Ищу работу машинистки.  
Печатаю со скоростью 4000 знаков в минуту.  
Правда, такая белиберда получается!

Мало открыть файл в текстовом редакторе и начать набирать. Нажимать на клавиши надо осмысленно.

### 2.3.1. Печатаем текст

При наборе книги/статьи/заметки основное вовсе не команды, а сам текст. Правила очень просты.

**Комментарии** Всё что следует за знаком «%» включительно является комментарием.

Большие закомментированные сегменты мешают работать с основным текстом, и поэтому их следует исключать из рабочего файла. Но при желании можно можно воспользоваться окружением `comment` из пакета `verbatim`.

---

<sup>5</sup>Расположение букв похоже на расположение букв в `cp1251`, но полностью не совпадает — чистая случайность.

**Разделение слов** Пробельные символы используются в  $\text{\LaTeX}$  для разделения слов. Пробелы в начале строки игнорируются. Символ перевода строки так же воспринимается как пробел. Если в конце строки сразу за последним словом вставить знак комментария:

```
экранировка перевода стр%
оки
```

то разделения слов не происходит. Иногда этот приём может оказаться полезным.

**Разделение абзацев** Для того чтобы начать следующий абзац необходимо оставить пустую строку:

```
текущий абзац закончился

следующий абзац начался
```

Число пустых строк между абзацами не имеет значения.

### 2.3.2. Пунктуация

Напечатанный текст обезличивается. Нет эмоций — только буквы. Единственное что остаётся — это знаки пунктуации и, возможно, смайлики ☺.

Запятую, точку, точку с запятой, двоеточие, многоточие, скобки, кавычки, восклицательный и вопросительные знаки следует «прижимать» к словам. Не надо оставлять пробелов, а то  $\text{\LaTeX}$  «подумает», что так и надо.

**Пробелы** Расстояние между словами  $\text{\LaTeX}$  выбирает по своему усмотрению для максимально равномерного заполнения страницы. Но иногда необходимо сделать указать размер пробела руками:

- ~ — неразрывный пробел, т. е. по этому пробелу не производится перенос предложения на другую строку,
- \, — маленький нерастяжимый пробел,
- \\_ — нормальный нерастяжимый пробел.

В основном, указывать размеры пробелов надо в случае набора каких-либо сокращений, например, так следует набирать ФИО:  
 Ф.\,А.~Миля "—— нежеже <<отрывать>> ИО от Ф\@. Ещё примеры:  
 т.\,е., г.~Новосибирск, рис.~1 и~т.\,д.\ и~т.\,п.

$\text{\LaTeX}$  считает, что после точки предложение заканчивается, если эта точка стоит не после заглавной буквы. Растяжимость пробелов между предложениями и между словами существенно разная. Поэтому если точка случается в середине предложения, то после неё следует явно вставить пробел «\\_» или неразрывный пробел «~».



Может случиться, что точка следует сразу за заглавной буквой и означает именно конец предложения (как в примере происходит с буквой  $\Phi$ ). Для этого перед такой точкой следует добавить коррекцию в виде команды «\@».

**Дефисы, минусы и тире** В издательских системах, основанных на  $\TeX$ 'е различают дефис «-» (hyphen), короткое тире «-» (en-dash), длинное тире «—» (em-dash) и знак минуса «-».

Чтобы получить на печати дефис, короткое или длинное тире, надо набрать один, два или три знака «-», соответственно.

При подключении пакета **babel** с опцией **russian** появляются дополнительные команды позволяющие более строго следовать русским печатным традициям.

Дефис используют в составных словах (кто-то, где-нибудь), короткое тире рекомендуется для указания диапазона чисел (10--15, 2001--2006), длинное тире означает обычное тире ( $\LaTeX$  "---- это круто), минус может существовать только в формулах ( $a-b=c$ ).

Пакет `\textpkg{babel}` вводит дополнительные команды для написания тире. Для составных/двойных фамилий следует использовать конструкцию "~~~, например, уравнение Клайперона"~~~Менделеева, композитор Римский"~~~Корсаков. Чтобы длинное тире не отрывалось от предыдущего слова и вокруг него создавались правильные пробелы вместо — следует употреблять "----, т.е. к трём тире надо добавить двойную кавычку. Прямая речь должна начинаться с команды "----\*:

"----\* Я сказал.

Правила могут показаться немного запутанными, но к ним быстро привыкаешь, и они того стоят.

**Переносы** В большинстве случаев  $\LaTeX$  грамотно переносит слова. Но в случае сложных слов, которые пишутся через дефис, перенос происходит только по дефису. Аналогично проблемы возникают когда слово частично состоит из английских букв, а частично из кириллицы.

Прямо в тексте перенос можно указать с помощью команды `\-`, например: дель\ -та-функ\ -ция, `\TeX`но\ -ло\ -гия.

При наличии русского языка в `\textpkg{babel}` вместо дефиса в сложном слове можно поставить команду "=", например, дельта"=функция. В этом случае переносы будут сделаны корректно без подсказки.

Для часто упоминаемых слов можно задать шаблон переноса с помощью команды `\hyphenation`{образ—цы пе—ре—но—са дель—та=—функ—ция}. Обычно, образцы переноса лучше определять в заголовке документа. Следует понимать, что образцы автоматически не склоняются, поэтому надо предусмотреть всевозможные варианты окончаний.

С помощью команды `\hyphenation` можно запретить перенос слова в нежелательных местах, просто не указав место разрыва. В тексте запрет переноса можно оформить с помощью инструкции `\mbox`{нет переноса}.

**Многоточие** Многоточие печатается с помощью команды `\ldots`. Если многоточие идёт после точки, то необходимо вставить неразрывный пробел `~`.

**Ударение** В русском языке длительность ударного гласного примерно в 1.5–2 раза длиннее безударного. Если ударение поставить не в том месте, то слово будет звучать совсем по другому.

В корне `\textbf{зар—}` "— `\textbf{зор—}` под ударением пишется гласная в соответствии с произношением, без ударения "— `\textbf{а}`.

`\emph`{Исключения:} зор\'янка, озар\'ять.

„Лапки“ и «Ёлочки». В пакете `babel` кроме всего прочего определены традиционные русские кавычки.

Если в начале или в конце текста встречаются внутренние и внешние кавычки, то они должны различаться между собой рисунком.

Он сказал: <<А пойду—ка я и подпишусь на „Linux Format“>>.

## 2.4. Структурная логика

Л<sup>A</sup>T<sub>E</sub>X ориентирован на логическую разметку документа. Можно конечно «сказать», что данный кусок текста следует напечатать размером 20 пунктов, выровнять по левому краю и сделать отступ после него в два интервала, но проще указать, что это заголовок раздела.

### 2.4.1. Титульный лист

Создания титульного листа это отдельная задача в которой визуальная составляющая обычно превалирует над структурной. В этом случае следует воспользоваться окружением `titlepage`. При инициализации этого окружения создаётся чистая

страница, которой присваивается номер один, а содержание этой странице полностью определяется фантазией автора. Но, в любом случае, это следует делать после написания самого текста. Обычно, достаточно стандартного заголовка:

```
\title{\LaTeX, Unix и русский стиль}
\author{Е. \,М. ~Балдин\thanks{e-mail: E.M.Baldin@inp.nsk.su}}
\date{2006}
\maketitle
```

Команда `\maketitle` создаёт стандартный титульный заголовок, используя информацию о названии документа (`\title`), авторе (`\author`) и даты написания текста (`\date`). Команда (`\thanks`) правильным образом позволяет оформить подстрочное примечание на титульной странице. Если авторов более чем один, то их можно перечислять разделяя командой (`\and`) — в этом случае список авторов печатается в виде таблицы.

В статьях (производные от класса **article**) вслед за заголовком следует обязательная аннотация, которая оформляется с помощью окружения **abstract**.

### 2.4.2. Секционирование

Часто бывает полезно сразу за титульной страницы вывести оглавление с помощью команды `\tableofcontents`. Но для этого в тексте должно присутствовать логическое разбиение на разделы.

```
\subsection{Секционирование}
\label{sec:base:sec}
```

Часто бывает ...

Команды секционирования образуют строгую иерархию. Самыми старшими по «званию» являются разделы `\part{Часть}` и `\chapter{Глава}`. Это большие куски текста и, соответственно, их применение обосновано только в книгах, поэтому они не определены в классах производных от **article** и **report**, зато определены в классе **book**.

Далее по старшинству следуют:

```
\section{Раздел}
\label{ex:section}

\subsection{Подраздел}
\label{ex:subsection}

\subsubsection[Подподраздел]{Что-то более мелкое чем подраздел}
\label{ex:subsubsection}

\paragraph{Параграф}
```

```
\label{ex:paragraph}
```

```
\subparagraph{Подпараграф}
```

```
\label{ex:subparagraph}
```

Если воспользоваться необязательным параметром команды секционирования, то он замещает основной заголовок при печати оглавления и создания колонтитулов.

Команды секционирования печатают заголовок необходимым шрифтом и нумеруют раздел. Если нет желания, что-бы название раздела попало в оглавления и нумерация без надобности, то к команде секционирования следует добавить символ «\*», например, `\section*{Приложение}`.

### 2.4.3. Перекрёстные ссылки

Одной из основных причин по которой ЛАТ<sub>E</sub>X вытеснил обычный Т<sub>E</sub>X из текстовых редакторов Т<sub>E</sub>Xников является механизм нумерации и создания ссылок.

Чтобы сослаться на раздел в нём необходимо оставить метку `\label{метка}`. А затем можно на этот раздел сослаться:

```
В разделе \ref{ex:section} на странице \pageref{ex:section} ...
```

Когда ссылки идут через метку, то номер раздела и номер страницы определяется ЛАТ<sub>E</sub>X автоматически. Причём автоматическая нумерация свойственна не только командам секционирования точно так же можно сослаться на формулы, таблицы, картинки и листинги программ. Для этого необходимо оставить метку `\label` в соответствующем окружении.

### 2.4.4. Сложные документы

Всё можно хранить в одном файле — это ничему не противоречит. Более того само понятие файл для пользователя не так уж и необходимо. Но уж если файл есть, то почему бы не разделить большой текст на несколько частично независимых кусков.

С помощью команды `\input{имя файла}` можно вставлять другой tex-файл в документ. ЛАТ<sub>E</sub>X просто добавляет содержимое по месту команды, считывая файл либо до конца, либо до первой встретившейся инструкции `\endinput`. Совершенно не важно в какой части документа встречается `\input`. Бывает довольно удобно вынести преамбулу в отдельный файл. В имени файла можно опустить расширение `.tex`.

Для включения текста можно применить другой способ:

```
\input{preheader}
\includeonly{
% intro ,
  base ,
% presentation
```

```
}  
\begin{document}  
%введение  
\include{intro}  
%базовые команды  
\include{base}  
%Презентация  
\include{presentation}  
\end{document}
```

Декларация `\include` позволяет включить только `tex`-файл (при написании имени расширение `.tex` опускается). В преамбуле с помощью команды `\includeonly` можно перечислить какие части надо подключить при текущей сборке. При этом сохраняется правильная нумерация страниц и можно сослаться на *не* включённые в эту сборку разделы. Довольно актуально в случае больших текстов в процессе их создания, так как значительно ускоряет компиляцию.

## Врезка: За букровку «ё» замолвите слово

Пара слов в поддержку буквы «ё». Эту букву незаслуженно забывают при наборе текстов. Более того, некоторые деятели ратуют за её полное упразднение. Однако, коль уж Вам довелось узнать русский язык, то говорить и писать на нём следует правильно. Наличие буквы «ё» в тексте значительно облегчает процесс чтения. Особенно это актуально при быстром чтении.

### *Поставь букву «ё» на её место!*

Для проверки правописания следует использовать словарь Александра Лебедева. Этот словарь построен на основе словаря русского языка для **ispell**, первоначально составленного Нилом Далтоном (Neal Dalton) в 1992 г. После тщательной проверки и исправления примерно 4000 ошибок в словаре Нила Далтона, в словарь были добавлены отсутствовавшие в нём правила образования форм существительных, прилагательных, причастий, наречий, изменены правила формирования окончаний глаголов, так что affix-файл можно считать переписанным заново. Одновременно в словарь было добавлено большое число слов.

Отличительной чертой данного словаря является то, что в него включена полноценная поддержка буквы «ё». В современных дистрибутивах GNU/Linux словарь Александра Лебедева является основным русским словарём для **ispell**. К сожалению в подавляющем большинстве случаев словарь «собран» без поддержки «ё». Мантейнерами пакета ошибочно предполагается, что «е» и «ё» это одинаковые буквы. При проверке правильными являются такие слова как «ежик», «елка» и тому подобное. Правильный выход: убедить мантейнера/самому стать мантейнером пакета. Не правильный, но гораздо более простой: локально пересобрать пакет с полноценной поддержкой «ё».

Словарь постоянно совершенствуется, дополняется и корректируется. Последнюю версию словаря можно найти на авторской страничке Александра Лебедева: <http://semiconductors.phys.msu.ru/~swan/orthography.html>

## Врезка: T<sub>E</sub>X-лого

Когда Д. Э. Кнут создавал T<sub>E</sub>X — он много думал. Причём думал не только об алгоритмах и кодировке. В частности он нашёл время подумать о том, как назвать своё произведение. T<sub>E</sub>X читается как «тех». Последняя буква вовсе не английская буква «икс», а греческая «хи». Так же он продумал и правила изображения этого названия. С тех пор в T<sub>E</sub>X-сообществе возникла мода на создание T<sub>E</sub>X-лого.

T <sub>E</sub> X	<code>\TeX</code>
L <sup>A</sup> T <sub>E</sub> X	<code>\LaTeX</code> или <code>\LATEX</code>
L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub>	<code>\LaTeXe</code>
A <sub>M</sub> S-T <sub>E</sub> X	<code>\AMSTeX</code> или <code>\AmSTeX</code>
МЕТАFONT	<code>\METAFONT</code> или <code>\MF</code>
VibT <sub>E</sub> X	<code>\VIBTeX</code> или <code>\VibTeX</code>
N <sub>C</sub> C	<code>\NCC</code>

Таблица 2.1. Распространённые T<sub>E</sub>X-лого

Команда `\NCC` определена в пакете **ncclatex**. Остальные команды заведомо определены в пакете **texnames**.

## Набор математики

Полиграфисты относят математические работы к каторжным. . .

---

Д. Э. Кнут. Математическая типография.

Иногда от незнакомых с  $\text{T}_\text{E}_\text{X}$  технологиями людей приходится слышать, что  $\text{L}_\text{A}_\text{T}_\text{E}_\text{X}$  годится *только* для набора математики. При знакомстве же с истинными  $\text{T}_\text{E}_\text{X}$  технологиями возникает понимание, что  $\text{L}_\text{A}_\text{T}_\text{E}_\text{X}$  настолько хорош, что с его помощью можно набирать *даже* математику.

Набор математики всегда считалась вершиной типографского искусства. Дело в том, что формулы для концентрации информации и дополнительной выразительности в отличие от обычного текста являются многоуровневыми. Д. Э. Кнут к своей программе компьютерной типографии создал язык для описания формул. После короткого периода обучения пользователь в состоянии читать и набирать формулы на этом языке практически любой сложности.

$\text{L}_\text{A}_\text{T}_\text{E}_\text{X}$  не единственная программная среда, использующая  $\text{T}_\text{E}_\text{X}$ -нотацию. Эта же нотация рекомендуется при наборе всех сколько-нибудь сложных формул на страницах Википедии (<http://ru.wikipedia.org> статья «Википедия:Формулы»).

Становлению  $\text{T}_\text{E}_\text{X}$  как стандарта для набора формул в значительной степени способствовало Американское математическое сообщество (The American Mathematical Society — AMS), которое в начале восьмидесятых годов прошлого столетия субсидировало разработку расширения  $\text{T}_\text{E}_\text{X}$  известного как  $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\text{T}_\text{E}_\text{X}$ . В 1987 году наработки  $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\text{T}_\text{E}_\text{X}$  были добавлены в  $\text{L}_\text{A}_\text{T}_\text{E}_\text{X}$  в виде пакета **amsmath**. Вместе с **amsmath** в  $\text{L}_\text{A}_\text{T}_\text{E}_\text{X}$  было добавлено множество улучшений, позволяющих набирать действительно изощрённую математику. Поэтому при использовании в тексте математики в шапке документа следует в обязательном порядке загружать пакет **amsmath**:

```
\usepackage { amsmath }
```

В дальнейшем предполагается, что этот пакет уже загружен.



Полностью описать *все* команды языка описания формул в рамках короткой статьи нереально, так как математика, как и способы её описания, безгранична. Поэтому основное внимание будет уделено базовым правилам и русскому стилю в формулах. В любой сколько-нибудь большой книге по ЛАТЭХ будет полный список всех команд. Если серьёзно работать с математикой, то подобная книжка в любом случае понадобится.

### 3.1. Набор формул

При формировании текста формулы подразделяются на *строчные* и *выносные*. Строчные формулы набираются внутри абзаца вместе текстом. По описанию формулы ЛАТЭХ создаёт бокс, который обрабатывается наравне с обычными текстовыми боксами. Как правило, строковые формулы это небольшие вставки, вроде  $E = mc^2$ . Выносные или *выключенные* формулы выводятся за пределы абзаца.

Строчная формула в тексте ограничивается<sup>1</sup> с помощью символа доллара \$«формула»\$ или с помощью команд-скобок \ («формула» \). При наборе предпочтительно использовать второй вариант оформления, так как он позволяет легко определить где начинается, а где кончается формула. «Долларовое» (\$) окружение лучше тем, что оно чуть-чуть короче, кроме этого команда \$ *крепкая*<sup>2</sup> в отличии от команд-скобок.

Однострочные выносные формулы формируются с помощью окружения **equation**. Так как в этом случае формула вынесена за пределы абзаца, то её можно пронумеровать. Например:

```
\begin{equation}
  \label{eq:math:ex1}
  \int\limits_{-\infty}^{\infty}
    e^{-x^2/2}dx=\sqrt{2\pi}
\end{equation}
```

$$\int_{-\infty}^{\infty} e^{-x^2/2} dx = \sqrt{2\pi} \quad (3.1)$$

Нумерация формул удобна для того, чтобы позже в тексте на неё можно было легко сослаться с помощью команды `\eqref{eq:math:1}`<sup>3</sup>. Если же формул немного и не хочется никакой нумерации, то можно воспользоваться окружением **equation\***<sup>4</sup>.

<sup>1</sup>Есть более формальное оформление строчной формулы как окружения: `\begin{math}` «формула» `\end{math}`. Но в силу понятных причин никто подобное описание не использует в пользу кратких обозначений.

<sup>2</sup>Когда начинаешь изучать команды ЛАТЭХ, то довольно быстро сталкиваешься с понятиями «хрупкости»/«крепкости». Крепкие команды в отличии от хрупких можно использовать в качестве аргументов других команд. С другой стороны хрупкие команды тоже можно использовать как параметры, защитив их с помощью команды `\protect`. Эти понятия в большинстве своём пережитки прошлого и их постепенно изживают, но пока их следует иметь в виду.

<sup>3</sup>Метка выставляется с помощью команды `\label`.

<sup>4</sup>К **equation** добавляется звёздочка. Подобный приём в создании команд применяется достаточно часто. Команда со \* обычно не нумеруется и не отображается ни в каких автоматически составляемых списках.

При создании выключенной формулы размер шрифта для улучшения читаемости немного увеличивается. ЛАТЭХ имеет несколько стилей для оформления математических формул. При желании можно выбрать необходимый стиль в ручную:

`\displaystyle` — стиль, используемый для выносных формул,

`\textstyle` — стиль строчных формул,

`\scriptstyle` — в этом стиле набираются индексы,

`\scriptscriptstyle` — индексы второго уровня.

С помощью этих команд можно увеличить размер шрифта для формул внутри абзаца, или заставить индексы выглядеть как базовые символы. Для примера сравните:

```
\begin{equation*}
\frac{1}{1+
\frac{1}{1+
\frac{1}{2}}}}
\end{equation*}
```

$$\frac{1}{1 + \frac{1}{1 + \frac{1}{2}}}$$

```
\begin{equation*}
\frac{1}{\displaystyle 1+
\frac{1}{\displaystyle 1+
\frac{1}{\displaystyle 1}
{\displaystyle 2}}}}
\end{equation*}
```

$$\frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{2}}}}$$

Пробелы в формулах отмечают только конец команды, а сами по себе смысла не имеют — ЛАТЭХ, как правило, гораздо лучше знает как сформировать результат.

## 3.2. Кириллица в формулах

Всё дело в имеющихся шрифтах — они красивые, разнообразные, но в большинстве своём англоязычные. В настоящее время кириллические математические шрифты в «дикой природе» отсутствуют, поэтому приходится пользоваться их текстовыми версиями.

Стиль **mathtext** (пакет **t2**), позволяет использовать кириллицу в формулах без дополнительных ухищрений. Стиль может быть подключён с опцией *warn* — в этом случае он сообщает обо всех случаях использования кириллических букв в формулах. **mathtext** следует загружать до **babel** и/или **fontenc**.

```
\usepackage [ warn ] { mathtext }
```

```
\[
v_{\text{cp}}=\frac{S_{\text{конец}}-S_{\text{начало}}}{\delta t}
\]
```

$$v_{\text{cp}} = \frac{S_{\text{конец}} - S_{\text{начало}}}{\delta t}$$

Здесь для создание выключенной формулы используется команда `\[«формула»\]` — краткий аналог окружения **equation\***. В отличии от латиницы русские буквы в формулах печатаются прямым шрифтом — это было сделано специально. Чтобы изменить это умолчание в преамбуле следует добавить команду для переопределения шрифта:

```
\DeclareSymbolFont{T2Aletters}{T2A}{cmr}{m}{it}
```

Стиль **amstext** (загружается автоматически при загрузке **amsmath**) определяет команду `\text`, которая позволяет вставлять в формулу обычный текст. Текст может быть и русским:

```
\[v_{\text{cp}}=
\frac{\text{конец пути}-
\text{начало пути}}{\text{время в пути}}\]
```

$$v_{\text{cp}} = \frac{\text{конец пути} - \text{начало пути}}{\text{время в пути}}$$

Преимущество такого подхода заключается в том, что внутри команды `\text` пробелы воспринимаются как нормальные символы и слова не сливаются. Использование `\text` предпочтительно и для целей переносимости.

### 3.3. Школьная математика

Математика в школе — это явление, через которое проходит каждый. Именно поэтому фактически любой вменяемый россиянин умеет обращаться с дробями, знает теорему Пифагора, с лёгкостью решает квадратные уравнения и что-то слышал про интеграл и производную. Разберёмся с этим поподробнее.

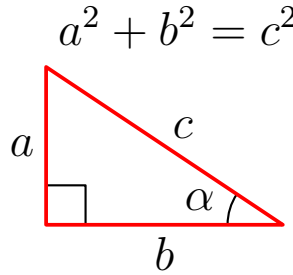
#### 3.3.1. Индексы

Букв в латинском алфавите не так уж и много, а научных понятий без числа. Один из способов отличать обозначения друг от друга, это индексы, как верхние, так и нижние:

```
\[A_{\text{нижний индекс}}\quad
B^{\text{верхний индекс}}\quad
C_n^k\]
```

$$A_{\text{нижний индекс}} \quad B^{\text{верхний индекс}} \quad C_n^k$$

Обратите внимание, что если в индексе ровно один знак, то фигурные скобки вокруг него можно и нужно опустить. Теперь можем записать теорему Пифагора:  $\backslash(a^2+b^2=c^2\backslash)$



### 3.3.2. Математические символы

Кроме символов латиницы и кириллицы математики используют множество самых разнообразных значков. Да и латиница не так уж и проста. Если воспользоваться пакетом **amsfonts** то она может стать:

```
\begin{itemize}
\item \(\ABCD\) "--- обычной,
% \item \(\mathbf{ABCD}\) "--- жирной,
\item \(\mathbb{ABCD}\) "--- ажурной,
% \item \(\mathcal{ABCD}\) "---
% прописной.
\end{itemize}
```

- $ABCD$  — обычной,
- $\mathbf{ABCD}$  — ажурной,

Это далеко не все возможные шрифтовые стили которые можно применять в математической моде. Но лучше особо не перегружать формулы всякой «готикой» (например, команда  $\backslash\mathfrak$ ).

Не единой латиницей жив математик. Традиционно везде, где только можно, используются греческие буквы. В  $\text{\LaTeX}$  присутствует полный набор и за исклю-

Греческие символы											
$\alpha$	$\backslash\alpha$	$\beta$	$\backslash\beta$	$\gamma$	$\backslash\gamma$	$\delta$	$\backslash\delta$	$\epsilon$	$\backslash\epsilon$	$\zeta$	$\backslash\zeta$
$\zeta$	$\backslash\zeta$	$\eta$	$\backslash\eta$	$\theta$	$\backslash\theta$	$\iota$	$\backslash\iota$	$\kappa$	$\backslash\kappa$	$\lambda$	$\backslash\lambda$
$\lambda$	$\backslash\lambda$	$\mu$	$\backslash\mu$	$\nu$	$\backslash\nu$	$\xi$	$\backslash\xi$	$\omicron$	$\backslash\omicron$	$\pi$	$\backslash\pi$
$\pi$	$\backslash\pi$	$\rho$	$\backslash\rho$	$\sigma$	$\backslash\sigma$	$\tau$	$\backslash\tau$	$\upsilon$	$\backslash\upsilon$	$\varphi$	$\backslash\varphi$
$\varphi$	$\backslash\varphi$	$\chi$	$\backslash\chi$	$\psi$	$\backslash\psi$	$\omega$	$\backslash\omega$	$\Gamma$	$\backslash\Gamma$	$\Delta$	$\backslash\Delta$
$\Delta$	$\backslash\Delta$	$\Theta$	$\backslash\Theta$	$\Lambda$	$\backslash\Lambda$	$\Xi$	$\backslash\Xi$	$\Pi$	$\backslash\Pi$	$\Sigma$	$\backslash\Sigma$
$\Sigma$	$\backslash\Sigma$	$\Upsilon$	$\backslash\Upsilon$	$\Phi$	$\backslash\Phi$	$\Psi$	$\backslash\Psi$	$\Omega$	$\backslash\Omega$		

чением трёх букв начертание вполне привычное. Для исправления непривычных начертаний эти буквы были переопределены с помощью пакета **amssymb**:

```
% Переопределение \kappa, \epsilon и \phi на русский лад
```

```
\renewcommand{\kappa}{\varkappa}
\renewcommand{\epsilon}{\varepsilon}
\renewcommand{\phi}{\varphi}
```

Спецсимволов в  $\text{\LaTeX}$  великое множество. В стандартной поставке  $\text{\TeX Live}$  идёт «Всеобъемлющий список символов  $\text{\LaTeX}$ » (The Comprehensive LaTeX Symbols List — файл `symbols-a4.pdf`) в котором перечислено 3300 распространённых символов, используемых пользователями  $\text{\LaTeX}$ . Почти наверняка любой операнд, который вам нужен, там уже есть. Ниже будут перечислены только та часть символов, которая с моей точки зрения может пригодиться в наборе школьной математики. Пакет `amssymb` для использования обязателен.

«Школьные» символы									
$\hat{a}$	<code>\hat{a}</code>	$\bar{a}$	<code>\bar{a}</code>	$\vec{a}$	<code>\vec{a}</code>	$\dot{a}$	<code>\dot{a}</code>	$\tilde{a}$	<code>\tilde{a}</code>
$\pm$	<code>\pm</code>	$\mp$	<code>\mp</code>	$\times$	<code>\times</code>	$\cdot$	<code>\cdot</code>	$\div$	<code>\div</code>
$\forall$	<code>\forall</code>	$\wedge$	<code>\wedge</code>	$\neg$	<code>\neg</code>	$\forall$	<code>\forall</code>	$\exists$	<code>\exists</code>
$\leq$	<code>\leq</code>	$\geq$	<code>\geq</code>	$\ll$	<code>\ll</code>	$\gg$	<code>\gg</code>	$\neq$	<code>\neq</code>
$\equiv$	<code>\equiv</code>	$\sim$	<code>\sim</code>	$\simeq$	<code>\simeq</code>	$\approx$	<code>\approx</code>	$\propto$	<code>\propto</code>
$\parallel$	<code>\parallel</code>	$\perp$	<code>\perp</code>	$\angle$	<code>\angle</code>	$\triangle$	<code>\triangle</code>	$\sphericalangle$	<code>\sphericalangle</code>
$\infty$	<code>\infty</code>	$\ell$	<code>\ell</code>	$\sum$	<code>\sum</code>	$\prod$	<code>\prod</code>	$\emptyset$	<code>\emptyset</code>

Для соответствия русским традициям два символа были переопределены:

```
% Переопределение \le и \ge на русский лад
\renewcommand{\le}{\leqslant}
\renewcommand{\ge}{\geqslant}
```

### 3.3.3. Дроби

Дроби формируются с помощью команды `\frac`<sup>5</sup>:

```
\[
дробь=\frac{числитель}{знаменатель}
\]
```

$$\text{дробь} = \frac{\text{числитель}}{\text{знаменатель}}$$

Как и практически вся математика в  $\text{\LaTeX}$  дробь записывается как читается само выражение.

### 3.3.4. Корни

Для рисования знака корня используется команда

<sup>5</sup>От слова `fraction` — дробь.

```
\sqrt [«степень» ] { «подкоренное выражение» }
```

Степень можно опустить. В этом случае рисуется обычный квадратный корень.

```
\[
\overline{
\underline{\Large
\sqrt [3]{a}+\sqrt [2]{b}+\sqrt [99]{g}
}
}
\]
```

$$\overline{\underline{\sqrt[3]{a} + \sqrt[2]{b} + \sqrt[99]{g}}}$$

Обратите внимание, что знак корня размещается в соответствии с размерами подкоренного выражения. Если в выражении присутствует только один корень, то это самое разумное поведение, но в случае нескольких корней, как вышеприведённом примере, то необходимо выравнивание.

Для выравнивания по высоте используется команда `\mathstrut`<sup>6</sup>. В результате её применения вставляется невидимый символ нулевой толщины высотой в точности равной высоте круглой скобки:

```
\[\Large
\sqrt [3]{\mathstrut a}+
\sqrt [2]{\mathstrut b}+
\sqrt [99]{\mathstrut g}
\]
```

$$\sqrt[3]{a} + \sqrt[2]{b} + \sqrt[99]{g}$$

### 3.3.5. Квадратное уравнение

И наконец вершина школьной математики — это решение квадратного уравнения  $ax^2 + bx + c = 0$ :

```
\[
x_{1,2}=\frac{-b\pm\sqrt{b^2-4ac}}{2a}
\]
```

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Теперь можно смело писать методички по школьной математике ☺.

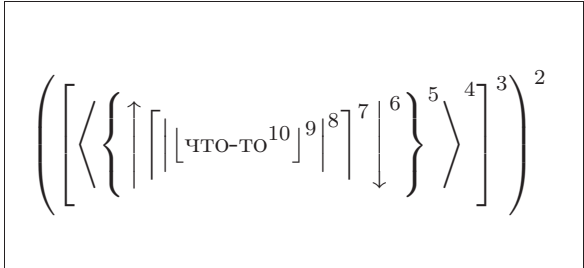
### 3.3.6. Скобки

Для визуальной группировки символов внутри формулы скобки вещь незаменимая. Особенно здорово, если скобки автоматически подбирают свой размер под вы-

<sup>6</sup>От английского strut — подпорка или страта.

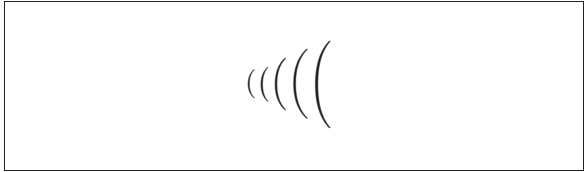
ражение, которое они окружают. Парные команды `\left` и `\right` включают режим подобной подстройке:

```
\[\left(
  \left[
  \left\langle
  \left\{
  \left\uparrow
  \left\lceil
  \left|
  \left\lfloor
  \text{что-то}^{10}
  \right\rfloor^9
  \right|^8
  \right\rceil^7
  \right\downarrow^6
  \right\}^5
  \right\rangle^4
  \right]^3
  \right)^2\]
```



Иногда хочется размер выставить в ручную, тогда перед скобкой можно выставить одну из следующих команд:

```
\[ ( \big( \Big( \bigg( \Bigg( \]
```



Эстетсы в зависимости от ситуации в конце команды могут добавить модификатор позиционирования разделителя как левого — `l` (отбивка как для `\left`), правого — `r` (отбивка как для `\right`) и среднего — `m`.

### 3.3.7. Функции

Все символы в математической моде печатаются курсивом, поэтому названия функций для выделения печатаются прямым шрифтом. Кроме смены шрифта функции с обеих сторон должны правильно «отбиваться» пробелами, иначе будет некрасиво. При загрузке русского языка с помощью пакета **babel** кроме стандартных имён функций доопределяется несколько сокращений применяемых в русскоязычной литературе. Среди часто употребляемых функций можно упомянуть: `cos`, `arccos`, `sin`, `arcsin`, `tg`, `arctg`, `ctg`, `arcctg`, `sh`, `ch`, `th`, `cth`, `exp`, `ln`, `log`, `lim`, `min` и `max`. В математической моде эти функции можно использовать в качестве команд:

```

\begin{equation*}
\begin{split}
&\log_2 10 = \ln 10 / \ln 2 \simeq 3.32 \\
&\lim_{x \rightarrow 0} \frac{\sin x}{x} = 1 \\
&(a+b)^n = \sum_{k=1}^n C_n^k a^k b^{n-k}
\end{split}
\end{equation*}

```

$$\log_2 10 = \ln 10 / \ln 2 \simeq 3.32$$

$$\lim_{x \rightarrow 0} \frac{\sin x}{x} = 1$$

$$(a+b)^n = \sum_{k=1}^n C_n^k a^k b^{n-k}$$

Обратите внимание на обработку индексов для функции `\log` (логарифм) и `\lim` (предел). Для доопределения новых функций правильнее всего воспользоваться в преамбуле командой `\DeclareMathOperator`:

```

% В преамбуле — определение новых функций
\DeclareMathOperator{\log-like}{log-like}
\DeclareMathOperator*{\lim-like}{lim-like}

```

В зависимости от варианта команды индексы отображаются как для логарифма (команда без звёздочки) или как для предела (команда со звёздочкой).

### 3.3.8. Производная и интеграл

В старших классах в конце обучения чуть-чуть касаются понятий интегрирования и дифференцирования. Возможно для того, чтобы правильно подсчитать сдачу в магазине, эти знания не являются необходимыми. Но для изучения физики и, как следствие, химии и биологии без интегралов никак — поверьте мне на слово.

Производная обычно отмечается штрихом. В физике производная по времени выделяется точкой, для того чтобы отличать её от производной по координате. Можно честно написать `\frac{d F(x)}{dx}`. Для частной производной вместо буквы *d* используется спецсимвол `\partial`:

```

\[ f' \quad f'' \quad \dot{f} \quad \ddot{f} \quad \frac{df}{dx} \quad \frac{\partial f}{\partial x}
\]

```

$$f' \quad f'' \quad \dot{f} \quad \ddot{f} \quad \frac{df}{dx} \quad \frac{\partial f}{\partial x}$$

Производная есть обратная функция от интегрирования:

```

\[ \frac{d}{dx} \int F(x) dx = F(x)
\]

```

$$\frac{d}{dx} \int F(x) dx = F(x)$$

Приглядевшись к имеющемуся здесь примеру, можно отметить, что в отличие от русских математических традиций представленный здесь интеграл не прямой,



а наклонный. Это можно исправить, например, загрузив пакет **wasysym** с опцией **integrals**. К сожалению получающиеся интегралы «не смотрятся». Поэтому пока лучше использовать начертания по умолчанию в надежде, что в будущем ситуация изменится к лучшему.

Неопределённый интеграл хорошо, но с определённым тоже надо работать. Качественное оформление пределов интегрирования важно для восприятия формулы.

```
\[
\int_0^{\infty}\quad
\int\limits_0^{\infty}\quad
\sum_{i=1}^n\quad
\sum\nolimits_{i=1}^n\quad
\]
```

$$\int_0^{\infty} \quad \int_0^{\infty} \quad \sum_{i=1}^n \quad \sum_{i=1}^n$$

По умолчанию пределы размещаются справа от интеграла. Ситуацию можно поправить с помощью команды **\limits**. Команда **\nolimits** делает всё ровно наоборот.

### 3.4. Перенос формул

В русскоязычной литературе принято, что при переносе строчной формулы на другую строку знак, по которому формула разрывается дублируется на следующей строке. Например так:

```
a + b =
= c
```

По умолчанию этого не происходит. Проще всего решить эту проблему с помощью следующего макроса<sup>7</sup>, который необходимо определить в преамбуле:

```
% перенос формул в тексте
\newcommand*{\hm}[1]{#1\nobreak\discretionary}{%
{\hbox{$\mathsurround=0pt #1$}}}
```

Здесь определена команда **\hm**, которую следует добавлять в местах потенциального переноса формулы, примерно, так:  $(a + b \hm{=} c)$ . Сделать это можно во время окончательной доводке текста. В любом случае для полировки рукописи ручная работа необходима.

Разрыв математических формул при переносе предпочтителен на знаках отношения ( $=$ ,  $>$ ,  $<$ ,  $\leq$ ,  $\geq$ ,  $\neq$ ,  $\simeq$ ); во вторую очередь на отточии, знаках сложения и вычитания; в третью — на знаке умножения в виде косога креста. Не рекомендуется разбивать формулу на знаке деления и на каких-либо других знаках, кроме упомянутых выше.

<sup>7</sup>Рецепт от Евгения Миньковского из fido7.ru.tex.

### 3.5. Заключение

Изложенных правил и приёмов вполне хватит для набора в рамках школьной математики. Для более изощрённых формул требуются более продвинутые приёмы и конструкции. Всё это будет, но чуть попозже.

## Врезка: Вики

«Движок» который использует Википедия для отображения формул называется WikiTeX. Основной сайт проекта, естественно, представляет из себя вики по адресу <http://wikisophia.org/>. Используя это программное обеспечение в связки с L<sup>A</sup>T<sub>E</sub>X, можно не только сносно отображать математические формулы на WWW без особых ухищрений, но и отрисовывать шахматные партии, химические формулы, фейнмановские диаграммы, нотные записи и многое другое.

T<sub>E</sub>X сразу разрабатывался как программа, которая может формировать изображения для разных устройств, даже для тех, о которых на момент создания этого текстового процессора профессору Д. Э. Кнуту ничего известно не было. Поэтому T<sub>E</sub>X обретается в самых неожиданных местах.

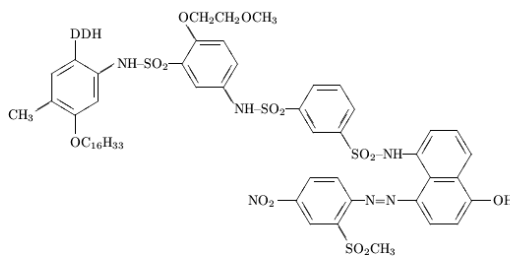
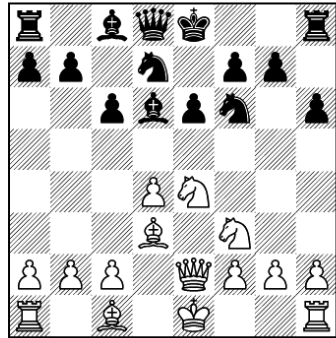
<b>Classes</b>	[edit]
The following classes have been implemented:	
<b>Amsmath</b>	[edit]
Plenary <b>AMS-LaTeX</b> , including commutative diagrams ( <a href="#">doc</a>   <a href="#">template</a> ).	
$\forall s, t \in \omega^* [s \geq t \Leftrightarrow s \text{ is an initial sequence of } t]$	
<b>Chem</b>	[edit]
<b>XyMTeX</b> is an intuitive chemistry package by Shinsaku Fujita ( <a href="#">doc</a>   <a href="#">template</a> ).	
	
<b>Chess</b>	[edit]
<b>Skak</b> by Torben Hoffmann supports SAN and FEN notation ( <a href="#">doc</a>   <a href="#">doc</a>   <a href="#">doc</a>   <a href="#">template</a> ).	
	

Рис. 3.1. WikiTeX за работой.

*Сделайте так, чтобы Ваш форум или вики заговорил на языке L<sup>A</sup>T<sub>E</sub>X.*

## Графика

Q. Как быстро написать курсовую на ЛАТ<sub>E</sub>X'e в которой кроме текста есть и графики?

A. Сделать для начала графики ☺

---

Вопрос и краткий ответ.

Вероятно, Т<sub>E</sub>X на текущий момент лучше других программ вёрстки умеет разбивать абзацы на строки. То есть удачнее всех разливать порции «клея» между «боксами», но подготовка графики выносится за рамки этого процесса. Почти . . .

С точки зрения Т<sub>E</sub>X картинка — это просто очень большой прямоугольник, который надо как-то разместить на странице. От пользователя нужны только размеры этого прямоугольника. Отображение же иллюстрации лежит на плечах драйверов. Самым востребованным форматом для представления графики в ЛАТ<sub>E</sub>X до сих пор является Encapsulated PostScript.

### 4.1. Encapsulated PostScript

Уже больше двадцати лет прошло с тех пор как никому не известная фирма Adobe Systems получила инвестиции от фирмы Apple на «обучение» лазерных принтеров молодому языку PostScript. Как следствие этот платформонезависимый язык с полностью открытой спецификацией стал безальтернативным стандартом. Даже сейчас PostScript фактически не имеет конкурентов в области допечатной подготовки. Поэтому почти все «уважающие себя» графические программы умеют экспортировать результаты своей деятельности в виде инструкций PostScript. Особенно это касается векторных графических редакторов, так как PostScript подразумевает векторную графику.

Encapsulated PostScript или кратко EPS — графический формат. Файлы в этом формате обычно имеют расширение eps. По сути дела это PostScript с некоторыми упрощениями и дополнительными договорённостями. Самая интересная с

точки зрения L<sup>A</sup>T<sub>E</sub>X договорённость — это обязательное наличие в заголовке информации о размере картинки, которая передаётся вместе с комментарием:

```
%!PS-Adobe-2.0 EPSF-2.0
%%Creator: dvips(k) 5.95b Copyright 2005 Radical Eye Software
%%Title: picture.dvi
%%BoundingBox: 127 464 430 667
%%DocumentFonts: SFRM1200 SFRM0800
%%EndComments
```

Первая строка комментария обычно содержит версию PostScript<sup>1</sup>. Вслед за комментарием `BoundingBox` идёт информация о размерах. Первые два числа соответствуют координатам левого нижнего угла картинки, а последние соответствуют координатам правого верхнего угла. Единицей измерения является «большой пункт» (`bp=1/72 in`), который примерно равен 0.351 мм. Для вёрстки текста указанной информации достаточно.

Чтобы из уже имеющегося одностраничного PostScript-файла сделать EPS необходимо и, как правило, достаточно добавить `BoundingBox`. Для вычисления искоемых размеров можно воспользоваться утилитой **ps2eps** из одноимённого пакета. Если же в стандартной поставке эта программа отсутствует, то можно напрямую воспользоваться программой Ghostscript — свободным программным интерпретатором PostScript:

```
> gs -q -dSAFER -dNOPAUSE -dBATCH -sDEVICE=bbbox «имя файла»
```

Размеры выясняются с помощью указания специального драйвера `bbbox`. Ключи `-q`, `-dNOPAUSE` и `-dBATCH` используются для подавления не нужной информации и вопросов со стороны программы. Ключ `-dSAFER` гарантирует что Ghostscript не будет производить никаких деструктивных действий<sup>2</sup>.

Ещё одной особенностью EPS-формата является возможность добавлять растровое изображения для предварительного просмотра. Это было сделано для случаев, когда программы не понимают PostScript, но что-то на месте дырки для картинки отобразить надо. Такое добавление идёт в разрез с принципиальной кросс-платформенностью PostScript и по возможности добавление картинки для предварительного просмотра следует избегать. Но в любом случае для операции с этим расширением, в том числе и для добавления/удаления можно воспользоваться утилитой **epstool** из одноимённого пакета.

В конце рассказа про EPS хотелось бы упомянуть о замечательной утилите **pstoedit** из, естественно, одноимённого же пакета. Не все, но некоторые из более-менее внятно созданных PostScript-файлов она ухитряется перевести в редактируемый век-

---

<sup>1</sup>Некоторые программы перед комментарием добавляют бинарный мусор. Не будем тыкать пальцем в драйвер вывода для PostScript-принтеров одной очень распространённой альтернативной операционной системы. Для полноценной работы с такими файлами этот мусор необходимо удалить.

<sup>2</sup>Отключается возможность выполнения таких команд, как удаление и переименование, а чтение файлов происходит в режиме `read-only`. Очень полезный ключ, если Ghostscript используется в качестве фильтра.

торный графический формат. Это упрощает работу с правкой файлов, которые не имеют исходников.

## 4.2. Как из растра сделать EPS

Одним из важных вопросов является конвертация растровых форматов в EPS. Растр гораздо проще создавать. Кое-где, например в случае снимков экрана, применение растра оптимальнее векторных форматов. Стандартные подходы, как в случае утилиты **convert** из пакета ImageMagick, не всегда дают оптимальные результаты.

Возможным и вполне разумным решением является замена традиционной линейки: **latex**→**dvips**→**[ps2pdf]** на **pdflatex**, который сразу из коробки поддерживает растровые форматы PNG<sup>3</sup> и JPEG<sup>4</sup> которые можно внедрять в формат PDF<sup>5</sup> на прямую. Массового перехода на эту технологию пока не видно, но заметное движение в эту сторону есть. У неё есть неоспоримые достоинства, но она не лишена недостатков. Рассказ о **pdflatex** выходит за рамки *этой* статьи.

Конвертацию из JPEG можно решить с помощью простой утилитки **jpeg2ps**, которую можно найти на любом CTAN<sup>6</sup>-архиве в директории **nonfree/support/jpeg2ps**. Утилита не преобразовывает JPEG-файл, а просто добавляет правильный eps-заголовок. Декомпрессия JPEG производится уже PostScript-интерпретатором<sup>7</sup>. К недостаткам утилиты можно отнести то, что в силу своей лицензии она не может распространяться со свободными дистрибутивами, а к достоинствам — отсутствие зависимостей.

Более комплексными решениями являются утилиты **sam2p** из одноимённого пакета и **bmeps**. Их так же можно найти на CTAN в директориях **graphics/sam2p** и **support/bmeps**, соответственно. **sam2p** является своеобразным комбайном, который поддерживает множество растровых графических форматов, в то время как **bmeps** фокусируется на PNG и JPEG. Обе эти программы позволяют получить вполне приличную eps-картинку для печати или для просмотра на экране. В обоих случаях необходимо поразбираться в ключах и настройках. С моей точки зрения **bmeps** является более удобным решением, производящим достаточно маленькие<sup>8</sup> по размеру eps-файлы, но и **sam2p** достаточно хорош.

Опять же на CTAN в директории **graphics/a2ping** можно взять довольно увесистый perl-скрипт **a2ping.pl**. Этот скрипт является своеобразной надстройкой над

---

<sup>3</sup>Portable Network Graphics — растровый графический формат использующий сжатие без потерь.

<sup>4</sup>Joint Picture Experts Group — самый популярный графический формат использующий сжатие с потерями.

<sup>5</sup>Portable Document Format — платформонезависимый формат электронных документов, созданный Adobe System.

<sup>6</sup>The Comprehensive TeX Archive Network. Центральный сайт <http://www.ctan.org>.

<sup>7</sup>Это стало возможным начиная с версии PostScript Level 2

<sup>8</sup>Это важно, так как мало какой растровый редактор может оптимально сохранить в EPS.

**Sam2p** и Ghostscript, что позволяет ему более-менее автоматически конвертировать из растра в PostScript и обратно.

Обзор внешних программ закончен. Далее слово «пакет» будет относиться к пакетам L<sup>A</sup>T<sub>E</sub>X, а не пакетам GNU/Linux-дистрибутива.

### 4.3. graphicx

Ответственным за создание «бокса» для размещения картинки является пакет **graphicx**<sup>9</sup>, а точнее команда `\includegraphics`:

```
% Эмблемы \TeX{} и \METAFONT{}, созданные
%Дуайном Бибби, взяты со странички Д.\,Э.~Кнута.

% Цветной пингвин взят из пакета \texttt{ps2pdf}
%от Rolf Niepraschk

\includegraphics[width=\textwidth]{title.1.eps}
```



В команде есть один обязательный параметр — вставляемая картинка. Необязательные параметры передаются с помощью пар «ключ»=«значение», разделяемых запятой. За подобный способ объявления параметров отвечает пакет **keyval**. Некоторые из поддерживаемых пакетом параметров перечислены ниже:

**bb** — позволяет исправить `BoundingBox` прямо в коде, не меняя eps. Значение представляет собой четыре числа кодирующие положение левого нижнего угла и правого верхнего, например: `[bb=127 464 430 667]`. Вместо одного **bb**, можно воспользоваться четвёркой ключей: `[bbllx=127,bbly=464,bbx=430,bbry=667]`, каждому из которых передаётся только одно значение.

Кроме перечисленных ключей для модификации `BoundingBox` можно использовать **viewport** — четыре числа значения описывают границы `BoundingBox`, где в качестве центра координат выбирается левый нижний угол уже существующего описания и **trim** — четыре числа значения описывают отступы от левой, нижней, правой и верхней границ.

**clip** — обрезает вставленную картинку по `BoundingBox`. Это необходимо сделать в случае изменения границ для «выкусывания» части картинки, иначе она будет «вылезать» за пределы выделенного её бокса. По умолчанию имеет значение `false`. Отсутствие значение у ключа **clip** при его упоминании эквивалентно значению `true`. Подобное поведение верно и для других логических переключателей.

---

<sup>9</sup>**graphicx** пришёл на смену пакету **graphics** — различия в последней букве. Команды из предыдущего пакета также можно использовать, но настоятельно не рекомендуется.

**angle** — поворачивает, картинку на указанный угол в градусах.

**origin** — определяет координаты центра вокруг которого вращается рисунок. Кроме непосредственно координат **origin** принимает и буквенные сокращения: **l**, **b**, **r** и **t** — соответствует центру вращения слева, снизу, справа и сверху. В этом случае выбирается середина указанной стороны. Возможны комбинации, задающие углы картинки: **lt**, **rt**, **rb** и **lb**. **c** — означает центр картинки.

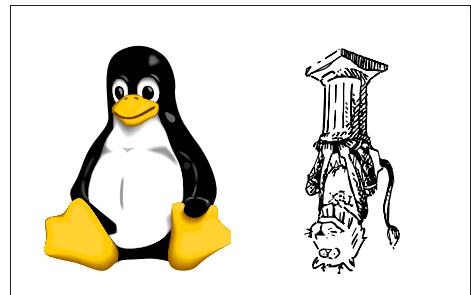
**width** — ширина вставляемой картинки.

**height** — высота вставляемой картинки.

**scale** — масштабный коэффициент.

**keepaspectratio** — логический переключатель. Модифицирует параметры высоты и ширины картинки в сторону уменьшения с целью сохранения естественных пропорций картинки.

```
\includegraphics[trim=110 0 105 100,clip,
  width=0.49\textwidth]{title.1.eps}
\hspace{0.5cm}
\includegraphics[viewport=0 0 100 200,clip,
  width=0.49\textwidth,
  height=3cm,keepaspectratio,
  angle=180,origin=c]{title.1.eps}
```



Аргументы `\includegraphics` интерпретируются слева направо. Для команд вращения и масштабирования порядок следования *имеет* значение.

## Определение своих правил

Пакет **graphicx** предоставляет возможность перед вставкой картинки вызвать внешнюю программу для её обработки. Например, так можно добавить возможность включения в документ png-файлов:

```
\DeclareGraphicsRule{.png}{eps}{.bb}{'bmerp -p3 -c #1}
```

Первый параметр определяет расширение нового формата графики, для которого задаются правила. В представленном примере это `.png`. Второй параметр указывает тип графики. После преобразования это будет `eps` — **dvips** по умолчанию ничего другого и не знает. Третий параметр определяет расширение файла откуда считывать параметры `BoundingBox`. Файл должен содержать одну строчку вида:

```
%%BoundingBox: 0 0 848 979
```

До вставки для каждой png-картинки необходимо создать такой файл, например, следующим образом:

```
bmerp -b «картинка».png «картинка».bb
```



Последний параметр определяет команду, которую следует выполнить для преобразования картинки. Команда должна выдавать результат на стандартный вывод. #1 соответствует имени обрабатываемого файла. Непосредственное выполнение команды происходит при трансляции dvi-файла.

Выполнение внешней команды является потенциально *опасной* процедурой, поэтому защита по умолчанию этого не позволяет. Для просмотра dvi-файла через **xdvi** следует использовать ключик `-allowshell`. В противном случае будет выдаваться запрос на исполнение команды каждый раз, когда встречается вставка по новым правилам. Для преобразования в PostScript в случае **dvips** также следует отключить защиту с помощью ключика `-R0`. Лучше всё-таки по возможности избегать вышеописанной процедуры и сразу готовить картинки в eps-формате.

Для трактования всех неизвестных драйверу расширений как eps следует применить команду:

```
\DeclareGraphicsRule{*}{eps}{*}{}{}
```

Это полезно в случае вставки картинок MetaPost, которые по умолчанию не имеют расширений. Если третий параметр равен «звёздочке», то это означает, что `BoundingBox` следует искать в том же файле, что и графику.

## 4.4. Плавающие объекты

Мало просто поместить картинку — её надо разместить красиво и по возможности она должна это делать самостоятельно. Просто `\includegraphics` для этого дела не очень подходит, так как размещение регулируется исключительно пользователем. Для этой цели в Л<sup>A</sup>T<sub>E</sub>X имеется специальная сущность: *плавающий объект* (`float`<sup>10</sup>). Если для этого объекта нет места на текущей странице, то он переносится на следующую.

Для размещения картинок стандартные классы определяют плавающий объект как окружение **figure**:

```
\begin{figure}[ht]
  \centering%центрируем картинку
  \includegraphics{«картинка»}
  \caption{«подпись»}\label{fig:metka}
\end{figure}
```

В качестве необязательного параметра окружению **figure** можно передать допустимые способы размещения плавающего объекта:

- h** — разместить по возможности здесь же,
- t** — разместить в верхней части страницы,

---

<sup>10</sup>Пакет, который позволяет создавать новые типы плавающих объектов, так и называется **float**. Вместо этого пакета в качестве замены можно использовать **floatraw**, созданный Ольгой Лапко.

**b** — разместить в нижней части страницы,

**p** — разместить на отдельной странице, где нет ничего кроме плавающих объектов.

Приоритет для размещения определяется порядком следования букв. Если первой следует буква **h**, то в случае неудачи  $\LaTeX$  размещает плавающий объект на *следующей* странице. Если же первыми следуют буквы **t** или **b**, то размещение организуется на текущей странице.

Для «красивого» размещения картинок  $\LaTeX$  опирается на некоторые значения по умолчанию, которые не всегда для текущего случая могут быть оптимальными. Поэтому, если очень хочется разместить картинку, например, внизу, то пожелание можно усилить с помощью восклицательного знака: `[b!]`.

## Управление плавающими объектами

Если плавающих объектов в документе немного, то всё будет хорошо без какого-либо вмешательства человека. Но если их много, то так или иначе надо будет управлять их размещением.

**clearpage** Если  $\LaTeX$  не справляется с размещением картинок, то он переносит их на следующую страницу. В какой-то момент может накопиться целая «толпа» таких перенесённых картинок и возникнет необходимость в их «насильственном» выводе в каком-то определённом месте. Для этого существует команда `\clearpage`. При вызове этой команды завершается текущая страница и выводятся все отложенные плавающие объекты и только потом продолжается обычный вывод текста. Единственная проблема этой команды в том, что текущая страница по ней обрывается. Чтобы избежать обрыва можно воспользоваться пакетом **afterpage**, точнее одноимённой командой из него:

```
\afterpage { \clearpage }
```

Команда `\afterpage` откладывает выполнение указанных в ней инструкций до конца текущей страницы.

**suppressfloats** Команда `\suppressfloats` полностью подавляет размещение плавающих объектов на текущей странице. В качестве необязательного параметра ей можно передать **t** или **b** — в этом случае запрет распространяется только на размещение плавающих объектов вверху или внизу страницы соответственно.

**placeins** Пакет **placeins** не даёт «утекать» плавающим объектам за установленные пределы. Барьер устанавливается с помощью команды `\FloatBarrier`.

Это бывает полезно, когда хочется чтобы все картинки не выходили за пределы своего раздела. В этом случае следует переопределить нужную команду секционирования для установки перед ней барьеров. В случае команды секционирования раздела (`section`) достаточно передать опцию `[section]` пакету при загрузке:

```
\usepackage [ section ] { placeins }
```

**endfloat** Часто при подготовке статей требуют их размещения после текста на отдельных страницах предваряя эту галерею списком иллюстраций. Пакет **endfloat** именно это и делает. Достаточно его загрузить.

### «Упаковка» картинок в один float

Для уменьшения «поголовья» плавающих объектов полезно размещать картинки группами. Например, чтобы разместить две картинки рядом можно применить команду `\parbox` или окружение **minipage**:

```
\parbox [ «позиционирование» ] { «ширина» } { «текст» }
```

```
\begin { minipage } [ «позиционирование» ] { «ширина» }
```

текст

```
\end { minipage }
```

В обоих случаях есть обязательный параметр ширина по которой формируется создаваемый бокс и необязательный «позиционирование» — расположение сформированного бокса относительно базовой линии по вертикали. Позиционирование может проводиться по центру (опция [c] — верно по умолчанию), по верхней линии ([t]) и по нижней линии бокса ([b]). Шаблон для двух рядом стоящих рисунков может иметь примерно следующий вид:

```
\begin { figure } [ ht ] \centering
  \parbox [ b ] { 0.49 \textwidth } { \centering
    \includegraphics { «рисунок-1» }
    \caption { «подпись-1» } \label { fig : metka - 1 } }
  \hfil \hfil %раздвигаем боксы по горизонтали
  \begin { minipage } [ b ] { 0.49 \textwidth }
    \centering
    \includegraphics { «рисунок-2» }
    \caption { «подпись-2» } \label { fig : metka - 2 }
  \end { minipage }
\end { figure }
```

Использование команды `\parbox` или окружение **minipage** зависит исключительно от личных предпочтений. С помощью них можно организовать и более сложные конструкции.

Для целей автоматизации упаковки можно использовать и специализированные пакеты:

**subfig** — организует группы из множества картинок. Относительно современный пакет.

**miniplot** — делает то же что **subfig**, хоть и менее изопрённо.

**figsize** — специализируется на автоматическом вычислении размеров картинок для размещения их в указанных пределах.

**dpfloat** — определяет новый тип плавающего окружения, занимающего сразу две страницы. Двойные иллюстрации на развороте.

## Картинки «в оборку»

Маленькие иллюстративные рисунки удобно делать в оборку с текстом, т. е. текст должен обтекать их. Такие картинки располагаются на внешней стороне страницы, т. е. слева для чётных и справа для нечётных страниц или в случае одностороннего режима печати.

Традиционно описываются два пакета для создания подобных рисунков: **floatflt** и **wrapfig**. **floatflt** более автоматизирован для размещения картинок, но он так же чаще «ломается» при большом числе плавающих объектов. Возможны даже «потери» картинок. Упомянутые пакеты определяют окружения **floatingfigure** и **wrapfigure**, соответственно.

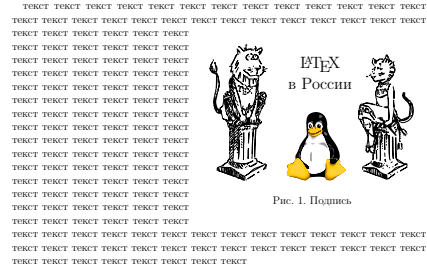


Рис. 1. Подпись.

Рис. 4.1. Картинка в оборку.

```
\begin{floatingfigure}[«размещение»]{«ширина»}
...
\end{floatingfigure}
```

Необязательный параметр «размещение» позволяет изменить алгоритм размещения картинки:

**rft** — размещать справа,

**lft** — размещать слева,

**vft** — слева для чётных и справа для нечётных страниц (по умолчанию).

```
\begin{wrapfigure}[«число строк в оборке»]
{«размещение»}{«ширина»}
...
\end{wrapfigure}
```

В отличие от окружения **floatingfigure** **wrapfigure** требует определить правила размещения картинки. Доступные варианты: справа (**{r}**), слева (**{l}**), с внешней стороны страницы (**{i}**) и с внутренней стороны страницы (**{o}**). Если вместо строчных букв передать заглавные, то включается запрет на сдвиг по вертикали — картинка должна быть размещена начиная с той строки абзаца, в которой она была определена.

Необязательный параметр «число строк в оборке» позволяет указать число строк текста, которые должны быть сбоку от картинки. При этом выносная формула считается за три строки текста. Если параметр не определён, то число строк вычисляется автоматически, к сожалению не всегда оптимально.

Свою процедуру размещения картинки в оборку с текстом предлагает так же и пакет **nccfloats**<sup>11</sup>:

```
\sidefig («ширина картинки»)(«ширина текста»)  
{\includegraphics {«картинка»}}{«текст»}
```

В этом случае предлагается передавать команде `\sidefig` и саму картинку и текст, помещаемый сбоку. Параметр «ширина текста» можно опустить. Тогда текст занимает всё оставшееся пространство. Подробности можно посмотреть в документации **nccfloats.pdf**.

## Подписи к рисункам

Для добавление подписи к рисунку используется команда `\caption`, которую можно использовать только внутри плавающих объектов. В качестве обязательного параметра передаётся текст подписи. При выводе подпись центрируется, если она достаточно мала. В противном случае подпись оформляется в виде абзаца. Текст подписи не должен содержать команд разрыва строки. Все хрупкие команды внутри подписи должны быть защищены с помощью команды `\protect`. `\caption` можно передать также необязательный параметр, который должен представлять собой краткую версию подписи, появляющуюся в автоматически создаваемых списках.

Оформление подписи жёстко привязано к стилю документа и изменить её без переопределения самой команды `\caption` не просто. Для модификации параметров следует воспользоваться пакетами **caption** или **scaption**. Оба упомянутые пакета позволяют «покрутить» множество ручек и снабжены исчерпывающей и очень объёмной документацией.

При включении русской опции `\usepackage[russian]{babel}` перед подписью выводится слово «Рис.» за которым идёт автоматически вычисляемый порядковый номер картинки. В качестве разделителя между счётчиком и подписью по умолчанию используется двоеточие. Для замены двоеточия на точку в преамбуле достаточно набрать, например, следующее:

```
\usepackage{scaption}  
% заменяем для рисунков ':' после номера рисунка на '.'  
\captiondelim { . } % после точки стоит пробел!
```

Кроме традиционного размещения подписи под картинками, подпись можно вынести, например на поля страницы. Не даром же стандартные классы имеют такие широкие поля. Пакет **mcaption** определяет окружение **margincap**:

<sup>11</sup>**nccfloats** входит в коллекцию **ncctools**, созданную А. И. Роженко

```
\begin{figure}[ht]
  \begin{margincap}{«Подпись»}
    \includegraphics{«картинка»}
  \end{margincap}
\end{figure}
```

В качестве обязательного параметра окружению передаётся подпись, а внутри определяется картинка.

## 4.5. Заключение

Странный получился рассказ. В статье, имеющей заголовок «Графика» не было ни слова сказано о том, как эту графику, собственно говоря, создавать. А ведь *есть* что сказать, но в данном случае информация о размещении и оформлении готовых картинок поважнее будет. Но и о том, как  $\text{\LaTeX}$  умеет рисовать тоже будет рассказано, но чуть попозже.

## Врезка: OpenSource шагает по стране. «Юзеры» наступают.

Посмотрите на картинку. Что это? Это TrX — простой векторный редактор, написанный под альтернативную операционную систему.

Чем же он интересен? Гххм, сложный вопрос. Ну да — его можно запустить в wine и от некоторых действий он даже не выпадает в осадок. Но это не Inkscape и отнюдь не xfig. Ещё один ничем не выдающийся велосипед, если бы не три «но».

Первое «но» в том, что этот «велосипед» (<http://sourceforge.net/projects/tpx/>) распространяется автором под открытой лицензией GPL.

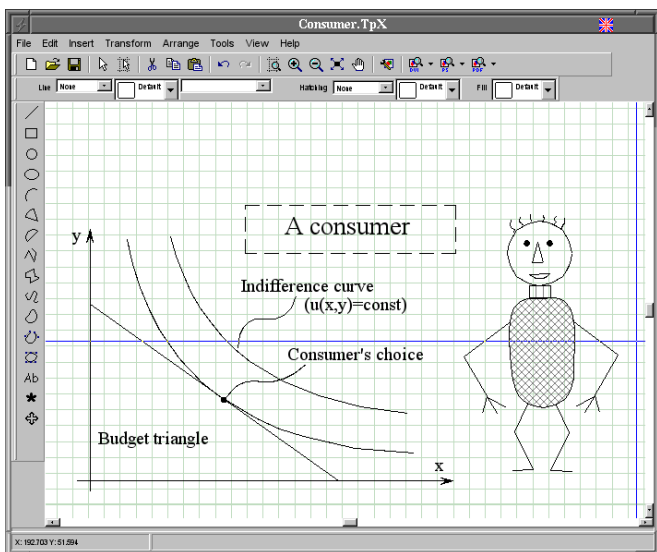


Рис. 4.2. TrX — простой векторный редактор.

не просто (её там нет) — отсутствие технической культуры, да диссертация написана в альтернативном текстовом редакторе — как это принято у экономистов. Но автор не технарь — он тот самый «юзер» о котором так любят рассуждать программисты. «Юзер» недовольный нехваткой инструментов настолько, что начал создавать их самостоятельно. Не просто создавать, а распространять результаты своего труда под свободной лицензией. И этой программой вполне можно пользоваться.

Правда, пока автор не освоит в совершенстве Lazarus, более-менее стабильной версии под Linux вряд-ли можно ожидать. Ну, естественно, если кто-нибудь ему не поможет, а то «юзеры» будут вынуждены взять власть в свои «очумелые» руки. ☺

Второе «но» заключается в специализации этого редактора — он предназначен для создания простых картинок с последующим внедрением в L<sup>A</sup>T<sub>E</sub>X. То есть это всё-таки специализированный велосипед.

«Но» третье и основное заключается в авторе. Это Александр Анатольевич Цыплаков (<http://www.nsu.ru/ef/tsy/>) — кандидат экономических наук и доцент Новосибирского государственного университета. Да, для разработки использовался Delphi, да информацию, что программа под GPL найти в коде

## Документация и программный код

+++ Ошибка Деления На Огурец.  
Переустановите Вселенную И Перезагрузитесь +++

Так зависает Гекс.

Источник: «Санта-Хрякус» от Терри Пратчетта

Программирование под Linux вполне естественное занятие. Написание документации неотъемлемая часть этого процесса.  $\LaTeX$  достоин быть включённым в технологическую цепочку по выпуску программного продукта.

Если вспомнить историю, то Д.Э. Кнут создал  $\TeX$  именно для целей представления кода и алгоритмов в своём глобальном пятитомнике «Искусство программирования».

### 5.1. Спецсредства

Чтобы украсить инструкцию надо добавлять в неё «пятна». Перегружать не стоит, но пару мыслей выделить вполне реально. Упомянутые ниже приёмы далеко не все имеющиеся в  $\LaTeX$  — это просто демонстрация возможностей.

#### 5.1.1. `keystroke`

Иногда в тексте необходимы фразы вида: «Для выхода из программы нужно нажать клавишу Esc.» Макрос `\keystroke`, определённый в одноимённом пакете **keystroke**, позволяет выделить название клавиши, примерно следующим образом:

Для продолжения нажмите  
`\keystroke{<<любую клавишу>>}`.

Для продолжения нажмите «любую клавишу».

В пакете определены так же многие клавиши, имеющиеся на стандартной клави-



атуре. Пакет очень прост и имеет зачатки интернационализации — его легко адаптировать для своих нужд.




























<code>\Spacebar</code>		<code>\Enter</code>		<code>\Return</code>	
<code>\Esc</code>		<code>\BSpace</code>		<code>\Tab</code>	
<code>\Alt</code>		<code>\AltGr</code>		<code>\Del</code>	
<code>\Shift</code>		<code>\PgUp</code>		<code>\PgDown</code>	
<code>\End</code>		<code>\Ctrl</code>		<code>\Home</code>	
<code>\Ins</code>		<code>\LArrow</code>		<code>\RArrow</code>	
<code>\UArrow</code>		<code>\DArrow</code>		<code>\PrtSc</code>	
<code>\Scroll</code>		<code>\Break</code>		<code>\NumLock</code>	
<code>\keystroke{A}</code>		<code>\keystroke{Я}</code>		<code>\keystroke{F1}</code>	

Рис. 5.1. Клавиши, определённые в `keystroke`

### 5.1.2. LCD-дисплей

LCD-дисплеи сейчас встроены даже в кофемолки. Они легко узнаваемы, поэтому нет необходимости копировать их вид в документацию с помощью фотографий — достаточно нарисовать что-то похожее. Изобразить вид дисплея можно с помощью пакета  $\LaTeX$  `lcd`.

```
\definecolor{darkgreen}{rgb}{0.22,0.26,0.19}
\definecolor{lightgreen}{rgb}{0.05,0.97,0.55}
\LCDcolors{darkgreen}{lightgreen}
\centering
\LARGE\textLCD{12}|Linux Format|\|[2mm]
\LCDcolors{lightgreen}{darkgreen}
\small\textLCD{12}|Linux Format|
```



Для определения цветов используется макрос `\definecolor` из пакета `color`. Команда `\LCDcolors` формирует цвет букв и фона, а макрос `\textLCD` выводит LCD-подобный текст на экран. `\textLCD` понимает стандартные команды изменения размера шрифта, поэтому его можно использовать совместно с обычным текстом внутри абзаца.

По умолчанию определены только латинские буквы, цифры и некоторые из стандартных символов. Для определения других символов можно воспользоваться макросом `\DefineLCDchar`. Макросу передаётся имя символа и битовая маска, определяющая картинку  $5 \times 7$  точек. Имя символа может быть однобуквенным, тогда соответствующая буква замещается новым рисунком, или многобуквенным, тогда созданный рисунок кодируется указанным словом в фигурных скобках. Другие размеры матрицы в пакете отсутствуют, но при желании его вполне можно доработать.

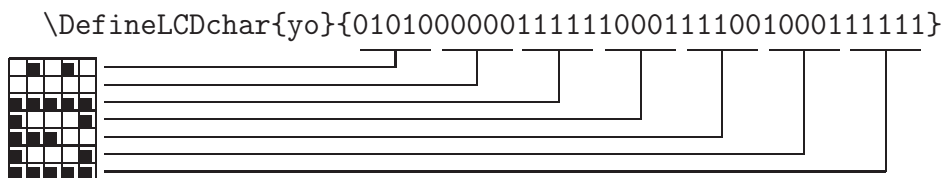
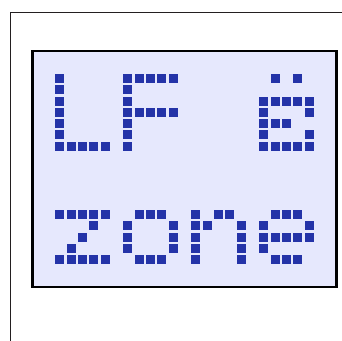


Рис. 5.2. Определяем букву «Ё» для LCD

Для эмуляции дисплея используется команда `\LCD` в качестве обязательных параметров ей передаётся число строк и число столбцов, за которыми следует содержание строк, разделённых каким-то разделителем. В приведённом примере в качестве разделителя используется вертикальная черта, но вместо неё может быть любой символ:

```
\DefineLCDchar{yo}{01010000001111110001111001000111111}
\definecolor{lightblue}{rgb}{0.9,0.91,0.99}
\definecolor{darkblue}{rgb}{0.14,0.2,0.66}
\LCDcolors{darkblue}{lightblue}
\LCDframe
\setlength{\LCDunitlength}{1.5mm}
\LCD{2}{4}|LF {yo} |
      |zone |
```



### 5.1.3. Битовые поля

Для описания сетевых протоколов, а так же для бинарных форматов данных удобнее всего представить последовательность битов графически, то есть в виде таблицы. Это специализация пакета **bytefield**. В пакете определено одноимённое окружение **bytefield** в качестве обязательного аргумента которому передаётся ширина таблицы в битах:

```
\begin{bytefield}{«битовая ширина поля»
  «битовые поля»
\end{bytefield}
```

В окружении **bytefield** работают команды `\wordbox` и `\bitbox`, которые формируют поля занимающие ширину таблицы или только часть её, соответственно:

```
\wordbox [«рамка»]{«число строк»}{«текст»}
\bitbox [«рамка»]{«число занимаемых битов»}{«текст»}
```

Не обязательный параметр «рамка» позволяет сформировать обрамление для текущего битового поля. Значение по умолчанию `[lrbt]` означает, что рамка рисуется со всех сторон поля: **l** — слева, **r** — справа, **t** — сверху и **b** — снизу. Строки разделяются двойной обратной чертой `\\`.

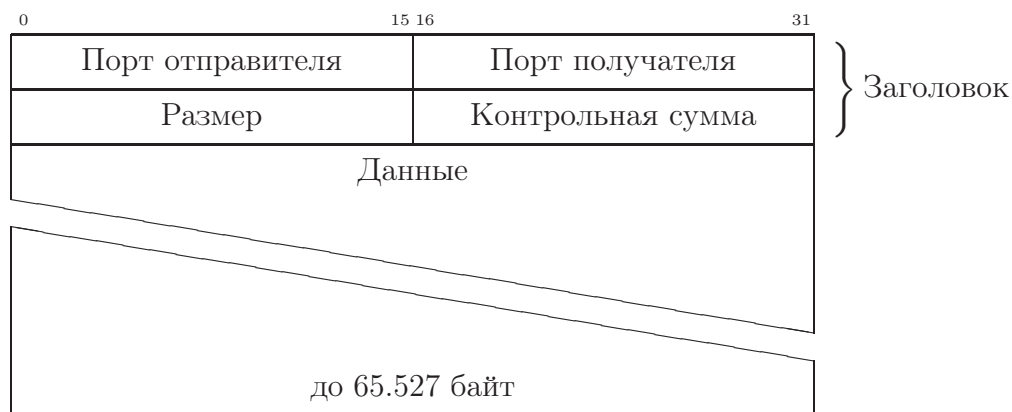


Таблица 5.1. Формат пакета UDP

Примерно следующим образом описывается формат пакета сетевого протокола UDP<sup>1</sup>:

```

\begin{bytefield}{32}
\bitheader{0,15,16,31}
\wordgroup{Заголовок}
\bitbox{16}{Порт отправителя}\bitbox{16}{Порт получателя}
\bitbox{16}{Размер}\bitbox{16}{Контрольная сумма}
\endwordgroup
\wordbox[lrt]{1}{Данные}
\skippedwords
\wordbox[lrb]{1}{до 65{.}527 байт}
\end{bytefield}

```

Кроме уже упомянутых команд создания полей при описании формата UDP использовалась команда нумерации столбцов `\bitheader`, конструкция для создания группы `\wordgroup` и макрос `\skippedwords` для формирования «разрыва».

В качестве обязательного аргумента команде `\bitheader` передаётся список нумеруемых битов, при этом можно передавать диапазоны чисел, например, `{0-31}`. В пакете определены два окружения для группировки битовых полей `\wordgroup` и `\wordgroupl` — отличие этих команд в том, что для первой заголовок группы вводится справа, а для второй — слева. Для более подробной информации следует обратиться к документации пакета.

## 5.2. Форматирование кода

$\text{\LaTeX}$  может использоваться не только для набора математики. Хотя набор математики безусловно вершина типографского искусства, но есть масса задач, где

<sup>1</sup>User Datagram Protocol — это сетевой протокол для передачи данных в сетях IP.



Для загрузки пакета **listings** необходимо добавить в заголовок следующие инструкции:

Listing 5.1. Заголовок listings

```
\usepackage{listings}
% подгружаемые языки — подробнее в документации listings
\lstloadlanguages {[LaTeX]TeX, bash, MetaPost, Fortran, Perl, C++, make}
% включаем кириллицу и добавляем кое-какие опции
\lstset{language=[LaTeX]TeX, % выбираем язык по умолчанию
        extendedchars=true, % включаем не латиницу
        escapechar=|, % |«выпадаем» в ЛАТЭХ|
        frame=tb, % рамка сверху и снизу
        commentstyle=\itshape, % шрифт для комментариев
        stringstyle=\bfseries} % шрифт для строк
```

Сразу после загрузки пакета рекомендуется «подгрузить» используемые в тексте языки программирования<sup>3</sup> с помощью макроса `\lstloadlanguages`. В квадратных скобках перед названием языка можно указать желательный диалект.

Команда `\lstset` позволяет устанавливать значения по умолчанию. Некоторые из полезных умолчаний перечислены ниже:

- Для того чтобы можно было печатать кириллицу, например в комментариях, следует определить переменную `extendedchars=true`<sup>4</sup>.
- Опция `escapechar` позволяет при наборе кода пользоваться услугами ЛАТЭХ напрямую. Всё, что находится между выбранными символами, обрабатывается средствами ЛАТЭХ. Естественно, если выбранный символ (в данном случае «|») используется в отображаемом языке, то могут возникнуть проблемы при компиляции. Для того чтобы обнулить `escapechar` достаточно ничего не писать за знаком равно при следующем переопределении.
- Инструкция `frame=<POSITION>` позволяет рисовать рамку вокруг сегмента кода. На вход принимаются буквы `t` — обрамление сверху, `b` — снизу, `l` и `r` —

<sup>3</sup>Текущая версия пакета 1.3с поддерживает следующие языки (в скобках указаны диалекты): АВАР, АСЛ, Ада (83, 95), Алгол (60, 68), Ant, Assembler (x86masm), Awk (gnu, POSIX), bash, Basic (Visual), C (ANSI, Handel, Objective, Sharp), C++ (ANSI, GNU, ISO, Visual), Caml (light, Objective), Clean, Cobol (1974, 1985, ibm) Comal 80, csh, Delphi, Eiffel, Elan, erlang, Euphoria, Fortran (77, 90, 95), GCL, Gnuplot, Haskell, HTML, IDL (empty, CORBA), inform, Java (empty, AspectJ), JVMIS, ksh, Lisp (empty, Auto), Logo, make (empty, gnu), Mathematica (1.0, 3.0), Matlab, Mercury, MetaPost, Miranda, Mizar, ML, Modula-2, MuPAD, NASTRAN, Oberon-2, OCL (decorative, OMG), Octave, Oz, Pascal (Borland6, Standard, XSC), Perl, PHP, PL/I, Plasm, POV, Prolog, Promela, Python, R, Reduce, Rexx, RSL, Ruby, S (empty, PLUS), SAS, Scilab, sh, SHELXL, Simula (67, CII, DEC, IBM), SQL, tcl (empty, tk), TeX (AllaTeX, common, LaTeX, plain, primitive), VBScript, Verilog, VHDL (empty, AMS), VRML (97), XML, XSLT.

<sup>4</sup>Если это не работает, то необходимо обновить пакет до последней версии или сменить дистрибутив ЛАТЭХ на более подходящий.

слева и справа, соответственно. В случае `frame=trbl` будет нарисована простейшая одинарная рамка. Опция `frame=` эквивалентен отказу от обрамления. Если вместо прописных букв указать заглавные `frame=TRBL`, то рамка будет двойная. В пакете есть возможность сделать рамки посложнее.

Все команды, определённые в пакете `listings`, начинаются с префикса `lst`. Команда для включения небольших кусочков кода `\lstinline !код!` аналогична по действию команде `\verb!текст!`.

Сегмент кода оформляется с помощью окружения `lstlisting`:

```
\begin{lstlisting}[language=Perl,
  caption={Включение сегмента кода}]
# проверка для перезаписи
if (open(CHECK,"<$file")) {
  $cmd=$term->
  readline("Overwrite (yes/NO): ");
if (lc($cmd) ne "yes") {die;}
close(CHECK);}

\end{lstlisting}
```

Listing 5.2. Включение сегмента кода

```
# проверка для перезаписи
if (open(CHECK,"<$file")) {
  $cmd=$term->
  readline("Overwrite (yes/NO): ");
if (lc($cmd) ne "yes") {die;}
close(CHECK);}
```

Необязательный параметр может принять опции, специфичные для оформления этого куска кода. Например, опция `language` позволяет установить язык программирования отличный от выбранного по умолчанию, а `caption` создаёт подпись к фрагменту кода.

Файлы можно включать с помощью команды `\lstinputlisting` :

```
%установка значений по умолчанию
\lstset{numbers=left , language=MetaPost ,
  backgroundcolor=\color{yellow} ,
  frame=shadowbox , rulesepcolor=\color{black}}

%вставка файла
\lstinputlisting[firstline=16, lastline=24,
  emph={forsuffixes , text , bpath} , emphstyle={\color{red}} ,
  emph={ [2] fill , unfill } , emphstyle={ [2] \bfseries \underbar } ,
]{ intro.mp}
```

```
16 \vardef drawshadowed(expr dx,dy)(text t) =
17   \fixsize(t);
18   \forsuffixes s=t:
19     \fill bpath.s shifted (dx,dy);
20     \unfill bpath.s;
21     \drawboxed(s);
22 %   \draw pic(s) withcolor red; %цвет текста
23   \endfor;
24 \enddef;
```

С помощью опций `firstline` и `secondline` можно указать строки, которые следует вывести. В зависимости от выбора языка форматирование существенно меняется. Инструкция `numbers=left` нумерует строки слева.

Для работы с цветами лучше загрузить уже упоминавшийся ранее пакет `color`. Цвета хороши для выделения каких-то ключевых слов и подложки, за которую отвечает опция `backgroundcolor`. Возможности для определения своих «словариков» предоставляет опция `emph=<список ключевых слов>`. В начале списка может идти его метка в квадратных скобках, таким образом можно поддерживать одновременно несколько списков. С помощью опции `emphstyle` можно определить способ выделения ключевых слов.

Обычно код располагается прямо по месту основного текста, так как обсуждение исходников можно не прерывать в самом коде, благо есть комментарии. Но при желании можно воспользоваться опцией `float`, чтобы из фрагмента кода получился полноценный «плавающий» объект.

Пакет с учётом диалектов поддерживает свыше сотни распространённых языков программирования и разметки. Так что, скорее всего, Вам не придётся определять свой язык с помощью инструкции `\lstdefinlanguage`. Но если очень хочется, то и это возможно.

## 5.3. Представление алгоритмов

Собственно говоря, именно то, ради чего Д. Э. Кнут и создал `TeX`. Поэтому пакеты для облегчения записи алгоритмов в `LaTeX` были с самого его рождения. На текущий момент число даже стандартных пакетов, подпадающих под эту тематику, больше десятка. Здесь рассмотрена только малая часть из них.

### 5.3.1. `algorithms`

Пакет `algorithms` ориентирован на написание алгоритмов, а не на представление кода. Это позволяет отрешиться от форматирования и сосредоточиться на основной задаче. Пакет определяет окружение `algorithmic`. Для использования в преамбуле следует загрузить одноимённый стиль. Если необязательный аргумент определён, то осуществляется нумерация строк. Если аргумент равен 1, то нумеруются все строки, если 2 — то каждая вторая, а далее по индукции.

Команда `\STATE` определяет простое утверждение. Условный оператор представлен командами `\IF{<условие>}`, `\ELSIF{<условие>}`, `\ELSE` и `\ENDIF`. Циклы представлены операторами `\FOR` и `\FORALL`, которые закрываются командой `\ENDFOR`. Аналогично присутствуют пары `\WHILE{<условие>}` — `\ENDWHILE`, `\REPEAT` — `\UNTILL{<условие>}` и бесконечный цикл `\LOOP` — `\ENDLOOP`. Кроме уже перечисленных конструкций определены предварительное условие для корректного выполнения алгоритма `\REQUIRE`, постусловие, которое должно выполняться при корректной работе алгоритма, `\ENSURE`, возвращение результата `\RETURN`, промежуточная печать `\PRINT` и комментарий `\COMMENT`.



**Algorithm 1** Пример использования пакета **algorithm**

```

\begin{algorithmic}[1]
\IF{\(i\leqslant 0\)} \STATE \(\i\gets 1\) \ELSE
\IF{\(i\geqslant 0\)} \STATE \(\i\gets 0\)
\COMMENT{смысла в этом алгоритме не ищите}
\ENDIF
\ENDIF
\ENSURE \(\i\geqslant 0\)
\FORALL{\(\xi \in \mathcal{A}\)}
\STATE \(\mathcal{B}\gets \xi^2\)
\ENDFOR
\RETURN \(\mathcal{B}\)
\end{algorithmic}

```

```

1: if  $i \leq 0$  then
2:    $i \leftarrow 1$ 
3: else
4:   if  $i \geq 0$  then
5:      $i \leftarrow 0$  {смысла в этом
                       алгоритме не ищите}
6:   end if
7: end if
Ensure:  $i \geq 0$ 
8: for all  $\xi \in \mathcal{A}$  do
9:    $\mathcal{B} \leftarrow \xi^2$ 
10: end for
11: return  $\mathcal{B}$ 

```

Собственно говоря, всё. Псевдокод автоматически разбивается на строки и форматируется в соответствии с общепринятыми представлениям. Очевидно также, что навыки набора математики будут здесь очень кстати. Подробности по настройке пакета следует выяснять в документации к нему: `algorithms.pdf`.

Для того чтобы из объекта **algorithmic** сделать «плавающий объект» можно воспользоваться окружением **algorithm**, для использования следует в преамбуле загрузить одноимённый стиль. Внутри **algorithm** можно использовать команды `\caption` и `\label`.

### 5.3.2. Клоны **algorithm**

Используя имеющиеся наработки пакета **algorithm**, был создан **algorithmicx**. Этот пакет предоставляет более расширенный набор команд. Кроме этого пользователю предоставляются команды, которые позволяют сформировать свои алгоритмические конструкции. Автор так же предоставил вариант форматирования отступов принятый в Pascal, что позволяет относительно легко переводить программы на этом языке к виду, годному для красивой распечатки. Пакет использует те же окружения, что и **algorithm**, что приводит к несовместимости этих двух пакетов.

Решение схожей функциональности предоставляет пакет **algorithm2e**. Форматирование C-подобно. Предоставлен избыточный набор конструкций и возможность самому создавать новые структуры. Есть зачатки локализации. Пакет использует окружение **algorithm**, что не позволяет работать совместно с одноимённым пакетом **algorithm**.



### 5.3.3. `clrscope`

Пакет `clrscope` представляет возможность набирать псевдокод, как это делали авторы книги «Алгоритмы: построение и анализ» Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест и Клиффорд Штайн<sup>5</sup>. Для работы с пакетом необходимо загрузить одноимённый стиль. Прекрасный пример того, как можно адаптировать  $\LaTeX$  для нужд создания книг по программированию.

```
\begin{codebox}
\Procname{
  $\proc{Сортировка методом вставок}$}
\li \For $j \gets 2$ \To $\id{length}[A]$
\li \Do $\id{key} \gets A[j]$
\li $i \gets j-1$
\li \While $i > 0$ and $A[i] > \id{key}$
\li \Do $A[i+1] \gets A[i]$
\li $i \gets i-1$ \End
\li $A[i+1] \gets \id{key}$ \End
\end{codebox}
```

```
СОРТИРОВКА МЕТОДОМ ВСТАВОК
1  for  $j \leftarrow 2$  to  $length[A]$ 
2      do  $key \leftarrow A[j]$ 
3           $i \leftarrow j - 1$ 
4          while  $i > 0$  and  $A[i] > key$ 
5              do  $A[i + 1] \leftarrow A[i]$ 
6                   $i \leftarrow i - 1$ 
7           $A[i + 1] \leftarrow key$ 
```

Рис. 5.3. Пример использования пакета `clrscope`

### 5.3.4. `pseudocode`

Профессора Дональд Л. Крехер (Donald L. Kreher) и Дуглас Р. Стинсон (Douglas R. Stinson) написали книгу «Combinatorial Algorithms: Generation, Enumeration and Search». Специально для этой книги в целях написания псевдокода они создали пакет, который так и назвали: `pseudocode`. Дональд Л. Крехер использовал одноимённое окружение и в своей следующей книге по алгоритмам, выпущенной уже 2005 году. Пакет развивается и поддерживается.

## 5.4. Заключение

К сожалению в книгах по  $\LaTeX$  редко рассматриваются структуры полезные для представления программных текстов или псевдокода. Здесь я попытался восполнить этот зияющий пробел. Тема настолько обширна, что разрабатывать её можно почти бесконечно.  $\LaTeX$  сам по себе код, поэтому программистам, по идее, должно быть уютно в его окружении.

---

<sup>5</sup>*Introduction to algorithms*, Second Edition Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein

```

\begin{pseudocode}{C2F\_таблица}
    {\text{от}, \text{до}}
\PROCEDURE{C2F}{c}
\COMMENT{Преобразование
    $\circ C$ to $\circ F$}
f \GETS {9c/5} + 32
\RETURN{f}
\ENDPROCEDURE
\MAIN
x \GETS \text{от}
\WHILE x \leq \text{до} \DO
\BEGIN
\OUTPUT{x, \CALL{C2F}{x}}
x \GETS x+1
\END
\ENDMAIN
\end{pseudocode}

```

**Algorithm 1.1:** C2F\_ТАБЛИЦА(от, до)

```

procedure C2F(c)
  comment: Преобразование °C → °F
   $f \leftarrow 9c/5 + 32$ 
  return (f)

main
   $x \leftarrow \text{от}$ 
  while  $x \leq \text{до}$ 
  do  $\left\{ \begin{array}{l} \text{output } (x, \text{C2F}(x)) \\ x \leftarrow x + 1 \end{array} \right.$ 

```

Рис. 5.4. Пример использования пакета `pseudocode`

## Исходники $\text{\LaTeX}$ и контроль версий

$\text{\LaTeX}$ -исходник тоже представляет собой код. И как всякий код он достоин включение в систему контроля версий. Часто бывает любопытно узнать версию текущего документа и последний момент его обновления. Если в качестве системы контроля версий используется Subversion или `svn`, то для начала следует загрузить пакет `svn`<sup>6</sup>.

```

\usepackage{svn}
\SVN $Date$
\SVN $Rev$

```

При этом в текст следует добавить метки, предваряемые командой `\SVN`. Для интерполяции меток в системе Subversion при обновлении файла следует выполнить команды вида:

```

> svn propset svn:keywords "Date_Rev" «имя файла»
> svn commit -m "интерполяция_меток"

```

При этом `svn` передаётся информация какие именно метки требуется обновлять при выполнении `commit`. В данном случае это метки `Date` и `Rev` — дата и версия, соответственно. Более подробную информацию можно получить с помощью команды

```

> svn help propset

```

<sup>6</sup>Если же в вашем проекте используется CVS (Concurrent Versions System), то следует воспользоваться пакетом `rcs`. За подробностями следует обратиться к документации пакета.

## 5 Документация и программный код

Команда `\SVN $Date$` определяет команды `\SVNDate` и `\SVNTime`, ответственные за календарную дату и время. Все остальные команды вида `\SVN $Keyword$`, где `Keyword` — одна из интерполируемых меток `svn`, определяют команды `\SVNKeyword`.

После интерполяции метки будут выглядеть примерно следующим образом:

`\SVN $Date: 2006-11-25 21:02:20 +0600 $`

`\SVN $Rev: 265 $`

Документ обновлён `\SVNDate\ \SVNTime`

Текущая версия `\SVNRev`

Документ обновлён 25 ноября 2006 г. 21:02:20
---

Текущая версия 265
--------------------

Схожую функциональность предоставляет пакет **svninfo**.

## Вёрстка I

Хороший набор — это плотный набор, «дырявый» же набор плохо читается, так как дыры нарушают связанность строки и тем самым затрудняют восприятие мысли.

---

Ян Чихольд.

Волшебных текстовых процессоров не существует. Телепатией программы пока не обладают. Они, естественно, делают то, что им сказано, но вкус и чувство прекрасного у них полностью отсутствует. В конце занимательного приключения по созданию текстов частенько приходится брать управление в свои руки, дабы навести лоск на почти готовое произведение.

Вёрстка — составление страниц (полос) газеты, журнала, книги определённого размера из набранных строк, заголовков, иллюстраций и т. п. в соответствии с разметкой или макетом. В этой главе разберёмся с тем как задавать размеры, что такое макет полосы набора и как «удерживать» текст в рамках дозволенного.

### 6.1. Определённые «размеры» и переменные «длины»

Л<sup>A</sup>T<sub>E</sub>X поддерживает переменные типа «длина» для определения расстояния. Например, ранее уже упоминалась команда `\textwidth` — это переменная, хранящее значение длины, равной ширине текста.

Для создания переменной типа длина следует воспользоваться командой `\newlength`. В качестве обязательного параметра передаётся имя переменной. При создании переменной присваивается нулевая длина, так что следующим шагом необходимо приравнять её чему-то:

```
\newlength{\MyLen}
\setlength{\MyLen}{1cm plus 2.5fill minus 5mm}
\addtolength{\MyLen}{5em}
Длина \linline!\MyLen! равна \the\MyLen.
```

Длина `\MyLen` равна 82.89214pt  
plus 2.5fill minus 14.22636pt.

Длина в  $\LaTeX$  это не просто какой-то определённый размер — это более сложная структура с указанием границ возможного сжатия и растяжения. Границы растяжения определяются с помощью инструкции `plus`, а сжатия — `minus`. При формировании абзацев  $\TeX$  использует эту информацию для максимально «красивого» заполнения.

Команда `\setlength` эквивалентна оператору присваивания. В свою очередь команда `\addtolength` позволяет увеличить переменную на указанную величину, которая может быть отрицательной. Макрос `\the` позволяет «развернуть» переменную длины для вывода на печать.

$\LaTeX$  «говорит» в терминах англо-американской системы мер. Эта система отживает своё, но её наследие будет ещё долго проявляться и портить жизнь современному «метрическому» миру. Для определённости следует знать, что один дюйм (in) равен 2.54 сантиметра и в нём умещается 72.27 пунктов ( $1\text{ pt} \approx 0.35\text{ mm}$ ). Метрические величины представлены привычными сантиметрами (cm) и миллиметрами (mm). Кроме упомянутых  $\LaTeX$  умеет оперировать с размерами в больших пунктах (bp), пунктах Дидо (dd), пиках (pc) и цецero (cc) — традиционные единицы измерения, используемые в типографиях. Минимальной ненулевой единицей длины в  $\LaTeX$  является приведённый пункт (sp), который составляет  $1/65536$  от одного пункта.

Кроме определённых единиц измерения длины можно задавать так же и в относительных: `1ex` соответствует высоте строчной латинской буквы x, а `1em` — ширине прописной латинской буквы M. Эти величины меняются вместе со сменой шрифта, что позволяет задавать автоматически масштабирующиеся горизонтальные промежутки не привязанные к конкретному размеру и типу шрифта. Например, широкий пробел, задаваемый с помощью команды `\quad`, определяется как `\hspace{1em}`.

```
\setlength{\MyLen}{1ex}
Высота x равна \the\MyLen\par
\Large \setlength{\MyLen}{1ex}
Высота x равна \the\MyLen
```

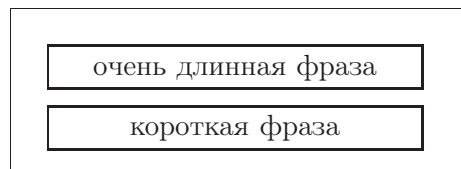
Высота x равна 4.71341pt

Высота x равна 7.43707pt

Интересной инструкцией является длина `fill` — это *бесконечность*.  $\TeX$  поддерживает операции с бесконечностями, причём оперирует тремя видами бесконечностей: `fil`, `fill` и `filll`, где `fil`  $\ll$  `fill`  $\ll$  `filll`. С помощью этих сущностей производится центрирование боксов и более сложные выравнивания.

Если хочется узнать ширину текста, то можно воспользоваться командой `\settowidth`:

```
\settowidth{\MyLen}{очень длинная фраза}
\addtolength{\MyLen}{1em}
\centering
\framebox[1.2\MyLen]{очень длинная фраза}\par
\framebox[1.2\MyLen]{короткая фраза}
```



Аналогично команда `\settoheight` позволяет выяснить высоту текста над базовой линией, а `\settodepth` — глубину под базовой линией. При использовании длины можно добавить перед ней множитель.

А теперь немного «магии» из английского FAQ по  $\text{\LaTeX}$ :

```
\makeatletter
\newcommand{\maxwidth}{%
\ifdim \Gin@nat@width > \linewidth
\linewidth
\else
\Gin@nat@width
\fi
}
\makeatother
```

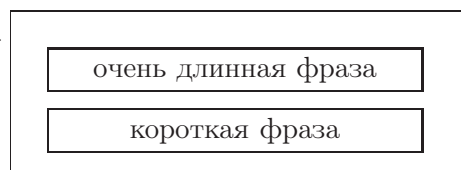
Эта конструкция определяет переменную длины `\maxwidth` таким образом, что при вставке картинки:

```
\includegraphics [ width=\maxwidth ] { «картинка» }
```

ширина картинки становится равной минимальной из двух возможных значений: «естественной» ширине картинки (размер в `BoundingBox`) или ширине строки. Это позволяет вывести картинку в натуральную величину при условии, что она не вылезает за рамки дозволенного и записывать её в эти рамки, коли она за них вылезает.

**calc** В дополнение к стандартным возможностям пакет **calc** расширяет базовые операции с длинами. Фактически **calc** вводит арифметические операции в привычной со школы инфиксной записи.

```
\setlength{\MyLen}{
(1em+\widthof{очень длинная фраза})*\real{1.2}}
\centering
\framebox[\MyLen]{очень длинная фраза}\par
\framebox[\MyLen]{короткая фраза}
```



При загрузке **calc** `\setlength` и `\addtolength` переопределяются так, что в качестве аргумента после этого можно передавать арифметические выражения. Кроме арифметики в **calc** определяются макросы `\widthof{текст}`, `\heightof{текст}` и `\depthof{текст}` — ширина, высота и глубина текста.



Рис. 6.1. Определение ширины (width), высоты (height) и глубины (depth).

При умножении длины на число длина должна стоять до числа ( $4\text{mm} * 2$  — верно, а  $2 * 4\text{mm}$  — не верно). Делить и умножать можно только на целые числа. Действительные числа вводятся с помощью уже использованного в примере макроса `\real` и отношения длин, вычисляемого с помощью команды:

```
\ratio {«длина»}{«длина»}
```

Подробное описание пакета можно найти в документации `calc.pdf` из коллекции `tools`.

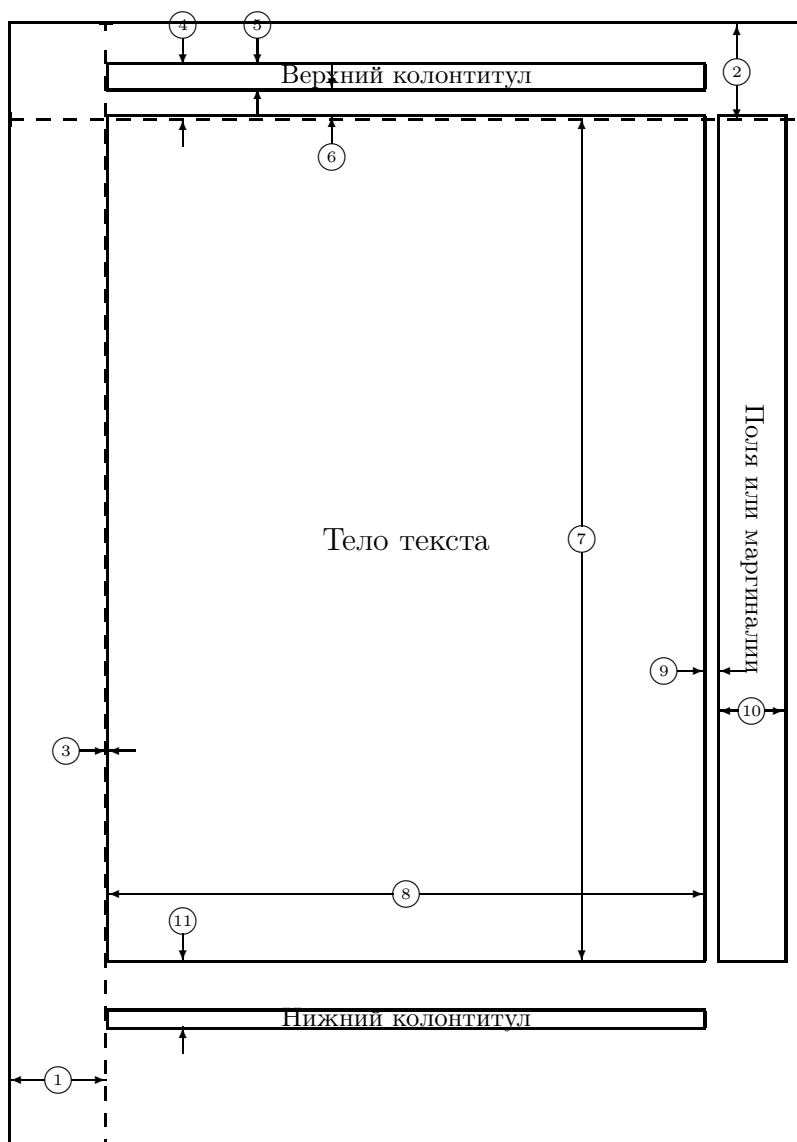
## 6.2. Скелет страницы

На рис. 6.2 приведён результат выполнения команды `\layout` из одноимённого пакета. Основное место на странице занимает текст — тело текста. Справа и слева от текста расположены поля. Поля обычно остаются пустыми, но иногда они используются для заметок (маргиналий или фонариков). В верхней и нижней части страницы расположены, соответственно, верхний и нижний колонтитулы. Колонтитул представляет собой справочную строку, помогающую ориентироваться в структуре текста.

Совокупность размеров и расположений указанных полей, а так же вид и содержание колонтитулов называется *макетом полосы набора*. На рисунке пунктирной линией изображены *поля драйвера* (1 и 2) относительно которых выстраиваются все остальные поля. По договорённости отступы до полей драйвера равны одному дюйму. Переопределив `\hoffset` и `\voffset` (по умолчанию они равны нулю), можно легко сдвинуть полосу набора целиком по горизонтали и вертикали, соответственно.

Ниже перечислены параметры, которые управляют макетом полосой набора:

- Тело теста характеризуется высотой `\texthight` (7) и шириной `\textwidth` (8). При многоколоночной вёрстке ширина колонки равна `\columnwidth`. Переменная `\linewidth` принимает значение равное длине строки текущего текста.
- `\oddsidemargin` (3) добавляется слева в случае односторонней печати. При двухсторонней печати полосы набора для чётных и нечётных страниц различаются. В этом случае для нечётных слева опять же добавляется `\oddsidemargin`, а для чётных `\evensidemargin`.



1	один дюйм + <code>\hoffset</code>	2	один дюйм + <code>\voffset</code>
3	<code>\oddsidemargin = 2pt</code>	4	<code>\topmargin = -41pt</code>
5	<code>\headheight = 18pt</code>	6	<code>\headsep = 21pt</code>
7	<code>\textheight = 635pt</code>	8	<code>\textwidth = 448pt</code>
9	<code>\marginparsep = 12pt</code>	10	<code>\marginparwidth = 49pt</code>
11	<code>\footskip = 50pt</code>		<code>\marginparpush = 6pt</code> (not shown)
	<code>\hoffset = 0pt</code>		<code>\voffset = 0pt</code>
	<code>\paperwidth = 597pt</code>		<code>\paperheight = 845pt</code>

Рис. 6.2. Макет полосы набора класса `scrartcl` (опция `a4paper`). Результат выполнения команды `\layout` из пакета `layout`.



- Верхний колонтитул располагается на расстоянии `\topmargin` (4) от поля драйвера, имеет высоту `\headheight` (5), а тело текста отступает от колонтитула на расстояние `\headsep` (6).
- `\footskip` позиционирует базовую линию нижнего колонтитула относительно последней строки текста.
- Поля для заметок имеют ширину `\marginparwidth` (10) и отступают от тела текста на расстояние `\marginparsep` (9). Ещё одна опция управляет минимальным расстоянием между заметками: `\marginparpush`.

**Выбор размера бумаги** Физический размер бумаги описывается параметрами `\paperwidth` и `\paperheight`. Стандартные базовые классы L<sup>A</sup>T<sub>E</sub>X (`article`, `book`, `report` и `letter`) по умолчанию предполагают, что для печати используется бумага формата `letter`. Очевидно это умолчание не годится для России, где стандартом является формат A4 (210×297 мм). Обычно, установить правильный формат можно с помощью передачи параметра `a4paper` при выборе класса документа:

```
\documentclass[a4paper,12pt,oneside]{scrbook}
```

Для создания небольших брошюрок со страницей размера A5 (половина размера A4) используется опция `a5paper`.

**Ориентация** Для портретной и альбомной ориентации вообще-то требуют разные макеты полосы набора. Альбомная ориентация может использоваться по умолчанию в некоторых специализированных классах (например, `slides`).

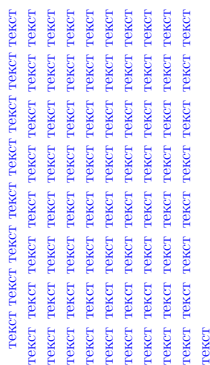


Рис. 6.3. `landscape` в действии.

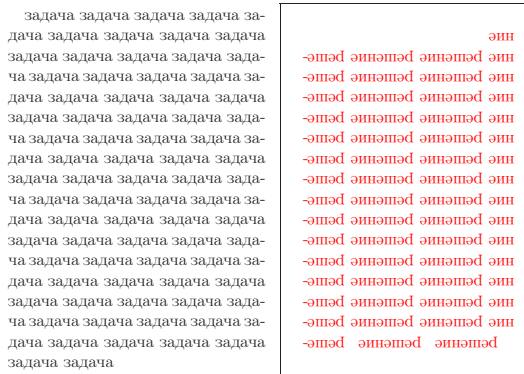


Рис. 6.4. `rotpages` в действии.

Если повернуть требуется только текст без изменения полей колонтитулов, то проще всего воспользоваться окружением `landscape` из пакета `landscape`. Всё, что находится внутри этого окружения поворачивается на 90 градусов против часовой стрелки. При использовании `pdflatex` для изменения ориентации не только текста, но и страницы (дабы не наклонять при чтении текста голову на бок) при загрузке

пакета следует передать ему опцию `pdftex` или воспользоваться пакетом-надстройкой `pdfscape`.

Лучше не менять параметры макета во время набора, но если очень хочется, то можно воспользоваться стандартным пакетом `portland`. `portland` позволяет на лету менять портретную ориентацию на альбомную и обратно, переопределяя соответствующие управляющие макетом переменные. Команды `\portrait` и `\landscape` работают как переключатели между этими режимами. В пакете определены и одноимённые с переключателями окружения. Это довольно низкоуровневые пакет и пользоваться им надо аккуратно.

Если по какой-то причине потребуется развернуть текст на 180 градусов<sup>1</sup> можно воспользоваться пакетом `rotpages`. В пакете определены два переключателя режима `\rotboxpages` и `\endrotboxpages`, которые указывают границы повернутого текста. Пакет умеет разворачивать не только страницы, но и колонки. Подробности в документации `rotpages-doc.pdf`

### 6.3. Меняем макет

Причина по которой не искушённый Т<sub>Е</sub>Xник начинает менять макет полосы набора обычно заключается в том, что он использует стандартные классы Л<sup>A</sup>T<sub>E</sub>X — один из четвёрки `article`, `book`, `report` или `letter`. Это очень древние классы и на них много чего «завязано», поэтому они прибывают в абсолютно замороженном состоянии. Поэтому лучше для начала найти себе класс по душе. Например, `scrartcl`, `scrbook`, `scrreprt` и `scrletter` — классы из коллекции КОМА-Script, которые дублируя функциональность стандартной четвёрки, ориентируются на европейских пользователей и размер листа А4. А. И. Рожено в рамках коллекции `ncclatex` (N<sub>C</sub>C) предоставляет класс `ncc`, ориентированный на русских Т<sub>Е</sub>Xников.

#### 6.3.1. Двигаем размеры

Выбор, естественно, не ограничивается упомянутыми выше классами — он огромен. Поэтому, прежде чем что-то изменять следует внимательно изучить уже имеющиеся решения. И даже если вы знаете что делаете, лучше не переопределять в ручную переменные управляющие размерами макета полосы набора. Правильным решением будет воспользоваться одним из уже имеющих специализированных пакетов, например: `geometry` или `vmargin`.

- Макет лучше не менять.

`geometry` Пакет `geometry` меняет размеры прямо в процессе загрузки стилевого файла, например так:

```
\usepackage [ height =25cm , a4paper , hmargin = {3cm , 2cm } ] { geometry }
```

<sup>1</sup>Например, для написания задачника с решениями, чтобы усложнить обучающемуся чтение решения сразу после прочтения задачи.

Разбор параметров выполняется с помощью пакета **keyval**, который уже упоминался в главе 4 (Графика). В качестве параметров можно передавать выражения, если загружен пакет **calc**.

Список воспринимаемых пакетом опций очень обширен. В документации к пакету **geometry.pdf** подробное описание всех имеющихся опций занимает свыше пять страниц текста. Использование этого пакета для изменения размеров полей является предпочтительнее, чем изменяя размеров напрямую. Обилие принимаемых параметров позволят выразить любую идею по формату полосы набора в наиболее естественной форме, не ошибившись при расчётах. Подробное описание пакета к сожалению выходит за все разумные рамки на объём статьи. К счастью документация очень хорошая и чрезвычайно подробная.

**vmargin** Пакет **vmargin** управляет размерами макета через выставку полей. Перед выставкой полей следует задать формат листа:

```
\setpapersize [«ориентация»] {«формат листа»}
```

В качестве обязательного параметра команды можно указать один из следующих форматов: A0, A1, ..., A9, B0, B1, ..., B9, C0, ..., C9, USletter, USlegal и USexecutive. Необязательный параметр может принимать значения **landscape** (альбомная ориентация) или **portrait** (портретная ориентация по умолчанию). Для не стандартных форматов листа можно задать размеры с помощью ключевого слова **custom** команды:

```
\setpapersize {custom} {«ширина»} {«высота»}
```

После того как удалось определиться с размером страницы можно задавать размеры для полосы набора с помощью одной из следующих команд:

```
%Полоса набора с колонтитулами
\setmargins {«ширина поля слева»} {«высота поля сверху»} %
           {«ширина текста»} {«высота текста»} %
           {«высота»} {«отступ»} %верхний колонтитул
           {«высота»} {«отступ»} %нижний колонтитул
%Полоса набора без колонтитулов
\setmargnohf {«ширина поля слева»} {«высота поля сверху»} %
            {«ширина текста»} {«высота текста»} %
%Установка размеров без изменения колонтитулов
\setmarg {«ширина поля слева»} {«высота поля сверху»} %
        {«ширина текста»} {«высота текста»} %
```

Кроме перечисленных в пакете **vmargin** определены их аналоги: `\setmarginrb`, `\setmargnohfrb`, `\setmarginrb`. Отличие этих команд от вышеупомянутых заключается что в качестве параметров вместо ширины и высоты тела текста им передаются ширина правого поля и высота нижнего поля. Подробно пакет описан в документации **vmargin.pdf**.

### 6.3.2. Стили страницы

Полоса набора это не только размеры — это ещё и наполнение колонтитулов. Обычно наполнение колонтитулов определяется в классе документа. Лучше без необходимости ничего не менять.

Простейший способ изменить стиль страницы, это воспользоваться командой:

```
\pagestyle {«стиль страницы»}
```

Если стиль надо переопределить только для текущей страницы, то следует воспользоваться командой `\thispagestyle{стиль страницы}`.

Есть три стандартно определённых стиля:

**empty** Страница выводится без каких-либо колонтитулов — только текст.

**plain** Выводится только номер страницы в нижнем колонтитуле.

**headings** В верхнем колонтитуле выводится номер страницы и информация, определяемая классом документа.

Если и это наполнение не устраивает, то можно определить свой собственный стиль. Пакет **fancyhdr** специализируется как раз на этом.

**fancyhdr** Чтобы воспользоваться возможностями пакета необходимо загрузить стилевой файл и с помощью `\pagestyle` выбрать стиль

А.В.Тор	— 112 —	Статья
Статья А.В.Тор		
текст текст		

Это просто демонстрация — xii — возможностей **fancyheadings**

Рис. 6.5. Заполняем колонтитулы с помощью **fancyhdr**.

Пакет **fancyhdr** позволяет управлять содержимым колонтитула. Возможности: верхний и нижний колонтитул разбивается на три независимых части, многострочные колонтитулы, колонтитулы, вылезавшие по ширине за `\textwidth`, декоративные линейки, разные колонтитулы для чётных и нечётных страниц, отдельные колонтитулы для специальных полос (начала глав, страницы, отведённые под плавающие объекты). Простейший пример для определения колонтитулов с помощью **fancyhdr**:

```

\usepackage{fancyhdr}
...
\begin{document}
\pagestyle{empty} %очищаем стиль страницы
\pagestyle{fancy} %включаем пользовательский стиль
\lhead{A.B.Top} %верхний колонтитул слева
\chead{---~\arabic{page}~---} % там же по центру
\rhead{Статья} %верхний колонтитул справа
% аналогично для оформления нижнего колонтитула
\cfoot{---~\roman{page}~---}
\lfoot{\hspace{0.7cm}Это просто демонстрация}
\rfoot{возможностей \textbf{fancyheadings}}

```

Переменная `page` содержит в себе номер страницы. Подробности о том что можно сотворить с колонтитулами следует искать в документации к пакету `fancyhdr.pdf`.

## 6.4. Причёсываем текст

После того как границы определены посмотрим что можно сделать для красивого размещения текста на странице. Часть забот по следованию правилам вёрстки  $\LaTeX$  берёт на себя. Вам, например, не требуется следить за единообразием оформления полос. Но есть дефекты которые компьютеру заметить не под силу, например, так называемые коридоры.

«Коридоры» графически разделяют текст абзаца или полосы на некоторое подобие неожиданных колонок, а они могут осложнять восприятие текста, из-за чего «коридоры» надо устранять как дефект набора.

Рис. 6.6. Дефект набора (коридор). Пример из [12].

**Строка** Когда строка с точки зрения  $\TeX$  становится слишком разряженной (Underfull) или слишком сжатой (Overfull), то в log-файле появляются предупреждения, начинающиеся со слов вида:

```

Overfull \hbox (26.1765pt too wide) in paragraph at lines 347--356
[[[]\T2A/cmr/m/n/12 Список вос-при-ни-ма-е-мых па-ке-том оп-ций
очень об-ши-рен. В до-ку-мен-та-ции geometry.pdf

```

Если при выборе класса документа передать ему опцию `draft`:

```
\documentclass[ draft , a4paper , 12pt , oneside ]{ scrbook }
```

то такие проблемные места будут отмечаться прямо в тексте. Обычно, подобные

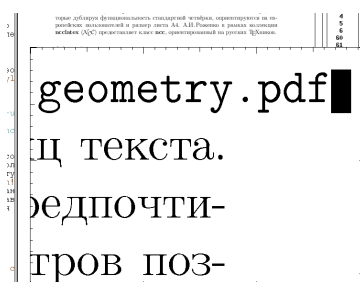


Рис. 6.7. Переполненная строка. Включена опция `draft`.

переполнения связаны с тем, что  $\text{\LaTeX}$  не знает как перенести какое-либо слово. В этом случае следует обучить его, что и где можно переносить, как это было показано в главе 2 (Базовые элементы). В крайнем случае можно насильно разорвать строку с помощью команды `\linebreak` или `\\`. В отличие от `\linebreak` команда `\\` не заставляет остаток строки выравниваться по правому полю.

Если можно редактировать текст, то для исправления дефектов набора лучше переделать предложение так, чтобы в новой инкарнации тест не создавал проблем для чтения.

**Горизонтальные пробелы** Расстояние между словами можно изменить с помощью горизонтальных промежутков. Горизонтальные промежутки создаются с помощью команды `\hspace`. В качестве параметра команде передаётся длина. Вариант команды `\hspace*` который отличается от основной тем, что создание пробела не игнорируется даже тогда, когда пробел приходится на начало или конец строки.

Существует несколько определённых по умолчанию горизонтальных пробелов:

`\quad` — горизонтальный промежуток шириной `1em`. Так же есть `\qquad` — удвоенный `\quad`, и `\endspace` — половина от `\quad`.

`\hfill` — бесконечный горизонтальный промежуток. Два `\hfill` подряд в два раза больше, чем один. Так же есть более «маленькая» бесконечность — `\hfil`.

`\hrulefill` — то же, что и `\hfill`, но заполненный промежуток подчеркивается.

Аналогично есть команда заполняющая всё точками — `\dotfill`.

**Страница** Проблемы могут возникнуть и при формировании страниц. В крайнем случае можно воспользоваться командами принудительного завершения страницы `\pagebreak` или `\newpage`. Отличие первой команды от второй в том, что после формирования страницы полоса выравнивается по нижней кромке — это может привести к неоправданному растяжению страницы. Если проблему можно решить путём увеличения/уменьшения страницы на одну-две строки, то лучше воспользоваться следующими макросами:

```
\newcommand{\longpage}{\enlargethispage{\baselineskip}}
\newcommand{\shortpage}{\enlargethispage{-\baselineskip}}
```

Команда `\longpage` увеличивает тело текста текущей страницы на одну строку, а `\shortpage`, соответственно, уменьшает. Длина `\baselineskip` служит для определения *интерлиньяжа* — междустрочного пробела.

**Висячая строка** Одним из самых неприятных дефектов набора является «висячая строка». Висячая строка — это концевая строка абзаца, стоящая первой на странице, или начальная строка абзаца, стоящая на странице последней. Этих артефактов следует всячески избегать. Для подавления этого эффекта в заголовке документа следует определить две переменные:

```
%подавление висячих строк.
\clubpenalty=10000
\widowpenalty=10000
```

**Вертикальные просветы** По аналогии с командой `\hspace{длина}` вертикальные промежутки организуются с помощью команды `\vspace{длина}`. Модификация команды `\vspace*{длина}` создаёт вертикальный просвет, которые не игнорируется даже если просвет попадает на начало или конец страницы.

Вертикальные просветы так же имеют свои умолчания:

`\bigskip` — вертикальный промежуток равный примерно `\baselineskip`. Так же имеются `\medskip` — половина от `\bigskip`, и `\smallskip` — четверть от `\bigskip`.  
`\vfill` — бесконечный вертикальный промежуток. Два `\vfill` подряд в два раза больше, чем один. Так же есть более «маленькая» бесконечность — `\vfil`.

**Печать через две строки** До сих пор временами встречаются требования вида: «Предоставить диплом, напечатанный через две строки» — пережиток эпохи печатных машинок. Для решение этой проблемы лучше всего воспользоваться пакетом `setspace`. В пакете определена команда `\doublespacing`, которая выполняет искомое действие. Так же в `setspace` определены макросы `\onehalfspacing` и `\singlespacing` — печать через полторы и одну строку, соответственно. Для вертикальной разрядки небольшого фрагмента текста лучше воспользоваться одноимёнными окружениями или окружением `spacing`:

```
\begin{spacing}{2.5}
  <<Этот текст, напечатан с
  интервалом в две с
  половиной строки>>.
\end{spacing}
```

```
«Этот текст, напечатан с интервалом в две
с половиной строки».
```

В качестве основного параметра окружению **spacing** передаётся число строк через которое следует печатать текст.

## 6.5. Послесловие

В этом тексте присутствует далеко не вся информация необходимая при вёрстке текста. Несмотря на то, что L<sup>A</sup>T<sub>E</sub>X позволяет верстать книги любителям без помощи профессионалов, но лучше при любой возможности спрашивать у этих профессионалов совета. Понимание *что, где и зачем* надо исправлять в случае L<sup>A</sup>T<sub>E</sub>X чрезвычайно важно, потому что *как это следует сделать*, как правило, и так очевидно.



# Путеводитель по классам L<sup>A</sup>T<sub>E</sub>X

Классы у людей определяются выбором оружия. . .

Википедия. Из описания игры Tremulous.

Класс документа — это первое, что требуется указать при наборе. В то же время *первое* вовсе не значит *важное*. Заключительный выбор класса почти всегда может повременить до окончания основного набора. С другой стороны выбирать всё равно придётся.

## 7.1. Зачем нужны эти классы?

Класс документа выбирается с помощью команды `documentclass`. Её нужно и можно выполнить ровно один раз в самом начале документа:

```
\documentclass[a4paper,12pt]{article}
```

В качестве обязательного аргумента указывается имя класса, которому через запятую передаются необязательные параметры. Класс определяется в файле с расширением `.cls`. В дистрибутиве L<sup>A</sup>T<sub>E</sub>X TeX Live 2005 присутствует 175 уникальных файлов с подобным расширением. Естественно, это далеко не все существующие на белом свете классы тем более, что никто не мешает создать свой личный класс. Но для начала лучше воспользоваться одним из имеющихся.

Класс определяет вид и структуру документа. Класс — это база, которую можно править с помощью подключаемых стилевых файлов. В классе задаётся геометрия страницы и определены команды секционирования. Класс сам по себе может быть как просто небольшой модификацией уже имеющегося класса, так и принципиально



Рис. 7.1. papertex

новой реализацией представления печатного или электронного  $\TeX$ нического слова. В качестве примера последнего можно привести молодой (2007 год) пока пакет **papertex** (рис. 7.1 — пример от автора класса Ignacio Llopis) который позволяет применять  $\LaTeX$  в деле вёрстки газет<sup>1</sup>, то есть для того, для чего  $\LaTeX$  в проекте вовсе не предназначался. Сам пакет можно найти на любом CTAN архиве в директории `{CTAN}/macros/latex/contrib/papertex/`.

## 7.2. Классовая база

Исторически сложилось, что  $\LaTeX$  начался с 6 классов: **article** (статья), **book** (книга), **report** (отчёт), **proc** (доклад), **letter** (письмо) и **slides** (слайды).

По идее статьи следовало набирать в **article**. В этом классе определены команды секционирования вплоть до `section` (раздел). Одним из желательных элементов оформления является предисловие (окружение `abstract`). В классе **book** присутствует расширенный набор команд секционирования в который добавлена команда `chapter` (глава). Так же в классе **book** присутствует базовый набор команд для оформления титульного листа, предисловия и оглавления. Класс для создания отчёта **report** является упрощённой версией класса **book**. Отчёты те же книги, только читают их по необходимости, а не по желанию. Класс **proc**, предназначенный для создания тезисов докладов, в свою очередь является модификацией класса **article**, причём основное отличие состоит в обязательной двухколоночной печати. Для написания писем был создан класс **letter** набор команд в этом классе существенно отличается от уже перечисленных, в частности для писем нет нужды в командах секционирования. Класс **slides** — простой и быстрый способ сделать презентацию. С помощью этого класса не удастся создать пёстрого фона и головокружительных эффектов смены слайдов, зато позволяет сосредоточиться на основном — на тексте.

Особняком от этих классов стоит класс **minimal**, который является болванкой для создания и тестирования новых классов и идей. В классе **minimal** не определено никаких специальных команд — там всё по минимуму. В качестве побочного эффекта документ, выбирающий этот класс транслируется  $\LaTeX$  значительно быстрее.

С этих классов всё начиналось, но не закончилось. Следует понимать, что базовые классы далеко не так хороши, как хотелось бы. После трансляции текста сразу возникает желание взять в руки «электронный надфиль» и пройтись по настройкам класса. Это цена за то, что эти классы являются базой. На них ссылаются и их модифицирует множество других классов и пакетов, поэтому их развитие было заморожено. Для начального набора сгодится и это, но для конечной вёрстки лучше подобрать что-то более подходящее или придётся серьёзно модифицировать значения по умолчанию.

Часто набор необязательных параметров для стандартных классах используется и в других классах, например, в целях совместимости. Некоторые из полезных опций перечислены ниже:

---

<sup>1</sup>На текущий момент этот класс не годится для вёрстки чего-нибудь более серьёзного чем школьная газета, но ведь надо начинать с простого.

**10pt|11pt|12pt** — установка базового размера шрифта. Обычно этих трёх значений хватает.

**a4paper** — установка размера листа бумаги. Следует использовать всегда, так как по умолчанию  $\LaTeX$  использует размер листа letter.

**draft** — режим черновой печати для «отлавливания» проблем вёрстки. В этом режиме не внедряются картинки (вместо них вставляются прямоугольники нужного размера) и отмечаются строчки, где алгоритм разбиения абзаца на строки даёт осечку.

**oneside|twoside** — форматирование документа для односторонней и двухсторонней печати, соответственно.

**twocolumn** — печать в две колонки.

## 7.3. Классификация

Число классов постоянно растёт, поэтому не следует думать, что всё исчерпываются классами перечисленными ниже.

### 7.3.1. Модификации и улучшения базы

Всем не нравятся стандартные классы и всякий старается их улучшить. Кто-то убирает какой-то конкретный недостаток, как это сделано в наборе классов `extsizes` (`extarticle`, `extbook`, `extletter`, `extproc`, `extreport`), которые отличаются от стандартных только возможностью указать базовый размер шрифта отличный от обычного 10-12pt<sup>2</sup>. Есть наборы классов, которые делались с какой-то определённой целью. Примером такого подхода являются классы от AMS (`amsart`, `amsbook`, `amsproc`), которые были предназначены для публикации в журналах Американского математического сообщества. Классы из набора `ntgclass`<sup>3</sup> представляют собой «героическую» попытку немецкоговорящих голландцев сделать то же, что и в стандарте, но существенно разными способами.

### КОМА-Script

В последнее время всё больше внимания обращает на себя набор классов КОМА-Script. В этот раз хорошо постарались немцы. Следует учитывать, что европейские традиции полиграфии (в основном французские), всё-таки к нам ближе, чем американские на которые традиционно ориентировалось  $\LaTeX$ -сообщество. Для статей предполагается использовать `scrartcl`, для книг `scrbook`, для писем `scrlttr2`, а для отчётов `scrreprt`.

---

<sup>2</sup>Кроме 10pt, 11pt и 12pt классы из набора `extsizes` поддерживают 8pt, 9pt, 14pt, 17pt и 20pt. Смена размера базового шрифта приводит к принципиально иному дизайну документа

<sup>3</sup>В набор `ntgclass` входят классы для набора статей (`artikel1`, `artikel2` и `artikel3`), для набора книг (`boek` и `boek3`), писем (`brief`) и отчётов (`rapport1` и `rapport3`).

В отличие от стандарта классы из KOMA-script позволяют использовать базовые размеры шрифта в 9pt, 14pt и 17pt. Огромные поля, имеющие место в стандартных классах, в классах KOMA-script значительно уменьшены. Претерпели изменения и другие элементы. Если оформление по умолчанию не кажется адекватными, то KOMA-script предоставляет обширный набор высокоуровневых настроек. Подробная документация на более чем двухстах страницах «The KOMA-Script bundle» (`scrguien.pdf`) позволяет подстроить все необходимые параметры.

## NC<sub>S</sub>

Ещё один вариант в качестве замены стандартным классам — это использование пакета *NC<sub>S</sub>*. Очень подробно об этом пакете написано в замечательной книге от создателя *NC<sub>S</sub>* Александра И. Роженко: «Искусство верстки<sup>4</sup> в L<sup>A</sup>T<sub>E</sub>X'e», 2005 (ISBN 5-901548-25-6).

Для использования следует загрузить класс `ncc` и передать ему желаемый стиль оформления в качестве параметра: `article` (статья — используется по умолчанию), `preprint` (препринт), `book` (монография) или `report` (отчёт). Дальнейшие подробности об использовании этого класса можно почерпнуть в краткой инструкции к пакету: `ncclatex.pdf`.

### 7.3.2. Поддерживаем стандарты

Стандарт подразумевает наличие подробного описания, которое и является его сущностью. То, что написано на бумаге в виде набора непротиворечивых правил, может быть переведено на язык машины. Далее можно забыть про эти правила, так как помнить все нюансы — работа *для* машины<sup>5</sup>.

Константин Корилов создал и активно поддерживает пакет `eskdx`, который представляет собой набор классов и стилей предназначенный для верстки документации в соответствии с требованиями «Единой системы конструкторской документации». Основу коллекции составляют три класса: `eskdtext` (для текстовой документации), `eskdbtab` (для чертежей и схем) и `eskdgraph` (для документов, разбитых на графы). Внятная документация на русском (`eskdx.pdf`) приятно дополняет картину. На CTAN пакет находится в директории `{CTAN}/macros/latex/contrib/eskdx/`, а домашняя страничка пакета находится здесь: <http://lostclus.linux.kiev.ua/eskd<sub>x</sub>/>.

`eskdx` относительно молодой пакет. Ранее аналогичная попытка была предпринята Вячеславом Фёдоровым, в результате которой на свет появился пакет `eskd`

СОДЕРЖАНИЕ	
<b>1</b>	<b>Общие сведения</b> <span style="float:right">4</span>
1.1	О коллекции <code>eskd<sub>x</sub></code> <span style="float:right">4</span>
1.2	Возможности коллекции <span style="float:right">4</span>
<b>2</b>	<b>Базовые принципы использования</b> <span style="float:right">5</span>
2.1	Пример простого документа <span style="float:right">5</span>
2.2	Описание классов <span style="float:right">6</span>
2.2.1	Общие описания всех классов <span style="float:right">6</span>
2.2.2	Описание класса <code>eskd<sub>text</sub></code> <span style="float:right">8</span>
2.2.3	Описание класса <code>eskd<sub>g</sub>raph</code> <span style="float:right">9</span>
2.2.4	Описание класса <code>eskd<sub>bt</sub>ab</code> <span style="float:right">10</span>
2.3	Информация о документе <span style="float:right">11</span>
2.4	Титульный лист <span style="float:right">13</span>
2.5	Заполнение граф основной надписи и дополнительных граф <span style="float:right">14</span>
2.6	Рубрикации <span style="float:right">17</span>
2.7	Пояснения, ссылки, вложения и формулы <span style="float:right">18</span>
2.8	Лист регистрации изменений <span style="float:right">18</span>
2.9	Чертежи и схемы <span style="float:right">18</span>
2.10	Спецификации <span style="float:right">19</span>
2.11	Спецификации прикладного метода <span style="float:right">19</span>
2.12	Лист утверждения <span style="float:right">19</span>
2.13	Количество рисунков, таблиц, приложений, и т.д. <span style="float:right">20</span>
<b>3</b>	<b>Тонкая настройка</b> <span style="float:right">21</span>
3.1	Управление стилями страниц <span style="float:right">21</span>
3.2	Настройка шрифтов <span style="float:right">22</span>
3.3	Настройка титульного листа <span style="float:right">23</span>
3.4	Управление заголовками рубрикации <span style="float:right">24</span>

Рис. 7.2. Страница документации к `eskdx`

<sup>4</sup>Да, да именно «верстки» — букву «ё» опять обидели.

<sup>5</sup>То есть тупая, нудная и не интересная.

(без «х» на конце). В отличие от класса Константина Корикина класс `eskd.cls` требует обязательной установки шрифтов из коллекции `psyr`<sup>6</sup>.

Стандарт для написания документов описывающих стандарты. Что может быть ещё более стандартным? Класс `isov2.cls` из пакета `iso` является стандартом для стандартов. Документация к пакету `isoman.pdf` подробно описывает все технические тонкости в деле подготовки документов по стандартам ISO. Аналогично для создания документации ISO 10303 есть свой пакет `iso10303`.

### 7.3.3. Пишем письма

Класс `letter` является стандартным для  $\LaTeX$  и как следствие никто им не пользуется. Часто стандарт для написания писем создаётся автором самостоятельно. Так, например, любит делать Кнут. Это очень неплохо работает в силу того, что структура письма не сильно сложна. Как следствие в  $\LaTeX$  имеется огромное число альтернатив для `letter`.

С точки зрения английского FAQ по  $\LaTeX$  (<http://www.tex.ac.uk/faq>) класс `newlrm` является наиболее продвинутым. `lrm` расшифровывается как `letter` (письмо), `fax` (факс) и `memoranda` (служебная записка). Документация представляет собой текстовый `README` и набор примеров использования.

Хорошо документированный класс `akletter` так же является хорошим шаблоном для старта. Документация `lettereng.pdf` кроме краткой инструкции так же включает и формальное описание структуры письма.

Упомянутый выше пакет КОМА-script предоставляет прекрасную замену стандартному классу `letter` в виде `scrlttr2`. Набор классов `ntgclass` так же предоставляет свой вариант в виде класса `brief`.

Кроме более-менее общих решений полно и частных. Например, для внутренней переписки университета города Падуя есть специальный пакет `cdpbundl`, содержащий целых три класса.

### 7.3.4. Верстаем книги

Написание книги это очень долгий процесс и первоначальную «набивку» текста можно начать со стандартного класса `book`. С другой стороны структура книги может быть очень сложной и правильный выбор базового класса позволит несколько облегчить процесс созидания.

В качестве улучшенного стандартного класса `book` можно использовать класс `octavo`. Класс `scrbook` из КОМА-script так же является хорошей альтернативой для `book`. Структура и основные команды копируют стандартный класс. Значения же параметров по умолчанию более адекватны для Европейской полиграфии.

---

<sup>6</sup>Шрифты из коллекции `psyr` авторами больше не поддерживаются и не развиваются. Основные проблемы этого пакета не технические, а лицензионные. В связи с чем этот пакет отсутствует в дистрибутивах  $\LaTeX$ . В дополнение к абсолютно не решаемым лицензионным там хватает и технических проблем. Последнюю версию этого пакета можно взять по адресу: <ftp://ftp.vsu.ru/pub/tex/font-packs/psyr/>.

Описание класса **memoir** (`memman.pdf`) представляет собой книгу о создании книги, превышающую по объёму *триста* страниц. Там есть всё, начиная от формальной структуры печатной книги, советов по оформлению электронных копий, заканчивая, собственно описанием класса. Все элементы структуры и управляющие размеры показаны в виде рисунков и схем. Класс не является надстройкой над чем-либо — это произведение искусства, созданное с нуля. Документацию следует пролистать хотя бы просто для ознакомления. Класс развивается до сих пор. Новые возможности описываются в дополнении к основной документации (`memmanadd.pdf`) и на текущий момент дополнение почти достигло объёма в сто страниц.

Если не требуется написать книгу, а нужно распечатать мегабайтный текст на дешёвеньком принтере в режиме экономии тонера, то для этого дела вполне может подойти класс **sfms** — простенько и строки через два интервала.

### 7.3.5. Создаём отчёты

Отчёт — не книга, но и здесь есть свои правила и структура. Для начала можно воспользоваться **scrrprt** из КОМА-script, как замена стандартному классу **report**.

Инженер-электронщик Eli Billaueг сделал ЛАТЭХ-класс для бумаг в HiTech-стиле и назвал его, соответственно, **hitec**. Простенько и со вкусом. Есть, естественно, и частные решения. Хочется заключить контракт с Американским правительством — стандартная форма 298, обеспечиваемая классом **sfms**, будет вполне кстати. Класс **manual** из пакета **nassflow** даст возможность пообщаться со структурой под названием «Center for the Automation of Weapon and Command Systems, Royal Netherlands Navy».

### 7.3.6. Делаем презентации

В начале предполагалось, что цель презентации в распространении нужной информации от одного человека ко многим. Поэтому во главу угла ставился текст, а «украшательства» сводились к простой рамке. Стандартный класс **seminar** и **sides** вполне годились для этого.

Но время суровых докладчиков прошло и «рюшечки» вышли на первый план. ЛАТЭХ может предоставить и «рюшечки», но лучше всё-таки помнить о смысле.

Класс **prosper** создан как улучшенный **seminar** и поддерживает не только оверлеи, гиперссылки и шаблоны оформления, но и «стандартный» набор динамических эффектов доступных через формат pdf<sup>7</sup>. Вспомогательный класс **ppr-prv** позволяет создать печатную версию электронных слайдов класса **prosper**.

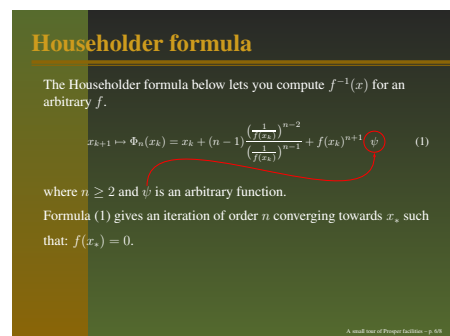


Рис. 7.3. **prosper** в действии

<sup>7</sup>Для просмотра динамических эффектов требуется Acrobat Reader, раскрытый на полный экран.



В пакет **texpower**, входит класс **powersem**, который по сути дела просто загружает **slides**, а всю работу по созданию презентации выполняет стиль **texpower**. Возможности этого пакета сравнимы с **prosper** — просто он немного другой.

В последнее время в деле создания презентации всё популярнее и популярнее становится относительно молодой, но довольно мощный класс **beamer**. Активная поддержка сообщества при создании этого пакета позволила автору **beamer** Тилу Тантау (Till Tantau) собрать в одном месте не мало тем для слайдов. Ключевой особенностью этих тем является разнообразие. Наличие широкого выбора стандартных тем позволяет быстро выбрать обрамление для презентации. Более чем 200 страниц документации ускоряет решение любой возникшей проблемы.

Даже если использовать  $\LaTeX$  в качестве WYSIWYM редактора, то и его вполне можно настроить для создания презентаций. Для этого следует воспользоваться ещё одним достаточно молодым, но уже вполне функциональным классом **powerdot**, в комплекте с которым идут настройки для  $\LaTeX$ .

Класс **talk**, в отличие от упомянутых выше пакетов, позволяет пользователю определить более одного стиля слайдов для презентации. Резкая смена стиля во время доклада — иногда нужно и такое.

Для создания настенного постера в первом приближении можно воспользоваться пакетом **aposter**, который позволяет работать с большими форматами бумаги. Канонического класса, который бы решал все проблемы при изготовлении постера в  $\LaTeX$  на текущий момент нет. Возможно, ближе всего к идеалу подошёл класс **sciposter** из одноимённого пакета.

### 7.3.7. Защищаем диссертации

Раньше были курсовые и дипломные работы, а теперь куда не плюнь везде диссертации. Каждый уважающий, не сильно уважающий и вообще не уважающий себя университет имеет свой уникальный стиль оформления диссертации. Если хочется написать свой класс, то в качестве отправной точки можно выбрать класс **ucthesis** от Калифорнийского университета (UC Berkeley).

Станислав Кручинин озадачился судьбами русскоговорящих диссертантов и создал класс **disser**. Пакет с одноимённым названием можно взять на CTAN в директории `{CTAN}/macros/latex/contrib/disser`. Следует учесть, что пользовательская документация на текущий момент отсутствует. С другой стороны диссертанты по идее люди не глупые и разберутся в имеющихся примерах.

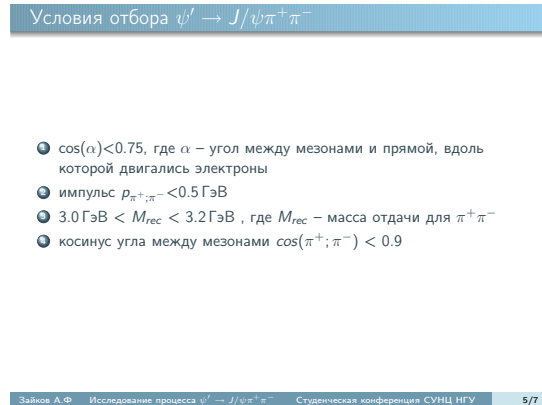


Рис. 7.4. **beamer** — справится даже школьник. Изготовлено учеником 11 класса А.Ф. Зайковым самостоятельно. На ошибки не фыркаем! Школьник всё-таки.

### 7.3.8. Организуем резюме

Написание резюме или curriculum vitae довольно популярный вид деятельности в современном мире. Для резюме нет общепринятого стандарта, но некоторые указания существуют.

Европейская комиссия рекомендует определённый формат для составления резюме и этот формат полностью реализуется с помощью класса `eurorescv`. Класс `vita` представляет из собой конструктор для создания резюме. Не смотря на отсутствие документации разобраться по имеющимся примерам для IT-специалиста и певца не составляет сложности. Класс `curve` напротив обладает качественной документацией. Механизм рубрик позволяет классу `curve` поддерживать несколько резюме разной направленности и легко переключаться между ними. Современный класс `morderncv` рекомендуется как гибкое и простое средство создания резюме как современного вида, так и классической формы. Пакет можно взять на CTAN в директории `{CTAN}/macros/latex/contrib/moderncv`.

Следует отметить, что классы определяют многое, но далеко не всё. Поэтому после выбора класса можно подключить стили, которые серьёзно меняют внешний вид документа. Стиль `currvita` позволяет создавать резюме в окружении стандартных классов.

### 7.3.9. Журнальные и конференционные классы

Каждый серьёзный научный журнал и крупная конференция имеет свой L<sup>A</sup>T<sub>E</sub>X-класс. Обычно, этот класс лежит где-то на официальном сайте, например, журналы, издаваемые издательством МАИК «Наука/Интерпериодика», должны следовать правилам выложенным здесь: <http://www.maik.ru/pub/tex/>. Но довольно много журнальных классов можно найти в стандартном дистрибутиве L<sup>A</sup>T<sub>E</sub>X. Например, класс `asaetr` используется в American Society for Agricultural Engineers (ASAE). Maple Technical Newsletter можно создавать с помощью класса `mtn`. Классом `jpsj2` отметились японцы. Из русскоязычных журналов замечен только «Сибирский журнал вычислительной математики» — класс `sibjnm`.

Следует отметить класс `elsart` обязательный для подготовке журнальных публикаций в издательстве Elsevier. Класс `nature` позволит подготовить pdf-файл для журнала Nature. Часто при создании публикаций для журналов или конференций используются небольшие модификации класса `revtex4`.

Для объединения разных документов в один, например, для оформления трудов конференции, может пригодиться класс `combine`.



Рис. 7.5. Класс `moderncv`



### 7.3.10. Всякая всячина

Далеко не все классы подчиняются уже перечисленной классификации. За рамки темы вышли классы для составления календарей, обложек для CD (**cd**), вопросников (**qcm**), объявлений о занятиях (**assignment**), концертных программ (**ConcProg**), программ курса (**courseoutline** и **coursepaper**), рабочего журнала для биологов (**labbook**), пьес (**stage**), тибетских карточек (**pecha**), карточек для запоминания иностранных слов (**flashcards**), и для многого другого. Старые и давно не поддерживаемые классы могут не собраться в новом окружении, но исходники доступны и всегда можно довести их до необходимой кондиции.

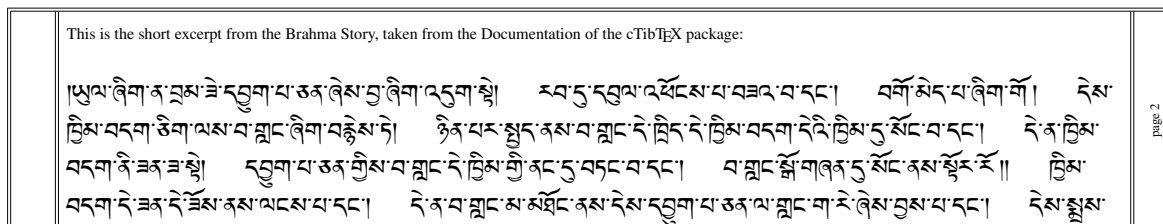


Рис. 7.6. Что-то определённо тибетское — класс **pecha**.

## Заключение

Классов много — места в статье мало. Малая толика из упомянутых классов будет рассмотрена в последующих статьях цикла. Это не является проблемой, так как в чём-чём, а в отсутствии документации к пакетам, L<sup>A</sup>T<sub>E</sub>X обвинить невозможно, ну, почти.

**Врезка: CTAN**

Q. Что такое CTAN?

A. CTAN — это международный файловый архив. Аббревиатура CTAN расшифровывается как Comprehensive TeX Archive Network. Цель CTAN — собрать всё, что относится к  $\TeX$  и его производным в одном месте. Основные сайты, составляющие CTAN — это:

- <ftp://ftp.dante.de/tex-archive/>
- <ftp://ftp.tex.ac.uk/tex-archive/>
- <ftp://ctan.tug.org/tex-archive/>

Зеркала CTAN разбросаны по всему свету, в частности, в России находятся:

- <ftp://ftp.chg.ru/pub/TeX/CTAN/>
- <ftp://ftp.nsu.ru/mirrors/ftp.dante.de/tex-archive/>

Пакеты, это как мясо, жилы и даже жир, которыми обрастает скелет в виде стабильного ядра  $\TeX$ . Такая модель разработки (ядро плюс множество расширений) довольно успешна. Существуют достаточно много развитых сообществ, которые исповедуют тот же путь. Взять тот же CPAN<sup>8</sup> для perl CRAN<sup>9</sup> для R и PyPI<sup>10</sup> для python — один в один модель CTAN, тем более, что именно с него она и копировалась.

---

<sup>8</sup>Comprehensive Perl Archive Network.

<sup>9</sup>Comprehensive R Archive Network

<sup>10</sup>Python Package Index

# Делаем презентации I

Существует три разновидности людей: те, кто видит;  
те, кто видит, когда им показывают;  
и те, кто не видит.

Леонардо да Винчи

Хочется показать свою крутизну? Подкупи слушателей. Хочется донести свою идею? Сделай нормальную презентацию.

При этом вовсе не нужно аляповатого фона, мультипликации при смене слайдов, но необходим разборчивый текст и картинки к месту. Вполне можно ограничиться «прозрачками» и стандартным «оверхэдом». Если слайд требует от аудитории размышления, то на него следует не пожалеть как минимум пяти минут. В противном случае все ваши усилия напрасны.

## 8.1. slides

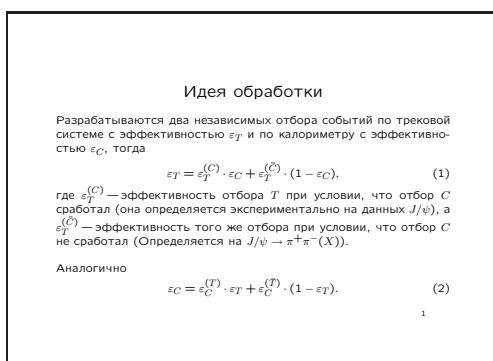


Рис. 8.1. **slides** — это просто  
слайды по умолчанию. Слайды создаются с помощью окружения **slide**. Всё.

Динозавр среди классов  $\text{\LaTeX}$  специализирующихся на презентациях. Идея очень простая. В качестве класса документа выбирается **slides**. В результате базовый размер шрифта автоматически увеличивается. Это позволяет прочесть стандартный текст на экране и избавиться от одного из смертных грехов докладчика — желания уместить слишком много информации на одной страничке. Здесь по умолчанию ничего с этим не выйдет. Опция класса **landscape** позволяет выбирать альбомную ориентацию для страницы.

```

\documentclass[a4paper,landscape]{slides}
...
\begin{document}
\begin{slide}
  \begin{center}
    \Large Идея обработки
  \end{center}
...
\end{slide}
\end{document}

```

Класс **seminar** похож на **slides** и лишь чуть-чуть более современен (1993 г.), но не в пример лучше документирован (файл **sem-user.pdf**) и кроме стандартного окружения **slide** имеет простейший набор команд для создания рамок.

Если надо что-то сделать по быстрому из уже готового текста с целью просто продемонстрировать какую-то идею, то **slide** и **seminar** вполне для этого подойдут.

## 8.2. Немного о PDF

PDF — Portable Document Format открытый платформеннонезависимый формат для описания документов созданный фирмой Adobe Systems в 1993 году. В январе 2007 года началась процедура стандартизации PDF, как стандарта ISO. В 2006 году была опубликована версия стандарта под номером 1.7. Файл в PDF-формате может представлять собой комбинацию векторной графики, текста и растровых изображений (фотографий, снимков экрана и тому подобное). В стандарте PDF предусмотрена возможность создания гиперссылок, заполняемых форм и интерактивных вставок на JavaScript. Начиная с версии 1.6 декларируется возможность описания 3D интерактивных документов — что бы это ни значило звучит заманчиво, но к сожалению пока рано использовать эти возможности.

С точки зрения формата для представления презентации PDF удовлетворяет необходимым условиям, таким как:

- Простота создания. Это сила качественных открытых форматов — рано или поздно их начинают поддерживать все кому не лень.
- Переносимость. Везде найдётся программа просмотра PDF.
- Элементы интерактивности. Документ может представлять собой не только плоскую последовательность страниц.

### 8.2.1. Простота создания

Допустим, что тем или иным способом был получен PostScript-файл презентации. Из него с помощью **ghostscript**, точнее с помощью скрипта **ps2pdf** (man ps2pdf) можно получить нормальный PDF:

```
> ps2pdf «файл.ps» «файл.pdf»
```

Получить PDF можно и напрямую из исходников с помощью программы **pdflatex**. Эта программа отличается от  $\text{\LaTeX}$  в основном только тем, что в качестве выходного формата получается PDF. При использовании **pdflatex** следует учитывать, что графика должны быть либо в виде pdf (вектор), либо png/jpeg (растр). **pdflatex** не умеет обрабатывать eps-файлы, за исключением картинок созданных с помощью MetaPost.

В PDF можно внедрять векторные шрифты Type1. Это позволяет отображать готовые документы независимо от набора имеющихся шрифтов. Отображение на экране особенно при низких разрешениях зависит исключительно от качества внедрённых шрифтов. Парадокс качества: чем хуже разрешение, тем больший объём работы надо проделать с векторным шрифтом, чтобы он выглядел приемлемо. К счастью в случае презентаций это не является проблемой, так для читабельности на большом экране размер шрифта нужно значительно увеличить. Это эффективно увеличивает разрешение до сравнимого с разрешением лазерного принтера под который и оптимизированы наиболее популярные векторные шрифты Computer Modern (пакет cm-super).

Ни в коем случае для отображения на экране не стоит использовать растровые шрифты в формате Type3. Шрифты cm-super (в  $\text{\TeX}$ Live есть заведомо) обязательно должны быть установлены.

Если вдруг по какой-то причине pdf нужно преобразовать в PostScript, то лучше воспользоваться утилитой **pdftops** из пакета **xpdf**:

```
> pdftops [-eps] «pdf-файл»
```

Если необходимо получить картинку в формате EPS, то следует использовать ключик `-eps`.

### 8.2.2. Переносимость

Везде есть Adobe Reader и Ghostscript. Если этого где-то нет, то оно легко может там появиться. Adobe Reader предоставляется всем желающим самой Adobe Systems. Как следствие в смысле поддержки всех расширений формата PDF эта программа условно «впереди планеты всей». Поэтому презентацию, скорее всего, придётся показывать с помощью неё.

Одной из раздражающих особенностей Adobe Reader, мешающей использовать эту программу при работе над документом, является то, что в нём отсутствует возможность перезагружать изменённый документ. Эту проблему можно частично решить с помощью сторонних программ **pdfopen** и **pdfclose** (заведомо присутствуют в дистрибутиве  $\text{\TeX}$ Live):

```
> pdfclose —file «файл.pdf»
# обновляем «файл.pdf»
> pdfopen —file «файл.pdf»
```

Ghostscript и программа просмотра с его использованием так же есть везде. Ghostscript отображает PDF как обычный «плоский» документ, то есть об интерактивных «эффектах» можно забыть. Зато проблем с обновлением текста нет: нажал «.» (точку) и картинка обновилась.

**xpdf** (<http://www.foolabs.com/xpdf/>) для просмотра PDF доступен только для систем где есть X Window. Начиная с версии 3.02 **xpdf** поддерживает структуру PDF вплоть до 1.7. **xpdf** используется как «движок» и для других программ просмотра, например, для **kpdf**. Обновить документ можно с помощью клавиши «г». Очень удобен при просмотре в процессе подготовке документа.

### 8.2.3. Интерактивность

Зависит исключительно от стиля который используется для подготовки PDF. Присутствует весь простейший джентльменский набор: гиперссылки, различные виды переходов со слайда на слайд и анимация. Есть и ограниченная возможность демонстрировать клипы и внедрять в презентацию звуки.

## 8.3. beamer

Время шло, компьютеры матерели, появились проекторы и захотелось чего-то разноцветного. Так появилось новое поколение презентационных классов.

С помощью пакета **beamer** в принципе можно создавать «прозрачки», как это делается посредством **slides**, но основное его предназначение — электронная презентация. Пакету чуть более трёх лет, но он очень активно развивается и на сегодня это, пожалуй, лучший пакет для презентаций в  $\text{\LaTeX}$ . Автор Тилл Тантау (Till Tantau) оказался очень восприимчивым к предложениям сообщества относительно своего проекта. У **beamer** есть масса стандартных стилей, исчерпывающее описание на более чем двухстах (200) страницах ([beameruserguide.pdf](#)) и домашняя страничка <http://sourceforge.net/projects/latex-beamer>.

**beamer** можно использовать как с **pdflatex** так и со связкой **latex + dvips + ps2pdf**. При желании можно использовать **beamer** в связке с **LyX**.  $\text{\TeX}$ Live включает в себя **beamer** по умолчанию. Для установки в дистрибутиве Debian следует выполнить:

```
> sudo apt-get install latex-beamer
```

После установки в начале преамбулы выбираем класс **beamer**, примерно так:

```
\documentclass[hyperref={unicode=true}]{beamer}
\usepackage[koi8-r]{inputenc}
```

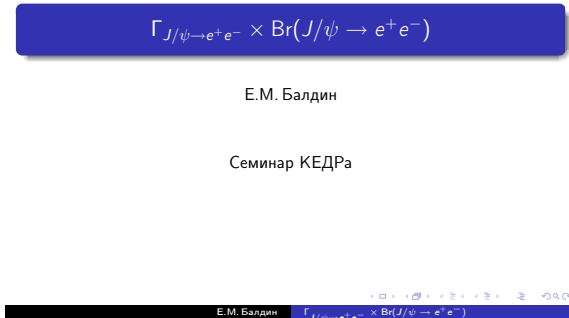
Класс **beamer** по умолчанию загружает пакет **hyperref**. Если в документе планируется использовать этот пакет со значениями отличными по умолчанию, то их следует передать как необязательный параметр команды выбора класса. Если текст

представлен в кодировке UTF-8, то это также необходимо указать при загрузке **beamer**:

```
\documentclass[utf8]{beamer}
\usepackage[utf8]{inputenc}
```

Теперь можно выбрать тему для презентации и определить заголовок для титульного листа. Единицей представления для **beamer** является окружение **frame**:

```
% выбор темы
\usetheme{Madrid}
\useoutertheme{shadow}
\title{«Заголовок»}
\date{«Дата или место проведения»}
\author{«Автор»}
\begin{document}
% титульная страница
\begin{frame}
  \titlepage
\end{frame}
```



Окружению **frame** можно передать необязательный параметр **t**, который «прижимает» текст к верхней части слайда.

Теперь можно приступить к самой презентации. Как и в обычных статьях в **beamer** можно применять команды структурной разметки типа **section**. Эти команды должны идти за пределами окружения **frame**. Структурная разметка в частности полезна для быстрого доступа, например, через оглавление. Оглавление создаётся с помощью стандартной команды `\tableofcontents`. Этой команде можно передать необязательный параметр **pausesections**, чтобы оглавление разворачивалось не сразу, а по ходу дела.

```
%структурная разметка
\section{Теория}
\begin{frame}
  %заголовок слайда
  \frametitle{Теоретическая
    зависимость (Азимов и др.)}
  ...
  \alert{GBee}... \alert{Gee}
  ...
\end{frame}
```

Теория

Теоретическая зависимость (Азимов и др.)

$$\frac{d\sigma^{e^+e^-}}{d\Omega} = \frac{1}{M^2} \left\{ \frac{9}{4} \frac{\Gamma_{e^+e^-}^2}{\Gamma M} \left(1 + \frac{3}{4}\beta\right) (1 + \cos^2\theta) \text{Im}f - \right.$$

$$\left. - \frac{3\alpha}{2} \frac{\Gamma_{e^+e^-}}{M} \left(1 + \frac{11}{12}\beta\right) \left[ (1 + \cos^2\theta) - \frac{(1 + \cos^2\theta)^2}{(1 - \cos\theta)} \right] \text{Re}f + \right.$$

$$\left. + \frac{\alpha^2}{4} \left(1 + \frac{13}{12}\beta\right) \frac{(3 + \cos^2\theta)^2}{(1 - \cos\theta)^2} \right\},$$

где

$$f = \left( \frac{\frac{M}{2}}{-W + M - \frac{i}{2}} \right)^{1-\beta}, \quad \beta = \frac{4\alpha}{\pi} \left( \ln \frac{W}{m_e} - \frac{1}{2} \right).$$

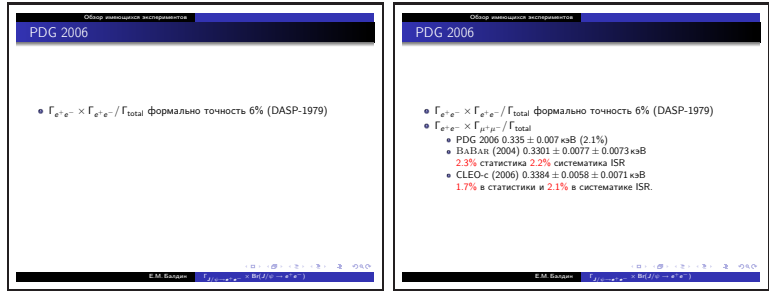
Подгоночная функция:

$$mLum \times (GBee \times jpsiee + Gee \times (inter1 + inter2) + bhabha).$$

Для создания заголовка текущего слайда используется команда `\frametitle`. Команда `\alert` является аналогом `\emph`. По умолчанию выделенный сегмент просто отображается красным цветом, но при желании `\alert` всегда можно переопределить.

**Оверлеи** В процессе представления очень полезны *оверлеи* — составные слайды, которые как бы накладываются друг на друга. Для создания простейшего оверлея используется команда `\pause`.

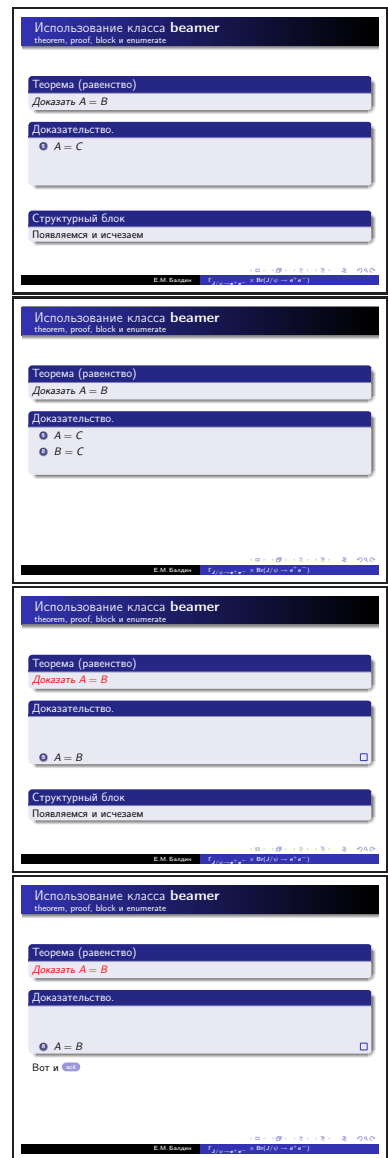
```
\begin{itemize}
  \item ...
  \pause
  \item ...
\end{itemize}
```



В **beamer** предусмотрена масса способов работы с оверлеями. Рассмотрим некоторые из них:

```
%создание своей теоремы
\newtheorem{rustheorem}{Теорема}

\begin{frame}
\frametitle{Использование
            класса \textbf{beamer}}
%подзаголовок
\framesubtitle{theorem, proof,
              block и enumerate}
%теорема
\begin{rustheorem}[равенство]
\color<3-4>[rgb]{1,0,0}
\{Доказать \((A=B)\)}
\end{rustheorem}
%доказательство
\begin{proof}
\begin{enumerate}
\item<-2> \((A=C\))
\item<2> \((B=C\))
\item<3,4> \((A=B\))\qedhere
\end{enumerate}
\end{proof}
%последняя фраза
\uncover<4->{Вот и \beamerbutton{всё}}
%манипуляция с блоком
\begin{block}<1,3>{Структурный блок}
Появляемся и исчезаем
\end{block}
\end{frame}
```





Для работы с оверлеями в **beamer** добавлен ещё один способ передачи параметров командам `< >` — меньше/больше. Таким образом команде передаётся список оверлеев на которых она должна действовать. То есть команда `\color<3–4>` раскрашивает текст в указанный цвет с 3го по 4ый оверлей. Список можно передавать через запятую или как интервал. Список вида: `-3,5-9,12,17-` означает, что команда действует для оверлеев из интервалов: от начала до 3го, от 5го до 9го, для 12го, от 17 и до конца.

Некоторые команды переопределены так, что могут воспринимать списки оверлеев. Примером таких команд являются:

- `\color{текст}` — цвет текста.
- `\item` — определена внутри перечислений к которым относятся окружения `itemize` и `enumerate`.
- Окружение **theorem**. Команда `\newtheorem` позволяет легко создавать свои теоремы.
- Окружение **prof**. Если есть теорема, то должно быть и доказательство. В конце доказательства традиционно добавляется квадратик — знак QED (quod erat demonstrandum — что и требовалось доказать). Команда `\qedhere` размещает QED в той же строке, где она указана. Иначе для QED будет отведена своя собственная строка, что не желательно.

В классе **beamer** определены так же и новые команды воспринимающие список, такие как:

- `\alert{текст}` — выделение текста.
- `\only` или `\visible` — добавление текста только для указанного списка оверлеев.
- `\invisible` — команда комплементарная `\only`.
- `\uncover` — тоже, что и `\only`, только резервируется место под текст даже на тех слайдах, где он отсутствует.
- `\alt<список>{текст}{альтернативный текст}` — для указанного списка оверлеев выводится «текст» иначе «альтернативный текст».
- Окружение **block** — именованный блок. Окружение во многом аналогично окружению **theorem**.

**Гиперссылки** Для создания гиперссылки для начало следует установить метку или якорь в нужном месте. Это можно сделать с помощью команды `\label`. После этого с помощью команды `\hyperlink` организуется гиперссылка:

```
\label{metka}
...
\hyperlink{metka}{«Гиперссылка»}
```

Вместо обычного текста можно использовать фактически любую ЛАТЭХ-структуру, например, команду создания «кнопки» `\beamerbutton`. Более общей командой для установки метки является команда:

```
\hypertarget<«номер оверлея»>{«метка»}{«текст»}
```

С помощью неё можно указать не только структурную единицу, но и на какой именно оверлей следует сослаться.

**Программный код** Для представления программного кода необходимо использовать окружения типа `verbatim` или `lstlistings`. Для того чтобы код на слайде отобразился правильно окружению `frame` необходимо передать опцию `fragile`. Оформление кода может выглядеть, например, так:

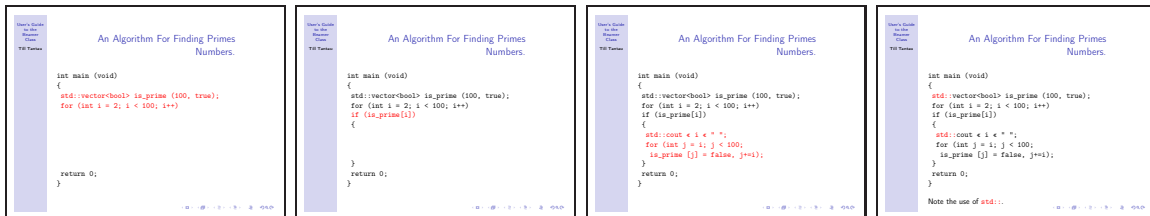
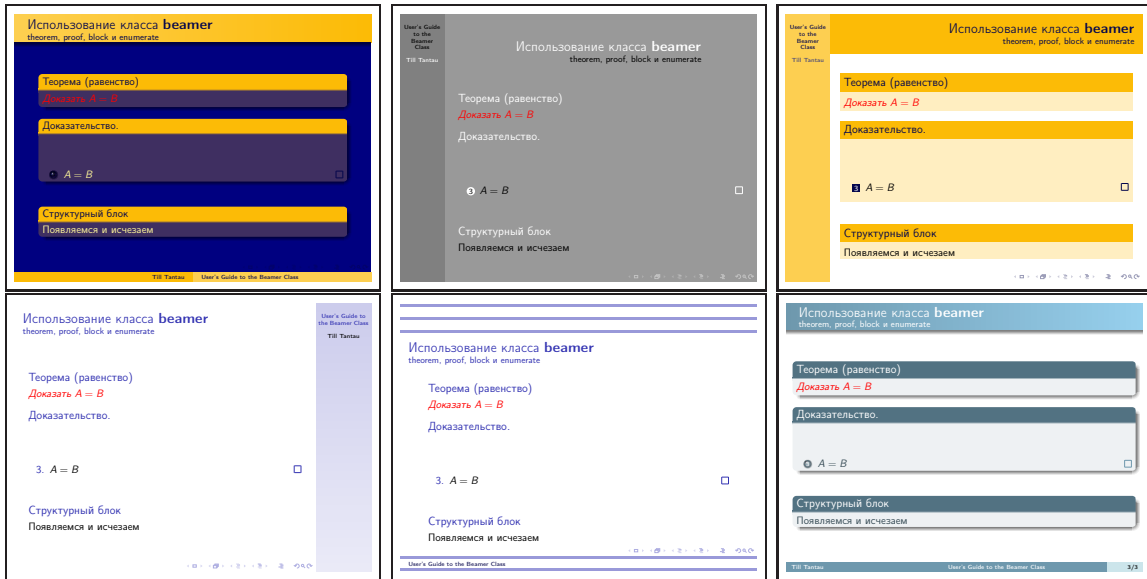


Рис. 8.2. Представление программного кода (тема Hannover)

```
\begin{frame}[fragile]
%определяем более короткие команды
\newcommand{\un}{\uncover}
\newcommand{\al}{\alert}
\frametitle{An Algorithm For Finding Primes Numbers.}
\begin{semiverbatim}
\un<1->{\al<0>{int main (void)}}
\un<1->{\al<0>{\{\}}}
\un<1->{\al<1>{\al<4>{std::vector<bool>is_prime(100,true);}}}
\un<1->{\al<1>{\for (int i = 2; i < 100; i++)}}
\un<2->{\al<2>{\if (is_prime[i])}}
\un<2->{\al<0>{\{\}}}
\un<3->{\al<3>{\al<4>{std::cout << i << " " ;}}}
\un<3->{\al<3>{\for (int j = i; j < 100;}}}
\un<3->{\al<3>{\is_prime [j] = false , j+=i);}}}
\un<2->{\al<0>{\}}}
\un<1->{\al<0>{\return 0;}}}
\un<1->{\al<0>{\}}}
\end{semiverbatim}
\end{frame}
```

Рис. 8.3. Примеры разных тем **beamer**. Малая часть от того что есть.

```

\end{semiverbatim}
\visible<4->{Note the use of \alert{\texttt{std::}}.}
\end{frame}

```

**Выбор и настройка темы** В **beamer** темы разбиваются на пять классов:

- Именные темы — концепция презентации. Для выбора темы используется команда `\usetheme`. Обычно создатель именной темы, просто выбирает в ней соответствующие цветовую, шрифтовую и декоративные темы. В **beamer** на начало 2007 года есть следующие именные темы: AnnArbor, Antibes, Bergen, Berkeley, Berlin, Boadilla, CambridgeUS, Copenhagen, Darmstadt, Dresden, Frankfurt, Goettingen, Hannover, Ilmenau, JuanLesPins, Luebeck, Madrid, Malmoe, Marburg, Montpellier, PaloAlto, Pittsburgh, Rochester, Singapore, Szeged и Warsaw.
- Цветовые темы — палитра презентации. Для выбора темы используется команда `\usecolortheme`. Можно выбрать из следующего набора палитр: albatross, beaver, beetle, crane, dolphin, dove, fly, lily, orchid, rose, seagull, seahorse, sidebartab, structure, whale и wolverine.
- Шрифтовые темы — выбор подмножества шрифтов. Для выбора темы используется команда `\usefonttheme`. Существуют следующие шрифтовые темы: professionalfonts, serif, structurebold, structureitalics serif и structuresmallcaps serif.
- Текстовые и структурные декорации — темы определяющие как выглядят перечисления, теоремы и выделения. Для выбора темы используется команда `\useinnertheme`. Можно выбрать следующие варианты декораций: circles, inmargin, rectangles, rounded.

- Внешние декорации — темы определяющие вид заголовков и обрамления слайда. Для выбора темы используется команда `\useoutertheme`. Существуют следующие типы обрамлений: `infolines`, `miniframes`, `shadow`, `sidebar`, `smoothbars`, `smoothtree`, `split` и `tree`.

Никто не мешает так же создать свою собственную тему и назвать её именем своего города или страны. Подробности о том как это делается следует искать в документации к пакету.

**Ускорение компиляции** При подготовке презентации можно использовать опцию `draft` при выборе класса. Это немного ускорит компиляцию. Так же можно указывать какие именно слайды следует включать при компиляции (похоже на `\includeonly`):

```
\includeonlyframes{ex1,ex3}
\frame[label=ex1]
{Этот слайд будет включён при компиляции.}
\begin{frame}[label=ex2]
Аналогично ex2.
\end{frame}
\frame{А вот этого слайда не будет.}
```

Использование меток позволяет выводить уже имеющиеся слайды ещё раз с помощью команды `\againframe`:

```
%ex1 будет выведен ещё раз
\againframe{ex1}
```

**Печать слайдов** На самом деле размер слайдов всего 128 мм на 98 мм, то есть большие буквы получается просто уменьшение размера листа бумаги. Для печати проще всего в Adobe Reader для растягивания страницы на А4 установить соответствующую опцию печати. Как вариант чтобы заведомо всё печаталось нормально можно воспользоваться стилевым файлом `pgfpages` из пакета `pgf`:

```
\usepackage{pgfpages}
\pgfpagesuselayout{resize to}[a4paper,border shrink=5mm,landscape]
```

Здесь слайд растягивается на страницу А4 в альбомной ориентации с отступом от краёв в 5 мм. Если хочется распечатать по два слайда на страницу, то необходимо передать следующие настройки:

```
\pgfpagesuselayout{2 on 1}[a4paper,border shrink=5mm]
```

**Мультимедиа** Пакет **beamer** включает стилевой файл **multimedia**. Загрузив этот файл можно воспользоваться командами `\movie` и `\sound` — включение клипа и звука в презентацию. К сожалению пока эта возможность ограничена тем, что поддерживает её только Adobe Reader в сборке для Windows и MacOS. Поддержка мультимедиа есть в стандарте PDF, поэтому её рано или поздно научится воспроизводить `xpdf` если Adobe System не почешется. Подробности об использовании этих команд можно посмотреть в пользовательской документации к пакету.

В пакете **beamer** предусмотрена возможность создания анимации на основе созданных слайдов. Команда

```
\animate<«список оверлеев»>
```

позволяет автоматически проигрывать последовательность слайдов. Для того чтобы эта возможность работала необходимо Adobe Reader раскрыть на весь экран.

## 8.4. Правила хорошей презентации

Создание презентации — это очень тяжёлое занятие и не следует жалеть о потерянных минутах для наведения блеска. Делая же презентацию следует не забывать об эмпирических правилах:

- Один слайд требует не меньше одной минуты.
- Один слайд со смыслом требует не менее пяти минут.
- Времени всегда не хватает.
- Не следует «пихать» в презентацию больше слайдов чем получится рассказать по времени. Перебор по времени только раздражает слушателей.
- Каждый слайд должен иметь свой заголовок (`\frametitle`)
- В один слайд можно поместить около 20–40 слов и заведомо не больше 80.
- Полезно использовать `block`, `theorem`, `proof` и `example`. Эти окружения структурируют текст и помогают выделять основные мысли.
- Для разных аудиторий правила могут отличаться.

## Справочно-поисковый аппарат издания

На этом же этаже располагалось книгохранилище. По поводу его размеров рассказывали, что в глубине, в полукилометре от входа, идёт вдоль стеллажей неплохое шоссе, оснащённое верстовыми столбами.

«Понедельник начинается в субботу»  
 Аркадий и Борис Стругацкие.

Книги делятся на те, что читаются один раз и те, что многократно перечитываются. Наличие информации о структуре книги повышает ценность любого текста. Отсутствие этой информации — прямой намёк, что после прочтения произведение следует забыть и выбросить.

Справочно-поисковый аппарат издания позволяет читателю облегчить и ускорить поиск имеющихся в книге объектов. В качестве элементов из которых складывается справочно-поисковая система можно упомянуть рубрикации, оглавление, колонтитулы, ссылки, подстрочные примечания, алфавитный указатель и библиографию.

Этот аппарат существует исключительно для читателя и он достаточно трудоёмок при создании. Но сложности не должны пугать истинных энтузиастов в деле создания текстов, так как их преодоление значительно повышают ценность серьёзного произведения.

### Пример рубрикации

А. В. Тор

#### Содержание

1. Раздел .....	1
1.1. Подраздел .....	1
1.1.1. Подподраздел .....	1
Заключение .....	1
А. Приложение .....	1

#### 1. Раздел

Основной элемент рубрикации.

#### 1.1. Подраздел

Вспомогательный элемент рубрикации.

#### 1.1.1. Что-то более мелкое чем подраздел

Вспомогательный для вспомогательного. В содержании выводится краткая версия заголовка.

**Параграф** Важный параграф.

**Подпараграф** Параграф чуть менее важный.

#### Раздел, отсутствующий в содержании

Всяко бывает. Иногда и такое нужно.

#### Заключение

Заключение в отличие от, скажем, раздела 1.1 на странице 1 нумеровать не надо, но в содержание отразить необходимо.

#### А. Приложение

Рис. 9.1. Пример рубрикации и оглавления

## 9.1. Рубрикация и оглавление

Нужны ли книге оглавление или содержание? Любой скажет: что за вопрос, конечно, нужны. И не только в книге научной и деловой. В любой.

---

А. Э. Мильчин «Культура издания»

Для оформления разделов в основном используются команды секционирования `\section`, `\subsection`, `\subsubsection`, `\paragraph` и `\subparagraph`.

Команды перечислены в порядке убывания значимости при рубрикации. Обычно, любой сколько-нибудь сложный текст следует начинать с планирования структуры, то есть создать шаблон, примерно следующего вида:

```
\documentclass[a4paper,12pt]{ncc}
\usepackage[warn]{mathtext}
\usepackage[T2A]{fontenc}
\usepackage[koi8-r]{inputenc}
\usepackage[english,russian]{babel}
\usepackage{indentfirst}
\title{Пример рубрикации}
\author{А. \ ,В. ~Тор}
\begin{document}
\maketitle{}
\tableofcontents{}

\section{Раздел}
\label{sec:section}
Основной элемент рубрикации.

\subsection{Подраздел}
\label{sec:subsection}
Вспомогательный элемент рубрикации.

\subsubsection[Подподраздел]{Что-то более мелкое чем подраздел}
\label{sec:subsubsection}
Вспомогательный для вспомогательного. В содержании выводится
краткая версия заголовка.

\paragraph{Параграф}
\label{sec:paragraph}
Важный параграф.

\subparagraph{Подпараграф}
\label{sec:subparagraph}
```

Параграф чуть менее важный.

```
\section*{Раздел, отсутствующий в содержании}
```

Всяко бывает. Иногда и такое нужно.

```
\section*{Заключение}
\label{sec:afterwords}
\addcontentsline{toc}{section}{Заключение}
```

Заключение в отличие от, скажем, раздела `\ref{sec:subsection}` на странице `\pageref{sec:subsection}` нумеровать не надо, но в содержание отразить необходимо.

```
\appendix
```

```
\section{Приложение}
\label{appendix}
```

```
\end{document}
```

Результат компиляции кода представлен на рис. 9.1. Кроме самих заголовков разделов, созданных с помощью команд секционирования, в начале создаётся оглавление. За создание оглавления отвечает команда `\tableofcontents`. При каждой компиляции информация о разделах собирается в файле с тем же именем, что и у `tex`-файла, но с расширением `toc`. При следующей компиляции обновлённая информация о разделах используется для создания оглавления.

Уровень до которого информация отображается в оглавлении можно поменять изменив значение переменной `tocdepth`, например, так:

```
\setcounter{tocdepth}{2}
```

В этом случае будет показана информация о разделах вплоть до второго уровня. Раздел типа `\section` соответствует первому уровню секционирования, `\subsection` второму и так далее.

Кроме оглавления также можно вывести список иллюстраций `\listoffigures` и таблиц `\listoftables`. Информация об иллюстрациях и таблицах автоматически собирается в файлах с расширениями `lof` и `lot`.

Для добавления какой-то информации в оглавления в обход команд секционирования можно воспользоваться командой

```
\addcontentsline{toc}{«уровень раздела»}{«строка в оглавлении»}
```

У этой команды три аргумента. Первый аргумент соответствует расширению файла (`toc`, `lof` или `lot`), куда добавляется «строка в оглавлении». Уровень раздела определяется именами команд секционирования, то есть `section`, `subsection` и так да-



лее. Команды  $\LaTeX$  при передаче в файлы списков следует защищать командой `\protect`, дабы избежать проблем с «хрупкими» инструкциями.

Команда `\appendix` отделяет приложение от основного текста. После её вызова правила нумерации разделов изменяются. `\appendix` тоже является командой секционирования.

## 9.2. Ссылки и примечания

Иногда не хочется разбивать канву повествования и в то же время необходимо как-то вставить пояснение. Это можно сделать просто сославшись на какой-то другой фрагмент текста с помощью ссылки или вставить пояснение на этой же странице с помощью подстрочного примечания.

### 9.2.1. Механизм ссылок

В примере, демонстрирующем работу команд секционирования вслед за каждой такой командой ставилась метка с помощью инструкции `\label`. Метка представляет собой последовательность ASCII-символов. При компиляции документа информация о имеющихся метках добавляется в файл с расширением `aux`. Для того чтобы извлечь эту информацию, то есть номер раздела (команда `\ref`) или номер страницы (`\pageref`) необходимо пропустить текст через `latex` ещё раз.

Для того чтобы можно было сослаться на внешний документ следует воспользоваться пакетом `xr`. В этом случае в преамбуле необходимо добавить примерно следующие инструкции:

```
\usepackage{xr}
\externaldocument [EXT-]{externaldoc}
```

Это позволяет получить доступ к меткам файла `externaldoc.tex`. Обращение к меткам, как и обычно, осуществляется помощью команд `\ref`/`\pageref`, только перед именем метки добавляется префикс `EXT-`. Можно обойтись и без префикса, так как этот параметр является опциональным, но в этом случае повышается вероятность конфликта из-за одинаковых меток.

Ссылаться можно не только на разделы. Метки внутри нумерованных окружений, типа `equation` (выключенные математические формулы<sup>1</sup>) или `theorem` (теоремы), принимают их номер. Это так же касается рисунков (окружение `figure`) и таблиц (окружение `tabular`). В этом случае `\label` должна следовать сразу за командой `\caption`, формирующей подпись к плавающему объекту.

Общего рецепта как делать ссылки на электронные ресурсы нет. Проще всего использовать команду `\url` из одноимённого пакета:

---

<sup>1</sup>Для правильной ссылки на номера формул вместо команды `\ref` следует использовать инструкцию `\eqref`

Моя WWW-страничка находится по адресу  
`\url{http://www.inp.nsk.su/~baldin/}`.

Моя WWW-страничка находится по адресу `http://www.inp.nsk.su/~baldin/`.

Ссылка печатается В адресной строке должны отсутствовать символы %, #, ^ и она не должна заканчиваться символом \. Если есть желание уйти и от этих ограничений, то аналогично команде `\verb` инструкцию можно использовать и так: `\url!http://www.адрес.ru!`.

В плане создания и управления гиперссылками также интересен пакет **hyperref** который предоставляет схожую функциональность и позволяет создавать гиперссылки в pdf-документах, но это уже совсем другая история:

```
%загрузка пакет hyperref
\usepackage [ unicode=true ] { hyperref }
```

## 9.2.2. Подстрочные примечания

Подстрочное примечание формируется с помощью команды `\footnote`. Правила оформления примечаний прописываются в определении класса. Без особых на то причин менять эти правила не стоит. Примечание, если позволяет место, печатается на той же странице где помещена ссылка. Текст и примечание отделяются разделительной линией.

Примечание можно добавлять и внутри окружения **minipage**, но тогда оно печатается внутри окружения:

```
\begin{minipage}{1.0\linewidth}
  Ссылки раз\footnote{Сноска.} и
  два\footnote[26]{Подстрочное примечание.}.
\end{minipage}
```

Ссылки раз<sup>a</sup> и два<sup>z</sup>.

<sup>a</sup>Сноска.

<sup>z</sup>Подстрочное примечание.

Необязательный параметр `\footnote` позволяет присвоить примечанию значение по выбору пользователя.

Для того чтобы можно было сделать сноску внутри заголовка раздела необходимо защитить инструкцию `\footnote` командой `\protect`:

```
\section{Заголовок\protect\footnote{Подстрочное примечание.}}
```

Команда создания подстрочного примечания является «хрупкой».

Иногда в сложных ситуациях, например, когда требуется сделать подстрочное примечание внутри бокса, для формирования сноски требуется прибегнуть к независимым командам создания ссылки и создания примечания:

```
\footnotemark [ num ]
\footnotetext [ num ] { «сноска» }
```

Необязательный параметр `num` как и в случае `\footnote` позволяет формировать свою нумерацию. Для хранения текущего номера ссылки используется счётчик `footnote`.

### 9.3. Колонтитулы

Правила формирования колонтитулов целиком зависят от выбранного класса документа. Если же хочется изменить значения по умолчанию, то проще всего выбрать стиль страницы `myheadings` и сформировать колонтитулы:

```
\pagestyle{myheadings}
\markboth{«левый колонтитул»}{«правый колонтитул»}
```

Если печать односторонняя, то достаточно воспользоваться командой `\markright`, которая имеет только один аргумент.

Для полного управления содержимым колонтитулов лучше всего подходит пакет `fancyhdr`. Подробно об этом пакеты было рассказано в разделе 6.3.2.

### 9.4. Библиография

Книги создаются не в безвоздушном, или бескнижном, пространстве.

---

А. Э. Мильчин «Культура издания»

Хорошая книга представляет собой ценность. На даже самая лучшая книга не в состоянии охватить абсолютно все аспекты рассматриваемой в ней темы. Книги существуют в книжном пространстве. Всегда можно найти что-то, на чём книга основывалась, что-то, что развивает основную идею и что-то, что позволяет взглянуть на главную тему с другой стороны. Список литературы книгу только украшает.

Для ссылок на литературу используется команда

```
\cite[«комментарий»]{«список меток»}
```

Метки либо формируются автором самостоятельно внутри окружения `thebibliography`

```
Полезно почитать книгу \cite[Роженко]{rozenko-2005}.
```

```
\begin{thebibliography}{9}
\bibitem{rozenko-2005}Роженко А.И. Искусство вёрстки в
\LaTeX'e. \newblock — Новосибирск: Изд. ИВМиМГ СО РАН,
2005. 398~с.
\end{thebibliography}
```

Полезно почитать книгу [1, Роженко].

## Список литературы

- [1] Роженко А.И. Искусство вёрстки в  $\text{\LaTeX}$ 'е. — Новосибирск: Изд. ИВМиМГ СО РАН, 2005. 398 с.

Рис. 9.2. Пример библиографической ссылки.

либо используется механизм  $\text{\bibTeX}$ . Команда `\newblock` позволяет логически разделить разные по смыслу элементы. В качестве обязательный аргумента окружения `\thebibliography` требуется передать текст соответствующий самой широкой метки для выравнивания. То есть, если список литературы содержит меньше 10 записей, то достаточно передать однобуквенную фразу, например, «9», а в случае двузначного числа книг в списке уже потребуется «99» и так далее.

Как и в случае с перекрёстными ссылками информация о списке литературы заносится в файл с расширением `aux`. То есть для правильного его отображения необходимо два прохода  $\text{\LaTeX}$ .

Для того чтобы можно было использовать кириллицу в метках для цитирования литературы, а именно иметь возможность написать что-то вроде:

```
\cite{Котельников-2004}
...
\bibitem{Котельников-2004}
```

Следует воспользоваться пакетом `citehack`:

```
\usepackage{citehack}
```

Из названия пакета очевидно что это хак со всеми вытекающими. Не следует им злоупотреблять.

### 9.4.1. $\text{\bibTeX}$

Можно список литературы оформлять руками. Есть какой-никакой стандарт, например, тот-же ГОСТ 7.80-00<sup>2</sup> или ГОСТ 7.1-84. Берём и просто следуем подробной инструкции. Но, далеко не все издательства подчиняются этому стандарту в котором, например, нет информации как нужно оформлять `www`-ссылки. Да и вообще список сопутствующей литературы это нечто большее, чем просто довесок к статье или книге — это вполне самостоятельный фрагмент информации, который очень полезно уметь представлять по разному.

---

<sup>2</sup>ГОСТ на оформление библиографического указателя, принятый в 2000 году. Правила оформления могут нравиться или не нравиться, но это всё-таки хоть какой-то стандарт.

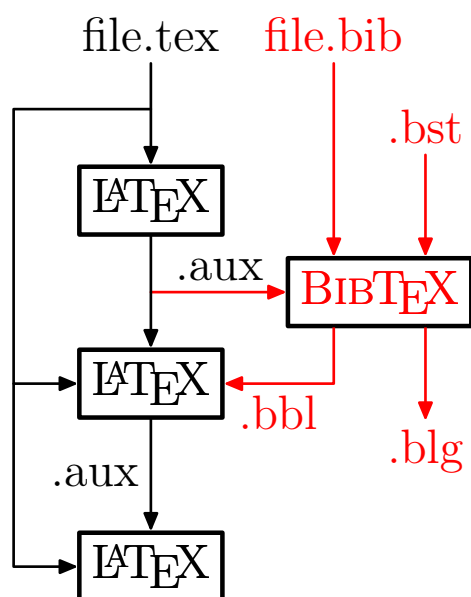


Рис. 9.3. Конвейер  $\text{\LaTeX}+\text{\BibTeX}$ . `tex` —  $\text{\LaTeX}$ -исходник, `bib` — библиографическая база, `bst` — стилевой файл для библиографии, `blg` — log-файл  $\text{\BibTeX}$ , `bbl` — отсортированный список литературы, `aux` — информация о ссылках.

Для решения этой проблемы Орен Поташник разработал программу  $\text{\BibTeX}$ , которая сама формирует окружение `\thebibliography`, получая информацию из текстовой библиографической базы. Структура библиографической базы  $\text{\BibTeX}$  является довольно распространённым форматом, который использует в том числе и Google Scholar (<http://scholar.google.com/>), не говоря уж о том, что основной архив электронных препринтов <http://xxx.lanl.gov> предоставляет библиографическую информацию исключительно в виде записей  $\text{\BibTeX}$ .

Из программного обеспечения, позволяющего работать с  $\text{\BibTeX}$  следует упомянуть встроенный в Emacs пакет RefTeX и JabRef <http://jabref.sourceforge.net/>. Тот, кто не освоил Emacs и кому не нравится Java может поискать программные пакеты gBib и KBib для Gnome и KDE, соответственно. Простой конвертер `bibtex2html` позволяет получить список литературы и html-виде. Естественно, и простое редактирование текстового файла руками так же никто не отменял.

Как правило, библиографическая база в формате  $\text{\BibTeX}$  хранится в файле с расширением `bib`. Перед тем как с помощью команды `\bibliography` подключить базу к  $\text{\LaTeX}$ -исходнику нужно выбрать стиль сортировки библиографии:

```

\ bibliographystyle {«стиль»}
\ bibliography {«имя bib-файла»}

```

В  $\text{\LaTeX}$  есть четыре стандартных стиля для формирования списка литературы:

**plain** — открытый стиль. Библиографические записи помечаются порядковыми номерами и сортируются в алфавитном порядке. Чтобы правильно отсортировать библиографию на русском языке необходимо вместо `bibtex` воспользо-

## Список литературы

- [1] *Гуссенс, М.* Путеводитель по пакету  $\LaTeX$  и его расширению  $\LaTeX 2\epsilon$ : Пер. с англ. / М. Гуссенс, Ф. Миттельбах, А. Самарин. — М.: Мир, 1999. — 606 с.
- [2] *Львовский, С. М.* Набор и вёрстка в системе  $\LaTeX$ . / С. М. Львовский. — М.: МЦНМО, 2003. — 448 с. — 3-е изд., испр. и доп.
- [3] *Котельников, И. А.*  $\LaTeX$  по русски. / И. А. Котельников, П. З. Чеботаев. — Новосибирск: Сибирский Хронограф, 2004. — 496 с. — 3-е изд., перераб. и доп.
- [4] *Роженко, А. И.* Искусство вёрстки в  $\LaTeX$ 'е. / А. И. Роженко; Под ред. А. Алексеева. — Новосибирск: Изд. ИВМиМГ СО РАН, 2005. — 398 с.

Рис. 9.4. Список литературы оформленный с помощью  $\text{Vi}\TeX$ . Стиль **gost780u**.

ваться командой **bibtex8** указав с помощью ключа `--csfile` соответствующее правило сортировки<sup>3</sup>.

**unsrt** — не сортирующий стиль. В отличии от **plain** порядок представления списка литературы определяется порядком цитирования библиографии в тексте.

**alpha** — алфавитный стиль. Вместо нумерации библиографии используются имена меток. Литература сортируется по меткам.

**abbrv** — аббревиатурный стиль. Вместо полных имён авторов, названий месяцев и журналов печатаются сокращения. Сортировка и нумерация соответствует стилю **plain**.

Максим Поляков разработал стили для  $\text{Vi}\TeX$  соответствующие ГОСТ 7.80-00 и ГОСТ 7.1-84: **gost780s/gost71s** аналогичные **plain** и **gost780u/gost71u** аналогичные **unsrt**. Описание этих стилей представлено в стандартной документации в виде файлов **gost780.pdf** и **gost71.pdf**. Многие журналы принимающие публикации в  $\LaTeX$  имеют свои собственные  $\text{Vi}\TeX$ -стили. В стандартной поставке  $\text{TeXLive}$  2007 идёт более 200 различных библиографических стилей.

В дополнение к команде цитирования `\cite` в случае подключения библиографической базы можно использовать инструкцию `\nocite`. Команда `\nocite` не создаёт никакой ссылки в тексте, но упомянутая запись отображается в списке литературы.

Сама по себе база состоит из записей вида:

```
@book{Gussens – 1999,
  author = {М. Гуссенс and Ф. Миттельбах and А. Самарин},
  title = {Путеводитель по пакету \LaTeX{} и его
           расширению \LaTeXe: Пер. с англ.},
```

<sup>3</sup>В стандартной поставке  $\LaTeX$  есть правило для сортировки для кодовой страницы `sr866` `sr866rus.csf`. На основе этого файла можно создать правило для другой кодовой страницы.

```

year = {1999},
isbn = {5-03-003325-4},
publisher = {Мир},
address = {М.},
numpages = {606},
language = {russian},
OPTnote = {}
}

```

После знака «коммерческое at» @ идёт тип записи. В фигурных скобках вслед за меткой через запятую перечисляются пары ключ-значение. В<sub>И</sub>Т<sub>Е</sub>Х поддерживает определённый набор типов записей, каждому из которых соответствуют свои обязательные и необязательные поля<sup>4</sup>. Если не заполнено обязательное поле, то при компиляции В<sub>И</sub>Т<sub>Е</sub>Х генерирует ошибку. Имеются следующие стандартные типы записей (В<sub>И</sub>Т<sub>Е</sub>Х не чувствителен к регистру):

- **Article** — статья в журнале. Обязательные поля: author, title, journal, year. Необязательные поля: volume, number, pages, month, note, annote.
- **Book** — книга. Обязательные поля: author или editor, title, publisher, year. Необязательные поля: volume, number, series, address, edition, month, note, annote.
- **Booklet** — брошюра. Обязательное поле: title. Необязательные поля: author, howpublished, address, month, year, note, annote.
- **Conference** или **InProceedings** — статья опубликованная в трудах конференции. Обязательные поля: author, title. Необязательные поля: crossref, booktitle, pages, year, editor, volume, number, series, address, month, organisation, publisher, note, annote.
- **Proceedings** — труды конференции. Обязательные поля: title, year. Необязательные поля: booktitle, editor, volume, number, series, address, month, organisation, publisher, note, annote.
- **InBook** — ссылка на часть книги, то есть на её главу, раздел или просто на определённый набор страниц. Обязательные поля: author или editor, title, chapter, publisher year. Необязательные поля: volume или number, series, type, address, edition, month, pages, note, annote.
- **InCollection** — часть книги со своим заглавием. Обязательные поля: author, title, booktitle. Необязательные поля: crossref, pages, publisher, year, editor, volume или number, series, type, chapter, address, edition, month, note, annote.

---

<sup>4</sup>Если к названию необязательного поля добавить ОПТ (note→OPTnote), то такие поля игнорируются, даже если присутствуют в записи.

- **Manual** — техническая документация. Обязательное поле: `title`. Необязательные поля: `author`, `organistaion`, `address`, `edition`, `month`, `year`, `note`, `annotate`.
- **PhdThesis** — диссертация. Обязательные поля: `author`, `title`, `school`, `year`. Необязательные поля: `address`, `month`, `note`, `annotate`.
- **MastersThesis** — дипломная работа. Обязательные и необязательные поля копируют **PhdThesis**.
- **TechReport** — отчёт. Обязательные поля: `author`, `title`, `institution`, `year`. Необязательные поля: `type`, `numer`, `address`, `month`, `note`, `annotate`.
- **Unpublished** — неопубликованный авторский текст. Обязательные поля: `author`, `title`, `note`. Необязательные поля: `month`, `year`, `annotate`.
- **Misc** — то, что не подходит для других типов записей. Обязательные поля отсутствуют. Необязательные поля: `author`, `title`, `howpublished`, `month`, `year`, `note`, `annotate`.

Значение полей в большинстве случаев понятно из их названия. Исключением, пожалуй является поле **crossref**, в качестве значения которого можно передать ссылку на другую запись из которой ВІВТѢХ при трансляции возьмёт значения всех недостающих полей записи. Из особенностей следует упомянуть, что авторы в поле **author**, разделяются с помощью союза **and**.

Кроме перечисленных стандартных полей при использовании библиографических ГОСТ-стилей **gost780u/gost71u** и **gost780s/gost71s** можно использовать поля **numpages** — число страниц и **language** — влияет на сортировку при выборе стиля типа **plain**. Допустимые значения для поля **language** — `russian`, `ukrainian` и `english` по умолчанию.

## 9.5. Алфавитный указатель

Указатель значительно повышает ценность любой книги, если она из тех, что читают не только на сквозь, но и выборочно.

---

А. Э. Мильчин «Культура издания»

Для создания «полуфабриката» алфавитного указателя в преамбуле помещается инструкция

```
\makeindex
```

Слова, которые нужно поместить в указатель, отмечаются с помощью команды **\index**.

```
Указатель \index{Предметный указатель} ...
```



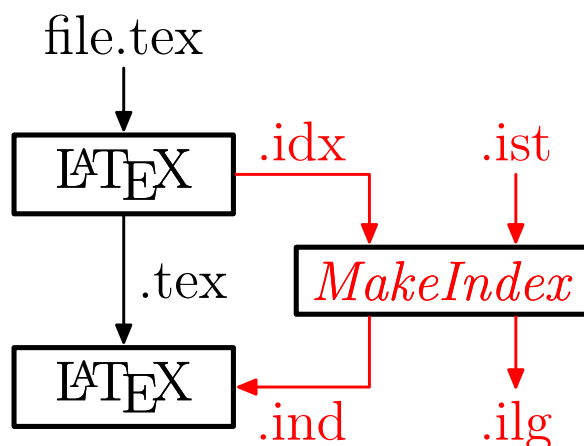


Рис. 9.5. Конвейер  $\text{LATEX} + \text{MakeIndex}$ .  $\text{tex}$  —  $\text{LATEX}$ -исходник,  $\text{idx}$  — не отсортированный индекс (полуфабрикат),  $\text{ist}$  — стилевой файл для указателя,  $\text{ilg}$  — log-файл  $\text{MakeIndex}$ ,  $\text{ind}$  — отсортированный указатель.

Сама по себе команда  $\backslash\text{index}$  игнорируется, но всё, что в ней отмечается вместе с информацией о положении команды, заносится в файл с расширением  $\text{idx}$ . Имя  $\text{idx}$ -файла по умолчанию соответствует имени основного документа. Полуфабрикатом  $\text{idx}$ -файл является потому, что записи хранятся в нём в не отсортированном виде. С помощью программ сортировки **rumakeindex** получается уже отформатированный правильным образом указатель в файле с расширением  $\text{ind}$ . Этот файл уже можно вставить в документ:

```
 $\backslash\text{input} \{ \langle \text{ind-файл} \rangle \}$ 
```

Это же делает и команда  $\backslash\text{printindex}$  из пакета **makeidx**. В дополнение к подключению индекса команда  $\backslash\text{printindex}$  проверяет существование индексного файла и не даёт  $\text{LATEX}$  генерировать ошибку в случае его отсутствия. Пакет **makeidx** содержит ещё несколько полезных команд для создания индекса, поэтому его в любом случае имеет смысл загрузить.

Программа **rumakeindex** является простейшим скриптом где с помощью **sed** кириллические буквы из внутреннего представления  $\text{LATEX}$  переводятся в  $\text{ko18}$ -г и правильным образом сортируются с использованием стандартного механизма **makeindex**. Если необходимо отсортировать индексный файл для включения в текст, использующий другую кодовую страницу или просто не устраивает правила сортировки, то этот скрипт легко переделывается.

Оригинальная программа сортировки индекса **makeindex** (подробная документация представлена в файле **makeindex.dvi**) была написана довольно давно и, естественно, не учитывала национальных особенностей других языков кроме английского и немецкого. К сожалению она оказалась достаточно гибка, чтобы полностью отказаться от неё в пользу другого механизма сортировки уже нормально поддерживающего интернационализацию. Наиболее вероятным претендентом на замену уже долгое время является **xindy** (<http://www.xindy.org/>), который поддержива-

## Предметный указатель

*MakeIndex*, 110

Указатель, 110–111

makeindex, см. rumakeindex

rumakeindex, 110, 111

xindy, 111

Рис. 9.6. Пример готового предметного указателя.

ет множество языков из коробки вплоть до клингонского, но до сих пор отсутствует в основных L<sup>A</sup>T<sub>E</sub>X-дистрибутивах.

Аргумент команды `\index` может содержать любые символы кроме `!`, `"`, `@` и `|`. Их специальное значение проявляется только внутри команды. Чтобы убрать специальное значение этих символов внутри `\index` необходимо перед ними добавить символ `"` (двойную кавычку). На рис. 9.6 представлен пример простейшего указателя. Ниже будут раскрыты методы его создания.

Для формирования многоуровневых иерархических указателей используется разделитель в виде восклицательного знака `«!»`:

```
% на страницах 110 и 111
\index{Указатель!rumakeindex}
```

Команды `\index` с одинаковыми аргументами группируются в одну запись с полным списком страниц.

Символ вертикальной черты `«|»` используется для отделения видимого аргумента от управляющих знаков. Команда `\see` (перекрёстная ссылка на другую запись), определённая в пакете **makeidx**, в индексе должна идти вслед за этим разделителем:

```
\index{Указатель!makeindex|see{rumakeindex}}
```

Так же с помощью вертикальной черты можно сформировать указатель на диапазон страниц:

```
% на странице 110
\index{Указатель|({
  много текста
% на странице 111
\index{Указатель|)})}
```

Иногда номер страницы нужно как-то выделить. Команды выделения текста также должны идти после вертикальной черты:

```
% Выделяем страницу 110 жирным шрифтом
\index{MakeIndex@\textit{MakeIndex}textbf}
```

Конструкция «ключ»@«запись» используется для правильной сортировки внутри \index. По «ключу» производится сортировка, а «запись» выводится в предметном указателе.

## Заключение

Написать любую книгу безумно тяжело. Сделать её полезной почти невозможно. Справочно-поисковый аппарат издания это лишь инструмент на этом пути. Но инструмент неоднократно проверенный грамотным человечеством. Можно путь пройти и без него, но с ним будет гораздо интереснее.

## Всё о таблицах

Классификация химических элементов, позволяющая выявить зависимость их различных свойств от атомной массы.

---

Таблица Менделеева — пожалуй, самая полезная из таблиц.

Таблица — это особая форма передачи содержания. Далеко не всякое содержание следует облекать в форму таблицы. Но для малых объёмов однородных значимых данных таблицы бывают лучшим способом отображения.

### 10.1. Немного теории

Содержимое таблицы организуется в колонки (графы) и горизонтальные строки таким образом, что каждый элемент является составной частью и строки, и колонки. Таблица состоит из следующих основных элементов: нумерационного и тематического заголовков (номер таблицы и её название), головки<sup>1</sup> (заголовочная часть таблицы), хвоста (вся остальная часть таблицы без головки), боковика (первая слева графа таблицы) и прографки (хвостовая часть таблицы без боковика).

<b>Боковик</b>	<b>Прографка</b>	<b>Головка</b>
<b>ИЛИ</b>	Истина    Ложь	
Истина	Истина    Истина	
Ложь	Истина    Ложь	<b>Хвост</b>

Таблица 10.1. Структура таблицы.

Заполняя таблицу текстовыми или цифровыми данными полезно следовать правилам:

---

<sup>1</sup>В «издательском словаре-справочнике» А. Э. Мильчина утверждается, что попытка замены этого термина по той причине что он якобы является жаргонным не привилась.

- Выносить данные общие для каждого элемента графы в её заголовок, а общие для каждого элемента строки в заголовок боковика.
- По возможности употреблять числа не более чем из четырёх значащих цифр. Для этого более многозначные числа следует округлять. Общий множитель следует вынести в заголовок. То же самое нужно сделать и с единицами измерения.
- Всегда перед знаком, отделяющим целую часть числа от дробной, должна быть цифра. То есть правильно писать «0.1», но не «.1».
- Проставлять вместо отсутствующих данных многоточие «...», «Нет свед.». Если данных в принципе быть не может, то лучше отметить это с помощью тире «—».
- Не следует использовать знаки, означающее «то же, что и предыдущее значение». Лучше повторить число.

Честно говоря, описание таблиц с помощью ЛАТЭХ-разметки может показать не очень удобным. Действительно когда в перемешку идут данные и управляющие структуры результат может выглядеть не очень красиво. Спасает только то, что большие таблицы с однородной структурой можно создавать с помощью скрипта, а маленькие таблицы не являются особой проблемой. Как правило, гораздо больше времени чем оформление таблицы занимает поиск и верификация данных.

## 10.2. `tabbing`

Если точно известны ширина столбцов и таблица относительно простая, то можно воспользоваться окружением `tabbing`.

```
\begin{tabbing}
MMMM \= MMMM \=      \kill
\textbf{\ ИЛИ} \> Истина \> Ложь  \\
Истина \> Истина \> Истина \\
Ложь   \> Истина \> Ложь
\end{tabbing}
```

<b>ИЛИ</b>	Истина	Ложь
Истина	Истина	Истина
Ложь	Истина	Ложь

Первая строчка устанавливает положение табуляторов с помощью команды `\=`. Команда `\kill`, завершающая управляющую строку, даёт понять текстовому процессору, что её не надо печатать. Далее идёт обычный текст, где переход к следующей табуляции осуществляется с помощью команды `\>`, а перевод строки завершается комбинацией `\\`. Всё просто — точно так же таблицы печатали с помощью обычной печатной машинки, только табуляция задавалась «железным», а не программным способом.

К использованию окружения `tabbing` следует подходить с известной долей осторожности. Окружение формирует абзац, состоящий из отдельных строк, в котором

нет месту переносам и многострочным элементам. Зато это позволяет  $\LaTeX$  легко перенести часть таблицы на следующую страницу. Абсолютно всё приходится делать своими руками, но в этом есть своеобразная прелесть. Часто **tabbing** становится базой для определения других более специализированных окружений.

Некоторые стандартные команды  $\LaTeX$  внутри **tabbing** переопределены. Это в частности касается команд переноса ( $\backslash-$ ) и акцентирования ( $\backslash'$  и  $\backslash\prime$ ). Для сохранения/воспроизводства текущей табуляции следует воспользоваться командами  $\backslashpushtabs/\backslashpoptabs$ .

### 10.3. tabular и array

Самым популярным окружением для отображения таблиц в  $\LaTeX$  является **tabular**. Окружение **array** фактически полностью повторяет функционал **tabular**, но в отличие от последнего работает в математической моде — полезно для создания матриц, которые по своей сути тоже обычные таблицы.

```
\centering
\begin{tabular}[c]{l|ll}
\textbf{\ ИЛИ} & Истина & Ложь \\ \hline
Истина & Истина & Истина \\
Ложь & Истина & Ложь
\end{tabular}
```

<b>ИЛИ</b>	Истина	Ложь
Истина	Истина	Истина
Ложь	Истина	Ложь

Данные делятся на ячейки с помощью символа «логическое И»  $\&$ . Переход на следующую строку контролируется стандартной командой переноса строки  $\backslash\backslash$ . В качестве необязательного параметра команды переноса строки можно указать дополнительный сдвиг по вертикали. Вертикальный размер каждой из строк автоматически выставляется в соответствии с высотой текста.

**tabular** создаёт единый объект — таблицу. Как и в случае картинок таблицу удобно заключать в плавающее окружение **table**:

```
\begin{table}[ht]
\centering%центрируем таблицу
\begin{tabular}[«позиционирование таблицы»]{«формат столбцов»}
«тело таблицы»
\end{tabular}
\caption{«подпись»}\label{tab:metka}
\end{table}
```

Это позволяет автоматически создать нумерационный заголовок таблицы и добавить тематический. Необязательный аргумент окружения **tabular** даёт возможность указать как позиционировать всю таблицу по вертикали по отношению к окружающему тексту:

**t** — выравнивание по верхней строке, то есть верхняя строка таблице будет расположена на одном уровне со строкой где эта таблица размещена,

- c** — выравнивание по центру,
- b** — выравнивание по нижней строке.

Ширина столбцов вычисляется автоматически по заданному формату, который задаётся через обязательный аргумент окружения. Каждому из столбцов должна соответствовать своя буква:

- l** (*left*) — выравнивание по левому краю,
- c** (*center*) — выравнивание по центру,
- r** (*right*) — выравнивание по правому краю,
- r**{«ширина»} — задание колонки определённой ширины. В случае жёстко заданной длины слишком длинный текст может разбиваться на несколько строк.

Если формат столбцов повторяется, то для сокращения записи можно воспользоваться следующей спецификацией:

```
*{n}{«формат столбца или столбцов»}
```

Где *n* — число повторений. Это своеобразный цикл.

```
\centering
\begin{tabular}{|*3{c|}}
\textbf{ИЛИ} & Истина & Ложь \\ \hline \hline
Истина & Истина & Истина \\
Ложь & Истина & Ложь
\end{tabular}
```

<b>ИЛИ</b>	Истина	Ложь
Истина	Истина	Истина
Ложь	Истина	Ложь

Разделительные линии между столбцами задаются с помощью вертикальной черты |. Две вертикальные линии || формируют двойной разграничитель. Горизонтальные линии создаются с помощью команды **\hline**. По аналогии с двойной вертикальной чертой две команды формируют двойную горизонтальную линию. Инструкция @{} позволяет вставить между столбцами любой символ указанный в качестве обязательного аргумента. При этом подавляются околостолбцовые промежутки, добавляемые в по умолчанию автоматически. Это можно быть полезно в случае если один столбец представляет собой какую-то измеренную величину, а второй её ошибку — в этом случае вместо разделительной черты между ними лучше вставить знак ±.

```
\centering
\begin{tabular}{c|p{2cm}@{${\pm}$}r|}
\textbf{ИЛИ} & Истина & Ложь \\ \cline{2-3}
Истина & Истина & Истина \\ \cline{1-1}\cline{3-3}
Ложь & Истина & Ложь \\ \cline{2-2}
\end{tabular}
```

<b>ИЛИ</b>	Истина	±	Ложь
Истина	Истина	±	Истина
Ложь	Истина	±	Ложь

Для того чтобы отчеркнуть только часть столбцов можно воспользоваться командой `\cline{диапазон столбцов}`.

Окружение `array` в дополнение к стандартным типам столбцов, используемых в `tabular`, добавляет два новых:

`m{«ширина»}` — то же, что и `r{«ширина»}`, но добавляется вертикальное выравнивание содержимого по центру клетки ,

`b{«ширина»}` — то же, что и `m{«ширина»}`, но вертикальное выравнивание содержимого идёт по нижней базовой линии последней строки.

Более подробную информацию об использовании окружения `array` следует искать в файле документации `array.pdf` из пакета `tools`.

Подробнее о том как *должна* выглядеть таблица и как это достигается можно узнать, например, в статье «Publication quality tables in L<sup>A</sup>T<sub>E</sub>X» (`booktabs.pdf`), написанной Симоном Фиром (Simon Fear) для пакета `booktabs`. Этот пакет для тех, кто любит везде наводить лоск.

### 10.3.1. К вопросу о разделительных линиях

По характеру оформления линейками таблицы бывают закрытые (глухие), полузакрытые, открытые. Два настоятельных совета:

- Никогда не следует использовать вертикальные линии в таблице.
- Двойные линии в оформлении таблицы также лишние.

Если данные в таблице настолько разные, что хочется их поделить вертикальной линией, то лучше сделать две таблицы. Но опять же никто не может запретить использовать разделительные линии.

Стиль `hhline` из коллекции `tools` определяет команду `\hhline`, которая позволяет управлять созданием двойной рамки, не создавая ненужных пересечений. Подробности в документации `hhline.pdf`.

Пакет `arydshln` необходимо в случае использования пунктирных разделительных линий. Документация к этому пакету `arydshln-man.pdf` более чем исчерпывающая.

### 10.3.2. Клетки

Для объединения рядом расположенных по горизонтали клеток можно воспользоваться командой `\multicolumn{n}{формат колонки}{текст}`:

```
\centering
\begin{tabular}{|*{3}{c|}}
\textbf{ИЛИ} & Истина & Ложь \\ \hline
Истина & \multicolumn{2}{c}{Истина} \\
Ложь & Истина & Ложь
\end{tabular}
```

<b>ИЛИ</b>	Истина	Ложь
Истина	Истина	
Ложь	Истина	Ложь



Первый обязательный параметр соответствует числу объединённых колонок, второй — формату получившейся объединённой колонки, третий — текст.

Для объединения клеток по вертикали можно воспользоваться пакетом **multirow** в котором определяется набор одноимённых команд:

```
\centering
\begin{tabular}{|*{3}{p{1.2cm}}|}
\textbf{ИЛИ} & Истина & Ложь \\ \hline \hline
Истина & \multirow{2}{1.2cm}{Истина} & Истина \\
Ложь & & Ложь
\end{tabular}
```

<b>ИЛИ</b>	Истина	Ложь
Истина	Истина	Истина
Ложь		Ложь

Подробнее об использовании команд пакета написано в README к нему. В пакете определены две команды с двумя и тремя обязательными аргументами:

```
\multirow {«число строк»}{«ширина»}{«текст»}
\multirow {«число строк»}*{«текст»}
```

В качестве первого аргумента передаётся число строк, которые займёт объединённая клетка, далее можно выбрать либо автоматическое вычисление ширины, или указать её самостоятельно. Число строк может быть отрицательным. В этом случае объединяются ячейки сверху от команды. Подобное может потребоваться, чтобы согласовать свою работу с пакетом **colortbl**:

```
\centering
\begin{tabular}{c>{\columncolor{yellow}}cc}
\backslashbox{два}{один} & Истина & Ложь \\
\rowcolor{yellow}
Истина & & Истина \\
Ложь & \multirow{-2}*{Истина} & Ложь
\end{tabular}
```

	один	Истина	Ложь
два		Истина	Истина
		Ложь	Ложь

Пакет **colortbl** предназначен для раскрашивания таблицы в разные цвета. В документации **colortbl.pdf** подробно излагаются принципы работы пакета.

Команда `\backslashbox{текст}{текст}`, делящая клетку на две части наклонной чертой, определена в пакете **slashbox**. Иногда так оформляют клетку на перекрестии боковика и головки. С пакетом идёт пример **slashbox.tex**.

Автор кириллических шрифтов **lh** Ольга Лапко для более изошрённой работы с таблицами предлагает пакет **makecell**. В это пакете определена команда `\makecell`, которая «создаёт окружение одноколоночной таблицы с предопределёнными общими параметрами выключки, интерлиньяжа и вертикальных отбивок вокруг. Её удобно использовать для многострочных ячеек. Дополнительный аргумент команды позволяет изменить выключку таблицы». В пакете документация на английском языке (**makecell.pdf**) дублируется русской документацией (**makecell-rus.tex**).

### 10.3.3. Выравнивание чисел

Таблица часто строится вокруг чисел. Поэтому не удивительно, что внимание к выравниванию чисел для целей упрощения восприятия данных должно быть повышенным.

Пакет **dcolumn** из коллекции **tools** добавляет ещё одну спецификацию к формату столбцов таблицы:

```
\centering
\begin{tabular}{|c|l|D{.}{,}{5}|}
0.3141 & 0.3141 & 0.3141 \\
3.141 & 3.141 & 3.141 \\
31.415 & 31.415 & 31.41 \\
\end{tabular}
```

0.3141	0.3141	0,3141
3.141	3.141	3,141
31.415	31.415	31,41

Новая спецификация имеет формат:  $D\{\text{delim}\}\{\text{output}\}\{\text{nfrac}\}$ , где *delim* — символ или набор символов по которому происходит выравнивание (обычно это точка или запятая), *output* — символ который замещает *delim* при компиляции (например, бывает нужно заменить точку на запятую), *nfrac* — максимальное число позиций в дробной части числа (при отрицательном значений число позиций не фиксируется). Подробности в документации **dcolumn.pdf**.

Пакет **rccol** обладает схожей функциональностью, что и **dcolumn**, но дополнительно позволяет округлять значения. К сожалению мне не удалось заставить его нормально работать если в качестве разделителя используется точка. **rccol** не смотря на информацию в документации **rccol.pdf** признаёт только запятую. Возможно, это недоразумение.

Пакет **warpcol** предоставляет общую процедуру формирования формата числовых колонок. В документации к пакету **warpcol.pdf** представлены примеры как добиться различных эффектов при выравнивании.

### 10.3.4. Доступ к данным

CSV (от англ. Comma Separated Values значения, разделённые запятыми) — это текстовый формат, предназначенный для представления табличных данных. Для доступа к этим данным напрямую можно воспользоваться пакетом **csvtools**. Документация к пакету **csvtools.pdf** достаточно подробна, но лучше к данным доступаться с помощью скрипта результатом действия которого является сам *tex*-файл.

### 10.3.5. Клоны tabular

**tabularx** из коллекции **tools** — расширение **tabular** на предмет автоматического вычисления ширины колонок, имеющих форматный определитель *X*.

```

\centering
\begin{tabularx}{\textwidth}{|D{.}{,}{4}|X|c|}
0.3141 & очень длинная строка & 0.3141 \\
3.141\footnote{В окружение tabularx можно
добавлять подстрочное примечание}
& 3.141 & 3.141
\end{tabularx}

```

0,3141	очень длинная строка	0.3141
3,141 <sup>a</sup>	3.141	3.141

---

<sup>a</sup>В окружение tabularx можно добавлять подстрочное примечание

В качестве первого аргумента окружения **tabularx** передаётся ширина таблицы. При компиляции  $X$  преобразуется в  $r\{\text{ширина}\}$ , где вместо ширины выставляется автоматическим образом вычисленная величина так чтобы в целом ширина таблицы оставалась неизменной. Если в таблице присутствует более одного столбца, имеющих формат  $X$ , то для формирования таблицы может потребоваться несколько проходов. Все подробности, как обычно, можно узнать в документации к пакету (`tabularx.pdf`).

Пакет **tabulary** так же является модификацией стиля **tabular**, точнее **array**. Пакет был создан для автоматического расчёта ширины колонок таблицы с целью минимизации высоты таблицы. Подробности в документации к пакету `tabulary.pdf`. Пользоваться одноимённым окружением следует с некоторой долей осторожности.

**ctable** альтернативный взгляд на оформление таблицы. Вместо окружения используется команда `\ctable`, которая объединяет в себе функциональность **tabular** и **table**. В дополнение этим особенностям в `\ctable` можно вставлять команды для создания подстрочных примечаний для таблицы. Примеры и документацию к пакету можно найти в файле `ctable.pdf`.

## 10.4. Многополосные таблицы

Окружение **tabular** и его производные всем хороши за исключением того, что они не могут занимать больше одной страницы. Для вёрстки на несколько страниц обычно используют одно из двух окружений: **supertabular** и **longtable** определённых в одноимённых стилях. Оба эти окружения обладают схожей функциональностью, но разными способами оформления. Оба предоставляют возможность создания стандартной шапки и стандартного окончания появляющихся в начале, на каждой новой странице и в конце таблицы. Подробности можно отыскать в документации к пакетам: `supertabular.pdf` для **supertabular** и `longtable.pdf` для **longtable** из коллекции **tools**.

**longtable** в отличие от **supertabular** гарантирует неизменность ширины столбцов на протяжении всей таблицы, что достигается за несколько проходов  $\text{\LaTeX}$ . Неизменность ширины столбцов в зависимости от ситуации может рассматриваться как преимущество, так и как недостаток.

В пакете **supertabular** кроме окружения **supertabular** определено окружение **mpsupertabular**, которое каждый отдельный кусок таблицы на своей странице

заключает в окружение **minipage**, что позволяет использовать подстрочные примечания прямо в таблице.

Более молодым и, возможно, более правильным является пакет **xTAB**. По сути дела это расширение **supertabular** с некоторыми улучшениями и исправлениями. Подробная документация доступна в файле `xTAB.pdf`

## 10.5. Вывод

Вывод — таблица без линеек или организованный в колонки и строки материал, который не разделён линейками. Оглавление `\tableofcontents`, списки иллюстраций `\listoffigures` и таблиц `\listoftables` тоже являются таблицами.

```
\centering
\begin{tabular}{p{0.7\textwidth}@{r}}
Глава 1\dotfill & 2 \\
Глава 2\dotfill & 10 \\
Глава 3\dotfill & 124
\end{tabular}
```

Глава 1 .....	2
Глава 2 .....	10
Глава 3 .....	124

Команда `\dotfill` формирует *отточия* (заполнение пространства точками).

Окружение **listliketab** из одноимённого пакета специализируется на создании таблиц, оформленных как перечисление. Это окружение будет очень кстати при оформлении вопросников:

```
\storestyleof{enumerate}
\begin{listliketab}
\newcounter{tabenum}\setcounter{tabenum}{0}
\newcommand{\nextnum}{\addtocounter{tabenum}{1}\thetabenum.}
\begin{tabular}{L>{\bf}l@{~или~}>{\bf}l@{~или~}>{\bf}l}
\nextnum & Красный & & зелёный & & голубой \\
\nextnum & Низкий & & средний & & высокий \\
\nextnum & Радостный & & грустный & & удивлённый
\end{tabular}
\end{listliketab}
```

1. **Красный**    или **зелёный**    или **голубой**
2. **Низкий**    или **средний**    или **высокий**
3. **Радостный** или **грустный**    или **удивлённый**

Документация с примерами находится в файле `listliketab.pdf`

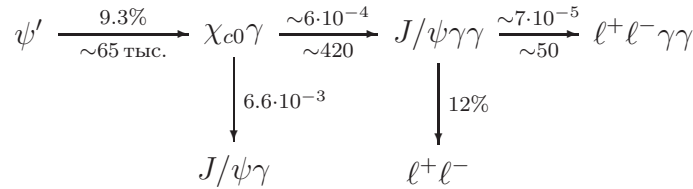
## 10.6. И это тоже таблицы?

Пакеты **tableaux** и **tabvar** созданы для исследования области определения и поведения функции. Пригодится при изучении или написании методички по началам матанализа. Молодцы французы!

$x$	$-\infty$	$-\sqrt[3]{2}$	$0$	$1$	$+\infty$
$f'(x)$	-		0	- 0 +	
$f(x)$	$+\infty$	$\searrow$	$0$	$\searrow$	$+\infty$
			$\searrow$	$+\infty$	$\searrow$
			$-\infty$	$\searrow$	$+\infty$
				$\frac{3}{2}$	$\searrow$
					$+\infty$

**tabvar** является более современной версией стиля **tableaux**. Поэтому подробности и примеры лучше всего искать в его документации **tabvar.pdf** и **demo.tex**.

Оформляется вовсе не как таблица, но что-то «табличное» в этом есть: пакет **pb-diagram** пригодится, когда нужно нарисовать простенькую диаграмму:



## 10.7. В заключение о таблицах

Таблица — это один из самых сложных текстовых элементов. Сложных не в том смысле, что её сложно оформлять — сложно добывать для неё подходящие данные. Очень легко сделать бессмысленную таблицу. Слишком много информации не оставит по прочтению никакого следа. Всегда надо думать как и что представлять, но хорошая таблица, стоит тысячи слов.

# Начала программирования

Когда придумываешь что-то сам, высок шанс ничего не придумать. Но когда живёшь чужим умом, уж точно ничего не придумаешь. Никогда не делай того, что делают другие. Это на 100% обрекает на неудачу.

Герш Ицкович Будкер.

Л<sup>A</sup>T<sub>E</sub>X позволяет не просто набирать текст — он позволяет его программировать, а, следовательно, перекладывать часть своей работы на компьютер. Привычка думать — одна из самых необычных особенностей разумного человека. Она позволяет экономить силы и время.

## 11.1. Создаём свои ...

... команды, окружения и прочее. Наверняка, возникшая в процессе набора, простенькая надоедливая проблема решена и не один раз. С другой стороны при нарастающей квалификации проще бывает изобрести этот велосипед заново в удобной на текущий момент форме:

```
\newcommand{\ee}{\ensuremath{e^{+}e^{-}}\xspace}
```

Часто новые команды создаются для комбинаций используемых исключительно в математическом окружении. Команда `\ensuremath` обеспечивает это окружение не зависимо от текущего режима:

`\(J/\psi\to\ee\)` является одним из подвидов `\ee`-рассеяния.

$J/\psi \rightarrow e^+e^-$  является одним из подвидов  $e^+e^-$ -рассеяния.

Команда `\xspace` из одноимённого пакета добавляет в конце команды пробел в случае, если за командой нет знаков препинания, то есть избавляет от необходимости самому вставлять явный пробел после команды.

Имеются три команды, которые позволяют создавать свои или переименовать уже имеющиеся макросы:

```
\newcommand{«команда»}[N][«зн. по ум.»]{«определение»}
\renewcommand{«команда»}[N][«зн. по ум.»]{«определение»}
\providecommand{«команда»}[N][«зн. по ум.»]{«определение»}
```

`\newcommand` определяет новую команду. Если такая команда уже была, то при компиляции генерится ошибка. `\renewcommand` напротив переопределяет уже существующую команду. В свою очередь `\providecommand` создаёт новую команду, если на момент описания такой команды не было, и ничего не делает, если она уже существовала.

В каждом из этих макросов есть два обязательных параметра — это имя команды и её описание. Если команде необходимо передать параметр/параметры, то первый необязательный аргумент должен принять значение от одного (1) до девяти (9). В разделе 3.4 обсуждалась команда для дублирования знака в формуле при переносе её на следующую строку (`\(a + b \hm{=} c\)`):

```
\newcommand*{\hm}[1]{#1\nobreak\discretionary}{%
  {\hbox{$\mathsurround=0pt #1$}}}
```

Вместо знака решётки (#) с цифрой после него при компиляции макроса подставляется соответствующий параметр. В данном случае параметр был только один и можно сказать, что его значение сохраняется в «переменной» #1.

Звёздочка (\*) в конце макроса `\newcommand` налагает на передаваемый параметр команды `\hm` дополнительное условие: в передаваемом тексте не должно быть пустых строк и команды `\par`. В некоторых случаях это упрощает отладку кода.

Наличие второго необязательного параметра в макросах определения новых команд позволяет первый параметр создаваемой команды определить как параметр по умолчанию:

```
\newcommand{\exmp1}[1][умолчанию]%
  {<<значение по #1>>}
Сравните \exmp1{} и \exmp1[требованию].
```

Сравните «значение по умолчанию» и «значение по требованию».

Для определения нового окружения используется команда `\newenvironment`, например:

```
\newenvironment{outlined}{\hrule\begin{center}}%
  {\end{center}\smallskip\hrule}
\begin{outlined}
  Выделенный текст.
\end{outlined}
```

Выделенный текст.

Формальное описание этой команды похоже на описание `\newcommand`:

```
\newenvironment {«окружение»} [N] [«зн. по ум.»] %
                {«код открывающий окружение»} %
                {«код закрывающий окружение»}
```

Точно так же, как и в случае `\newcommand`, созданному окружению можно передавать параметры. Подставлять параметры можно только в коде открывающему окружение. Кроме команды создания нового окружения можно так же переопределить уже имеющиеся с помощью аналогичной команды `\renewenvironment`.

В разделе 8.3, посвящённом описанию презентационного класса `beamer` упоминалось об ещё одной возможности создавать новые именованные окружения с помощью команды `\newtheorem`:

```
\newtheorem{Техmpl}{Пример}
\begin{Техmpl}[Теорема Пифагора]\label{th:1}
  Пифагоровы штаны во все стороны равны.
\end{Техmpl}

\begin{Техmpl}\label{th:2}
  Мудрость ограничена, а глупость бесконечна.
\end{Техmpl}

Можем сослаться первую теорему:~\ref{th:1},
а можно и на вторую:~\ref{th:2}
```

**Пример 1 (Теорема Пифагора).**  
*Пифагоровы штаны во все стороны равны.*

**Пример 2.** *Мудрость ограничена, а глупость бесконечна.*

Можем сослаться первую теорему: 1, а можно и на вторую: 2

Команда `\newtheorem` имеет две формы:

```
\newtheorem {«теорема»} [«существующая теорема»] {«заголовок»}
\newtheorem {«теорема»} {«заголовок»} [«имя счётчика»]
```

Каждая из форм имеет по два соответствующих обязательных аргумента и одному необязательному. В первом случае это имя уже существующей теоремы с которой следует иметь совместную нумерацию. Во втором случае в качестве необязательного параметра передаётся имя уже существующего счётчика на основе которого строится нумерация. О том, что такое счётчики и как их определять, речь пойдёт далее.

## 11.2. Счётчики и другие переменные

«Другие переменные» уже обсуждались в разделе 6.1 „Определённые «размеры» и переменные «длины»“. Операции с этими переменными выполнялись с помощью команд `\newlength`, `\setlength` и `\addtolength`. Аналогично в ЛАТ<sub>E</sub>X представлена и целочисленная арифметика с использованием счётчиков в качестве переменных:



```
\newcounter{MyCount}\setcounter{MyCount}{5}
Значение MyCount равно \arabic{MyCount},
или ~\alph{MyCount}, или \Asbuk{MyCount}.\par
\addtocounter{MyCount}{1550}
\arabic{MyCount} эквивалентно \Roman{MyCount}.
```

Значение MyCount равно 5,  
или е, или Д.  
1555 эквивалентно MDLV.

Новый счётчик создаётся с помощью команды `\newcounter`. При создании новый счётчик инициализируется нулём. Создание счётчика является глобальной операцией, то есть при компиляции информация о его создании не исчезнет, даже если новый счётчик был определён внутри окружения. Для присвоения счётчику другого значения используется команда `\setnewcounter`, а для изменения на какое-то определённое число — `\addtocounter`.

В отличие от длин, основная роль которых помнить размеры какого-то определённого блока, счётчики используются для отображения какой-либо структурной информации. Поэтому особое внимание уделяется написанию счётчиков. Чтобы просто отобразить численное значение счётчика с помощью арабских цифр используется команда `\arabic{счётчик}`. Для римской числовой нотации необходимо воспользоваться командой `\Roman` и `\roman` — заглавные и строчные буквы, соответственно. Счётчик может быть представлен так же буквой алфавита: `\alph` — латинская строчная, `\asbuk` — кириллическая строчная и `\asbuk` — кириллическая заглавная.

В стандартных классах уже определён набор счётчиков в которых хранятся номера страницы (счётчик `page`), раздела (соответственно, счётчики `part`, `chapter`, `section`, `subsection`, `subsubsection` и т. д.), подстрочного примечания (счётчик `footnote`), плавающих окружений (счётчики `figure` и `table`) и формул (`equation`). При создании счётчика также автоматически создаётся команда с префиксом `\the` перед именем счётчика. Вызов такой команды выводит номер счётчика. При выводе номера раздела, плавающего объекта, уравнения и тому подобного используются именно такого рода команды, поэтому, переопределив `\the`-команду, можно немного изменить стиль, например, следующая команда предписывает в дальнейшем маркировать все страницы в римском стиле:

```
\renewcommand{\thepage}{\Roman{page}}
```

На базе счётчиков можно организовывать иерархические структуры, то есть можно указывать зависимости:

```
\newcounter{Main}\addtocounter{Main}{10}
\newcounter{Dep}[Main]\addtocounter{Dep}{10}
Было: \theMain.\theDep\par
\stepcounter{Main}
Стало: \theMain.\theDep
```

Было: 10.10  
Стало: 11.0

При создании нового счётчика можно создать связь с уже существующим, указав имя существующего счётчика в качестве необязательного параметра. В примере

выше счётчик **Dep** зависит от счётчика **Main**. Эта связь проявляется в том, что если увеличить значение базового счётчика (**Main**) на единицу с помощью команды `\stepcounter`, то подчинённый счётчик (**Dep**) обнуляется. Обычно, новый счётчик устанавливаются в подчинение счётчикам разделов (**section**).

Команда `\refstepcounter{счётчик}` отличается от `\stepcounter`, тем, что помимо обнуления всех зависимых счётчиков, `\refstepcounter` определяет значение, выводимое командой ссылки `\ref`, как текст, создаваемый `\the`-командой:

```
% окружение "Задача"
\newcounter{Problem}[section]
\renewcommand{\theProblem}{\thesection.\arabic{Problem}}
\newenvironment{Problem}[0]{%
  \par\refstepcounter{Problem}%
  \theProblem\,}%
{\par}%
```

Здесь определено окружение **Problem** и одноимённый счётчик. Счётчик **Problem** зависит от счётчика раздела. Вывод счётчика `\theProblem` переопределён как номер раздела за которым следует уже сам счётчик. Внутри окружения счётчик **Problem** увеличивается на единицу с помощью команды `\refstepcounter{счётчик}`. Результат использования нового окружения представлен в следующем примере:

```
\begin{Problem}\label{ex:1}
  Задача раз
\end{Problem}
\begin{Problem}\label{ex:2}
  Задача два
\end{Problem}
Ссылки на раз~\ref{ex:1} и два~\ref{ex:2}.
```

```
11.2.1 Задача раз
11.2.2 Задача два
Ссылки на раз 11.2.1 и два 11.2.2.
```

При работе с переменными **Л<sup>A</sup>T<sub>E</sub>X** также могут помочь следующие пакеты:

- calc** — макропакет из коллекции **tools** для арифметических вычислений, уже упоминавшийся в разделе 6.1. Этот пакет переопределяет команды типа `\newcounter` так, что в них можно использовать арифметические выражения, хоть и с некоторыми ограничениями. Подробности в файле `calc.pdf`.
- ifthen** — макропакет в котором определена команды условного перехода `\ifthenelse` и цикла `\whiledo`. Подробности в файле `ifthen.pdf`. Так же можно присмотреться к усовершенствованной версии этого пакета **xifthen**.
- fmtcount** — представляет различные форматы (двоичный, восьмеричный, шестнадцатеричный и т. д.) отображения счётчиков (`fmtcount.pdf`).
- multido** — определяет оператор цикла `\multido` (`multido.pdf`).
- tokenizer** — позволяет разбивать тестовые списки на элементы (`tokenizer.pdf`).
- totpages** — даёт возможность узнать число страниц в документе и тому подобную информацию (`totpages.pdf`).

**xkeyval** — улучшенная версия пакета **keyval**, который позволяет передавать/принимать в качестве параметров пары значений «key=value» (`xkeyval.pdf`).

### 11.3. Создаём свой пакет

Предположим, что вы уже владеете искусством программирования в среде ЛАТЭХ. Для того чтобы распространить свои наработки следует организовать исходники в удобном для дальнейшей поддержки, передаче и установке виде. Хотя можно и не стараться, если вас не интересует результат.

*Внимание:* Политика создания названий команд в ТЭХподобной среде такова, что для новых пакетов необходимо придумывать новые команды. Это сделано для обеспечения абсолютной совместимости сверху вниз. К сожалению подобная политика в случае бездумного использования слов может привести «захватыванию» подходящих сочетаний<sup>1</sup>.

Знать как правильно устроен пакет полезно и новичку, так как эффективное обучение программированию напрямую связано с изучением уже существующего кода.

В ЛАТЭХ сообществе принято распространять свои пакеты и документацию к ним в виде автономных файлов с расширением `dtx` (`dtx`-файлы). Для автоматической установки пакетов используются инструкции, записанные в файлах с расширением `ins` (`ins`-файлы). Для более подробной информации следует обратиться к инструкции «How to Package Your ЛАТЭХ Package», созданной Скотом Пакиным (Scott Pakin). Файл `dtxtut.pdf`, как обычно, можно найти в стандартной поставке ЛАТЭХ или на CTAN. Вместе с документацией идут файлы примеров `[c]skeleton.dtx` и `[c]skeleton.ins`.

За работу с `dtx`-файлами отвечает пакет **doc** и сопутствующая ему утилита DOCSTRIP (файл `docstrip.pdf`). Основная идея пакета **doc** состоит в совмещении кода с документацией, что облегчает поддержку и развитие пакета.

#### 11.3.1. Установочный `ins`-файл

Для извлечения кода и документации из `dtx`-пакета следует написать специальный установочный файл. Набор инструкций достаточно стереотипен:

```
% Стандартный копирайт по выбору (рекомендуется LPPL/GPL)
%%
%% Первый шаг — загрузка DOCSTRIP.
\input docstrip.tex
%% Подробный отчёт о каждом шаге хорош только когда пакет
%%отлаживается.
\keepsilent
```

<sup>1</sup>Примером может служить пакет **listings**, где вместо подходящего по названию окружения **listing** используется **lstlisting**.

```

%% Директория в которую устанавливается пакет. Имя
%%директории является относительным по отношению к
%%базовой директории $(ТЕХМФ).
\usedir{tex/latex/{«имя пакета»}}
%% Определение преамбулы, которая вставляется во все
%%сгенерированные файлы. Обычно, это информация об авторе
%%и пожелания пользователям.
\preamble
  Текст преамбулы
\endpreamble
%% Извлечение файлов пакета из dtx. Основной шаг,
%%который может повторяться несколько раз.
\generate{\file{«извлекаемый файл»}\from{«dtx-файл»}{метка}}
  ...
  ...
%% Информация для пользователя. Всегда что-то полезно сказать
%%после установки.
\obeyspaces
\Msg{*****}
\Msg{*      Здорово, что вы поставили этот пакет.      *}
\Msg{*      Прочитайте документацию перед использованием!      *}
\Msg{*****}
%% Метка конца установочного файла.
\endbatchfile

```

### 11.3.2. Пакетный dtx-файл

Пакетный dtx-файл содержит в себе и код с комментариями, и текст описания пакета. Структура dtx-файла после прогона через **latex** позволяет получить печатную документацию. Код с комментариями тоже может стать частью документации. Это шаг по направлению к «грамотному программированию» (*literate programming*).

Наличие комментариев в коде заставляет для получения результата повторять процедуру компиляции дважды. Первый раз отрабатывается ЛАТЭХ-код, а затем комментарии. Во втором случае знак % перед комментарием игнорируется и текст комментария передаётся на вход ЛАТЭХ, если он (комментарий) не окружён командами `\iffalse-fi`.

#### Пролог

В начале следует, естественно, добавить информацию об авторе:

```

%\iffalse meta-comment
% Этот текст не обрабатывается ЛАТЭХ'ом. Слово meta-comment
%добавлено просто для удобства чтения кода человеком и

```

```
%означает, что этот текст предназначен именно для него (человека).
%\ fi
```

В ins-файле в команде `\generate` использовался параметр «метка». Это говорит DocStrip, что следует выбирать строки, которые следуют за комментарием и конструкцией `<метка>` или между тегами `<*метка>` и `</метка>`. Далее идёт код заголовка пакета соответствующего метке «метка»:

```
% \iffalse
%<метка> \NeedsTeXFormat{LaTeX2e}
%<метка> \ProvidesPackage{«имя пакета»}
%<метка> [ <ГГГГ>/<ММ>/<ДД> v<версия> <краткое описание> ]
```

Строчку «`<ГГГГ>/<ММ>/<ДД> v<версия> <краткое описание>`» нужно заменить на дату, версию и краткое описание, соответственно.

Закончить пролог необходимо следующими словами, создающими основную документацию:

```
%<*driver>
\documentclass{ltxdoc}
\usepackage{«имя пакета»}
\begin{document}
\DocInput{«dtx-файл»}
\end{document}
%</driver>
%\ fi
```

Это единственная часть относящаяся к документации, которая не начинается со знака комментария (%).

В прологе можно указывать ещё некоторое количество инструкций уточняющих формат создаваемой документации.

## Пользовательская документация

Прежде всего следует учесть, что подавляющий объём описаний для пакетов  $\LaTeX$  сделан на английском языке. Для этого есть довольно веские основания, связанные с размером англоязычной аудиторией.

Написание документации для dtx-пакета ничем не отличается от написания обычного  $\LaTeX$ -документа за исключением, что не следует забывать о знаке комментария (%) в начале строки

```
% \title{Пакет \textsf{«имя пакета»}}
% \author{«Ваше имя» \\\ \texttt{«Ваш e-mail»}}
% \maketitle
% текст документации
```

К стандартным  $\LaTeX$ -командам секционирования уровня **paragraph** добавляются `\DescribeMacro{макрос}` и `\DescribeEnv{окружение}`

## Код с комментариями

Очевидно, что лучшая документация для программиста это сам код, но для нормального человека описательный текст предпочтительнее. Проблема совмещения кода и описания является основной причиной возникновения «грамотного программирования» (literate programming).

После окончания пользовательской документации идёт сам код:

```
%\StopEventually{\PrintIndex}
%\begin{environment}{«имя окружения»}
% Описание окружения.
% Аналогично, существует окружение macros, для описания новых команд.
%_..._ \begin{macrocode}
Здесь идёт код, вида:
\newenvironment{«имя окружения»}{начало}{окончание}
%_..._ \end{macrocode}
%\end{environment}

...

%\Finale
\endinput
```

Команда `\StopEventually{}` отмечает начало кода и принимает в качестве параметра команду, которую следует выполнить в конце документации, например, распечатать алфавитный указатель `\PrintIndex`.

Любой код следует окружать с помощью окружения **macrocode**. Это позволит включить его в печатную документацию. Есть две особенности, для этого окружения, которые следует учитывать:

- между `%` и `\begin{macrocode}` должно быть ровно четыре (4) пробела. Аналогичное правило действует и для `\end{macrocode}`,
- внутри этого окружения не должно быть текста, начинающегося с `%`.

Внутри окружений **environment** и **macros** может быть несколько вставок кода и текста.

### 11.3.3. Пакетирование

Часто L<sup>A</sup>T<sub>E</sub>X-пакеты распространяются в виде одного dtx-файла. Существует способ включить установочный ins-файл в файл пакета:

```
%<*batchfile>
\begingroup

Содержание ins-файла
```

```
\endgroup
%</batchfile>
```

Следует только убрать заключительную команду `\endbatchfile`, для того чтобы  $\text{\LaTeX}$  мог скомпилировать всё остальное.

Всё. Теперь для распространения свой пакет лучше всего поместить на CTAN. Для загрузки следует обратиться к ресурсу <http://www.ctan.org/upload>. Всегда необходим краткий README с описанием. Собранный документация в виде pdf-файла так же является хорошим тоном.

## 11.4. Напутствие

Документируйте каждый шаг. Пишите как можно больше качественного текста, так как его мало не бывает. Живучесть программы определяется не только кодом, но и описанием. «Светлое будущее» за грамотным программированием (literate programming).

Вот и закончился  $\text{\LaTeX}$ -цикл в журнале. Честно говоря, я сам за это время узнал много чего нового для себя. Надеюсь мне удалось поделиться этими знаниями с читателями. В этой информации нет никакой чёрной магии — всё просто и логично. Эта информация полезна, так как позволяет автоматизировать одно из самых сложных ремёсел человеческой цивилизации — создание книг. Пишите тексты большие и маленькие — они не пропадут.

# Литература

- [1] *Кнут, Д. Э.* Всё про T<sub>E</sub>X / Д. Э. Кнут. — М.: Вильямс, 2003. — 560 с.
- [2] *Кнут, Д. Э.* Всё про МЕТАФОНТ / Д. Э. Кнут. — М.: Вильямс, 2003. — 384 с.
- [3] *Кнут, Д. Э.* Компьютерная типография / Д. Э. Кнут. — М.: Мир, 2003. — 686 с.
- [4] *Грэтцер, Г.* Первые шаги в L<sup>A</sup>T<sub>E</sub>X / Г. Грэтцер. — М.: Мир, 2000. — 172 с.
- [5] *Гуссенс, М.* Путеводитель по пакету L<sup>A</sup>T<sub>E</sub>X и его расширению L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> / М. Гуссенс, Ф. Миттельбах, А. Самарин. — М.: Мир, 1999. — 606 с.
- [6] *Гуссенс, М.* Путеводитель по пакету L<sup>A</sup>T<sub>E</sub>X и его графическим расширениям / М. Гуссенс, С. Ратц, Ф. Миттельбах. — М.: Мир, 2002. — 621 с.
- [7] *Гуссенс, М.* Путеводитель по пакету L<sup>A</sup>T<sub>E</sub>X и его Web-приложениям / М. Гуссенс, С. Ратц. — М.: Мир, 2001. — 604 с.
- [8] *Балдин, Е. М.* Компьютерная типография L<sup>A</sup>T<sub>E</sub>X / Е. М. Балдин. — СПб.: БХВ-Петербург, 2008. — 304 с.
- [9] *Львовский, С. М.* Набор и вёрстка в системе L<sup>A</sup>T<sub>E</sub>X / С. М. Львовский. — М.: МЦНМО, 2003. — 448 с.
- [10] *Котельников, И. А.* L<sup>A</sup>T<sub>E</sub>X по-русски / И. А. Котельников, П. З. Чеботаев. — Новосибирск: Сибирский Хронограф, 2004. — 496 с.
- [11] *Роженко, А. И.* Искусство верстки в L<sup>A</sup>T<sub>E</sub>X'e / А. И. Роженко; Под ред. А. С. Алексеева. — Новосибирск: Изд. ИВМиМГ СО РАН, 2005. — 398 с.
- [12] *Мильчин, А. Э.* Издательский словарь-справочник / А. Э. Мильчин. — М.: ОЛМА-Пресс, 2003. — 500 с.