Краткая инструкция по использованию TEX и LATEX

1. Общие сведения.

Следует иметь в виду, что полное описание TEX занимает более 500 страниц, поэтому в данной инструкции приведены лишь минимальные сведения.

Основные этапы работы с TEX, LATEX209 или LATEX2e таковы:

а) Надо изготовить текстовый файл (имя файла должно иметь расширение `.tex`), который содержит Ваш английский, русский или русско-английский текст, а также команды TEX. Например:

```
This is {\it the} example (пример):  $\pi^2$.  \bf Это кусочек (part) TEX файла.
```

Результат будет выглядеть следующим образом:

This is *the* example (пример): π^2. **Это кусочек (part) TEX файла.**

В этом примере команда `\it` включает курсивный шрифт, а фигурные скобки ограничивают область ее действия словом "the". Аналогично, команда `\bf` включает жирный шрифт. Запись `$\pi^2$` соответствует символу "пи", возведенному в квадрат. Эти и другие команды описаны в следующих разделах.

б) Запустить TEX. Он превратит Ваш текстовый файл в промежуточный DVI-файл (driver independent, имя файла будет иметь расширение `.dvi`). Этот файл содержит графический образ, созданный по исходному текстовому файлу, но не может быть напечатан или просмотрен непосредственно.

в) После того, как DVI-файл создан, можно:

1) Просмотреть на экране результаты ("view") с помощью программы `dviscr` или `dviwin`.
2) Напечатать результаты на принтере ("print") с помощью программы `dvidot` или `dvihplj`.
3) Создать файл в формате PostScript с помощью программы `dvips`.

Эти этапы обязательны как в случае использования TEX, так и LATEX. В действительности LATEX является просто некоторой надстройкой над TEX. При этом существуют команды, действующие *только* в TEX, действующие *только* в LATEX, и *общие* для TEX и LATEX.

2. Основные команды TEX.

Обычный текст (без формул) набирается обычным образом. Единственным существенным исключением являются кавычки, тире и специальные символы. Короткий дефис (-) набирается как один символ '–', черточка (–) набирается как два знака '–' '–' *подряд*, длинное тире (—) — как три знака '–' '–' '–'. Аналогично, две левые (' ') или правые (‘ ‘) кавычки *подряд* склеиваются в двойные кавычки (") или (“). Специальные символы $, %, &, # набираются как \$, \%, \&, \#. Ни один из этих символов, появившийся в тексте без "\", напечатан не будет. При этом символ "%" приведет к игнорированию следующих за ним символов вплоть до конца строки (это комментарий). Другие символы, скорее всего, вызовут сообщения об ошибках.

В конце файла *необходимо* поставить команду

```
\bye
```

(это единственная обязательная команда). Следует иметь в виду, что TEX различает большие и малые буквы в командах, поэтому нельзя писать "\BYE".

Чтобы подчеркнуть какое-либо слово или группу слов, можно использовать команду:

```
\underbar {слово1 слово2 }
```

В самом начале файла можно поставить команду

```
\magnification \magstep1
```

(или \magstephalf, \magstep2, \magstep3, \magstep4, \magstep5). Это увеличит размер букв и все расстояния в 1.2 (или 1.1, 1.4, 1.6, 2, 2.4) раза. Эта команда может появиться только один раз (нельзя иметь разное увеличение в разных частях документа). Размеры текста (без учета полей) определяются командами

```
\hsize 17 cm       (ширина)
\vsize 25 cm       (высота)
```

В любых командах, определяющих размер, могут использоваться следующие единицы измерения

```
cm mm in        (сантиметр миллиметр дюйм)
pt              (точка)
em ex           (ширина буквы M высота буквы x)
```

Перед каждой единицей измерения может появиться слово `true` — соответствующая длина не будет увеличиваться командой \magnification. Например:

```
\hsize 5.3 true in       \vsize 88.666 mm
```

Правила форматирования текста.

TEX полностью игнорирует все переходы на новую строку и избыточные (числом более одного) пробелы в Вашем файле. Он разбивает Ваш текст на абзацы и форматирует их по своему разумению. Границей между абзацами является команда

```
\par       или пустая (пробельная) строка.
```

TEX умеет переносить слова. Довольно редко встречаются слова, которые TEX переносить не умеет. Если такое слово встречается в Вашем документе часто, целесообразно в начале текста написать

```
\hyphenation{пе-ре-нос}
```

Если оно встречается редко, его следует напечатать (там, где оно стоит в тексте) в виде `пе\-ре\-нос`.

Неприятной особенностью функции, обеспечивающей переносы, является то, что она перестает переносить слово, которое примыкает к какому-либо символу, не являющемуся буквой. Например, она не знает, как перенести слова "А.Эйнштейн", "рождения-уничтожения", хотя и умеет переносить "Эйнштейн", "рождения", "уничтожения".

Форма абзаца

Форма простого абзаца определяется следующими командами

`\parindent 5.5 mm`	величина абзацного отступа для этого и последующих абзацев (может быть отрицательной)
`\noindent`	отсутствие отступа в *этом* абзаце
`\leftskip -8 mm`	левые поля для этого и последующих абзацев (левая граница текста сместится на -8 mm вправо, т.е. на 8 mm влево)
`\rightskip 2 cm`	правые поля для этого и последующих абзацев (правая граница текста сместится на 2 cm влево)
`\baselineskip 12pt`	расстояние между строками

Предупреждение: не пользуйтесь командами \leftskip и \rightskip для того, чтобы задать левые и правые поля всего документа. Для этого вполне достаточно команд \hsize и \hoffset (см. далее). Команды \leftskip и \rightskip нужны для того, чтобы внутри документа временно изменить левый или правый отступ.

Форма абзаца с выемкой (под картинку) определяется как вышеприведенными командами, так и командами:

`\hangindent -5 cm \hangafter 4`	в абзаце после 4 строк будет прямоугольная выемка ширины 5 см справа от текста.
`\hangindent 5 cm \hangafter 4`	в абзаце после 4 строк будет прямоугольная выемка ширины 5 см слева от текста.
`\hangindent -5 cm \hangafter -4`	в абзаце первые 4 строки образуют прямоугольную выемку ширины 5 см справа от текста.
`\hangindent 5 cm \hangafter -4`	в абзаце первые 4 строки образуют прямоугольную выемку ширины 5 см слева от текста.

Форма абзаца сложной формы определяется командой

```
\parshape 3 2cm 12cm 3cm 14 cm -1cm 5cm
```

здесь 3 — число строк в абзаце, 2 cm и 12 cm — отступ слева и длина первой строки, (3cm 14 cm) и (-1cm 5cm) — аналогичные данные для второй и третьей строки. Если в абзаце окажется менее трех строк, будет использована часть данных, если более трех строк — для лишних строк (начиная с четвертой) будут использованы данные последней (третьей) строки.

Если TEX не может отформатировать абзац в соответствии со своими представлениями о красоте, он пишет предупреждения двух типов: `underfull hbox` (строка несколько короче ширины текста) и `overfull hbox` (строка несколько вылезает за правую границу). К сожалению, помимо сообщений, он отмечает каждую слишком длинную строку черным прямоугольником, который *появится* и при печати. В этом случае лучше всего несколько поменять текст, чтобы сделать возможным форматирование. Если это нежелательно, все черные прямоугольники можно убрать одной командой

```
\overfullrule 0 mm
```

помещенной в начало Вашего файла. Наконец, можно изменить параметр, определяющий, насколько TEX может растягивать пробелы между словами:

```
\tolerance 900
```

Опыт показал, что числа порядка 1000 для нормальных текстов приводят к исчезновению слишком длинных строк. Злоупотреблять этой возможностью *не следует*.

В текст можно ввести горизонтальные и вертикальные промежутки фиксированной длины

`\hskip 5 pt`	горизонтальный пробел, в этом месте может произойти переход на новую строку (при этом "недоиспользованная" часть пробела не перейдет на новую строку).
`\vskip -0.5 cm`	вертикальный пробел, в этом месте может произойти переход на новую страницу (при этом "недоиспользованная" часть пробела не перейдет на новую страницу). В данном случае пробел отрицательный, т.е. последующая строка налезет на предыдущую.
`\kern 33 mm`	в середина абзаца — горизонтальный пробел, между абзацами — вертикальный. Переносов в этом месте не может быть.
`\ ( '\' и ' '(пробел) )` ..	пробел размером с букву, в этом месте возможен перенос.
`~ (символ '~')`	пробел размером с букву, перенос в этом месте невозможен.

Кроме того, существуют команды

```
\break  и  \eject
```
первая вызывает принудительный переход на новую строку внутри текущего абзаца (не обрывая его), вторая — принудительный переход на новую страницу.

Группы. Локальность команд.

Существуют команды, действующие в заранее определенной области (команда `\noindent` действует в пределах одного абзаца). Другие команды просто переустанавливают некоторые параметры (`\leftskip 2cm` установит левые поля текущего и последующих абзацев). Область действия таких команд можно ограничить, если использовать фигурные скобки. Группой называется все, заключенное между фигурными скобками. Большинство команд TEX действуют только в пределах данной группы. Допускается вложение групп друг в друга.

Строки специальной формы.

Существуют команды, позволяющие напечатать текст в одну строку (для печатания заголовков и т.п.)

`\centerline{текст}` ..... текст будет центрирован
`\leftline{текст}` ..... текст будет выровнен по левому краю
`\rightline{текст}` ..... текст будет выровнен по правому краю
`\line{текст}` ..... текст будет напечатан отдельной строкой

При использовании этих команд текст может содержать как пробелы фиксированной длины, так и промежутки неопределенной длины, определяемые командами

`\hfil` ..... этот пробел в строке действует как пружина неограниченной длины
`\hfill` ..... этот пробел в строке действует как пружина неограниченной длины с жесткостью бесконечно большей, чем у `\hfil`
`\hss` ..... этот пробел в строке действует как пружина неограниченной длины, причем длина может быть и отрицательной

Между абзацами можно вставлять вертикальные пружины, действующие по аналогичным правилам

```
\vfil, \vfill
```

Смысл этих пружин можно понять на таком примере — команда `\rightline{текст}` эквивалентна команде `\line{\hfill текст}`. Ясно, что использование большего числа пружин позволит получить строку произвольной конфигурации. Кроме того, пружины целесообразно использовать рядом с командами `\break`, `\eject`.

При составлении списков удобно пользоваться командами

```
\item{3).} текст третьего пункта
\itemitem{7.a} текст подпункта "a" седьмого пункта
```

Команда `\item` сдвигает левую границу текста на величину абзацного отступа, а содержимое фигурных скобочек выносят влево от новых полей текста. Внутри (и только внутри) каждого такого пункта можно ввести подпункт командой `\itemitem`, которая действует аналогично команде `\item`.

Окружение основного текста

Помимо основного текста TEX помещает на странице верхнюю и нижнюю строки. Если Вы не определите их содержание командами

```
\footline{текст}
\headline{текст}
```

то верхняя строка будет пустой, а нижняя будет содержать номер страницы. Команда

```
\nopagenumbers
```

отменяет нумерацию страниц. Номер страницы содержится в переменной `\pageno`, поэтому команда

```
\pageno 123   или   \pageno=123
```

устанавливает номер текущей страницы равным 123. Чтобы вставить в текст верхней или нижней строки номер страницы, надо в соответствующем месте напечатать

```
\the\pageno
```

TEX позволяет делать примечания с помощью команд

```
\footnote{метка}{текст примечания}
```

Для примечаний, маркируемых звездочками, в качестве метки следует ставить просто `**` (число звездочек произвольно), т.к. в стандартном шрифте звездочка `*` поднята вверх: *. Для примечаний, маркируемых цифрами, в качестве метки следует использовать

```
$^{3)}$
```

чтобы поднять метку 3) вверх: [3)] (см. главу о печатании формул).

Команда `\footnote` вставляются непосредственно в текст.

Кроме команд `\leftskip` и `\rightskip` (которые определяют левые и правые поля), существует команда

```
\topskip
```

определяющая верхние поля для каждой страницы. Наконец, существует возможность сдвигать всю сформированную страницу как целое командами

`\voffset` и `\hoffset`
в вертикальном и горизонтальном направлениях соответственно.

Переключение шрифтов.

Переключение на разные типы шрифтов (фонтов) одного размера (определенного командой \magnification) осуществляется командами

`\rm`	нормальный
`\bf`	жирный
`\it`	курсив
`\sl`	наклонный
`\tt`	типа пишущей машинки

При этом будут использоваться так называемые десятиточечные фонты, шкалированные (увеличенные) в соответствии с командой \magnification. Кроме того, команды

`\sevenrm` `\fiverm`

включат соответственно семиточечный и пятиточечный стандартный шрифт (размер букв пропорционален количеству точек).

Чтобы использовать другие шрифты, необходимо их подключить (определить). Это достигается командами вида:

```
\font\twelverm=cmr12
```

Эта команда подключает (определяет) двенадцатиточечный стандартный шрифт. Переключение на него достигается командой

```
\twelverm
```

Название шрифта может быть любым набором латинских *букв* (можно написать `\font\twerom=cmr12` и `\twerom` будет переключать шрифт). Тем не менее разумно использовать стандартные имена шрифтов (sevenrm, ninebf, fiveit и т.д.).

Ниже приведены доступные в данной версии шрифты, сгруппированные по типам

Нормальные (Roman) : Example Пример

```
cmr5, cmr6, cmr7, cmr8, cmr9, cmr10, cmr12, cmr17
```

Жирные (Bold) : **Example Пример**

```
cmbx5, cmbx6, cmbx7, cmbx8, cmbx9, cmbx10, cmbx12
```

Курсивные (Italic) : *Example Пример*

```
cmti7, cmti8, cmti9, cmti10, cmti12
```

Наклонные (Slanted) : *Example Пример*

```
cmsl8, cmsl9, cmsl10, cmsl12
```

Типа пишущей машинки (Type writer) : `Example Пример`

```
cmtt8, cmtt9, cmtt10, cmtt12
```

Существуют также и другие группы шрифтов

Упрощенные (Sans Serif)

```
cmss8, cmss9, cmss10, cmss12, cmss17
```

Курсивные упрощенные (Sans serif italic)

```
cmsi8, cmsi9, cmsi10, cmsi12, cmsi17
```

Математические (Math italic)

```
cmmi5, cmmi6, cmmi7, cmmi8, cmmi9, cmmi10, cmmi12
```

Другие

```
cmbxsl1, cmbxti1, cmitt10, cmssbx1, cmssdc1, cmmib10, cmsltt1
```

Список всех шрифтов можно найти в корне директории EMTEX в файле fontlist. Просмотреть все буквы какого-либо шрифта можно с помощью файла showfont.tex, расположенного там же.

Наконец, можно подключить фонт с заданным увеличением (тем самым можно фактически получить разное увеличение шрифтов в разных частях документа). Это делается командой

```
\font\bigital=cmti8 scaled \magstep3
```

Стандартных названий для увеличенных таким образом фонтов не существует, Вы можете использовать любые имена. Не следует злоупотреблять возможностями выбора фонтов, поскольку использование очень большого количества шрифтов приведет к переполнению памяти и TEX работать не будет.

Переменные в TEX

TEX позволяет ввести до (приблизительно) 200 целых переменных и до (приблизительно) 200 переменных с размерностью длины. Целая переменная вводится командой

```
\newcount\schet
```

где `\schet` — произвольное имя, состоящее из латинских *букв*. Аналогично,

```
\newdimen\dlin
```

определяет переменную размерности длины с именем \dlin. С переменными можно производить арифметические операции

\schet -3 (или \schet=-3)	\dlin 3.5cm (или \dlin=3.5cm)	присвоение
\advance\schet by -3	\advance\dlin by 3.5cm	сложение или вычитание
\multiply\schet by 4	\multiply\dlin by 7	умножение (только на целые)
\divide\schet by 2	\divide\dlin by 3	деление (только на целые)

Значение целой переменной печатается командой

```
\the\schet
```

Значение переменной с размерностью длины может быть использовано в командах типа

```
\hskip\dlin
```

Определение новых команд

Если в Вашем тексте часто появляется одна и та же последовательность символов и команд, то целесообразно определить новую команду, которая заменит всю эту последовательность. Например, команда

```
\def\vvv{\vskip 1 true mm \line{--- \hfill --- \hfill ---}\vskip 1 true mm }
```

определит новую команду, которая рисует три тире, окруженные дополнительными пробелами по 1 mm сверху и снизу. После такого определения достаточно написать

```
\vvv
```

и три тире появятся в Вашем документе.

Более того, команда может иметь параметры. Например, если Вам необходимо, чтобы каждый раздел Вашего документа начинался курсивным заголовком, выровненным по правому краю, достаточно написать

```
\newcount\nrazd       \nrazd=0
\def\zagol#1{\vskip 2 ex \advance\nrazd by 1
                \rightline{\it\the\nrazd).#1} \vskip 0.5 ex}
```

Очередной заголовок вводится командой

```
\zagol{Текст заголовка}
```

Благодаря введенной переменной \nrazd нумерация разделов ведется автоматически. Обратите внимание на фигурные скобочки вокруг текста заголовка. TEX рассматривает как параметр *первый объект* после команды. В данном случае объектом является группа, заключенная в скобки. В отсутствие скобок первый объект — это первая буква первого слова после команды.

Команда может иметь до 9 параметров. Например

```
\def\aaa#1#2#3{\line{#1 \hfill #2 \hfill #3}}
```

После чего ей можно воспользоваться так:

```
\aaa{Текст слева}{Текст посередине}{Текст справа}
```

Результат действия этой команды очевиден.

На этом краткий обзор обычных команд текстового режима TEX завершается. Существуют и другие команды (позволяющие рисовать таблицы, линии и т.п.). Они требуют довольно осторожного обращения и изложены в дополнительной главе 5.

Печатание формул изложено в следующей главе.

3. Печатание математических формул в TEX и LATEX

Все, сказанное в этой главе, справедливо как для TEX, так и для LATEX.

Все формулы должны быть заключены в "математические скобки". Они бывают двух видов:

```
$ ... $     и     $$ ... $$
```

для печатания формул внутри текста и отдельной строкой соответственно. Внутри математических скобок TEX работает в "математической моде" и не понимает большинство команд обычной (текстовой) моды, аналогично, команды математической моды не должны появляться в текстовой.

Простейшие команды

Все переменные ($a\ b\ \dots\ x\ y\ z$; $A\ B\ \dots\ X\ Y\ Z$) и знаки арифметических операций ($+ - */ =$) набираются как текстовой моде. Возведение в степень реализуется командой ^ , а нижний индекс — командой _ , например

x^{23}	x^{23}	A_{ilm}	A_{ilm}
$g^{(o)}_{ik}$	$g^{(o)}_{ik}$	${g_{ik}}^{(o)}$	${g_{ik}}^{(o)}$

Используя фигурные скобки, можно получить практически любой результат.

Принято символы некоторых функций (sin(x), cos(x) и т.п.) набирать нормальным (не курсивным) шрифтом, для этого существуют команды

```
\sin   \cos   \exp   \log   \ln   \max   \min   \det   \arcsin   \arccos   \dim
```

Переход на шрифт, в котором заглавные латинские буквы являются рукописными, осуществляется командой \cal

```
$ \cal ABCDEFGHIJKLMNOPQRSTUVWXYZ $
```

$\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$

Греческие буквы можно напечатать командами

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

Напоминание: эти команды работают только в математической моде, т.е. между скобками $ ... $ или $$ ... $$.

Операции отношения набираются командами:

>	$>$	<	$<$	=	$=$	\not=	$\neq$
\leq	$\leq$	\geq	$\geq$	\ll	$\ll$	\gg	$\gg$
\sim	$\sim$	\simeq	$\simeq$	\approx	$\approx$	\equiv	$\equiv$
\subset	$\subset$	\supset	$\supset$	\subseteq	$\subseteq$	\supseteq	$\supseteq$
\in	$\in$	\not\in	$\notin$	\ni	$\ni$		

Приставка \not может перечеркнуть любую из операций отношения.

Символы, помещаемые над буквами, вводятся следующими командами

\hat a	$\hat a$	\dot a	$\dot a$	\ddot a	$\ddot a$	\vec a	$\vec a$
\bar a	$\bar a$	\tilde a	$\tilde a$	\hat{abc}	$\hat{abc}$	\widehat{abc}	$\widehat{abc}$
\widetilde{abc}	$\widetilde{abc}$						

Символы, помещаемые в кружок, вводятся командами

\oplus	$\oplus$	\ominus	$\ominus$	\otimes	$\otimes$	\odot	$\odot$
\oslash	$\oslash$	\bigoplus	$\bigoplus$	\bigotimes	$\bigotimes$	\bigodot	$\bigodot$

Разнообразные точки рисуются командами

\cdots	$\cdots$	\ldots	$\ldots$	\vdots $\vdots$	\ddots $\ddots$	\cdot $\cdot$

Стрелочки разных видов рисуются командами

\leftarrow	$\leftarrow$	\Leftarrow	$\Leftarrow$	\rightarrow	$\rightarrow$	\Rightarrow	$\Rightarrow$
\longleftarrow	$\longleftarrow$	\Longleftarrow	$\Longleftarrow$	\longrightarrow	$\longrightarrow$	\Longrightarrow	$\Longrightarrow$
\uparrow	$\uparrow$	\Uparrow	$\Uparrow$	\downarrow	$\downarrow$	\Downarrow	$\Downarrow$
\leftrightarrow	$\leftrightarrow$	\Leftrightarrow	$\Leftrightarrow$	\updownarrow	$\updownarrow$	\Updownarrow	$\Updownarrow$
\nearrow	$\nearrow$	\searrow	$\searrow$	\swarrow	$\swarrow$	\nwarrow	$\nwarrow$

Треугольники вводятся командами

\triangleleft	$\triangleleft$	\triangleright	$\triangleright$	\bigtriangleup	$\bigtriangleup$	\bigtriangledown	$\bigtriangledown$

Скобки рисуются командами

(	(	)	)	[	[	]	]
\{	$\{$	\}	$\}$	\big(	$\big($	\big)	$\big)$
\big[	$\big[$	\big]	$\big]$	\big\{	$\big\{$	\big\}	$\big\}$
\Big(	$\Big($	\Big)	$\Big)$	\Big[	$\Big[$	\Big]	$\Big]$
\Big\{	$\Big\{$	\Big\}	$\Big\}$	\bigg(	$\bigg($	\bigg)	$\bigg)$
\bigg[	$\bigg[$	\bigg]	$\bigg]$	\bigg\{	$\bigg\{$	\bigg\}	$\bigg\}$
\Bigg(	$\Bigg($	\Bigg)	$\Bigg)$	\Bigg[	$\Bigg[$	\Bigg]	$\Bigg]$
\Bigg\}	$\Bigg\}$	\Bigg\}	$\Bigg\}$				

Предупреждение: в TEX есть возможность окружить произвольное выражение скобками переменной высоты, соответствующей его размерам (см. ниже). Однако, в ряде случаев скобки фиксированной высоты оказываются удобнее (например, если открывающаяся и закрывающаяся скобки в длинной формуле оказываются на разных строчках).

Другие символы вводятся командами

\times	$\times$	\ast	$\ast$	\star	$\star$	\div	$\div$
\pm	$\pm$	\mp	$\mp$	\bullet	$\bullet$	\circ	$\circ$
\bigcirc	$\bigcirc$	\perp	$\perp$	\parallel	$\parallel$	\angle	$\angle$
\cap	$\cap$	\cup	$\cup$	\vee	$\vee$	\wedge	$\wedge$

`\forall`	$\forall$	`\exists`	$\exists$	`\infty`	∞	`\partial`	∂
`\nabla`	∇	`\hbar`	$\hbar$	`\diamond`	$\diamond$	`\to`	$\to$

По естественным причинам перечислены лишь наиболее "полезные" символы.

Команды, связанные с особым расположением символов

Знаки суммы, произведения, интеграла, интеграла по контуру и предела могут быть получены командами, соответственно

`\sum` `\prod` `\int` `\oint` `\lim`

При этом верхние и нижние индексы рассматриваются как пределы (суммирования, интегрирования и т.д.) и могут быть помещены: а) над и под знаком б) справа от него (как обычные индексы). TEX сам выбирает способ размещения, но если он Вас не устраивает, команды `\limits` и `\nolimits` могут изменить его:

`\sum\nolimits_{i=0}^{\infty}[i(i+1)]^{-2}` $\sum_{i=0}^{\infty}[i(i+1)]^{-2}$

`\sum\limits_{i=0}^{\infty}[i(i+1)]^{-2}` $\sum\limits_{i=0}^{\infty}[i(i+1)]^{-2}$

Большую дробь с горизонтальной чертой деления можно получить командой `\over`, а размещение одного объекта над другим (фактически дробь без черты деления) — командой `\atop`

`x={a+b^2 \over c_2+d}` $x=\frac{a+b^2}{c_2+d}$

`A=\Big({a_1 \atop a_2}\Big)` $A=\Big(\begin{smallmatrix}a_1\\a_2\end{smallmatrix}\Big)$

`\sum_{i=1 \atop j,k=0}^\infty` $\sum\limits_{\substack{i=1\\j,k=0}}^{\infty}$

Обратите внимание на фигурные скобки, они *необходимы* для того, чтобы TEX знал, что является числителем, а что — знаменателем. Числитель (знаменатель) можно сдвинуть влево (или вправо) к краю дроби, если справа (или слева) от набора команд, формирующих числитель, вставить команду

`\hfill`

Она работает аналогично команде `\hfill` в текстовой моде.

Корень из некоторого выражения извлекается командами

`\sqrt{выражение}` и `\root{степень}\of{выражение}`

Например

`\sqrt{x^2+y^2}` $\sqrt{x^2+y^2}$

`\root{n+2}\of{ a-{b \over c+d} }` $\sqrt[n+2]{a-\frac{b}{c+d}}$

Окружить некоторое выражение скобками переменной высоты можно с помощью команд

`\left(` `\left[` `\left\{` `\left|` `\left\|` `\left.`
`\right(` `\right[` `\right\}` `\right|` `\right\|` `\right.`

Эти команды могут встречаться только парой (каждому \left должен соответствовать \right), но типы скобок слева и справа могут не совпадать. Символ `|` соответствует вертикальной черте, символ `\|` — двойной вертикальной черте, а точка — пустому ограничителю.

Простой пример

`V(x)=\left\{ {0, x<0 \atop 1, x>0} \right.` $V(x)=\left\{\begin{smallmatrix}0,x<0\\1,x>0\end{smallmatrix}\right.$

Эти команды имеют две неприятные особенности: во-первых, при разбиении длинной формулы на несколько строк приходится вставлять пустые ограничители (чтобы на каждой строке была пара команд), при этом размеры соответствующих скобок на разных строках могут оказаться разными. Во-вторых, для формул, сильно возвышающихся над строчкой, скобки не приподнимаются вверх, а рисуются симметрично вверх и вниз (т.е. слишком свисают вниз).

Провести под (над) некоторым выражением черту или поместить под (над) некоторым выражением фигурную скобку можно с помощью команд

`\underline` `\overline` `\overbrace` `\underbrace`

причем скобка может быть снабжена индексами

`\overline{x}+ \overline{y^2}` $\overline{x}+\overline{y^2}$

`\underbrace{x_1 \cdots x_n}_{a+i}` $\underbrace{x_1\cdots x_n}_{a+i}$

Обратите внимание, что надчеркивание происходит на разной высоте, в зависимости от высоты выражения. Этого можно избежать, если в каждое из выражений вставлять невидимый символ с максимальной высотой. Этот символ вставляется командой

`\mathstrut`

Команды, связанные с выравниванием

Нарисовать матрицу можно командой

```
$$ ...   \matrix{
   a11   &   a21   &   a31   &   ...   &   an1  \cr
   a12   &   a22   &   a32   &   ...   &   an2  \cr
   ...
   a1m   &   a2m   &   a3m   &   ...   &   anm  \cr
} ... $$
```

причем вместо a_{ij} можно подставить что угодно. Элемент матрицы можно сдвигать к левому или правому краю области, отведенной под него, командой \hfill (аналогично числителю или знаменателю дроби). Полученная матрица является полноправной частью формулы (ее можно приравнять чему-либо, извлечь корень, окружить скобками).

Номера формул вставляются командами

```
$$ формула \eqno (59a) $$
$$ формула \leqno (2.77) $$
```

при нумерации формул справа и слева соответственно. Все, находящееся *справа* от команд \eqno и \leqno, рассматривается как номер формулы. При этом для формул с большой высотой номер размещается на середине высоты.

Если необходимо выровнять по *одному* определенному месту (обычно по знаку равенства) несколько формул, расположенных одна над другой, нужно использовать команду

```
$$   \eqalign  {
     слева1   &   справа1   \cr
     слева2   &   справа2   \cr
     ...
     слеваm   &   справаm   \cr
}  $$
```

где "слева" и "справа" означают части формул слева и справа от точки выравнивания.

К сожалению, *внутри* команды \eqalign *нельзя* использовать команды \eqno и \leqno. Если все же есть необходимость пронумеровать эти выровненные формулы, надо использовать команды

```
$$   \eqalignno  {
     слева1   &   справа1   & (39a)   \cr
     слева2   &   справа2   &    \cr
     ...
     слеваm   &   справаm   &    (39d) \cr
}  $$
```

при нумерации формул справа, и

```
$$   \leqalignno  {
     слева1   &   справа1   &   \cr
     слева2   &   справа2   &   (666a)  \cr
     ...
     слеваm   &   справаm   &    (452s) \cr
}  $$
```

при нумерации слева. Все, находящееся между вторым знаком & и \cr, интерпретируется как номер формулы. Если для данной формулы номер не нужен, достаточно поставить пробел.

Напоминание: TEX игнорирует излишние пробелы и переходы на новую строку, поэтому в приведенных выше командах содержимое фигурных скобок можно распределять по строкам произвольным образом. В приведенных примерах каждая строка матрицы (каждая новая формула) записывается на отдельной строке только для удобства чтения.

TEX сам устанавливает размеры степеней и индексов в формулах. Его можно поправить командами

```
\displaystyle    \scriptstyle    \scriptscriptstyle
```

Они работают в точности как команды переключения шрифтов.

TEX сам рисует формулы, и в большинстве случаев делает это удачно. Иногда, однако, бывает необходимо вставить в формулу текстовое пояснение или пробел определенной длины. Это можно сделать командами, соответственно

```
\hbox{текст вставки }
\qquad      \quad       \   ('\' и ' ')      \;       \>        \,      \!
 ]    [       ] [         ] [                ][       ][        ][      ‖
```

Пробелы даны в порядке убывания, причем последний из них — *отрицателен*.

4. Основные команды LATEX209 и LATEX2e.

Предупреждение: во всех случаях, когда это возможно, следует пользоваться TEX, а не LATEX. Это связано с тем, что в LATEX заранее определено большое количество фонтов и макросов, часто тех, которыми Вы никогда пользоваться не будете. Поэтому вероятность переполнения памяти для большого и сложного документа очень высока.

Кроме того, команды TEX лаконичнее, гораздо реже интерферируют между собой и (в силу своей простоты), как правило, исполняются именно так, как Вы этого ожидаете.

Команды же LATEX представляют собой достаточно разветвленные макросы, которыми во вполне тривиальных случаях пользоваться, быть может, и удобнее, чем командами TEX, но в нетривиальных дают поведение, которое сложно интерпретировать.

Наконец, TEX гарантирует гораздо большую стандартизацию, чем LATEX (нередко взятый откуда-либо готовый файл в формате LATEX бесполезен ввиду отсутствия или несовместимости используемого стилевого файла).

Из сказанного очевидно, что автор является сторонником использования TEX, и надо быть готовым к тому, что изложение LATEX будет еще более неполным, чем изложение TEX.

Простейшие команды

Каждый LATEX209 файл обязан начинаться командой

`\documentstyle[11pt]{article}`

Вместо `[11pt]` может стоять `[12pt]` или `[]` (*без пробела* между скобками), вместо `article` может стоять `book`, `report`, `letter`. Это определяет, какие фонты и команды будут доступны из данного документа, т.е. так называемый "стиль" документа.

В LATEX2e команда `\documentstyle` заменена на

`\documentclass{article}`

Это единственное замечание, которое автор считает нужным сделать о LATEX2e.

После `\documentstyle` может стоять "преамбула" — набор команд (определяющих размер страницы и т.п.)

После преамбулы должна стоять команда

`\begin{document}`

После этой команды начинается собственно Ваш текст. Кончаться текст должен командой

`\end{document}`

Эти три команды *необходимы* для работы LATEX. Многие команды LATEX имеют аналогичную структуру (`\begin{название}` $\cdots$ `\end{название}`).

Форматирование текста. Шрифты

Правила набора и форматирования текста в основном аналогичны правилам TEX. Более того, ряд команд являются общими для TEX и LATEX:

`\leftskip` `\rightskip` `\topskip` `\hoffset` `\voffset`
`\parindent` `\noindent` `\hangindent` `\hangafter` `\parshape` `\par`
`\hsize` `\baselineskip`

Эти команды работают так же, как в TEX, но может возникнуть их интерференция с командами LATEX. Единицы измерения, используемые в них, совпадают с единицами TEX. Эти команды лучше помещать в тексте.

Размеры текста определяются командами

`\textwidth` `\textheight`

Эти две команды лучше помещать в преамбуле. Ниже приведена информация об области действия команд: `\hsize` работает в тексте, не работает в преамбуле; `\vsize` в тексте действует только на первую страницу, не работает в преамбуле; `\textwidth` работает только в преамбуле; команда `\textheight` работает в преамбуле, в тексте действует на все страницы, кроме первой; команды `\voffset, \hoffset` работают как в преамбуле, так и в тексте.

Горизонтальные и вертикальные промежутки, а также переходы на новую строку и страницу вводятся командами TEX

`\hskip` `\vskip` `~` `\ ('\' и ' ')` `\hfil` `\hfill` `\vfil` `\vfill` `\break` `\eject`

Можно создать целую группу строк специальной формы командами

`\begin{center}` строка_1 `\\` строка_2 `\\` $\cdots$ строка_m `\end{center}`
`\begin{flushleft}` строка_1 `\\` строка_2 `\\` $\cdots$ строка_m `\end{flushleft}`
`\begin{flushright}` строка_1 `\\` строка_2 `\\` $\cdots$ строка_m `\end{flushright}`

в первом случае строки будут центрированы, во втором — выровнены по левому краю, в третьем — по правому.

Нумерацию страниц можно отменить командой

`\pagestyle{empty}`

Неприятной особенностью этой команды является ее интерференция с командой `\maketitle` (см. ниже).

Двухколоночный вывод можно получить командой

`\twocolumn`

При этом расстояние между столбцами определяется командой

`\columnsep 0.8 cm`

При двухколоночном выводе команды типа `\rightskip, \hsize` определяют размеры колонки, а не всего текста. Выровнять обе колонки по нижнему краю можно командой

`\flushbottom`

Все эти команды лучше помещать в преамбуле.

Переключение шрифтов. В LATEX заранее определено большое количество шрифтов разного типа и разного размера. Переключение размера шрифтов осуществляется командами

```
\tiny \scriptsize \footnotesize \small \normalsize \large \Large \LARGE
\huge \Huge
```

эти команды работают аналогично командам переключения шрифтов в TEX, их выполнение сопровождается автоматическим переключением на стандартный (Roman) шрифт. Переключение типа шрифтов реализуется командами

```
\rm   \bf   \it   \sl   \tt   \sf
```

Обозначения такие же, как в TEX, \sf соответствует упрощенным шрифтам. В маловероятном случае нехватки существующих в LATEX шрифтов можно использовать команду

```
\font ··· scaled ···
```

которая работает в точности как в TEX.

Новая команда может быть определена двумя способами. Команда типа команды TEX определяется так

```
\newcommand{\vvv}[2]{{\large \it #1 :  }\hfill \par
   {\normalsize \bf #2 }\hfill \par }
```

В квадратных скобках указано число параметров. Эту команду можно использовать так

```
\vvv{заголовок}{комментарий}
```

Команда типа команды LATEX определяется так

```
\newenvironment{ddddd}{\begin{flushleft}\small \tt }{\end{flushleft}}
```

И может использоваться так

```
\begin{ddddd}
строка программы N 1 \\строка программы N 2 \\··· строка программы N m
\end{ddddd}
```

Команды, связанные с нумерацией.

В Latex существует две команды для составления списков

```
\begin{itemize}
\item элемент1  \item элемент2  ···  \item элементm
\end{itemize}
```

или

```
\begin{enumerate}
\item элемент1  \item элемент2  ···  \item элементm
\end{enumerate}
```

В качестве элементов могут выступать предложения произвольной длины. В первом случае каждый пункт метится кружком, во втором — нумеруется арабской цифрой.

Сноски вводятся командой

```
\footnote{текст сноски}
```

и нумеруются автоматически.

Особенностью стиля "article" является возможность сооружения заголовка и автоматической нумерации разделов документа. Командами

```
\title{заголовок}   \author{автор1 \and автор2 \and автор3}   \date{12 июня 1987}
```

определяется вид заголовка. Эти команды лучше поместить в преамбуле. Если не определить дату, будет использовано текущее время компьютера. Чтобы заголовок был напечатан, в начало текста надо поместить команду

```
\maketitle
```

Разделы в статье создаются командами:

```
\section{название}  \subsection{название}
\subsubsection{название}  \paragraph{название}
```

они приведены в порядке вложенности. Первые три вида автоматически снабжаются номерами. Выбор шрифтов для названий разделов осуществляется автоматически, но командой переключения шрифта, помещенной в текст названия шрифт можно изменить. (Это относится и к шрифтам заголовка).

Команда

```
\tableofcontents
```

сооружает оглавление статьи. Особенностью команды является то, что для того, чтобы гарантировать правильность оглавления, необходим *двукратный* прогон LATEX. При этом следует читать сообщения, которые выдает LATEX.

Существует возможность соорудить список литературы полуавтоматическим путем. Именно, в местах появления ссылок на источник следует ставить

```
\cite{Jonson}  или  \cite{Иванов}  или  \cite{Петров}
```

В фигурных скобках находится произвольная метка, позволяющая Вам и LATEX определить, о каком источнике идет речь. Привязка метки к номеру и сооружение списка литературы осуществляется командами

```
\begin{thebibliography}{3}
   \bibitem {Иванов} ссылка на Иванова
   \bibitem [!..?]{Jonson} Jonson reference
```

```
\bibitem {Петров} ссылка на Петрова
\end{thebibliography}
```

Число в фигурных скобках должно совпадать с числом ссылок. При этом ссылка на Иванова будет нумероваться как [1], ссылка на Петрова — как [2], а ссылка на Jonson — как [!..?]. Для выполнения этой команды также может потребоваться двукратный прогон LATEX.

Наконец, нумерация формул может также производиться автоматически, если вместо обычных математических скобок $$ ... $$ использовать команды

```
\begin{equation}... \end{equation}
```

каждая такая формула нумеруется справа, текущий номер формулы увеличивается на единицу. Кроме того, если в этой формуле поставить команду

```
\label{energiya}
```

то метка станет символическим обозначением номера формулы, и команда

```
\ref{energiya}
```

будет просто печатать этот номер.

Для выполнения этих команд также может потребоваться двукратный прогон LATEX.

Таблицы и картинки.

Удобнее всего нарисовать таблицу командой

```
\begin{tabular}{r ¦ c ¦¦ l r ¦ p {5cm }}\hline \hline
    a_11 & a_12 & a_13 & a_14 & b_1 \\ \hline
    a_21 & a_22 & a_23 & a_24 & b_2 \\ [5cm]
    a_31 & a_32 & a_33 & a_34 & b_3 \\
    a_41 \multicolumn{3}{¦¦¦ r ¦¦¦ }{a_4(2,3,4)}& b_4 \\
    a_51 & a_52 & a_53 & a_54 & b_5 \\ \cline{2-3}
    a_61 & a_62 & a_63 & a_64 & b_6 \\ \hline
\end{tabular}
```

Этот пример иллюстрирует возможности команды tabular. Заголовок определяет количество колонок (оно совпадает с количеством букв "r","c","l","p") буква "c" означает, что соответствующая колонка центрирована, "r" — выровнена по правому краю, "l" — про левому. Буква "p" соответствует фразе произвольной длины, которая помещается в абзац указанной ширины. Символ ¦ указывает на вертикальную черту между колонками, два символа — на двойную черту. Команда `\hline` дает горизонтальную черту во всю ширину таблицы, а `\cline` — горизонтальную черту между указанными номерами колонок. (`\cline{2-2}` проведет черту под одной второй колонкой). Из сказанного ясно, что элементы a_{ij} не должны быть слишком длинными, а b_j — наборы слов произвольной длины. Команда `\multicolumn` позволяет объединить некоторое (первый аргумент) количество колонок в одну ячейку, второй аргумент указывает на окружение этой объединенной ячейки (синтаксис такой же как в заголовке), а третий аргумент — содержимое ячейки.

Предупреждение: не следует оставлять пробельные строки внутри скобок tabular, это может вызывать ошибки или "хвосты" в конце таблицы.

Картинка может быть нарисована командой

```
\unitlength 1mm   \thicklines
\begin{picture}(130,70)
    \put(10,15){framebox(12,8){text}}
    \multiput(22,55)(10,8){4}{\line(-2,3){20}}
\end{picture}
```

Здесь проиллюстрирован общий принцип рисования: сначала задается универсальная единица длины и указывается, какими линиями рисуется картинка (`\thinlines` или `\thicklines`); в заголовке картинки определяется ее размер ($x \times y$); после этого некоторый объект командой

```
\put(37,45){объект}
```

помещается в точку с заданными координатами (все координаты даются в порядке (x, y) и отсчитываются от левого нижнего угла). Команда `\multiput` делает то же самое, но несколько раз; в первых круглых скобочках указано положение первой копии, вторая копия сдвигается относительно первой на вектор, координаты которого указаны во вторых круглых скобках, третья сдвигается на столько же относительно второй, и т.д. В фигурных скобках указано количество копий.

Объекты могут быть следующие

`\makebox(20,9){текст}`	текст в невидимой рамочке данного размера
`\framebox(34,13){текст}`	текст в рамочке данного размера
`\dashbox(10,33){текст}`	текст в штриховой рамочке данного размера
`\circle{7}`	окружность указанного диаметра ($d \leq 14$)
`\circle*{13}`	зачерненная окружность ($d \leq 6$)
`\oval(30,20)`	овал указанных размеров
`\line(2,-3){35}`	прямая в данном направлении и с данной проекцией на ось x
`\vector(1,2){27}`	стрелка в данном направлении и с данной проекцией на ось x

При этом координатой рамочки считается координата ее нижнего левого угла, а текст центрируется относительно рамочки. Позиционирование текста для \makebox и \framebox можно упростить, опустив размеры рамочки, в этом случае размеры рамки определяются исходя из размеров текста, y-координатой при этом будет низ строки, а x-координатой — для \makebox начало текста; для \framebox — левая граница рамки.

Координатой окружностей и овала считается их центр, координатой прямой и стрелки — их начало. Направления стрелок и прямых задаются "элементарными" векторами (в круглых скобках). Для направления (0,1) (по оси y) в фигурных скобках указана проекция на ось y, для всех остальных направлений — проекция на ось x. Для прямых допустимы следующие направления:

$(0,\pm1)$, $(\pm1,0)$, $(\pm1,\pm1)$, $(\pm1,\pm2)$, $(\pm1,\pm3)$, $(\pm1,\pm4)$, $(\pm1,\pm5)$, $(\pm1,\pm6)$,
$(\pm2,\pm1)$, $(\pm2,\pm3)$, $(\pm2,\pm5)$, $(\pm3,\pm1)$, $(\pm3,\pm2)$, $(\pm3,\pm4)$, $(\pm3,\pm5)$,
$(\pm4,\pm1)$, $(\pm4,\pm3)$, $(\pm4,\pm5)$, $(\pm5,\pm1)$, $(\pm5,\pm2)$, $(\pm5,\pm3)$, $(\pm5,\pm4)$, $(\pm5,\pm6)$
$(\pm6,\pm1)$, $(\pm6,\pm5)$.

Для стрелок допустимы следующие направления

$(0,\pm1)$, $(\pm1,0)$, $(\pm1,\pm1)$, $(\pm1,\pm2)$, $(\pm1,\pm3)$, $(\pm1,\pm4)$,
$(\pm2,\pm1)$, $(\pm2,\pm3)$, $(\pm3,\pm1)$, $(\pm3,\pm2)$, $(\pm3,\pm4)$,
$(\pm4,\pm1)$, $(\pm4,\pm3)$,

5. Опасные команды и трюки TEX

Предупреждение: приводимые в этой главе команды и трюки являются "опасными" в том смысле, что залезают довольно далеко во внутренние процессы, происходящие при работе TEX, и поэтому рассчитаны на человека, понимающего эти процессы и имеющего некоторый опыт в работе с TEX.

Ящики (boxes) и клей.

Грубо говоря, TEX работает следующим образом. В текстовой моде он считывает абзац и форматирует его как целое наилучшим (с его точки зрения) образом. В математической моде он считывает формулу и конструирует ее. Далее он собирает абзацы и формулы вместе до длины, несколько превышающей высоту страницы. Затем он оптимальным (по его мнению) образом определяет, где страницу оборвать, и записывает страницу на диск, выбрасывая ее из памяти. К строчкам, не попавшим на страницу, начинают добавляться вновь считываемые абзацы.

При этом процесс сборки и конструирования состоит в том, что основные элементы (горизонтальные и вертикальные "ящики", которые возникают либо неявно, либо явно определяются командами `\hbox` и `\vbox`) склеиваются между собой по вертикали и горизонтали "клеем" ("пружинами", которые также определяются либо неявно, либо командами `\hskip, \hfill` и `\vskip, \vfill`).

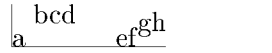
Каждый ящик имеет определенную ширину. Кроме того, он делится по горизонтали "базовой линией" на "верх" и "низ" и в соответствии с этим имеет определенную высоту и глубину. Элементарным ящиком является буква, и, когда мы помещаем слово "барокко" в исходный файл, TEX берет буквы, входящие в него и склеивает их по горизонтали так, чтобы базовые линии слились в одну линию. Это позволяет хвостикам букв "б" и "р" торчать вверх и свисать вниз на правильную величину, в результате чего получается слово "барокко". При этом клей между буквами предопределен свойствами шрифта и слегка зависит от текущего абзаца.

Правило склеивания горизонтальных ящиков по горизонтали в новый горизонтальный ящик такое же, как для склеивания букв. При этом высота и глубина результирующего ящика определяется как максимальная высота и глубина его составляющих.

Однако, существует возможность явным образом определить клей и задать отклонение базовой линии некоторых ящиков от общей базовой линии командами `\raise` и `\lower`. Команда

```
\hbox{\hbox{a}\hskip 1 mm \raise 3 mm \hbox{bcd}\hskip 4 mm
      \hbox{ef}\hskip -1 mm \lower -2 mm \hbox{gh}}
```

создаст ящик



Ширина ящика определяется суммированием ширин составляющих ящиков и клея между ними.

Клей может иметь допуски, это позволяет определять ящики фиксированных размеров

```
\hbox to 12 mm {a \hskip 5 mm plus 2 mm minus 1 mm
      b \hskip 6 mm plus 1 mm minus 2 mm c }
```

Если допуски не позволяют достичь установленной длины, то появится сообщение "underfull box" или "overfull box". Помещение в ящик пружины неограниченной длины (например, `\hfill`) гарантирует от сообщения "underfull box", а пружины с возможностью неограниченного сжатия (`\hss`) — и от сообщения "overfull box".

Существует возможность определить ящик нулевой ширины командой `\rlap{...}`. Легко понять, что эта команда является синонимом набора команд

```
\hbox to 0 mm{\hss ...}
```

Правила склеивания по вертикали практически такие же, за исключением того, что склеивать можно только горизонтальные ящики (по горизонтали можно склеивать как вертикальные, так и горизонтальные). Однако правила определения высоты глубины и ширины результирующего ящика довольно своеобразны.

Прежде всего, существует два типа вертикальных ящиков — \vbox и \vtop. У первого высота — это высота первого из ящиков, а глубина получается суммированием оставшихся вертикальных расстояний. У второго глубина есть глубина последнего из ящиков, а высота — сумма оставшихся расстояний.

Горизонтальный же размер вертикального ящика определяется либо как максимальный по всем горизонтальным ящикам, либо (если в вертикальном ящике хоть раз встречается текст, не заключенный в \hbox) — текущим значением величины \hsize (разумеется, ее можно переустановить внутри ящика). При этом с текстом, не заключенным в \hbox, TEX поступает так же, как в обычной текстовой моде — делает абзацный отступ и форматирует на ширину \hsize (при этом нельзя использовать команды \par или "пробельная строка" для того, чтобы кончить абзац и начать новый, следует использовать команду \vskip, например \vskip 0 mm). Команда

```
\hbox{\vbox{\hsize 20 mm это будет один абзац }
    \vtop{\hsize 25 mm \noindent это будет другой абзац }}
    \raise 11 mm \vtop{\hsize 30 mm \parindent 1.8 cm это будет третий по счету абзац
    \vskip 1 mm это будет четвертый по счету абзац }
}
```

создаст ящики

это будет один абзац

это будет другой абзац

это будет третий по счету абзац

это будет четвертый по счету абзац

При склеивании по вертикали каждый горизонтальный ящик можно подвинуть влево или вправо командами \moveleft и \moveright. Например команда

```
\vbox{\hbox{abc}\vskip 1 mm \hbox{de}\vskip -2 mm \moveleft -3 mm \hbox{fgh}
    \vskip -1 mm \moveright 2 mm \hbox{ij}}
```

создаст ящик

abc

de fgh

ij

Существуют еще и "переменные-ящики". Именно, можно определить ящик \jashik

```
\newbox\jashik
```

Заметьте, что тип его не фиксирован. После этого можно написать что-либо вроде

```
\jashik = \hbox{abcdef}
```

Этой командой Вы определяете текущее содержимое ящика. Далее Вы можете скопировать его в любое место, куда можно поместить ящик соответствующего типа. Это достигается командами

```
\box\jashik    или    \copy\jashik
```

Первая из команд обнуляет содержимое ящика после копирования, а вторая — нет. При этом Вам доступны три переменные — высота ящика \ht\jashik, глубина \dp\jashik и ширина \wd\jashik (они имеют размерность длины). Вы можете не просто считать их в какую-либо переменную, но и поменять их. При этом, разумеется, физические размеры ящика не поменяются (они определяются его содержимым), но при размещении ящика как целого TEX будет использовать установленные Вами размеры.

Таблицы

Для рисования таблиц предназначена команда \halign. Возможности этой команды можно понять на следующем примере

```
\halign{\offinterlineskip
    \vrule \strut \hskip 3 mm # \hfill &\vrule
    \hskip 3 mm \vtop{\noindent \hsize 3 cm # \strut}&\vrule
    \hskip 3 mm \vtop{\noindent \hsize 5 cm # \strut}\vrule
\cr % конец строки-образца
    \noalign{\hrule}
    \omit\span\omit\span\omit\vrule \strut \hfill Это заголовок
       таблицы \hfill \vrule \cr \noalign{\hrule}
    1 & 2 & 3 \cr \noalign{\hrule}
    4 & 5 & 6 \cr \noalign{\hrule}   7 & 8 & 9 \cr
    10 & 11 & 12 \cr \noalign{\hrule}
    \omit 1 & \omit 2 & \omit 3 \cr \noalign{\hrule}
    1 & 2 & \omit 3 \cr \noalign{\hrule}
    MM &\omit\span\omit \vrule \hfill Это объединенная
       (2+3) ячейка \hfill \vrule \cr \noalign{\hrule}
    NN & Этот текст отформатирован в абзац ширины 3 cm &
```

```
    Этот текст отформатирован в абзац ширины 5 cm \cr \noalign{\hrule }
}}
```

В результате получиться таблица

Это заголовок таблицы		
1	2	3
4	5	6
7 10	8 11	9 12

1	2	3
1	2	3
MM	Это объединенная (2+3) ячейка	
NN	Этот текст отформатирован в абзац ширины 3 cm	Этот текст отформатирован в абзац ширины 5 cm

Синтаксические правила легко понять из приведенного примера. Обязательным элементом при использовании `\halign` является строка, задающая образец. Тексту, помещаемому в колонку, соответствует символ #, границе между колонками — символ &, а концу строки — `\cr`. Далее следуют строки текста, которыми заполняется таблица. Ширина каждой колонки определяется просто как максимальная по всем строкам.

Текст, помещаемый в разные колонки, отделяется (как и в строке-образце) символами &, а конец строки маркируется `\cr` (тоже как в образце). Весь текст, окружающий символ # в данной колонке строки-образца, автоматически окружает текст данной колонки в каждой строке. Команда `\omit` позволяет опустить этот добавок. Команда `\strut` совершенно аналогична команде `\mathstrut` математической моды, она вводит невидимый символ максимальной высоты и глубины, благодаря этому линии проходят не вплотную к тексту.

Команда `\span` уничтожает границу между колонками и объединяет их (в данной строке) в одну. Команда `\noalign` позволяет вставить произвольный элемент, который может склеиваться по вертикали, между строками таблицы. Наконец, команда `\offinterlineskip` уничтожает клей, который TEX добавляет между строками, что делает вертикальные линии в таблице непрерывными.

Полезные мелочи

В TEX существует команда `\input`, позволяющая включать содержимое других файлов в обрабатываемый файл. Достаточно большие тексты удобно разбивать на куски и втягивать эти куски командами

```
\input filename    или    \input filename.tex
```

Обычный вопрос, который возникает при наборе книг в TEX — это как создать колонтитулы и обеспечить разные поля для четных и нечетных страниц. (В LATEX команды для этого уже определены).

Это очень несложно. Достаточно определить `\headline` (см. выше) следующим образом

```
\headline {\ifodd\pageno \global\hoffset 2 true cm \the\pageno
   \hfill Текст для нечетной страницы        \else
\global\hoffset -2 true cm Текст для четной страницы \hfill \the\pageno \fi }
```

Более того, с помощью условных выражений можно определить колонтитул для одной отдельной страницы:

```
\headline {\ifnum\pageno = 5 \hfill --- \the\pageno --- \hfill
\else Определение колонтитула для прочих страниц
```

в данном случае будет определен колонтитул для 5-й страницы.

Полезно привести команды, которые реализуют автоматическую нумерацию формул в статье. Это не самое изящное, зато самое простое решение. Единственным его ограничением является условие, что ссылка на формулу должна идти после формулы — разумеется, как правило в статьях это так и есть.

В начале файла надо поставить определения команд

```
\def\label#1{(\the\numequat)\xdef#1{\the\numequat }\global\advance\numequat by 1}
\def\refer#1{{\rm (#1)}}
```

После чего ими можно пользоваться в тексте:

```
$$ x=y \eqno \label{\mynum}$$
в формуле \refer{\mynum}x выражается через y.
```

Для рисования линий предназначены команды

```
\hrule width 50 mm height 1 mm depth 2 mm
\vrule width 50 mm height 1 mm depth 2 mm
```

Эти две команды делают, разумеется, одно и то же — рисуют прямоугольник с заданными размерами. Они отличаются тем, что `\hrule` может склеиваться с другими ящиками только по вертикали, а `\vrule` — только по

горизонтали. Кроме того, некоторые (либо все) размеры можно опустить, тогда `\hrule` проведет горизонтальную линию "естественной" толщины и размером `\hsize` (полная ширина текущего ящика), а `\vrule` — вертикальную, на полную высоту текущего ящика.

В TEX существует возможность заменять клей на повторенный много раз ящик. Ящик повторяется столько раз, сколько необходимо для того, чтобы набрать длину клея. Это полезно, например, для оглавлений (можно провести черту из точек от названия раздела до номера страницы) и осуществляется командой

```
\leaders\hbox{...}\hfill
```

Вместо `\hfill` можно поставить `\hskip`. Например, команды

```
\line{это лидер до конца строки \leaders \hbox{.}\hfill }
\line{это лидер длиной 5 cm \leaders \hbox{.-.}\hskip 5 cm \hfill }
```

дадут

это лидер до конца строки .....

это лидер длиной 5 cm .-..-..-..-..-..-..-..-..-..-..-..-..-..-..-.

Нередко в формулах возникает необходимость строить "многоэтажные" выражения с регулируемым расстоянием между "этажами". В некоторых случаях можно просто определить соответствующую матрицу и регулировать расстояния между ее строками с помощью линий нужной высоты (или глубины) и нулевой ширины (т.е. невидимых).

```
$a=\left( \matrix{a & b & c \cr d & e & f \cr
    \vrule width 0mm height 6 mm f & g & h \cr}\right) $
```

результат имеет вид

$$a = \begin{pmatrix} a & b & c \\ d & e & f \\ \vrule\ \text{width 0mm height 6 mm} & f & g & h \end{pmatrix}$$

Иногда, однако приходится использовать команды `\vbox` и `\vtop`, например

```
$A=\vtop{\hsize 9 mm \parindent 2 mm \noindent ssup \vskip -1 mm
    $\scriptstyle i \in I$ \vskip -1 mm $\scriptstyle x<3$ \vskip -1 mm
    $\scriptstyle \rho<55$ }  a_i(x,\rho) $
```

что даст

$$A = \operatorname*{ssup}_{\substack{i\in I\\ x<3\\ \rho<55}} a_i(x,\rho)$$

Нередко возникает задача поиска символа, имеющегося в каком-либо шрифте, но не определенного как специальная команда (типа команды `\pi`). При этом полезна команда `\char`. Так, команда

```
\char 45
```

печатает 45-ю букву текущего шрифта. Полезно еще раз напомнить, что список всех шрифтов содержится в файле fontlist, а просмотреть начертание всех букв любого из шрифтов можно с помощью файла showfont.tex.

Окружить произвольный объект рамочкой заданных размеров можно с помощью команд

```
\def\ramka#1#2#3{\vbox{\hrule\vfill\hbox{\vrule\hfill
    \vbox to #2 mm{\vfill \hbox to #1 mm{\hfill #3 \hfill}\vfill
    }\hfill\vrule}\vfill\hrule}}
```

После этого определения команда

```
\ramka{40}{20}{пример}
```

нарисует рамку $40mm \times 20mm$, внутри которой будет центрировано слово "пример".

Вставление картинок.

В принципе международным стандартом является вставление картинок в формате PostScript. К сожалению, ни программа предварительного просмотра на экране (dviscr или dviwin), ни программы печати на принтере (dvidot или dvihplj) не позволяют действительно вставить эти картинки — они просто оставляют пустое пространство размером с картинку. Только программа создания PostScript файла dvips действительно вставит PostScript-картинку в выходной PS файл.

Чтобы вставить в TEX-файл PS-картинку необходимо где-нибудь в самом начале Вашего TEX-файла поместить строчку:

```
\input epsf
```

После этого картинка вставляются в любое место текста командой

```
\leavevmode
\epsfbox{ris1.ps}
```

где ris1.ps — имя PS-файла, содержащего картинку.

Чтобы вставить в LATEX-файл PS-картинку необходимо поставить в его заголовке

```
\documentstyle[epsf]{book}
```

или

```
\documentstyle[12pt,epsf]{article}
```

После этого картинка вставляются в любое место текста командой

```
\begin{figure}
\leavevmode
\epsfbox{ris1.ps}
\caption{Caption 1.  }
\end{figure}
```

где ris1.ps — имя PS-файла, содержащего картинку.

С расширенными возможностями команды `\epsfbox` можно ознакомиться, прочитав заголовок файла `texinput\dvips\epsf.tex`. Подробное описание программы dvips и набора макросов epsf можно найти в файле `texinput\dvips\dvips.tex`.

В EMTEX имеются и дополнительные возможности. Во-первых, можно вставлять картинки в форматах .bmp и .pcx (к сожалению, только монохроматические, т.е. 1 точка — 1 бит). Это делается командой

```
\special{em:graph filename.pcx}
```

где filename.pcx — имя файла с картинкой. Такой способ включения рисунков имеет как достоинства, так и недостатки. Главное достоинство — программы dviscr (View) и dvidot/dvihplj (Print) распознают этот формат. Вы видите картинку на экране и получаете ее на печати без каких-либо дополнительных усилий. Главный недостаток — непереносимость (эту команду понимает только EMTEX) и зависимость от типа принтера (один и тот же файл с картинкой на разных принтерах будет давать картинку разных размеров, в зависимости от разрешения принтера; никакие параметры TEX (в частности, `\magnification`) никакого влияния на размер картинки не оказывают, Вы можете только помещать картинку как целое в то или иное место).

Кроме того, в EMTEX имеется возможность реализовать векторную графику, и определить команды типа 'moveto', 'lineto' (как в PostScript). Разумеется эти команды будет понимать только EMTEX (впрочем dvips их тоже понимает), зато это будет не зависящая от типа принтера картинка, которую к тому же можно масштабировать.

Например

```
\strut\special{em:point 1}\vskip 7 true mm
\strut\hskip 12 true mm\special{em:point 2}
\special{em:line 1,2}
```

Команда point отмечает текущее положение на странице как точку с данным номером, а команда line соединяет указанные точки прямой. Разумеется, набирать такие вещи вручную — довольно бессмысленное занятие, лучше воспользоваться рисовальными программами (см. ниже).

Наконец, автором написана подпрограмма, позволяющая рисовать картинки в TEX. В действительности это слегка модифицированный аналог макросов LATEX, рисующих картинки. Она и переносима (правда, кроме Вашего файла надо слать еще и файл `picture.tex`), и масштабируема, но возможности рисования в ней сильно ограничены.

Чтобы использовать эту подпрограмму, достаточно загрузить ее командой

```
\input picture
```

(файл picture.tex лежит в директории texdraw).

Синтаксис практически такой же, как и при рисовании картинок в LATEX. Например

```
\picture(150,80,0.8 mm)
\put(22,15){\linel(3,1)(43) }
\put(41,13){\vector(-4,3)(21) }
\thicklines(\magstep0)
\put(9,12){\dashline(1,5)(10) }
\put(85,42){\waveline(1,0)(4) }
\put(85,42){\waveline(0,-1)(2) }
\thinlines(\magstep0)
\put(85,20){\wavemline(1,0)(4) }
\put(120,42){\wavemline(0,-1)(2) }
\put(48,32){\waveline(-1,1)(3) }
\put(34,30){\circlef(8) }
\put(68,18){\circlen(9) }
\put(50,16){\circlel(9) }
\put(89,27){\frametxt(25,8)(1){Text N 1}}
\put(81,8){\frametxt(25,7)(0){Text N 2}}
\put(59,36){\mintxt(1){Text N 3}}
```

```
\put(30,62){\curva(15,-8)(10,5) }
\put(117,59){\dashcurva(-15,-10)(7,-9)(30,13) }
\multiput(40,75)(10,-5)*(6){\circlef(4)}
\put(2,2){\frametxt(146,76)(1){}}
\endpicture
```

Здесь параметры при команде `\picture` — это размер картинки ($x \times y$) и единица измерения; команда `\put(x,y){объект}` помещает объект в точку (x, y); команда `\multiput(x,y)(dx,dy)*(n){объект}` помещает n копий объекта со сдвигом на (dx, dy) в точку (x, y); `\linel(a,b)(c)` — линия направления (a, b) с проекцией на ось x равной c, $-6 < a, b < 6$; `\vector(a,b)(c)` — вектор направления (a, b) с проекцией на ось x равной c, $-4 < a, b < 4$; `\dashline(a,b)(c)` — штриховая линия направления (a, b) с проекцией на ось x равной c, $-6 < a, b < 6$; `\waveline(a,b)(c)` и `\wavemline(a,b)(c)` — волнистые линии направления (a, b), их фазы отличаются на π, количество периодов равно c, $-1 < a, b < 1$; `\circlef` — зачерненный кружок, `\circlen` — окружность, размер ее определяется целым числом, `\circlel` — окружность, размер которой определяется диаметром, `\frametxt` — текст, заключенный в рамке указанных размеров, `\mintxt` — текст, окруженный рамкой минимальных размеров, для обеих текстовых команд рамка либо видна (параметр (1)), либо не видна (параметр (0)), `\curva(a,b)(c,d)` — кривая, описываемая формулой $x(t) = a * t + c * t * (1 - t)$, $y(t) = b * t + d * t * (1 - t)$, $0 < t < 1$, `\dashcurva(a,b)(c,d)(e,f)` — штриховая кривая, полный период штриховки пропорционален e, размер штрихов пропорционален f. Кривыми лучше не пользоваться.

Следует понимать, что PS-картинка — это объект, внешний для TEX: он знает только размер PS-картинки. Аналогично, картинка из .bmp или .pcx файла — это совсем внешний объект для TEX, он не знает даже размера картинки, просто программы dviscr, dvidot и dvihplj системы EMTEX умеют добавлять bitmap-картинку в выходной файл или на экран. Векторная графика EMTEX — это свойство программ dviscr, dvidot, dvihplj и dvips в системе EMTEX. Только картинки TEX (и LATEX) — это действительно TEX'овские объекты, составленные из шрифтов TEX с помощью команд TEX.

В директории texdraw лежат программы, которые позволяют создавать картинки в TEX. Эти программы снабжены кратким описанием. Попробуйте ими воспользоваться.

6. Заключение.

Следует еще раз напомнить, что в инструкции приведены далеко не все команды (в особенности это касается LATEX). Тем не менее, с помощью приведенных команд практически всегда можно достичь желаемого результата.

Автор гарантирует, что каждая из приведенных здесь команд, использованная в гордом одиночестве, работает именно так, как написано. Однако, нередко наблюдается интерференция команд, которые по отдельности работают прекрасно (в особенности это имеет место в LATEX).

Опыт показывает, что первая выдача TEX (особенно формулы) вызывает невероятное изумление своим несоответствием с желаемым результатом. Со временем, однако, эти проблемы исчезают.