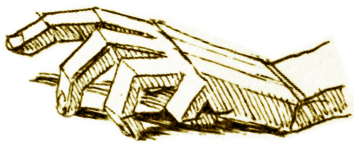
И. В. Романовский

КАК ГОТОВИТЬ ТЕКСТЫ

для системы

TEX



По эскизу Ганса Гольбейна, мл., 1540-е гг.

С.-Петербург

2013

1 Введение

Это не учебник и не справочник. Мне приходилось много самому разбираться в системе TEX и еще больше консультировать других. При этом обычно нет ни времени ни, к сожалению, желания превращать консультируемого в специалиста, а вопросы иногда бывают достаточно сложными, так что нужно выбирать какой-то компромисс. В конце концов, я стал собирать и систематизировать ответы на вопросы, не только по TEX-у, но и по полиграфии. Когда текст стал увеличиваться, захотелось, чтобы были красивые примеры; и я не всегда мог преодолеть свои литературные симпатии (и антипатии).

С 1996 г. я к этому тексту не притрагивался, а просто положил его на свою страницу. Несколько человек сказали мне, что они по этому тексту получили начальные сведения по TEX-у и текст показался им удобным. Кроме того, оказалось, что я использовал фрагменты этого текста при чтении курса по компьютерной полиграфии.

Поэтому я решил пересмотреть текст, убрав многочисленное «старье» и многое добавив. Читателю остается принять это *as is* и, если угодно, придти с дополнительными вопросами.

* * *

Система TEX предназначена для полиграфической подготовки сложных текстов. Она сама очень сложна и состоит из большого числа компонентов, которые имеются в разных вариантах. Нам целесообразно начать с несколько упрощенной структуры системы.

На уровне программ она состоит из двух частей — основного процессора `tex.exe` и драйверов `dviXX.exe`, где вместо `XX` подставляется что-нибудь конкретное в зависимости от выполняемого действия.

Основной процессор `tex.exe` осуществляет набор текста. Обычно этот текст задается одним или несколькими текстовыми файлами, как принято говорить, ASCII-файлами, с расширением `.tex` . Результатом работы является файл набранного текста с расширением `.dvi` (от DeVice Independent — независящий от устройства).

Есть много разновидностей основного процессора. Наиболее известен процессор `MikTEX` для персональных компьютеров типа `IBM PC`.

При обработке вашего текста процессор использует обычно некоторые заготовки, сильно облегчающие вашу и его работу. Этот набор необходимых заготовок содержится в *стилевом файле*, который может быть подготовлен многими способами. Наиболее известны следующие три стандартных варианта:

plain — Основной формат, разработанный автором системы Дональдом Кнутом.

amsplain — Формат Американского Математического общества, содержащий богатый выбор шрифтов, обозначений и конструкций, необходи-

мых для набора математических текстов. Сейчас он почти не используется по причинам. о которых будет сказано дальше.

lplain — Так называемый LaTeX, формат, разработанный Лесли Лампортом и являющийся наиболее распространенным при обычных использованиях. Мы будем следовать в основном LaTeX-у.

Основной процессор может различаться также по поддерживаемой им версии системы TeX. Сейчас система дошла до версии 3.1415926 (март 2008). Важно знать, что есть разные версии и у Латеха. Мы следуем версии 2ε, часто обозначаемой 2e.

Драйверы предназначены для вывода получившегося `.dvi`-файла на конкретное устройство. Упомянем следующие разновидности драйверов

dviscr выводит на дисплей компьютера. Эта программа часто называется *превьюером*.

dvidot выводит на точечный принтер, т. е. на наиболее распространенный у нас[1] матричный 9-точечный или 24-точечный принтер.

dvihplj выводит на лазерный принтер, наиболее характерным представителем которых считается принтер Laser Jet фирмы Hewlett–Packard (это и объясняет аббревиатуру в названии). Струйные принтеры также обслуживаются этим драйвером.

dvips выводит в языке PostScript, — это специальный язык программирования вывода на лазерный принтер и другие выводные устройства высокого разрешения. В некоторых лазерных принтерах и наборных устройствах имеется встроенный интерпретатор языка PostScript или возможность присоединения специального *картриджа*. На персональных компьютерах встречаются интерпретирующие программы, среди которых наиболее известен и распространен GhostScript. [2]

dvipdfm переводит файл в широко распространенный формат `pdf` — *portable document format* фирмы Adobe, которая обеспечивает свободный доступ к программе. читающей документы в этом формате.

2 Простейший набор текста

Текст нужно готовить каким-либо *самым простым* текстовым редактором, т. е. таким, который не вносит в текст специальных символов, например,

[1] Это было написано 15 лет назад. Сейчас такие принтеры встретишь редко, но они существуют благодаря их наднжности и дешевизне экспиатации.

[2] Посетив несколько американских университетов, я не увидел других способов вывода кроме PostScript-а. При этом вместо драйвера dviscr используется программа GhostView — специальная оболочка для GhostScript. Сам я начал активно пользоваться этим драйвером после того, как мне потребовалось вставлять в текст рисунки, выполненные в ПостСкрипте. Это оказалось очень удобно. Сейчас на персональных компьютерах используется интерпретатор GS с оболочкой GSview. Удобно также иногда рисовать на постскрипте символы, и использовать их в наборе. Я это позднее покажу.

разметки текста. Особенно опасен Word. Я сейчас очень доволен редактором Notepad++, в котором предусмотрена специальная подцветка конструкций TEX-а. Но многие хвалят специальную «теховскую оболочку», в которой есть подходящий текстовый редактор.

Набирая текст, не делайте переносов и не выравнивайте край. Все ваши попытки сделать *красивую машинопись*, имитируя привычки докомпьютерного делопроизводства, игнорируются программой, она имеет гораздо более изощренные возможности выравнивания текста, а слово, разбитое переносом, трактуется ею как два слова. Лучше уделите больше внимания тому, чтобы после каждого знака препинания был пробел, так как если пробел отсутствует,соседние слова соединяются в длинные конструкции,неразъединяемые при обработке.[3] Печально богатый опыт моих коллег заставляет добавить: не делайте пробела ПЕРЕД знаком препинания. Не смешивайте в одном слове *k*ириллицу и л**а**тиницу.

Абзац от абзаца отделяется хотя бы одной пустой строкой (это очень экономно, — пустая строка занимает в файле всего два символа).

Вот я пропустил пустую строчку, и начался новый абзац (английский термин — *paragraph*). Пустую строчку может заменить команда `\par`, в некоторых случяях такая замена необходима.

Видите, некоторые слова набраны наклонным шрифтом, который полиграфисты называют *курсивом*. Соответствующий английский термин — *italics*, этот шрифт был создан в XVI веке итальянским типографом Альдом Манунцием, который по преданию имитировал почерк Ф. Петрарки.

Для перехода на курсив нужно вставить в текст *контрольную последовательность* `\it` (от *italics*). Эта установка шрифта будет действовать до ее отмены. Отменой может быть установка другого шрифта аналогичной командой или выход из области действия установки.

Область действия устанавливается фигурными скобками { и }, — если вы заключите какую-то часть текста в эти скобки, все установки шрифта внутри скобок только внутри и будут действовать. Можно так же менять и размер шрифта. Фигурные скобки используются для выделения области действия очень широко, — есть только одно исключение, о котором будет сказано много позже.

Приведем теперь таблицу основных шрифтов

Код	Название	Демонстрация и пояснение
`\rm`	roman	Основной шрифт, антиква, — Computer Roman
`\bf`	**boldface**	**Полужирный вариант Computer Roman**
`\sl`	*slanted*	*Наклонный вариант Computer Roman*
`\sc`	SMALL CAPS	Заголовочный вариант Computer Roman
`\it`	*italics*	*Курсивный Computer Roman*
`\sf`	sansserif	Шрифт без засечек (without serifs)
`\tt`	`teletype`	`Машинописный шрифт`

[3] Надеюсь, что вы уже понимаете могучую силу ПЛОХИХ ПРИМЕРОВ, поэтому я не пишу, что эти склейки слов я вставил в текст нарочно.

Первоначально установлена антиква `\rm`.

Шрифт может быть разного размера. Размер шрифта измеряется в *пунктах* (пункт — 1/72 часть дюйма, но дюймы в разных культурах чуть-чуть различаются). Есть старинные традиции использования шрифтов и их названий, приведем некоторые из них

Название	Размер	Демонстрация и пояснение
Перл	5	Самый мелкий шрифт математических книг
Миньон	7	Мелкий шрифт подписей
Петит	8	То, что мы называем мелким шрифтом
Боргес	9	Шрифт газет
Корпус	10	Основной шрифт обычных книг
Цицеро	12	Шрифт детских учебников
Миттель	14	Заголовочный шрифт
—	17	Заголовочный
Текст	20	Титульный
—	25	Титул

Каждый шрифт каждого размера (есть термин *шрифторазмер*) имеет свое имя (точнее, два имени, первое — это, грубо говоря, имя файла, а второе — имя, данное ему в системе, мы будем говорить только о втором имени). Например, латиница без засечек кегля 14 имеет имя `cmrssxiv`, а телетайпный шрифт кегля 8 — имя `cmttviii`.

Однако, можно этих имен не знать и ими не пользоваться. При начале работы какой-либо размер устанавливается в качестве базового (по умолчанию это кегль 10, но можно выбрать 11 или 12, а теперь распространился и 14 — это шрифт диссертаций, совсем не потому, что диссертанты менее грамотны чем младшие школьники). По этому основному кеглю устанавливаются несколько вспомогательных шрифтов с удобными условными обозначениями в соответствии со следующей таблицей

Название	10	11	12
`\tiny`	5	6	7
`\scriptsize`	7	8	9
`\footnotesize`	8	9	10
`\small`	9	10	11
`\normalsize`	10	11	12
`\large`	12	12	14
`\Large`	14	14	17
`\LARGE`	17	17	20
`\huge`	20	20	25
`\Huge`	25	25	25

Учтите, что при обычном использовании этих команд установки размера шрифт автоматически переустанавливается в латинскую антикву. [4] Как

[4] Я говорю «обычно», т. к. сейчас используется новая система управления шрифтами,

вы уже видели, некоторые символы оказываются «изъятыми» из обычного обихода. Это знаки открывающей и закрывающей фигурной скобки, знак обратной косой черты и знак процента. Знак процента служит для вписывания в текст комментариев, — все, что идет от знака процент до конца строки, системой пропускается. Не пренебрегайте этой возможностью, удобно вставлять в текст ссылки на страницы оригинальной рукописи, спорные вопросы и недоделки, варианты набора текста и проч.[5]

Дальше будут видны причины, по которым из обращения изъят знак доллара (для набора математического текста). Для включения этих знаков в текст используются специальные контрольные последовательности:

`\{` {	`\}` }	`\backslash` \	`\%` %
`\$` $			

Некоторые символы имеют специальные начертания. Например, нужно различать разные варианты горизонтальной черточки — *дефис* и *тире* (о минусе речь пойдет дальше).

Знак	Название	Запись	Использование	Пример
-	дефис	-	Переносы, сост. слова	по-нашему
–	кор. тире	--	Числовые интервалы	32–54
—	норм. тире	---	Как тире	— Что это ?

В иностранных языках часто используются всякого рода надстрочные и подстрочные знаки, а также составные символы — *лигатуры*.

так наз. NFSS — New Font Selection Scheme, в которой каждая характеристика шрифта переключается независимо. Об этом речь пойдет значительно позднее.

[5] Правда, эта возможность приводит к неприятностям, если вы используете текстовый редактор, который сам меняет разбиение абзаца на строки.

Знак	Название	Запись	Пример
å	скандинавское a с кружком	`\aa`	ångstrem
ø	скандинавское перечеркнутое o	`\o`	Gøteborg
æ	французская лигатура $a+e$	`\ae`	Ælia
œ	французская лигатура $o+e$	`\oe`	Nœlle
ł	польское эль	`\l`	Połska
ß	немецкое эс-цет	`\ss`	Gauß
o̧	цедилла	`\c{o}`	Ça ira
ò	аксанф грав	`\‘{o}`	manière
ó	аксанф дегю	`\’{o}`	révérence
ô	сиркумфлекс	`\^{o}`	tête
ö	умляут	`\"{o}`	über
õ	тильда	`\~{o}`	señor
ō	надчеркивание	`\=o`	T̄ēs̄t̄
ȯ	верхняя точка	`\.{o}`	Ṫėṡt̄
ŏ	бреве	`\u{o}`	Ťest
ǒ	птичка	`\v{o}`	Holčik
ő	венгерский умляут	`\H{o}`	művelet
o͡	связка	`\t{o}`	bri͡ukva
o̩	нижняя точка	`\d{o}`	b̩y̩t̩e̩
o̲	подчеркивание	`\b{o}`	b̲y̲t̲e̲

Теперь *о переносах*. Места возможных переносов внутри слова определяются специальной программой по сложно устроенным таблицам переносов. Но никакая автоматика не может дать абсолютно приемлемого результата: иногда нужно избегать неблагозвучных сочетаний или ухудшать переносы для увеличения возможностей формирования абзаца. Кроме того, система «отказывается» разбивать слова, в которые вставлены небуквы (например, ударения), и приходится прибегать к *ручному управлению*. Команда `\-` служит для задания возможных переносов слова, например: `макро\-вставка`. [6]

Для кавычек в исходном варианте используются двойные апострофы ‘‘ и ’’, результат применения которых не соответствует традициям русской полиграфии. Знаки << и >> используются при наборе кириллицы для кавычек-«елочек». Эти знаки имеют ASCII-коды 244 и 245 соответственно, в модифицированной альтернативной кодировке это верхняя и нижняя половинки интеграла. Попутно введем собственное обозначение для знака номера и (стандартное) для знака параграфа

Знак	Название	Запись	Пример
§	параграф	`\S`	§ 5
№	знак номера	`\No`	№ 5
“	откр. кавычка	‘‘	“Empty” sky
”	закр. кавычка	’’	“Empty” sky
«	откр. кавычка	<<	«Пустое» небо
»	закр. кавычка	>>	«Пустое» небо

[6] Учтите, что уже упоминавшееся отсутствие пробела после знака препинания также делает склеенное “слово” сложным и непереносимым.

3 Конструкции размещения текста

Управление абзацным отступом. В некоторых случаях (например, после перечисления условий в теореме) хочется, чтобы текст, начатый с новой строки, шел без отступа. В этом случае может использоваться команда `\noindent`. Например:

```
Перед словом Перед вставлено \noindent.
```

Перед словом Перед вставлено \noindent.

Размер абзацного отступа может быть также изменен. Способ изменения требует разговора о том, в каких единицах задаются длины. Поэтому об этот речь пойдет позже.

Примечание. Для того, чтобы сделать примечание к тексту, нужно воспользоваться командой `\footnote{...}`, поместив внутри скобок текст примечания. Не беспокойтесь относительно нумерации, примечания нумеруются автоматически.[7]

Принудительный переход на новую строчку вызывается командой `\linebreak`. Команда `\newline` или `\\` служит для перехода на новую строчку без выключки, т. е. без выравнинания правого края строки по уровню полей за счет изменения межсловных пробелов. Сопоставьте окончания строчек с помощью `\linebreak`:

```
Шел по улице отряд —
Сорок мальчиков подряд:
Раз, два, три, четыре, и четыре на четыре,
И четырежды четыре и потом еще четыре.
```

и с помощью `\\`:

```
Шел по улице отряд —
Сорок мальчиков подряд:
Раз, два, три, четыре, и четыре на четыре,
И четырежды четыре и потом еще четыре.
```

Д. Хармс

Команда `\obeylines` позволяет пользоваться символом перехода на следующую строчку из исходного файла в качестве команды перехода на следующую строчку для набираемого текста, т. е. воспроизводить в наборе разбиение на строки из исходного файла. Область действия этой команды, как обычно, можно задать фигурными скобками.

Обстановки. Имеется специальная конструкция `environment`, которая используется для формирования особенных режимов обработки и расположения текста (*environment* — обстановка, окружение, среда). Каждая конкретная обстановка (возьмем для примера обстановку `verse`, предназначенную для набора стихов) заключается в специальные скобки `\begin{verse}` и `\end{verse}`. Например,

```
\begin{verse}
А для низкой жизни были числа,\\
```

[7] Хотя добиться, чтобы на каждой странице нумерация начиналась сначала, довольно трудно. Во всяком случае, не с этого надо начинать!

```
Как домашний, подъяремный скот,\\
Потому что все оттенки смысла\\ Умное число передает.\\
\makebox[70mm][r]{\it Л.~Мартынов}
\end{verse}
```

дает такой результат

> А для низкой жизни были числа,
> Как домашний, подъяремный скот,
> Потому что все оттенки смысла
> Умное число передает.
>
> *Л. Мартынов*

Надо, конечно, объяснить конструкцию `\makebox[70mm][r]{\it Л.~Мартынов}`. Для подписи сделан особый *ящик* длины 70 mm, в котором расположена справа (что определяется параметром `r`) текст, набранный курсивом.

Близкая по действию обстановка `quotation` служит для набора цитат

> Никогда не воображай себя иным, чем это может показаться другим, чтобы то, чем ты был или мог быть, было ничем иным, как то, чем ты был или мог показаться другим, будучи иным.
>
> *Л. Кэррол, Алиса в Зазеркалье*

Вот еще несколько обстановок, управляющих размещением текста

Обстановка `center` служит
для центровки каждой строки текста,
размещенного внутри.

Обстановка `flushright` служит
для сдвига текста,
размещенного внутри, к правому полю формата.

Обстановка `flushleft` служит
для сдвига текста,
размещенного внутри, к левому полю формата.

Упомянем еще обстановку `verbatim`, которая воспроизводит буквально весь текст между открывающей и закрывающей скобками. Именно с ее помощью показано, как набран отрывок из стихотворения Л. Мартынова.

Наряду с «обстановкой» `verbatim` существует команда `\verb` также предназначенная для буквального воспроизведения исходного текста. Для того чтобы отграничить параметр этой команды от окружающего текста принимается следующее соглашение: первый знак после названия команды принимается в качестве ограничителя параметра, его повторное появление служит концом параметра. Например, в тексте `\verb+Example @#+` параметром является текст между плюсами, а сам этот текст набран так `\verb|\verb+Example @#+|`.

Перечисления. Эта обстановка, имеющая несколько разновидностей, чрезвычайно важна, и мы посвятим ей отдельный параграф, а сейчас только назовем им. Есть три варианта перечислительной обстановки:

itemize Для перечисления в форме отдельных «пунктов», каждый из которых сопровождается особой меткой, выбранной вами или назначенной по умолчанию («пуля» • на первом уровне и минус — на втором и последующих).

enumerate Для перечисления в форме отдельных нумерованных «пунктов».

description Для перечисления в форме отдельных «пунктов», каждый из которых имеет отрицательный абзацный отступ.

Само это перечисление набрано в обстановке `description`. Важный частный случай перечислительной обстановки — библиография. О ней также речь пойдет позже.

Смена страницы. Аналогично командам перехода на новую строку вводятся команды перехода на новую страницу `\pagebreak` и `\newpage`. Первая из них стремится расположить текст на прерванной странице по возможности равномерно, вторая оставляет конец страницы незаполненным.

4 Формулы

Удобство набора сложных математических формул — одно из крупнейших преимуществ TEX-а. Многие особенности математического набора, о которых автор и читатель часто не знают или только догадываются, TEX поддерживает автоматически. Например, набор букв в формулах курсивом, а скобок и цифр — прямым шрифтом, смену кегля при наборе индексов и дробей, небольшую отбивку вокруг знаков отношений и арифметических действий, центровку числителя и знаменателя дроби по вертикальной черте, выбор размера этой черты, выбор размеров скобок и приставных знаков.

Математические формулы могут набираться двумя способами *в подборку* (английский термин *in the text mode*) и *выключенные*, т. е. помещенные в отдельную строку (английский термин *in the displayed mode*). Сравните, пожалуйста, одну и ту же формулу, набранную в подборку: $\sum_k \beta^k$ и выключенную:

$$\sum_k \beta^k.$$

Формула, набираемая в подборку, должна быть заключена с обеих сторон в знаки доллара: `$\sum_k\beta^k$`, выключаемая формула набирается в точности так же, но со сдвоенными знаками доллара: `$$\sum_k\beta^k$$`. Знак препинания после выключаемой формулы должен находиться внутри этого окаймления, т. к. иначе он появится как отдельный знак в начале следующей строки.

Обычно то, что может быть набрано одним символом имеющегося на клавиатире набора, — круглые и квадратные скобки, знаки плюс и минус (здесь соглашения про три разновидности тире не действуют), — так и набирается, но потребности математического набора значительно шире. Для многочисленных специальных знаков используются соответствующие контрольные последовательности, которые как правило выбраны очень удобно. Например, греческие буквы представляются просто их именами:[8]

\alpha	α	\eta	η	\nu	ν	\tau	τ
\beta	β	\theta	θ	\xi	ξ	\upsilon	υ
\gamma	γ	\iota	ι	o	o	\phi	ϕ
\delta	δ	\kappa	κ	\pi	π	\chi	χ
\epsilon	ϵ	\lambda	λ	\rho	ρ	\psi	ψ
\zeta	ζ	\mu	μ	\sigma	σ	\omega	ω

и аналогично для больших греческих букв (\Delta — Δ, \Xi — Ξ; к сожалению, не для всех, а только для букв, не имеющих латинских замен. Строчная *омикрон*, как видно из таблицы, также заменяется латинской курсивной). Некоторые греческие буквы имеют варианты начертаний, ко многим из них мы приучены русской полиграфией. Приведем эти варианты (при этом символы нижнего ряда требуют подключения специального пакета amssymb)

\varepsilon	ε	\varphi	φ	\vattheta	ϑ
\varkappa	$\varkappa$	\varrho	ϱ	\varsigma	ς

Знаки отношений и операций:

<	$<$	>	$>$	\leq	$\leq$	\geq	$\geq$
\neq	$\neq$	\equiv	$\equiv$	\approx	$\approx$	\times	$\times$
\in	$\in$	\subset	$\subset$	\cap	$\cap$	\cup	$\cup$
\to	$\to$	\cdot	$\cdot$	\setminus	$\setminus$	\circ	$\circ$
\pm	$\pm$	\mp	$\mp$	\forall	$\forall$	\exists	$\exists$
\div	$\div$	\wedge	$\wedge$	\vee	$\vee$	\neg	$\neg$

Знак \not перечеркивает любой следующий за ним символ. Например, a\not\to 0 дает $a \not\to 0$.

Обозначения для специальных функций принято набирать прямым шрифтом в отличие от курсивного шрифта для переменных. Обычно для таких стандартных обозначений определены *естественные* контрольные последовательности \sin, \exp, и т.д. Нужно только предупредить, что естественное для американских математиков обозначение тангенса — это \tan (и аналогично котангенса и арктангенса). Отсутствующие специальные обозначения вы можете определить самостоятельно, но об этом позднее!

Приведем еще обозначения для некоторых специальных значков, изобретенных математиками и/или взятых из других языков:

[8] Это представление используется только для формул. Для "настоящего" греческого текста есть специальные шрифты.

`\infty`	∞	бесконечность
`\partial`	∂	частная производная
`\aleph`	$\aleph$	алеф
`\nabla`	∇	набла
`\angle`	$\angle$	угол
`\langle`	$\langle$	левая угловая скобка
`\rangle`	$\rangle$	правая угловая скобка
`\emptyset`	$\emptyset$	пустое множество
`|`	$\|$	вертикальная черта для модуля
`\\|`	$\|\|$	две вертикальные черты для нормы

Конечно же, математических знаков запасено много больше и в основном TEXe и, особенно, в коллекции Американского Математического общества. Предусмотрены и другие алфавиты — готический, рукописный, греческий полужирный, с двойными линиями (*blackboard letters*), нотный, астрономический, химический, шахматный, и т.д.; мы ограничимся их упоминанием.

Многоточия бывают различных типов. Следует различать многоточие на уровне строки (по базовой линии, т. е. по низу нормальных букв), обозначаемое `\ldots` и используемое в перечислениях $a, b, \ldots, z$, и многоточие на среднем уровне (по уровню знаков операций), обозначаемое `\cdots` и используемое в действиях $a + b + \cdots + z$. Для сравнения неправильное использование двоеточий: $a, b, \cdots, z$ и $a + b + \ldots + z$.

Отметим еще, что при наборе формул пробелы игнорируются, поэтому при наборе нескольких идущих подряд формул их нужно набирать как отдельные формулы $x < y$, $y < z$, $i \in I$, $j \in I \times I$, а не единой формулой $x < y, y < z, i \in I, j \in I \times I$, с которой программе трудно справиться. Если же вы хотите добиться дополнительного пробела между элементами формулы (особенно в выключенных формулах), вы можете использовать специальные команды:

ничего	`\rhd\lhd`	$\rhd\lhd$
`\,`	`\rhd\,\lhd`	$\rhd\,\lhd$
`\:`	`\rhd\:\lhd`	$\rhd\:\lhd$
`\;`	`\rhd\;\lhd`	$\rhd\;\lhd$
`\!`	`\rhd\!\lhd`	$\rhd\!\lhd$
`\quad`	`\rhd\quad\lhd`	$\rhd\quad\lhd$
`\qquad`	`\rhd\qquad\lhd`	$\rhd\qquad\lhd$

Рекомендуется при наборе нескольких выключенных формул в одну строку делать отбивку в два квадрата `\qquad`, если формулам «тесно», то в один квадрат `\quad`, а если все равно тесно, то уже компоновать из более мелких отбивок нужный размер.

В конце выключенной формулы с помощью команды `\eqno` вы можете поставить в скобках ее номер

```
$$\sum_{j\in J}a_{ij}x_j=b_i,
\qquad i\in I\setminus I_0.\eqno(15)$$
```

$$\sum_{j\in J} a_{ij}x_j = b_i, \qquad i \in I \setminus I_0. \tag{15}$$

Я чуть-чуть поспешил с индексами. Сейчас они появятся.

5 Формулы с индексами и приставными знаками

Для индексов (нижних и верхних) используются знаки `_` — подчеркивание для нижнего индекса и `^` — для верхнего. Если индекс состоит более чем из одного символа, вся индексная группа заключается в фигурные скобки. Внутри скобок можно использовать те же индексные конструкции (размеры шрифтов пересчитываются по специальным правилам). Если требуется одновременно и верхний и нижний индекс, они записываются один за другим в произвольном порядке.

Примеры.

`a_n`	a_n	`a^n`	a^n
`a_n^k`	a_n^k	`{a_n}^k`	${a_n}^k$
`a_{n_k}`	a_{n_k}	`a_{n^k}`	a_{n^k}
`a^(n)`	$a^(n)$	`a_{i_kj_k}`	$a_{i_kj_k}$

Скобка в показателе степени в нижнем левом примере — это не ошибка, а демонстрация типичной ошибки.

Индексы около служебных слов пишутся с использованием тех же конструкций, однако, эффект их использования различен для формул выключенных и набранных в подборку. Сравните, при одном и том же исходном тексте `\lim_{n\to\infty}x_n` в строке набирается так: $\lim_{n\to\infty} x_n$, а в выключенной формуле:

$$\lim_{n\to\infty} x_n.$$

Приставными знаками называются знаки, меняющие свой размер в зависимости от высоты стоящей около них математической формулы. К ним относятся, в частности,

Сумма	`\sum`	$\sum a_n$	$\sum A^{B^R}$
Произведение	`\prod`	$\prod a_n$	$\prod A^{B^{R^T}}$
Объединение	`\bigcup`	$\bigcup a_n$	$\bigcup A^{B^R}$
Пересечение	`\bigcap`	$\bigcap a_n$	$\bigcap A^{B^R}$
Интеграл	`\int`	$\int a(x)\,dx$	$\int A^{B^{f^k(x)}}\,dx$

При использовании приставных знаков, как правило, требуется задавать область действия описываемой операции, — в нормальном жаргоне —

верхний и нижний предел, хотя часто ограничиваются именно областью, записываемой на месте нижнего предела:

$$\int_{-\infty}^{x}, \quad \sum_{i=1}^{n} \quad \prod_{k\in K}, \quad \sum_{k\in\bigcup_{i\in I} K_i}.$$

Здесь, естественно, используются те же способы, что и для задания верхних и нижних индексов. Обратите внимание на два разных способа размещения индексного текста — ровно снизу или сверху и со сдвигом вправо (как это сделано в индексном выражении внутри индексного выражения в последнем примере, так же располагаются и индексные выражения в формулах, набираемых в подборку). Можно сменить способ размещения заключив приставной знак в фигурные скобки:

$$\int_{-\infty}^{x}, \quad \sum\nolimits_{i=1}^{n} \quad \prod\nolimits_{k\in K}, \quad \sum\nolimits_{k\in\bigcup_{i\in I} K_i}.$$

Квадратный корень также относится к приставным знакам, хотя и представляет более сложную конструкцию из-за горизонтальной черты, надчеркивающей все подкоренное выражение: `\sqrt{a^2+b^{a+2^k}}` и как результат

$$\sqrt{a^2+b^{a+2^k}}.$$

Близки по типу к приставным знакам скобки и другие ограничители — вертикальная черта, знак нормы и др. Однако, их отличительная особенность в том, что они встречаются согласованными по размеру группами. Чаще всего эта группа состоит из пары ограничителей. Этот случай является основным, мы с него и начнем, а затем посмотрим, как к нему приводятся все остальные.

Ограничители переменного размера могут объединяться в пары в любом сочетании. Для того чтобы образовать такую пару нужно перед левым ограничителем поставить команду `\left`, а перед правым — `\right`. Например,

```
$$\left(a\right)\qquad \left(\sqrt{A^{A^a}\right)$$
```

дает следующий набор

$$\left(a\right) \qquad \left(\sqrt{A^{A^a}}\right)$$

Если требуется только один ограничитель (часто это связано с использованием объединяющей фигурной скобки), вместо второго ставится «пустой ограничитель», роль которого играет точка. Когда ограничителей больше двух, они набираются из пар. Нужно только иметь в виду, что высота каждой пары ограничителей определяется высотой текста между ними, и соответственно выбирать определяющие высоту части формулы. Например,

```
$$\left\{\left.a_i\right| \int_{-\infty}^{a_i}\leq 0\right\}$$
```

дает

$$\left\{a_i\left| \int_{-\infty}^{a_i}\leq 0\right.\right\}$$

тогда как

```
$$\left\{a_i\left| \int_{-\infty}^{a_i}\leq 0\right.\right\}$$
```

дает

$$\left\{a_i\left|\int_{-\infty}^{a_i}\leq 0\right.\right\}$$

Иногда ограничители вырастают до очень больших размеров. В этих случаях полезно постараться уменьшить высоту определяющего их выражения.

В перечислении простейших конструкций не обойтись без дробей, хотя дроби — это часть более сложных конструкций вертикальной организации текста. Для дробей мне больше нравится конструкция, используемая в LaTeX-е, — за командой `\frac` (от *fraction* — дробь) следуют сначала числитель, потом знаменатель:

```
$$\frac{1+\sum_k\rho^k}{\theta(\sin x-\sin2x)}.$$
```

$$\frac{1+\sum_k\rho^k}{\theta(\sin x-\sin 2x)}.$$

6 Таблицы и выводы

Вам может показаться странным термин *Выводы*, но ничего не сделать, — это принятый полиграфический термин. Цитируем *Справочник полиграфиста*

> Вывод — это цифровой или текстовой материал, сгруппированный в виде горизонтальных строк и вертикальных колонок, разделенных пробельным материалом.

Попросту вывод — это таблица без линеек.

Таблицы и выводы здесь не различаются. Разделяющие линейки ставятся в тех местах, где понадобятся. Но конструкций для формирования таблиц и выводов две — для обычного набора (обстановка `tabular`) и для математического (обстановка `array`). Ограничимся первой из них, вторая вполне аналогична.

При задании таблицы нужно указать в фигурных скобках способ выравнивания полей типовой строки. Формат строки таблицы задается форматной строкой, в которой `c` соответствует центруемому полю, `l` — полю, выравниваемому по левому краю, `r` — полю, выравниваемому по правому краю, `|` — вертикальной линейке, `@{...}` — более сложному заполнению междуколонного пространства. Далее следует заполнение полей, причем отдельные поля разделяются знаком амперсанда `&`, а набор строки (кроме последней) обозначается командой `\\`. Количество полей в конкретной

строке может быть меньше чем это предусмотрено в форматной, а превышение числа полей над заданным считается ошибкой. Команда `\hline` вырабатывает горизонтальную линейку.

Пример. Исходный текст

```
\begin{center}
\begin{tabular}{c|rrcl|}
\hline
\No & & Engl. & Deu. & Fr.\\
\hline
1 & 10   & ten      & zehn    & dix    \\
2 & 100  & hundred  & hundert & cent   \\
3 & 1000 & thousand & tausend & mille \\
\hline
\end{tabular}
\end{center}
```

Результат:

№		Engl.	Deu.	Fr.
1	10	ten	zehn	dix
2	100	hundred	hundert	cent
3	1000	thousand	tausend	mille

В случае необходимости при наборе конкретной строчки можно «захватить» несколько столбцов, переопределив формат получившегося столбца (эта конструкция позволяет переопределить формат и одного столбца). Для такого переопределения используется команда `\multicolumn{n}{f}{...}`, где `n` — число столбцов, `f` — формат получающегося столбца.

Табличная обстановка используется и для набора матриц и определителей. Вопрос об ограничителях прост, — используются конструкции `\left` и `\right`

```
$$\left(\begin{array}{cccc}
a_{11} & a_{12} & a_{13} & a_{14}\\
a_{21} & a_{22} & a_{23} & a_{24}
\end{array}\right)$$
```

$$\left(\begin{array}{cccc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{array}\right)$$

И, конечно, формулы с несколькими случаями тоже описываются с помощью обстановки `array`

```
$$f(x)=\left\{\begin{array}{ll}
0, & x<a, \\
\displaystyle\frac{x-a}{b-a} & a\leq x\leq b,\\
1, & x<a.
\end{array}\right.\eqno(A)$$
```

$$f(x) = \begin{cases} 0, & x < a, \\ \dfrac{x-a}{b-a}, & a \le x \le b, \\ 1, & x < a. \end{cases} \qquad (A)$$

А как в сложной группу формул сделать строчку из точек, если мы предусмотрели n случаев и нужно сделать вертикальное «и так далее»? В TEX-е имеется команда `\dotfill`, заполняющая точками все выделенное место. Для тех ситуаций, когда длина этого места не может быть определена из контекста, нужно поместить команду в ящик нужной ширины. Такой ящик можно создать уже упоминавшейся командой `\makebox[a][b]{c}`, у которой первый (не обязательный) параметр — это ширина ящина, третий — текст записываемый в ящик, а второй (тоже не обязательный) — параметр, задающий выравнивание текста внутри ящика. Но здесь можно прямо использовать данные таблицы, только с помощью команды

`\multicolumn{число столбцов}{выравнивание}{текст}`

объединить 2 столбца. Получим

$$\begin{array}{l} a_1(x) = x \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ a_n(x) = x^n + \ldots + x. \end{array}$$

7 Определение новых команд

В случае, если вам требуется часто набирать одно и то же сложное для набора выражение, вы можете сами определить необходимые команды. Команды бывают длинные (управляющие последовательности) и короткие — в один символ. Первоначально определяются длинные команды. Определение состоит из команды `\newcommand`, за которой следует в фигурных скобках название определяемой команды, затем, если нужно, в квадратных скобках число параметров команды, затем снова в фигурных скобках подставляемый вместо команды текст. Параметры обозначаются `#1`, `#2`, и т.д.

Например, определение

```
\newcommand{\PS}{PostScript}
```

вводит команду `\PS`. Каждый вызов этой команды будет заменяться словом `PostScript`, набираемым тем шрифтом, который установлен к моменту вызова. Если нужен вполне определенный шрифт, то следует установить этот шрифт прямо в команде и для локализации действия этой установки добавить пару фигурных скобок:

```
\newcommand{\PS}{{\sf PostScript}}
```

Теперь вызов `\PS\` вставит слово PostScript и не изменит установки шрифта для окружающего текста. Фактически вместо имени команды подставляется определяемый этой командой текст, а затем этот текст «исполняется»,

т. е. заново просматривается анализатором, и тексты подставляются, а команды исполняются.

Отметим, что после вызова команды нужно поставить знак \ для обеспечения пробела.

Короткие команды вводятся с помощью команды `\let`, за которой должны следовать обозначение этой команды, знак равенства и уже введенная ранее длинная команда.

Как отмечалось, можно вводить команды и с параметрами. Например, для того, чтобы написать частную производную, скажем, $\frac{\partial u}{\partial x}$ нужно написать следующий текст

```
\frac{\partial u}{\partial x}
```

Вводя макрокоманду

```
\newcommand{\dd}[2]{\frac{\partial#1}{\partial#2}}
```

пишем просто: `\dd{u}{x}`, и не составляет труда писать большие и громоздкие конструкции с частными производными:

```
$$\Delta u=\dd{^2u}{x^2}+\dd{^2u}{y^2}+\dd{^2u}{z^2}=0.$$
```

что дает

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = 0.$$

Упражнение. Догадайтесь сами, как набрана формула (вам нужно придумать два обозначения)

$$\sqrt[8]{891} \approx 2.337409006.$$

Теперь приведем несколько полезных макроопределений:

1. **Большие фигурные скобки** Скобки, меняющие размер в зависимости от высоты внутренней формулы, удобно задать макрокомандой

```
\newcommand{\GRB}[1]{\left\{#1\right\}}
```

Я написал «большие», т. к. изменение, как правило, идет в сторону увеличения. Аналогично определяются большие круглые и квадратные скобки, модуль и норма.

2. **Кратный интеграл**

```
\newcommand{\intint}{\int\!\!\int}
```

Обратите внимание на уменьшение пробела между знаками интеграла в правой формуле, а заодно на отступы перед дифференциалами:

$$\int\int f(x,y)dxdy \quad \to \quad \iint f(x,y)\,dx\,dy$$

3. **Определение или операция с надписью под знаком операции**. Определение

```
\newcommand{\defeq}{\mathrel{\mathop=^\Delta}}
```

дает в использовании

$$X \stackrel{\Delta}{=} (V^3 + 1)$$

Здесь интересно использование команды `\mathop` для временной классификации знака равенства (являющегося отношением) как приставного знака. Это действие вызывает вертикальное размещение «верхнего индекса» Δ. Дальше получившаяся конструкция классифицируется как отношение с помощью команды `\mathrel`.

Предложенная конструкция работает, конечно, и при наличии параметров. При использовании определения

```
\newcommand{\towith}[1]{\mathrel{\mathop{\longrightarrow}_{#1}}}
```

можно после контрольной последовательности в фигурных скобках поставить подписываемый текст, и, например, `f(x)\towith{x\to\infty}0` даст

$$f(x) \mathop{\longrightarrow}_{x\to\infty} 0$$

4. **Новые стандартные слова**. По аналогии с синусом, косинусом и другими текстовыми обозначениями приходится вводить новые обозначения. При введении такого обозначения нужно не только предусмотреть его написание прямым латинским шрифтом, но и дать ему те же «права», чтобы индексные выражения правильно располагались при наборе. Делается это, как выше, классификацией нового обозначения как приставного знака с помощью команды `\mathop`. Например, определение

```
\newcommand{\vraisup}{\mathop{\mbox{\rm vrai~sup}}}
```

выглядит в использовании так

$$\mathop{\mathrm{vrai\ sup}}_{x\in X} f(x).$$

Обратите внимание на оператор `\mbox{...}`, который создает внутри формулы «неформульную зону» для того, чтобы набрать поясняющие слова в текстовом режиме:

```
$$\Phi(x)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^x
\exp\left(-\frac{x^2}2\right)dx
>\frac12\quad\mbox{\rm для}\quad x>0.$$
```

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{x^2}{2}\right) dx > \frac{1}{2} \quad \text{для} \quad x > 0.$$

8 Условное исполнение

Д. Кнут применил очень интересную, в каком-то смысле «первобытную», конструкцию для логических переменных. Логическая переменная определяется как *условие*, при выполнении которого должны выполняться те или иные действия. Каждое такое условие имеет имя, состоящее из обязательного `\if` и произвольного дополнительного идентификатора, возьмем для определенности `OK`. Условие вводится командой: `\newif\ifOK` и при этом автоматически описываются две команды `OKtrue` и `OKfalse`, устанавливающие состояние выполнения и нарушения этого условия. Условие используется в конструкциях

```
\ifOK Исполняемый текст \fi
\ifOK| Текст ДА \else  Текст НЕТ \fi
```

Вот простое и элегантное использование условной конструкции для того, чтобы варьировать текст в зависимости от того, окончательный это вариант или еще рабочий. [9]

Описывается новое условие: `\newif\ifdraft` и устанавливается значение «черновик» (для значения «чистовик» достаточно «закомментировать» вторую строчку):

```
\newif\ifdraft
\drafttrue
```

После этого можно пользоваться конструкциями

```
\ifdraft
    текст для черновика
\else
    текст для окончательного набора
\fi
```

Исходный TEX включает несколько макрокоманд с параметрами и без, вырабатывающих условия. Рассмотрим некоторые из них.

Команда `\ifmmode` вырабатывает условие «идет набор математической формулы». Этим условием можно пользоваться для определения знака, который набирается только в математическом наборе, но может использоваться и независимо. Например,

```
\newcommand{\bullt}{\ifmmode\bullet\else$\bullet$\fi}
```

9 Окаймление входного файла

При использовании LATEX-а входной файл должен иметь несколько стандартных предложений в начале и в конце. Вот типичное оформление файла

[9] Получено по электронной почте 11 апреля 1993, автор — `Richard Tenney <rlt at cs.umb.edu>`.

```
\documentclass{article}
\usepackage[cp1251]{inputenc}
\usepackage[english,russian]{babel}
\usepackage{amssymb}
\usepackage{graphicx}
\title{{\bf КАК ГОТОВИТЬ ДОКУМЕНТЫ НА {\rm\TeX}-е}}
\author{{\bf И.В.Романовский}}
\date{}
\begin{document}
\maketitle
.....................................
\end{document}
```

Здесь строка с `documentclass` — это самое начало документа,[10] параметр `{article}` определяет класс издания, `\title{...}` и `\author{...}` — заголовок и автор, `\date{}` — такой вид параметра «дата» подавляет печать даты; если же вы «что-то» впишите в фигурные скобки, это «что-то» и будет напечатано, а при отсутствии этого параметра будет напечатана текущая дата (по установке в компьютере).

Между `\documentclass` и классом издания можно вставить в квадратных скобках некоторые факультативные установки. В частности, если в качестве базового кегля выбирается не 10 пунктов, то именно здесь и делается новая установка `\documentclass[12pt]{...}`. Если вы разработаете какие-то свои команды и уточнения, то можете объединить их в своем собственном стилевом файле, например, `beloved.sty` (расширение меняться не может) и предусмотреть его ввод при запуске программы, вставив его имя в тех же квадратных скобках `\documentclass[12pt,beloved]{...}`.

После этой строки перечисляются еще пакеты, используемые в данном тексте. Я использую 4 пакета. Первый определяет кодировку входного документа `input encoding` — я использую кодировку `cp1251` — это кириллица для Windows.

Второй — языковая настройка `babel`, названная так в честь библейской вавилонской башни. Настраивается на английский и русский тексты.

Третий — подключение коллекции математических символов Американского математического общества.

Четвертый — подключение команд для работы с графикой.

Строка `\begin{document}` — это открывающая скобка для документа, а закрывающей скобкой служит, конечно, `\end{document}`. Наконец, команда `\maketitle` вызывает печать заголовка, формируемого из перечисленных выше параметров. Команда `\documentclass` и все поле перед документом (эта часть документа называется *преамбулой*) имеют много дополнительных возможностей определения стиля издания, но об этом речь сейчас вести рано.

[10] Слово `documentclass` появилось в версии Латеха 2e, до этого использовалось слово `documentstyle`.

Отметим еще, что можно пользоваться включением файла в файл. Такое включение выполняется оператором `\input`, за которым следует имя файла. Такое включение удобно при формировании больших текстов и для использования своих собственных обозначений и описаний.

10 Числа и счетчики

Естественно, что в полиграфической системе нельзя обойтись без хотя бы элементарной арифметики. Арифметика нужна уже для счета страниц. На самом деле, вычислительные действия, производимые системой, очень серьезны, и пользователь располагает некоторыми возможностями доступа к этой системе. Начнем с целочисленной арифметики.

Некоторые параметризованные системные макрокоманды требуют в качестве параметра целого числа. Это число может быть задано непосредственно, а может быть выработано какой-либо конструкцией. Непосредственная запись может быть десятичной, восьмеричной или шестнадцатиричной.

Дессятичное число пишется как обычно. Перед восьмеричным числом ставится апостроф `'`. В записи шестнадцатиричных чисел кроме цифр используются большие буквы от `A` до `F`, а перед числом ставится двойная кавычка `"`. Одно из важнейших применений этих форм записей — для задания символов. Базовые таблицы символов в исходном TEX-е включают по 128 позиций нумеруемых от 0 до 127, или «по-восьмеричному» от `'0` до `'177`, а в более современном допускающем восьмибитовую входную кодировку — от `'0` до `'377`. Макрокоманда `\char###` определяет символ номер `###` из текущей таблицы шрифтов.

В частности, упоминавшаяся макрокоманда для знака номера выглядит так:

```
\newcommand{\No}{\char"9D}
```

Предположим, что нам понадобились старо-славянские буквы. их можно найти в так нвзываемой вашингтонской кириллице. Закажем себе нужные шрифты, например, определением

```
\font\wncn = wncyr10 \font\wncf = wncyr8
\font\wncin = wncyi10 \font\wncbn = wncyb10
```

Здесь два обычных шрифта, один полужирный и один курсивный. Определим команды установки нужного шрифта в качестве текущаго старого шрифта

```
\newcommand{\Wb}{\let\cof=\wncbn} \newcommand{\wf}{\let\cof=\wncf}
\newcommand{\Wi}{\let\cof=\wncin} \newcommand{\Wn}{\let\cof=\wncn}
```

и дальше определим буквы так, чтобы текущий старый шрифт устанавливался только на время набора одной данной буквы

```
\newcommand{\cof}{\wncn}
\newcommand{\Th}{{\cof\char'25}}\newcommand{\thi}{{\cof\char'35}}
\newcommand{\yt}{{\kern-1pt\cof\char'53\kern0.7pt}}
\newcommand{\ih}{{\cof\char'14}}
\newcommand{\titlo}{\rule[6pt]{1.5pt}{0.4pt}%
\rule[6.2pt]{2pt}{0.2pt}\rule[6.2pt]{1.5pt}{0.4pt}\kern-5pt}
```

Теперь можно прямо писать [11] (цитируя словарь В. Даля и используя рисунок М. Чебоксаровой):

Ѳ, буква ѳита, 34-я по ряду, въ церкв. 41-я; пишется, безъ нужды, въ греческихъ словахъ, замѣстъ *ф*; въ црквн. счетѣ: $\overline{\text{ѳ}}$ девять. В гречск. произношеньи ϑ напоминаетъ англиійское the, а нѣ когда писалось у нас также въ гречск. словахъ замѣстъ *т*, нпр. *ѳеатръ*, *ѳеорія*; да и понынѣ буква эта на запд. языкахъ замѣнена th, нпр. *Ѳеодор*, *Ѳома*, *ѳеологія* ипр. *Ѳита, ижица къ розгъ близится. У нея ротикъ ѳитою. Отъ ѳиты подвело животы.*

У нея ротикъ ѳитою.

Аналогично использует номера символов команда `\accent###`, набирающая символ номер `###` как акцент над следующей буквой.

В TEX-е есть и аппарат переменных. Вы можете определить новую целочисленную переменную — *счетчик* — с помощью команды `\newcounter`. Например,

```
\newcounter{rx}
```

описывает счетчик `rx` со значением, равным 0. Установка нового значения счетчика (присваивание) производится оператором `\setcounter`, а приращение значений оператором `\addtocounter`. Первым операндом в этих командах является имя счетчика, а вторым — число, и для того, чтобы использовать значение счетчика во втором операнде его надо «разыменовать» с помощью команды `\value`:

```
\setcounter{rx}{7}                %   rx :=  7
\addtocounter{rx}{7}              %   rx +:= 7    rx := rx + 7
\addtocounter{rx}{-3}             %   rx +:= -3
\setcounter{rx}{\value{ry}}       %   rx :=  ry
\addtocounter{rx}{\value{ry}}     %   rx +:= ry
```

Для вывода значения счетчика в текст используется автоматически порождаемая команда, состоящая из `\the` и названия счетчика: [12] в нашем примере `\therx`, текст печатается в установленном в данный момент формате. Для принудительной смены формата числа, записанного в счетчик, используются следующие команды:

[11] Обратите внимание на знак процента в определении титла, если его не поставить, то переход на следующую строчку будет рассматриваться как пробел. Впрочем, используя команду `rule`, мы немного забегаем вперед.

[12] Эта конструкция используется и для других типов переменных, о которых, может быть, еще будет идти речь.

`\arabic{rx}`	17	Арабские цифры
`\roman{rx}`	xvii	Малые римские цифры
`\Roman{rx}`	XVII	Большие римские цифры
`\alph{rx}`	q	Малые латинские буквы, до 27
`\Alph{rx}`	Q	Большие латинские буквы, до 27

Имеется возможность предусмотреть «иерархию» счетчиков, обеспечивающую автоматический сброс подчиненных счетчиков (например, параграфов, рисунков, таблиц и т. д., для их автономной нумерации внутри каждой главы).

С помощью макросов `\ifeven<число>` и `\ifodd<число>` можно проверить четное число или нечетное. Эта проверка особенно важна из-за ее распространенности (в двухполосном наборе, в двухколонном наборе полосы). Вот пример меняющегося через раз интерлиньяжа в восточном бейтовом строении стиха (в газелях, например):

```
\newcommand{\gaslin}{\addtocounter{glN}{1}\\\ifeven\theglN[4pt]}
```

В иерархии нумеруемых частей текста — главы, параграфы, подпараграфы — TEX использует счетчики, соответственно, `chapter`, `section`, `subsection`, ..., последовательно подчиненные друг другу. Для вызова их значений, нужным образом форматированных, используются команды `\thechapter`, `\thesection`, `\thesubsection`, ...

Например, в стиле `article` (где главы не предусмотрены), переменные `\thesection`, `\thesubsection` и `\thesubsubsection` описаны так (эта информация находится в файле `\tex\latex\base\article.cls`, там есть еще два этажа иерархии)

```
\renewcommand \thesection {\@arabic\c@section}
\renewcommand\thesubsection   {\thesection.\@arabic\c@subsection}
\renewcommand\thesubsubsection
     {\thesubsection.\@arabic\c@subsubsection}
```

Мы видим сравнительно просто описание переменной `\thesection`, в котором устанавливается, что значение счетчика `\section` будет преобразовано в формат `arabic` (некоторые детали я пока опускаю). Сейчас нам важно, что переменная `\thesubsection` состоит из аналогично формируемой части, к которой спереди присоединяются `\thesection` и точка. Это для того, чтобы, например, номер подпараграфа 5 из параграфа 3 выглядел как `3.5` — к этим номерам и непонятным использованиям знака `@` мы еще вернемся.

Два целых числа можно сравнить с помощью макроса `\ifnum<число><отношение><число>`, где параметр `<отношение>` может принимать значения $<$, $>$ или $=$. Например,

```
\ifnum\thepage>15 ... \fi
```

Конструкция из двух макрокоманд `\loop` и `\repeat` используется для организации циклов. При этом после `\loop` должна быть `\if`-овая команда, а макрос `\repeat` завершает конструкцию.

Типичный пример использования конструкции переключателя дает макрокоманда \today, которая использует три системных макроса \day, \month и \year, вырабатывающие по установкам внутри компьютера соответственно день, месяц и год данного пропуска программы.

```
\newcommand{\today}{\ifcase\month\or
January\or February\or March\or April\or May\or June\or
July\or August\or September\or October\or November\or December\fi
\space\number\day, \number\year}
```

Первая конструкция этой макрокоманды и есть переключатель. Он начинается с \ifcase, за которым следует целочисленный параметр, в данном случае вырабатываемый макросом \month. За ним следуют действия, соответствующие значению параметра 0, затем, разделяемые макросом \or значению 1, 2 и т. д. Конструкция кончается закрывающей скобкой \fi. Два других числовых параметра предваряются макросом \number, переводящим число в представляющую его строку. Вот другая форма представления числа, часто используемая в России:

```
\newcommand{\rustoday}{\number\day/%
$\underline{\overline%
{\mbox{\rm\uppercase\expandafter{\romannumeral\month}}}}$%
--\number\year}
\rustoday
```

Сегодня 27/$\overline{\underline{\text{IV}}}$–2014. Здесь, кажется, потребуются некоторые дополнительные объяснения. Команда \romannumeral аналогична команде \number, но переводит число в последовательность строчных римских цифр. Чтобы перевести эти буквы в прописные используется команда \uppercase{...}. Однако, прямая запись

```
\uppercase{\romannumeral\month}
```

работает неправильно, так как команда \uppercase{...} берет следующий за ней текст и воспринимает его как строку. Макрокоманда \expandafter и служит для задержки передачи параметра до исполнения следующей команды.

11 Размеры

Линейные величины (размеры) могут выражаться в разных единицах, и TEX допускает использование довольно разнообразных единиц с естественным переводом одних в другие (внутреннее представление одно, отличие только при определении самих размеров). Нам естественно начать с обычных для нас «метрических» единиц — сантиметра cm и миллиметра mm. Бытовая американская единица — это дюйм ≈2.54 см (inch), обозначаемый in. Для полиграфических целей эти единицы не очень удобны, и традиционно используются более мелкие *пункты*, размер которых $^1/_{72}$ часть

дюйма. В зависимости от того, какой дюйм взят за основу, меняется и размер пункта. Так, в русской полиграфии традиционно используется старофранцузский пункт `dd`, введенный основоположником измерений в полиграфии Ф. Дидо (F. Didot), в английской и американской полиграфии либо старо-английский пункт `pt` (это самая распространенная единица), либо современный пункт `bp` (называемый еще «большим пунктом»). Их размеры таковы:

$$1\,\mathrm{dd} = 0.377475\,\mathrm{mm}, \quad 1\,\mathrm{pt} = 0.35146\,\mathrm{mm}, \quad 1\,\mathrm{bp} = 0.352778\,\mathrm{mm}.$$

Производные от этих размеров — *пайка* или *пика* (pica, `pc`) — 12 основных пунктов и, аналогично, *цицеро* (cicero, `cc`) — 12 пунктов Дидо. [13]

Используя эти единицы, можно ввести в употребление размеры. *Размером* называется число, записанное в одной из принятых в TeXe систем записи (мы ограничимся обычными десятичными) и сопровождаемое единицей измерения: `10pt`, `3cc`, `17.2cm`.

Некоторые команды требуют размера в качестве параметра. Здесь мы можем уже ввести в обращение команды вертикальной и горизонтальной *отбивки*, т. е. дополнительного пробела;

`\vspace{4pt}`	Вертикальная отбивка в 4 пункта
`\vspace*{4pt}`	Вертикальная отбивка даже в начале страницы
`\hspace{4pt}`	Горизонтальная отбивка в 4 пункта
`\hspace*{4pt}`	Горизонтальная отбивка даже в начале строки

Можно вводить переменные типа размер, такая переменная описывается с помощью команды `\newlength`, ей можно присвоить требуемый размер командой `\setlength`, или увеличить ее значение командой `\addtolength`.

Некоторые параметры набора заранее определены как размеры, их значениями можно всюду пользоваться и даже менять внутри файла (однако, так, чтобы не ставить систему в «трудное положение»). Например, абзацный отступ определяется размером `\parindent`, а ширина текста — размером `\textwidth`.

Попробуем набрать текст с большим абзацным отступом (в $^2/_5$ ширины текста):

```
\setlength{\parindent}{0.4\textwidth}
```

Ему нравился большой размах, хотя сам он был человеком сдержанным.

Так, например, из женщин он ценил Жанетту с Искусственных минеральных вод, которая первая ввела таксу на каждую руку и ногу в отдельности.

— Это женщина, — говорил он.

Но допускал существование и других.

Ю. Тынянов, Малолетный Витушишников

[13] Pica (лат) — свод правил для вычисления времени Пасхи и других праздников с переменной датой.

Параметры размеров страницы могут устанавливаться только в начале документа, в «преамбуле» — до строки `\begin{document}`. Это, прежде всего, высота и ширина текста. Способ их задания упрощен — после имени параметра просто задается требуемый размер:

```
\textheight23cm
\textwidth16.3cm
```

Однако, как мы видели выше, использоваться эти параметры могут повсеместно. Другие параметры задают расположение текста на листе. Расположение задается двумя параметрами — отступом вниз от базовой точки страницы и отступом от нее вправо. Базовой точкой считается точка, отстоящая от физического левого верхнего угла на 1 дюйм (=72 пункта). Верхний отступ обозначается через `\topskip`, а левый отступ имеет два имени, — для левой и правой полос разворота. Если на развороте только одна полоса, то используется только `\oddsidemargin`, а размер `\evensidemargin` принимается равным ему. В случае двухполосного разворота нужно задать оба параметра.

Важный типографский знак, размер которого может задаваться непосредственно, — это сплошной прямоугольник, который называется *линейкой*, так как обычно используется с сильно различающимися шириной и высотой. Тем не менее, и вертикальная и горизонтальная линейка — это просто прямоугольник `\rule<длина><ширина>`. Пример полезного прямоугольника — «бабашка» — квадратный прямоугольник ■, который сейчас любят ставить в конце доказательства вместо традиционного раньше Q.E.D. — *Quod erat demonstrandum*:

```
\newcommand{\QED}{\rule{5pt}{5pt}}
```

Макрокоманда `\rule` имеет еще факультативный (optional) параметр, задающий подъем нижнего среза линейки над базовой линией набора строки. Этот параметр, когда он использукется, задается первым и в квадратных скобках. Пример: ▬▟ — эти прямоугольники немного сдвинуты «назад», это достигается командой `\kern`, параметр которой, размер сдвига вправо, может принимать и отрицательные значения

```
\newcommand{\dachshund}{\rule[3pt]{5pt}{3pt}\kern-1pt
\rule{15pt}{3pt}\kern-1.2pt\rule[3pt]{2.2pt}{6pt}}
```

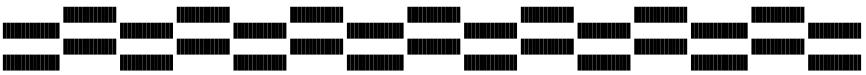
Нужно еще упомянуть два очень важных размера, которые зависят от текущих установок шрифтов: `em` и `ex`. Первый из них — это горизонтальный размер, задаваемый макросом `\quad` в текущем шрифте, а второй — вертикальный размер буквы x. Таким образом,

```
{{\Large M\rule{2em}{1ex}x} и {\small M\rule{2em}{1ex}x}
и {\small\tt M\rule{2em}{1ex}x}}
```

M▬▬x и M▬▬x и M▬▬x

Сделаем небольшое упражение на использование целочисленной и размерной арифметики.

```
\newcounter{spa}\newcounter{i}
\newcommand{\rulefill}{\leaders\hbox{\rule{0.5mm}{6pt}}\hfill}
\newcommand{\sleeper}{\ifodd\thespa\rulefill\else\hfill\fi
\addtocounter{spa}{1}}
\newcommand{\sleeperline}{\noindent\strut\setcounter{i}{15}
\loop\ifnum\thei>0\sleeper\addtocounter{i}{-1}\repeat\hfill\strut}
\sleeperline\vskip-6pt\sleeperline\vskip-6pt\sleeperline%
\vskip-6pt\sleeperline
```



Первая строчка этого текста описывает два новых счетчика, — в одном из них запоминается порядковый номер «шпалы», второй понадобится как индекс для организации цикла. Во второй строке определяется новый заполнитель — линейка высотой в 6 пунктов, состоящая из кусочков длиной в полмиллиметра. Самое интересное дальше: макрокоманда `\sleeper`, которая поочередно вызывает новый заполнитель и пробел. Эта процедура вызывается в макросе `\sleeperline` дальше (обязательно нечетное число раз, чтобы был эффект «шахматного» заполнения; мы выбрали 15). Макрокоманда `\strut` формирует линейку нулевой ширины. Уменьшение пробела между строками на 6 пунктов обеспечивает правильное прилегание линеек.

12 Ящики

При разработке TEX-а Кнут использовал плодотворную идею Кернигена и Ритчи, авторов системы `UNIX` и языка Си,, которую они развили в текстовом редакторе `TROFF`. Идея заключается в том чтобы представить компонуемый текст (первоначально формулу) в виде совокупности неких «ящиков», из которых по определенным правилам компонуются другие ящики: *Ящик дробь ящик*, *ящик в степени ящик* и т.д.[14]

В TEX-е все составляется из ящиков. Один тип ящиков — `\mbox{...}` — нам уже встречался, когда речь шла о тексте внутри формул. Этот тип ящиков, к нему относятся и несколько других разновидностей, введен только в LATEX-е. Он называется LR-box. Содержимое такого ящика всегда набирается в одну строку (*LR* означает *Left-to-Right*). Более изощренной формой является `\makebox{...}` с факультативными параметрами (в квадратных скобках), первый из которых задает ширину ящика (это размер), а второй — односимвольный — способ расположения текста внутри ящика: `c` для центровки (это значение принимается по умолчанию), `l` и `r` соответственно для сдвига влево и вправо. Угадайте, как это сделано:

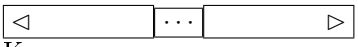
[14] Авторы, разумеется, используют термин *box*, а не *yashchik*, и это побуждает некоторых наших соотечественников использовать термин *бокс*. По этому поводу вот история, рассказанная мне Г. С. Цейтиным. Однажды он придумал новый тип организации памяти, который назвал «пузырем», и рассказал о нем на конференции. Один из слушателей спросил, как «пузырь» по-английски. «*A bubble*», — ответил Г.С. «Мы *баблы* тоже используем», — сказал тот в выступлении.

вот
где была
собака зарыта.

государство
— это
Я

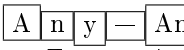
Аналогичные макросы `\fbox{...}` и `\framebox{...}` отличаются тем, что соответствующие ящики обводятся рамкой, причем не вплотную, а на некотором удалении. Например,

```
\framebox[2cm][l]{$\lhd$}\fbox{$\cdots$}\framebox[2cm][r]{$\rhd$}
```



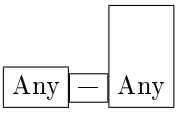
Как видите, ящики получились разной высоты и по-разному выровнены (их высота над базовой линией и глубина определяются высотой и глубиной содержащегося в ящике текста. Попробуем еще:

```
{\rm\fbox{A}\fbox{n}\fbox{y}\fbox{---}\fbox{Any}\fbox{.}}
```



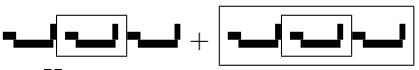
Линейка `\rule` — это еще один пример ящика. Линейка нулевой ширины, как сказано по другому поводу у Ю. Тынянова, «вида не имеет». Она используется для увеличения вертикального размера объемлющего ящика:

```
{\rm\fbox{Any}\fbox{---}\fbox{\rule{0pt}{30pt}Any}}
```



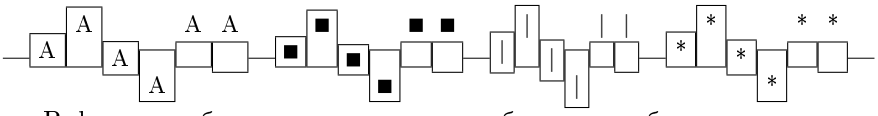
Отметим еще, что ящики можно запоминать, запоминание уже подготовленного ящика занимает дополнительную память, но сокращает затраты времени на обработку текста. Для того чтобы запоминать ящики нужно ввести «ящиковую переменную» и вписать в нее требуемое содержимое используя макрос `\savebox{}{...}`, первый параметр которого задает имя ящика, а второй — содержимое. Для использования нужно «вызвать ящик» макросом `\usebox{}{...}`. Например,

```
\newsavebox{\threedogs}
\savebox{\threedogs}{\dachshund\fbox{\dachshund}\dachshund}
\usebox{\threedogs} $+$ \fbox{\usebox{\threedogs}}
```



Наконец, последняя пока описываемая нами ящиковая конструкция такого типа из LaTeX-а — это вертикально перемещаемые ящики. Команда `\raisebox{}{...}`. создает ящик, который должен быть сдвинут наверх от обычного положения (измеряемого относительно базовой линии) на размер, заданный первым параметром. Два факультативных параметра, помещаемые после первого основного, задают высоту ящика над базовой линией основного текста и глубину под нею, соответственно. Чтобы увидеть их действие, окружим ящики рамками (учтите, что рамки добавляют свободного места вокруг ящиков) и обозначим базовую линию.

```
{\small\newcommand{\boxes}[1]{\rule{10pt}{0.2pt}\fbox{#1}%
\fbox{\raisebox{10pt}{#1}}\fbox{\raisebox{-3pt}{#1}}%
\fbox{\raisebox{-13pt}{#1}}\fbox{\raisebox{10pt}[3pt]{#1}}%
\fbox{\raisebox{10pt}[3pt][2pt]{#1}}}%
\boxes{A}\boxes{\QED}\boxes{$|$}\boxes{"}\rule{10pt}{0.2pt}}
```



В формате обычного текстового по-абзацного набора можно использовать специальные ящики типа `\parbox{...}{...}`, у которых первый параметр задает ширину текста, а второй — сам текст. Факультативный параметр, который ставится перед шириной, определяет способ вертикального выравнивания ящика. По умолчанию ящик центруется относительно базовой линии объемлющего текста, параметр `t` обуславливает совпадение базовой линии верхней строки текста внутри ящика с основной линией, параметр `b` — аналогичное совпадение для нижней строки.

```
{\small\parbox{20mm}{...}\hrulefill\parbox{20mm}{...}%
\hrulefill\parbox[t]{20mm}{...}\hrulefill\parbox[b]{20mm}{...}}
```

{small}
Короткий текст, выравниваемый по центру
Гораздо более длинный текст, и он тоже должен быть выровнен по центру
Текст, верхняя строка которого будет выровнена по базовой линии
Текст, нижняя строка которого будет выровнена по базовой линии

Теперь что-нибудь более изысканное:

Порт сонный,
Ночной
Плененный
Стеной;
Безмолвны
Спят волны, —
И полный
Покой.

Странный ропот
Взвился вдруг, —
Ночи шепот,
Мрака звук,
Точно пенье
И моленье
Душ, в кипеньи
Вечных мук.

Звук новый льется,
Бренчит звонок:
То пляс уродца
Веселый скок.
Он мрак дурачит,
В волнах маячит,
По гребням скачет,
Встав на носок...
В. Гюго, Джинны

13 Переменные типа ящик

$\mathcal{NB}$: Вставка про переменные типа ящик, их определение, размещение и использование

Работа с переменными типа ящик позволяет получить многочисленные полезные эффекты. Один из них — изменение и измерение размеров готового фрагмента текста. В первую очередь здесь должны быть названы макросы, использующие фантомы.

Фантом (*phantom*) — это призрак. [15] Макрокоманда `\phantom{...}`, аргументом которой может быть любой ящик, создает *пустой ящик* с теми же размерами.

Эта макрокоманда имеет две упрощенных разновидности:

`\hphantom{...}` имеет нулевые вертикальные размеры,

`\vphantom{...}` — нулевой горизонтальный размер.

Имеющаяся в TEX-е макрокоманда `\mathstrut=\vphantom(` вырабатывает вертикальные размеры открывающей скобки. Она очень удобна для выравнивания всяких надчеркиваний и т.п. в математических формулах. Например, если требуется набрать i с чертой сверху, то обычная конструкция `$\overline{i}$` даст слишком низкую черту: $\overline{i}$. Добавка `\mathstrut` поднимет черту до нужной высоты: `$\overline{\mathstrut i}$`: $\overline{\mathstrut i}$.

Макрокоманду `\vphantom` хорошо использовать в тех случаях, когда выбирается пара ограничителей одного размера, а отдельные ограничители этой пары находятся в разных строках, — тогда часть формулы, определяющую размер, можно вставить в другую часть формулы «фантомно».

А вот, скажем, для набора стихов Маяковского удобно использовать `\hphantom`. Например,

```
\hphantom{xxxxxxxxxxx}Начал кричать.\\
\hphantom{xxxxxxxxxxxНачал кричать.} Разве это осилите?!\\
\hphantom{xxxxxxxxxxx}Буря басит ---\\
\hphantom{xxxxxxxxxxxБуря басит ---} не осилить вовек.\\
\hphantom{xxxxxxxxxxx}Спасите! Спасите! Спасите! Спасите!\\
\hphantom{xxxxxxxxxxx}Там\\
\hphantom{xxxxxxxxxxxТам} на мосту\\
\hphantom{xxxxxxxxxxxТам на мосту} на Неве\\
\hphantom{xxxxxxxxxxxТам на мосту на Неве} человек!\\
\hphantom{xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx}\it Про это
```

Начал кричать.
Разве это осилите?!
Буря басит —
не осилить вовек.
Спасите! Спасите! Спасите! Спасите!

[15] Слово **phantom** согласно *Concise Oxford Dictionary* означает form without substance or reality.

Там
на мосту
на Неве
человек!
Про это

Отметим, что существуют и более изящные решения, использующие технику ящиков более изощренно и позволяющие не повторять предыдущий текст фантомно.

При наборе центрованного текста в строке, где присутствует какой-то еще текст (например, в центруемом колонтитуле с колонцифрами, поставленными с наружной стороны разворота) также используется `\hphantom`:

```
\framebox[0.5\textwidth]{100\hfill\bf Chapter 1\hfill%
\hphantom{100}}%
\framebox[0.5\textwidth]{\hphantom{100}\hfill{\it Much ado
about nothing}\hfill100}
```

100	**Chapter 1**	*Much ado about nothing*	100

Той же техникой программирования в TEX-е сделана и еще одна макрокоманда, которая набирает и включает в текст некоторый ящик, но считает его ящиком нулевых размеров. Она называется `\smash{...}` и также имеет параметром ящик.

Напишем теперь какой-нибудь замечательный текст, который нам хочется отчеркнуть слева вертикальной чертой, или даже двойной чертой.

> *Боже, Боже!*
> *Что случилось?*
> *Отчего же*
> *Все кругом*
> *Завертелось, закружилось*
> *И помчалось колесом?*

(а вы-то, наверное, учили этот текст в более поздней редакции!). Этот эффект получен макрокомандой (и сопутствующими определениями)

```
\newlength{\problemwd}\setlength{\problemwd}{\textwidth}
\addtolength{\problemwd}{-7mm}
\newsavebox{\problem}
\long\def\OptProb#1{
\savebox{\problem}{\begin{minipage}[b]{\problemwd}{#1}
\end{minipage}}
\IV\noindent\,\,\,\rule{0.75pt}{\ht\problem}\,%
\rule{0.25pt}{\ht\problem}\hfill\usebox{\problem}
\IV}
```

Как видите, определен новый размер, на 7 миллиметров у́же ширины полосы, и ящик для хранения отчеркиваемого текста. Когда этот текст сохранен

в ящике, можно измерить его ширину, высоту и глубину. Мы воспользовались командой `\ht` для определения его высоты, обеспечив предварительно нулевую глубину выравниванием мини-страницы по низу (параметром `[b]`). Соответствующие команды для глубины и ширины называются `\dp` и `\wd`. Остальное, наверное, ясно и без пояснений.

14 Перечисления

После размеров и ящиков можно уже рассмотреть очень полезную и важную конструкцию перечислительных списков. Она выполнена как обстановка и использует вспомогательные ящиковые конструкции, так что внутри нее некоторые другие полезные средства отказываются работать.

Как уже отмечалось раньше, есть три варианта перечислительной обстановки. Перечислим их в другом варианте оформления:

1. *itemize* Для перечисления в форме отдельных «пунктов», каждый из которых сопровождается особой меткой, выбранной вами или назначенной по умолчанию («пуля» • на первом уровне и минус — на втором и последующих).

2. *enumerate* Для перечисления в форме отдельных нумерованных «пунктов».

3. *description* Для перечисления в форме отдельных «пунктов», каждый из которых имеет отрицательный абзацный отступ.

Это перечисление сделано с помощью обстановки `enumerate`:

```
\begin{enumerate}
\item {\it itemize} ...
\item {\it enumerate} ...
\item {\it description} ...
\end{enumerate}
```

15 Упражнение на переопределение макросов

Рассмотрим всем известный пример (можно даже не смотреть результат, он легко предсказуем):

```
Вот дом,
\def\house{который построил Джек.\\[4pt]}\house
А вот пшеница,\\
\def\wheat{Которая в темном чулане хранится\\ В доме, \house}\wheat
А вот синица,\\
\def\tomtit{Которая часто ворует пшеницу,\\ \wheat}\tomtit
А вот кот,\\
\def\pussy{Который пугает и ловит синицу,\\ \tomtit}\pussy
```

```
А вот пес
\def\dog{без хвоста,\\ Который за шиворот треплет кота,\\ \pussy}\dog
А вот корова безрогая,\\ Лягнувшая
\def\cow{старого пса \dog}\cow
А вот старушка, седая и строгая,\\
\def\woman{Которая доит корову безрогую,\\ Лягнувшую \cow}\woman
А вот пастух,\\
\def\shep{Который бранится с коровницей строгою,\\ \woman}\shep
А это два петуха,\\
Которые будят по утрам пастуха, \shep
```

В этом решении на каждом шаге определяется новый макрос, который используется только два раза — для одного вызова и для подстановки в следующий. Естественно желание обойтись без этих новых макросов.

И действительно, использование макроса, соединяющего (*catenating*) строки, позволяет записать этот текст гораздо короче и проще. Х. Ханче-Ользен [16] предложил следующую форму этого макроса

```
\newtoks\a\newtoks\b\newtoks\res
........
\def\cat#1#2#3{{\edef\0{#3{\the#1\the#2}}\expandafter}\0}
.....
\cat\a\b\res
```

Параметрами этого макроса служат переменные типа `token`, которые можно завести с помощью макроса `newtoks`. Предлагаемый этим автором макрос соединяет строки, записанные в первой и второй переменной и записывает результат в третьей.

Мы несколько модифицировали для наших целей этот макрос, ограничившись двумя переменными — для накопленного текста и для добавляемого (в начало) — и добавив вызов накопленного текста с переводом строки.

```
\def\ncat#1#2{{\edef\0{#2{\the#1\the#2}}\expandafter}\0\the#2\\[4pt]}
```

Итак, получаем текст:

```
\verb|\newtoks\a\newtoks\b|
Вот дом,
\a={который построил Джек.}\the\a\\[4pt]}
А вот пшеница,\\
\b={Которая в темном чулане хранится\\ В доме,}\ncat{\b}{\a}
А вот синица,\\
\b={Которая часто ворует пшеницу,\\}\ncat{\b}{\a}
А вот кот,\\
\b={Который пугает и ловит синицу,\\}\ncat{\b}{\a}
А вот пес
```

[16] Harald Hanche-Olsen, 17 May 93, University at Stony Brook, NY. Автор пишет в своем письме: That use of `\expandafter` is a cute trick of my own, which lets me use the control sequence `\0` without stomping on anyone elses use of the same sequence. Я еще не способен пока это достойно прокомментировать.

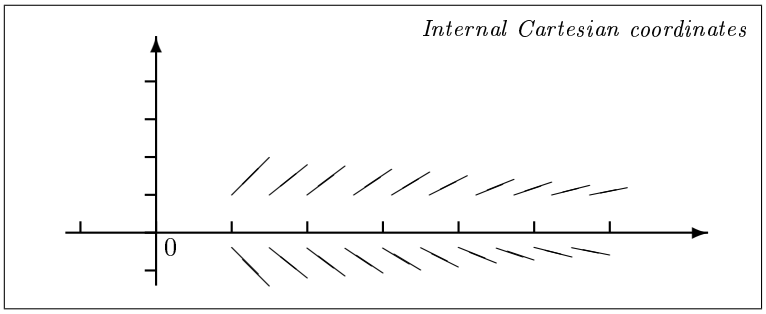
```
\b={без хвоста,\\ Который за шиворот треплет кота,\\}\ncat{\b}{\a}
А вот корова безрогая,\\ Лягнувшая
\b={старого пса\ }\ncat{\b}{\a}
А вот старушка, седая и строгая,\\
\b={Которая доит корову безрогую,\\ Лягнувшую }\ncat{\b}{\a}
А вот пастух,\\
\b={Который бранится с коровницей строгою,\\}\ncat{\b}{\a}
А это два петуха,\\
Которые будят по утрам пастуха, \the\a
```

16 Рисунки

LaTeX позволяет включать в текст небольшие чертежи, в которых можно рисовать прямые линии разной толщины, прямоугольники, круги и эллипсы, и вписывать любые текстовые фрагменты, изготовленные с помощью средств LaTeX-а. Эта возможность оформлена в виде *обстановки* `picture`. К моменту вызова обстановки должна быть установлена используемая в описаниях единица измерения, это делается так (настоятельно рекомендую и вам взять миллиметр в качестве стандартной единицы):

```
\setlength{\unitlength}{1mm}
.................
\begin{picture}(w,h)(xmin,ymin)
...................
...................
\end{picture}
```

При открытии обстановки указываются ширина и высота окна, измеренные в единицах `\unitlength` (параметры `w` и `h`), и координаты левого нижнего угла в принятой вами системе (декартовых) координат внутри данного рисунка.




```
\begin{picture}(100,40)(-20,-10)
\put(-20,-10){\framebox(100,40)[tr]
{\makebox(45,4)[bl]{\small\it Internal Cartesian coordinates}}}
\thicklines
\put(-12,0){\vector(1,0){85}}
```

```
\put(0,-7){\vector(0,1){33}} \put(1,-3){0}
\multiput(-10,0)(10,0){8}{\line(0,1){1.5}}
\multiput(0,-5)(0,5){6}{\line(-1,0){1.5}}
\thinlines
\put(10,-2){\line(1,-1){5}\line(5,-4){5}\line(4,-3){5}%
\line(3,-2){5}\line(5,-3){5}\line(2,-1){5}%
\line(5,-2){5}\line(3,-1){5}\line(4,-1){5}\line(5,-1){5}
}
\put(10,5){\line(1,1){5}\line(5,4){5}\line(4,3){5}
\line(3,2){5}\line(5,3){5}\line(2,1){5}
\line(5,2){5}\line(3,1){5}\line(4,1){5}\line(5,1){5}
}
\end{picture}
```

Разберемся подробно в операторах, создающих эту картинку, и рассмотрим их возможные варианты. Описание размеров вводит картинку 100×40 мм со сдвигом начала координат на 20 мм вправо и на 10 мм вверх от левого нижнего угла (проверьте по линейке)

```
\begin{picture}(100,40)(-20,-10)
.............................
\end{picture}
```

Оператор

```
\put(-20,-10){\framebox(100,40)[tr]
{\makebox(45,4)[bl]{\small\it Internal Cartesian coordinates}}}
```

берет в качестве базовой точку с координатами $(-20, -10)$ и рисует начиная с этой точки рамку размером 100×40 мм, причем то, что будет располагаться в этой рамке будет сдвинуто наверх и вправо (в соответствии с установкой `[tr]`). Это внутреннее изображение, в свою очередь представляет собой ящик без рамки размера 45×4 мм, содержимое которого будет сдвинуто вниз и налево (в соответствии с установкой `[bl]`). Само это изображение — это надпись. Такой парой вложенных ящиков удобно ее устанавливать внутри рисунка. При отсутствии указания на вертикальный или горизонтальный сдвиг изображение центруется.

Оператор `\thicklines` и появляющийся затем оператор `\thinlines` служат для установки ширины линий. Линии проводятся с помощью специальных шрифтов, так что возможны только две толщины, — жирные и тонкие.

Строки

```
\put(-12,0){\vector(1,0){85}}
\put(0,-7){\vector(0,1){33}} \put(1,-3){0}
```

рисуют систему координат. Оператор `\put(1,-3){0}` рисует 0 около начала координат, первый оператор рисует ординату, второй — абсциссу. У оператора `\vector(a,b){c}` параметры в круглых скобках определяют наклон

отрезка, параметр в фигурных скобках — его длину. Мы будем подробнее рассматривать эти параметры дальше, сейчас отметим только, что макрос `\vector(a,b){c}` отличается от макроса `\line(a,b){c}` тем, что ставит векторную стрелку в конце отрезка.

Теперь удобно рассмотреть операторы, рисующие веера линий.

```
\put(10,-2){\line(1,-1){5}\line(5,-4){5}\line(4,-3){5}%
\line(3,-2){5}\line(5,-3){5}\line(2,-1){5}%
\line(5,-2){5}\line(3,-1){5}\line(4,-1){5}\line(5,-1){5}
}
\multiput(0,-5)(0,5){6}{\line(-1,0){1.5}}
\put(10,5){\line(1,1){5}\line(5,4){5}\line(4,3){5}
\line(3,2){5}\line(5,3){5}\line(2,1){5}
\line(5,2){5}\line(3,1){5}\line(4,1){5}\line(5,1){5}
}
```

Первый веер задает набор линий с наклоном вниз, второй — с набором вверх. Обратите внимание на эффект неравномерности, возникающий в верхнем веере от переводов строки в описывающем веер тексте. В нижнем веере этот эффект подавлен благодаря знаку процента в конце строк.

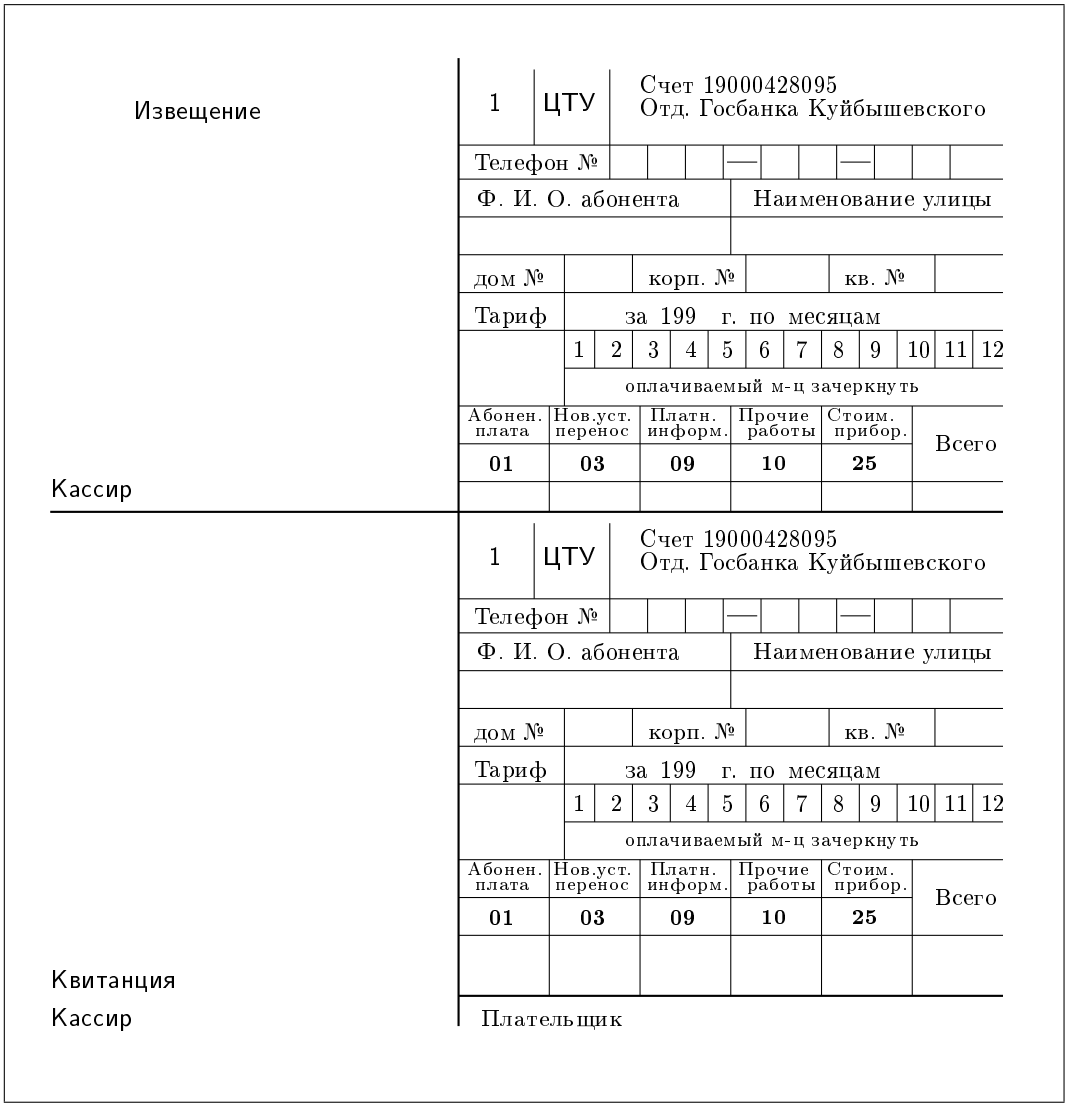
Наклон линии в операторе `\line(a,b){c}` определяется отношением чисел `a` и `b`, которые должны быть целыми числами, каждое из диапазона $-6:6$. Если параметр `a` отличен от 0, параметр `c` задает проекцию линии на ось ординат (а не длину самой линии). При нулевом значении параметра прямо задается длина линии (она же проекция на ось абсцисс).

Отметьте, что при одном операторе `\put` можно записать несколько изображений; каждый из них образует ящик, и эти ящики компонуются в горизонтальную цепочку (набираются в LR-режиме). Здесь присутствие лишних пробелов может вызвать неожиданный эффект. Отсутствие оператора `\put` аналогично `\put(0,0)`.

Наконец, отметки шкалы на координатных осях определяются в специальной циклической версии оператора `\put`. Цикл обеспечивается оператором `\multiput(x,y)(dx,dy){n}{...}`, который имеет параметрами начальную точку, где должен быть нарисовано изображение, приращение координат на каждом шаге цикла, число итераций и действие, которое должно быть выполнено. Сами действия в нашем примере повидимому не требуют комментария.

```
\multiput(-10,0)(10,0){8}{\line(0,1){1.5}}
\multiput(0,-5)(0,5){6}{\line(-1,0){1.5}}
```

Предусмотрена возможность рисования кругов, окружностей, четвертей окружностей. При этом диаметры кругов и окружностей ограничены и не очень велики. Во всех случаях с круглыми объектами базовой точкой объекта служит его центр. Таким образом, операторы `\put(20,10){\circle{5}}` и `\put(20,10){\circle*{3}}` рисуют концентрические окружность и круг с центром в (20,10). Можно рисовать «четверти овалов», под которыми понимаются прямоугольники с максимально закругленными углами.



Фрагмент рисунка, который появляется несколько раз и возможно в нескольких рисунках, может быть сохранен в сохранном ящике и затем использован. Это сильно экономит время на построение картинки, но ощутимо расходует память. Ящик для сохранения картинки должен быть заранее резервирован. Впрочем, использование всей техники хорошо посмотреть на практически важном примере бланка абонентной платы за телефон.

```
\newcounter{monno}
\newsavebox{\tbody}
\newcommand{\rmt}[3]{\put(#1,#2){{\small\rm #3}}}
\newcommand{\rpt}[3]{\put(#1,#2){{\scriptsize\rm #3}}}
```

```
\newcommand{\bpt}[3]{\put(#1,#2){{\footnotesize\bf #3}}}
\newcommand{\sft}[3]{\put(#1,#2){{\yf #3}}}
\setlength{\unitlength}{1mm}
\begin{center}
\begin{picture}(140,145)(0,0)
\savebox{\tbody}(65,65)[bl]{
\sft{4}{57}{1} % - (60,14)
\sft{11}{57}{ЦТУ}
\rmt{24}{59.5}{Счет 19000428095}
\rmt{24}{56.5}{Отд. Госбанка Куйбышевского}
%\rmt{24}{53.5}{коммерческого Лен. П.С.Б.}
\put(10,52.5){\line(0,1){10}}
\put(20,52.5){\line(0,1){10}}
\put(0,52.5){\line(1,0){72}}
\multiput(0,28)(0,5){5}{\line(1,0){72}}
% phone No
\rmt{2}{49.2}{Телефон \No}
\multiput(20,48)(5,0){10}{\line(0,1){4.5}}
\multiput(35.5,50.25)(15,0){2}{\line(1,0){4}}
% FIO
\rmt{2.5}{44.5}{Ф. И. О. абонента}
\rmt{39}{44.5}{Наименование улицы}
\put(36,38){\line(0,1){10}}
% house
\rmt{2}{34}{дом \No}
\rmt{25}{34}{корп. \No}
\rmt{51}{34}{кв. \No}
\put(14,33){\line(0,1){5}}
\put(23,33){\line(0,1){5}}
\put(38,33){\line(0,1){5}}
\put(49,33){\line(0,1){5}}
\put(63,33){\line(0,1){5}}
% Tariff
\rmt{2}{29}{Тариф}
\rmt{22}{29}{за\, 199\,\,\,\, г.\, по\, месяцам}
% Month
\put(14,18){\line(0,1){15}}
\put(14,23){\line(1,0){58}}
\setcounter{monno}{0}
\multiput(15.2,24.5)(4.9,0){12}
{\addtocounter{monno}{1}\rmt{0}{0}{\themonno}}
\multiput(18,23)(5,0){11}{\line(0,1){5}}
\rpt{22}{20}{оплачиваемый м-ц зачеркнуть}
\put(0,18){\line(1,0){72}}
% titles
\put(0,13){\line(1,0){60}}
\rpt{1.1}{16}
{Абонен.\,\,\,Нов.уст.\,\,\, Платн.\,\,\,\, Прочие\,\,\, Стоим.}
\rpt{2.2}{14}
{плата\,\,\,\,\, перенос\,\,\, информ.\,\,\, работы\,\,\, прибор.}
```

```
\put(0,8){\line(1,0){72}}
\bpt{4}{9.5}{01}
\bpt{16}{9.5}{03}
\bpt{28}{9.5}{09}
\bpt{40}{9.5}{10}
\bpt{52}{9.5}{25}
\rmt{63}{12}{Bcero}
}
% end savebox
\put(0,0){\framebox(140,145){}}
\sft{6}{80}{Кассир} \sft{6}{10}{Кассир}
\sft{6}{15}{Квитанция} \sft{17}{130}{Извещение}
\multiput(72,14)(12,0){5}{\line(0,1){18}}
\multiput(72,78)(12,0){5}{\line(0,1){14}}
\thicklines
\put(6,78){\line(1,0){126}} \put(60,14){\line(1,0){72}}
\put(60,10){\line(0,1){128}} \put(60,14){\usebox{\tbody}}
\put(60,74){\usebox{\tbody}}
\rmt{63}{10}{Плательщик}
\end{picture}
\end{center}
```

17 Плавающие объекты

Рисунки и таблицы (а иногда и программы) должны располагаться единым куском, без переноса части на другую страницу. Поскольку тип такого объекта нам сейчас несущественнен, мы будем говорить только о рисунках.

В том случае, если на странице использующей данный рисунок, осталось слишком мало места, TEX вынужден перенести рисунок на следующую страницу и оставить часть места на странице незаполненным. Между тем, часто бывает, что нет большой необходимости помещать рисунок именно в данном месте текста — он может появиться и немного раньше и значительно позднее. Имеется возможность объявить такой рисунок «плавающим».

Плавающий рисунок помещается в список таких объектов, и набор текста на текущей странице продолжается до нормального завершения. После перехода на новую страницу в первую очередь размещаются рисунки из списка плавающих рисунков, а затем, когда очередного рисунка уже нет или он не помещается в страницу, остаток страницы «добирается» обычным текстом.

18 Битовые мапы или изо-биточки

Различные модификации TEX-а позволяют вставлять в документ рисунки, подготовленные другими средствами. В этом параграфе мы увидим как использовать *битограммы*, т. е. по-точечно заданные изображения (англий-

ский термин *bitmap images*).[17]

Используемые нами программы Эберхарда Маттеса позволяют «импортировать» битограммы, подготовленные как файлы в графическом формате `.PCX`, — это, в частности, формат широко известного графического редактора PaintBrush. Рисунок, который вы видите слева, был скопирован ручным сканнером из книжки Альберта Капра *Johannes Gutenberg,* Leipzig, 1977, (Гутенбергу скоро исполнится 600 лет), подправлен и переведен специальным преобразователем форматов из формата `.TIFF` в `.PCX`. На титульном листе аналогично воспроизведен фрагмент с наброска Г. Гольбейна-мл. Сама конструкция вставки хорошо видна на примере. Для того чтобы вставить в текст битограмму, нужно в нем «захватить место», определив рисунок в смысле LaTeX-а и, указав внутри рисунка с помощью оператора `\put` левый ВЕРХНИЙ угол вставки, выполнить оператор `\special` с указанием требуемого `PCX`-файла. Итак,

Герб рода Гутенбергов

```
\begin{picture}(60,90)(0,0)
\put(-5,90){\special{em: graph gutenbrg.pcx}}
\put(0,0){\framebox(60,90)[b]{\it Герб рода Гутенбергов}}
\end{picture}}
```

Другой вариант вставки битограммы, причем не только `.PCX`, заключается в том, что особая программа `bm2mf` переводит это изображение в специальный «шрифт», которым затем печатается «текст». Здесь, конечно, мы ограничимся только упоминанием такой возможности.

[17] Опять не удержаться от брюзжания! Вместо того чтобы использовать исконно русское слово *биточки* с вариантами *цветочки* и *плито́чки* для многоцветных изображений, мы думаем, как перевести заграничное слово *pixel*, составленное из *picture* и *cell*.

Конечно, можно использовать очень удобное слово *растр*. Это слово происходит от латинского rastrum — *грабли* и в полиграфии означает решетку (часто, но не обязательно, прямоугольную), через которую фотографируется полутоновое изображение, которое тем самым «растрируется». Но желательно сохранение термина растр в его первоначальном смысле, и навешивание на тот же термин «битовой мапы» может вызвать путаницу в переводах, если потребуется говорить, например о представлении данного поточечного изображения в данной растровой системе.

19 Разное

1. Таблица составной кодировки

	0	1	2	3	4	5	6	7
00	◌̀	◌́	◌̂	◌̃	◌̈	◌̋	◌̊	◌̌
01	◌̆	◌̄	◌̇	◌̧	◌̨	I	⟨	⟩
02	“	”	◌̰	◌̴	◌̵	–	—	
03	◌̡	ı	ȷ	ff	fi	fl	ffi	ffl
04	◌̲	!	"	#	$	%	&	’
05	(	)	*	+	,	-	.	/
06	0	1	2	3	4	5	6	7
07	8	9	:	;	<	=	>	?
10	@	A	B	C	D	E	F	G
11	H	I	J	K	L	M	N	O
12	P	Q	R	S	T	U	V	W
13	X	Y	Z	[	\	]	◌̂	◌̲
14	‘	a	b	c	d	e	f	g
15	h	i	j	k	l	m	n	o
16	p	q	r	s	t	u	v	w
17	x	y	z	{	|	}	~	-
20	Ґ	Ғ	Ҕ	Ҥ	Һ	Ӂ	Ҙ	Љ
21	Ї	Қ	Ҟ	Ҝ	Æ	Ң	Ҥ	S
22	Ө	Ҫ	Ў	Ү	Ұ	Ҳ	Ҷ	Ҷ
23	Ҷ	Є	Ә	Ҧ	Ё	№	¤	§
24	ґ	ғ	ҕ	ҥ	h	ӂ	ҙ	љ
25	ї	қ	ҟ	ҝ	æ	ң	ҥ	s
26	ө	ҫ	ў	ү	ұ	ҳ	ҷ	ҷ
27	ҷ	є	ә	ҧ	ё	„	«	»
30	А	Б	В	Г	Д	Е	Ж	З
31	И	Й	К	Л	М	Н	О	П
32	Р	С	Т	У	Ф	Х	Ц	Ч
33	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
34	а	б	в	г	д	е	ж	з
35	и	й	к	л	м	н	о	п
36	р	с	т	у	ф	х	ц	ч
37	ш	щ	ъ	ы	ь	э	ю	я

2. Таблица латинской кодировки

	0	1	2	3	4	5	6	7
00	`	´	^	~	¨	˝	°	ˇ
01	˘	¯	·	¸	˛	I	⟨	⟩
02	“	”	‿	‟	‚	–	—	
03	◦	ı	ȷ	ff	fi	fl	ffi	ffl
04	␣	!	"	#	$	%	&	’
05	(	)	*	+	,	-	.	/
06	0	1	2	3	4	5	6	7
07	8	9	:	;	<	=	>	?
10	@	A	B	C	D	E	F	G
11	H	I	J	K	L	M	N	O
12	P	Q	R	S	T	U	V	W
13	X	Y	Z	[	\	]	^	_
14	‘	a	b	c	d	e	f	g
15	h	i	j	k	l	m	n	o
16	p	q	r	s	t	u	v	w
17	x	y	z	{	|	}	~	-

3. Таблица кодировки вашингтонской кириллицы

	0	1	2	3	4	5	6	7
00	Њ	Љ	Џ	Э	І	Є	Ъ	Ѣ
01	њ	љ	џ	э	і	є	ѣ	ѣ
02	Ю	Ж	Й	Ё	Ѵ	Ѳ	S	Я
03	ю	ж	й	ё	ѵ	ѳ	s	я
04	¨	!	”	Ђ	˘	%	´	’
05	(	)	*	ђ	,	-	.	/
06	0	1	2	3	4	5	6	7
07	8	9	:	;	«	ı	»	?
10	ˇ	А	Б	Ц	Д	Е	Ф	Г
11	Х	И	J	К	Л	М	Н	О
12	П	Ч	Р	С	Т	У	В	Щ
13	Ш	Ы	З	[	“	]	Ь	Ъ
14	‘	а	б	ц	д	е	ф	г
15	х	и	j	к	л	м	н	о
16	п	ч	р	с	т	у	в	щ
17	ш	ы	з	–	—	№	ь	ъ

4. Подключение другого шрифта

В случае необходимости можно включить в систему какой-либо новый шрифт. Например, готический. Он находится в файлах `eufm10.*`. Нужно представить системе этот шрифт и присвоить ему имя, а затем выполнить команду, состоящую из этого имени

```
\font\eufrak = eufm10
.......
\eufrak
```

	0	1	2	3	4	5	6	7
00	𝔡	𝔡	𝔣	𝔣	𝔤	𝔨	𝔱	𝔲
01								
02			‘	’				
03								
04		!					&	'
05	(	)	*	+	,	−	.	/
06	0	1	2	3	4	5	6	7
07	8	9	:	;		=		?
10		𝔄	𝔅	ℭ	𝔇	𝔈	𝔉	𝔊
11	ℌ	ℑ	𝔍	𝔎	𝔏	𝔐	𝔑	𝔒
12	𝔓	𝔔	ℜ	𝔖	𝔗	𝔘	𝔙	𝔚
13	𝔛	𝔜	ℨ	[		]	^	
14		𝔞	𝔟	𝔠	𝔡	𝔢	𝔣	𝔤
15	𝔥	𝔦	𝔧	𝔨	𝔩	𝔪	𝔫	𝔬
16	𝔭	𝔮	𝔯	𝔰	𝔱	𝔲	𝔳	𝔴
17	𝔵	𝔶	𝔷			"		1

Так и хочется воспроизвести что-нибудь немецкое:

Der Stein Problem war was im Nieren
Ist neue Zeit annigiliren...

Yu. V. Linnik

5. Определение для строки со знаком копирайта

Эта строка должна делаться так же как обычное подстрочное примечание, но без знака примечания в тексте и в самой строке. Оказалось проще всего взять в LaTeX-е определение примечания и сократить до нужных потребностей. Получилось так

```
\long\def\copyrght#1{\insert\footins{\footnotesize
    \interlinepenalty\interfootnotelinepenalty
    \splittopskip\footnotesep
    \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
    \hsize\columnwidth \@parboxrestore
   {\rule{\z@}{\footnotesep}\ignorespaces \copyright\,\,
   {#1}\strut}}}
```

К сожалению, сейчас показать не могу, это определение должно быть сделано в стилевом файле.