

ИНСТИТУТ ФИЗИКИ ВЫСОКИХ ЭНЕРГИЙ

М.В. Лисина

PLAIN TEX
ОСНОВНЫЕ ПОНЯТИЯ
И КАТАЛОГ КОМАНД

Под редакцией С.В. Клименко

Протвино, 1995 г.

Содержание

Введение	4
1. Основные понятия Т _E X'a	5
1.1. Грамматика Т _E X'a	5
Символы	5
Кавычки	5
Тире	5
Пробелы	5
Команды	6
Размеры	6
Группирование	6
1.2. Шрифты Т _E X'a	7
1.3. Как Т _E X читает входной файл	8
Боксы	8
Клей	10
Моды	11
1.4. Как Т _E X формирует строки и страницы	12
Строки	12
Страницы	14
1.5. Набор математических формул	15
1.6. Макроопределения	18
1.7. Программа вывода	20
2. Как пользоваться каталогом Т _E X'a	22
3. Каталог Т _E X'a	23
Список литературы	156

Введение

Уважаемый читатель! Представляем вам краткое описание основных понятий и полный каталог команд системы \TeX — популярной в научных кругах системы набора, предназначенной для издания книг высокого полиграфического качества, и особенно для книг, содержащих много математических формул. Эта система была создана классиком современного программирования Дональдом Кнутом и в настоящее время широко распространена во всем мире. Подготовить рукопись по правилам \TeX 'а не намного сложнее, чем напечатать ее на пишущей машинке, а типографское качество результата будет сравнимо с работой лучших наборщиков. А если учесть, что компьютерные файлы значительно легче изменять и перепечатывать, общий объем вашей работы, вероятно, будет существенно меньше, чем при традиционно используемых ножницах и клее.

Система \TeX основана на приблизительно 300 командах низшего уровня, которые называются примитивами, и обладает способностью макрорасширения, т. е. в ней можно создавать макрокоманды из примитивов и других макрокоманд. Поскольку большие наборы макрокоманд часто связаны с определенным форматом выходных документов, их иногда тоже называют форматами. Созданная Кнутом макронадстройка из примерно 600 макрокоманд образует формат plain \TeX . В настоящее время образовалось довольно много различных макронадстроек, таких, как \LaTeX , \AMS-TeX , \RNZZX и т. д., из которых особой популярностью пользуется \LaTeX , предназначенный для подготовки научно-технических документов стандартных стилей (статья, доклад, книга, препринт, ...). Если набора макрокоманд plain \TeX 'а (или какого-нибудь другого формата) Вам покажется недостаточно, не составляет особого труда определить свои собственные макрокоманды.

В настоящее время \TeX представлен версией 3.1*, расширенной для работы с текстами, подготовленными на двух языках — русском и английском.

Представляемое Вам руководство не является ни полным и всесторонним описанием plain \TeX 'а, ни учебником для начинающих — для этих целей есть другие книги (см. библиографию*). Мы стремились привести полный каталог примитивов, команд, специальных символов и ключевых слов \TeX 'а с некоторыми пояснениями, а иногда и с интересными примерами. Так же, как в начале словарей часто дается краткое описание грамматики языка, так и в нашем руководстве перед собственно каталогом приводится конспективное описание “грамматики” терминологии и принципов работы \TeX 'а.

Основной материал прямо или косвенно взят из фундаментальной книги Дональда Кнута “Все про \TeX ”, а большинство примеров — из книги Raymond Seroul “Le petit Livre de \TeX ”, перевод с французского которой выполнен В. А. Павловой. Автор благодарит В. Н. Поздрачева, А. В. Самарина и С. Н. Соколова за чтение рукописи и ценные замечания, а также З. А. Котову, и Э. Г. Плотноцкую за помощь в подготовке рукописи.

* Исчерпывающее описание системы приводится в [1], получить первое представление о системе можно по [3] и [4], применение \TeX 'а в ИФВЭ и особенности русскоязычного \TeX 'а описано в [5] и [6]. Представляет интерес и [7] с каталогом шрифтов, имеющихся в ИФВЭ.

1. Основные понятия Т_EX’а

1.1. Грамматика Т_EX’а

Входной информацией для Т_EX’а является файл, содержащий рукопись автора, специальные символы, команды и ключевые слова. Этот файл готовится с помощью любого текстового редактора и его имя должно иметь расширение **.tex**. На выходе Т_EX автоматически создает два файла: результат работы — файл с расширением **.dvi** (device-independent file), содержащий сформированный выходной документ в виде, не зависящем от типа устройства вывода, и протокол работы — обычно с расширением **.log**. При подготовке входного файла необходимо знать следующее.

Символы. В Т_EX’е можно использовать все стандартные символы ASCII, а также прописные и строчные буквы русского алфавита, однако 10 выделенных символов можно использовать только специальным образом. Эти *специальные символы* являются служебными и не могут быть напечатаны в тексте обычным образом. Следующая таблица показывает назначение специальных символов и способ их ввода для изображения в выходном документе.

Символ	Назначение	Ввод
\	Сигнальный символ	<code>\$\backslash\$</code>
{	Признак начала группы	<code>\${}</code>
}	Признак конца группы	<code>\$\}\$</code>
\$	Переключатель в математическую моду	<code>\\$</code>
&	Табулятор	<code>\&</code>
#	Признак параметра в макроопределениях	<code>\#</code>
^	Верхний индекс	<code>\^</code>
_	Нижний индекс	<code>_</code>
%	Символ комментария	<code>\%</code>
~	Неразрываемый пробел	<code>\~</code>

Кавычки. Обычные левые и правые кавычки, имеющиеся на клавиатуре дают ‘одинарные’ кавычки. Чтобы получить принятую в типографском тексте “двойную кавычку”, надо одинарную кавычку ввести дважды.

Тире. Не путайте различного вида тире, дефис и знак минус. Обычно существуют по меньшей мере четыре разных символа:

- дефис или знак переноса: -
- черточка или en-тире (тире длиной примерно в букву n): –
- тире или em-тире (тире длиной примерно в букву m): —
- знак минус: —

Дефисы используются в составных словах типа “мальчик-с-пальчик” и “N-кратный”. En-тире применяется для указания диапазонов чисел, типа “страницы 13–34”, “упражнения 1.2.6–1.2.8”. Em-тире служит знаком пунктуации в предложениях — это то, что мы обычно называем простым тире. Знак минуса используется в формулах. Добросовестный пользователь будет внимательно отличать эти четыре применения. Вот как это делается:

для дефиса надо печатать дефис (-)
 для en-тире — печатать два дефиса (--)
 для em-тире — печатать три дефиса (---)
 для знака минуса — печатать дефис в математической моде (\$-\$(см. Моды).

Пробелы. При наборе текста несколько пробелов подряд считаются за один пробел. Конец входной строки эквивалентен пробелу, а пустая строка обозначает конец абзаца. Пробелы после командных слов игнорируются. Принудительный пробел получается командой ‘_’, а пробел, на котором нельзя разрывать строку — символом ~.

При формировании выходного документа Т_ЕX увеличивает пробелы, если они расположены после точек, поэтому, если точка не оканчивает предложения, после нее надо поставить ‘_’ или ‘ ’.

Команды. Как уже говорилось, формат **plain** состоит из около 600 макрокоманд, определенных через команды низшего уровня, т. е. через *примитивы*. Все команды начинаются с символа \ (бэкслэш). Имена команд состоят либо только из букв, и тогда за ними должен следовать пробел, либо из одной не буквы, и тогда пробел необязателен. Прописные и строчные буквы в именах команд различаются (\b**ig**l и \B**ig**l — это разные команды). Имеются также слова, которые в определенном контексте имеют особый смысл. Такие слова называются *ключевыми*. Например, в конструкции \h**box** to 5cm{...} слово to — ключевое.

Размеры. Т_ЕX использует *размеры*, выраженные в различных единицах измерения, как традиционных полиграфических, так и метрических:

pt	point	пункт (расстояние между базовыми линиями в этом руководстве равно 12 pt)
pc	pica	пика (1 pc = 12 pt)
in	inch	дюйм (1 in = 72.27 pt)
bp	big point	большой пункт (72 bp = 1 in)
cm	centimeter	сантиметр (2.54 cm = 1 in)
mm	millimeter	миллиметр (10 mm = 1 cm)
dd	didot point	дидот-пункт (1157 dd = 1238 pt)
cc	cicero	цицero (1 cc = 12 dd)
sp	scaled point	суперпункт (65536 sp = 1 pt)

Кроме указанных выше, в Т_ЕX’е используются две единицы измерения, которые зависят от текущего шрифта: **em** — немного меньше, чем ширина заглавной буквы М текущего шрифта и **ex** — приблизительно высота строчной буквы x текущего шрифта.

Группирование. Иногда необходимо часть рукописи рассматривать как одну единицу и указать, где эта часть начинается, а где кончается. Такая часть называется *группой* и для ее задания используются *символы группирования* { и }. Все указания, которые Т_ЕX получил внутри группы, немедленно забываются, как только группа закончилась. Иными словами, команды внутри группы действуют *локально*.

1.2. Шрифты Т_EX'a

Когда Вы начинаете готовить входной файл для Т_EX'a, Вам надо знать, какими полиграфическими возможностями Вы можете располагать. В plain Т_EX'e можно получать символы из Computer Modern шрифтов, которые обеспечивают набор самых разнообразных документов. Символы в шрифте также называются *глифами*.

В каждом из Computer Modern шрифтов содержится 128 символов, хотя Т_EX может иметь до 256 символов на шрифт. Так, например, если Вы запрашиваете `\char'35` в шрифте `cmr10`, то получаете \AA . Текстовые шрифты включают *лигатуры* и *акценты*.

Plain Т_EX использует шестнадцать основных шрифтов:

<code>cmr10</code>	(Computer Modern Roman в 10 пунктах)	}	текст.
<code>cmr7</code>	(Computer Modern Roman в 7 пунктах)		
<code>cmr5</code>	(Computer Modern Roman в 5 пунктах)		
<code>cmbx10</code>	(Computer Modern Bold Extended в 10 пунктах)		
<code>cmbx7</code>	(Computer Modern Bold Extended в 7 пунктах)		
<code>cmbx5</code>	(Computer Modern Bold Extended в 5 пунктах)		
<code>cmsl10</code>	(Computer Modern Slanted Roman в 10 пунктах)	}	спец.
<code>cmli10</code>	(Computer Modern Text Italic в 10 пунктах)		
<code>cmth10</code>	(Computer Modern Typewriter type в 10 пунктах)		
<code>cmmi10</code>	(Computer Modern Math Italic в 10 пунктах)		
<code>cmli7</code>	(Computer Modern Math Italic в 7 пунктах)		
<code>cmli5</code>	(Computer Modern Math Italic в 5 пунктах)		
<code>cmsy10</code>	(Computer Modern Math Symbols в 10 пунктах)		
<code>cmsy7</code>	(Computer Modern Math Symbols в 7 пунктах)		
<code>cmsy5</code>	(Computer Modern Math Symbols в 5 пунктах)		
<code>cmex10</code>	(Computer Modern Math Extension в 10 пунктах)		

Формат plain для изменения шрифтов имеет следующие команды:

`\rm` — обычный прямой шрифт (roman)
`\sl` — наклонный (*slanted*)
`\it` — курсив (*italic*)
`\tt` — шрифт пишущей машинки (typewriter)
`\bf` — расширенный жирный шрифт (bold)

В начале работы получается прямой шрифт (`\rm`), если только Вы не указали другое.

Заметим, что два шрифта имеют “наклон”: наклонный шрифт — это, по существу, такой же шрифт, как прямой, но его буквы слегка наклонены в сторону, тогда как буквы курсива пишутся по-другому. Если после курсивного слова следует прямое, для компенсации уменьшения пробела между ними необходимо перед включением прямого шрифта поместить команду `\/` — так называемую *курсивную поправку*.

Шрифты различаются не только по форме, но и по величине символов. Например, шрифт, который вы сейчас читаете, называется “шрифтом в 10 пунктах”, потому что отдельные фигуры его конструкций, если их оценивать в печатных единицах, имеют величину в 10 пунктах.

Имеется возможность использовать шрифт нескольких различных размеров, увеличивая или сжимая изображение символов. Каждый шрифт имеет так называемый *проектный размер*, обычно присущий ему по умолчанию; например, проектный размер

шрифта `cmr9` — 9 пунктов. Существует также диапазон размеров, в которых Вы можете использовать определенный шрифт, уменьшая или увеличивая его размеры.

Оказалось удобным иметь шрифты с коэффициентами увеличения 1.2 и 1.44 (что есть 1.2×1.2), а возможно также с увеличением 1.728 ($= 1.2 \times 1.2 \times 1.2$) и даже выше. Тогда можно увеличивать размер напечатанного в целом в 1.2 или 1.44 раза и все еще оставаться внутри набора доступных шрифтов. Plain TeX содержит аббревиатуры `\magstep0` для масштаба 1000, `\magstep1` для масштаба 1200, `\magstep2` для 1440 и так далее до `\magstep5`. Например, чтобы загрузить шрифт `cmr10` в 1.2×1.2 его нормального размера, вы говорите `\font\bigtenrm=cmr10 scaled\magstep2`.

Это шрифт `cmr10` в нормальном размере (`\magstep0`).

Это шрифт `cmr10`, увеличенный в 1.2 (`\magstep1`).

Это шрифт `cmr10`, увеличенный в 1.44

(`\magstep2`).

Существует также `\magstephalf`, которое увеличивает в $\sqrt{1.2}$ раза, т.е., посередине между шагами 0 и 1.

При увеличении выходного документа увеличиваются все заданные размеры. Если требуется, чтобы какой-нибудь размер не менялся, его надо задавать в неизменяемых единицах с использованием ключевого слова `true`, например, `\hsize=115 true mm`.

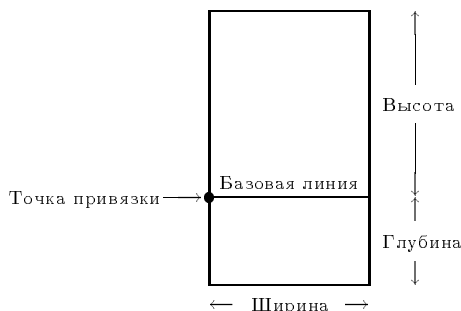
1.3. Как TeX читает входной файл

Боксы. Когда TeX читает входной файл, он преобразует входные данные в *список элементов*. Все вводимые символы подразделяются на 16 *категорий* с номерами от 0 до 15:

Номер	Значение	
0	Начало команды	(в этом руководстве <code>\</code>)
1	Начало группы	(в этом руководстве <code>{</code>)
2	Конец группы	(в этом руководстве <code>}</code>)
3	Математический ключ	(в этом руководстве <code>\$</code>)
4	Табулятор	(в этом руководстве <code>&</code>)
5	Конец строки	(в этом руководстве <code>CR</code>)
6	Параметр	(в этом руководстве <code>#</code>)
7	Верхний индекс	(в этом руководстве <code>^</code>)
8	Нижний индекс	(в этом руководстве <code>_</code>)
9	Игнорируемый символ	(в этом руководстве <code>\null</code>)
10	Пробел	(в этом руководстве <code>_</code>)
11	Буква	(<code>A — Z</code> , <code>a — z</code> , <code>A — Я</code> , <code>a — я</code>)
12	Другой символ	(не перечисленное выше или ниже)
13	Активный символ	(в этом руководстве <code>~</code>)
14	Символ комментария	(в этом руководстве <code>%</code>)
15	Ошибочный символ	(в этом руководстве <code>\delete</code>)

Элемент — это либо символ с номером категории, либо команда. Дальнейшая работа ведется уже с этими списками элементов. Чтобы описать построение строк и страниц из списка элементов, используются такие Т_ЕXнические термины, как *боксы* и *кле́й*.

Боксы в Т_ЕX'e — это двумерные объекты прямоугольной формы, имеющие три связанных измерения, называемые *высотой*, *шириной* и *глубиной*. Приведем схематическое изображение типичного бокса, которое показывает его так называемые точку привязки и базовую линию:



С точки зрения Т_ЕX'a отдельный символ шрифта является боксом; это один из простейших видов боксов. Разработчик шрифта решает, каковы высота, ширина и глубина символа, и как символ будет выглядеть, когда он в боксе. Т_ЕX использует эти размеры для того, чтобы склеить боксы вместе и, в конечном счете, определить местоположения точек привязки для всех символов на странице. Например, в шрифте `\rm plain` Т_ЕX'a (**cmr10**) буква *h* имеет высоту 6.9444 pt, ширину 5.5555 pt, а глубину равную нулю; буква *g* имеет высоту 4.3055 pt, ширину 5 pt, глубину 1.9444 pt. Только некоторые специальные символы, такие как круглые скобки, имеют сумму высоты и глубины действительно равную 10 pt, хотя **cmr10** и считается шрифтом в 10 пунктов. Разработчик шрифта также задает так называемую *курсивную поправку*: это число, которое определено для каждого символа и указывает, грубо говоря, насколько далеко символ выходит за правую границу своего бокса, плюс еще чуть-чуть на всякий случай. Например, курсивная поправка для *g* в **cmr10** равна 0.1389 pt, в то время как в **cmsl10** она 0.8565 pt. Эта поправка будет добавляться к обычной ширине, если вы сразу после символа вводите `\/`.

Часто встречается простой вид боксов, представляющий собой закрашенные прямоугольники заданного размера, которые называются *горизонтальными линейками* и *вертикальными линейками* и обозначаются `\hrule` и `\vrule`.

Все набранные Т_ЕX'ом страницы составлены из простых символьных и линейчатых боксов, соединенных в различных комбинациях. Т_ЕX соединяет боксы двумя способами, *горизонтально* и *вертикально*. Когда Т_ЕX строит *горизонтальный список* боксов, он располагает их так, что все точки привязки располагаются в одном горизонтальном ряду, поэтому базовые линии соседних символов будут лежать на одной прямой. Аналогично, когда Т_ЕX создает *вертикальный список* боксов, он выстраивает их так, что точки привязки располагаются в одной вертикальной колонке. Это легко понять, если представить бокс как бусину, просверленную по базовой линии. Тогда склеивание —

это протягивание “нитки” через эти “бусины” и закрепление “бусин” на “нитке”. Отдельные боксы в горизонтальном списке могут быть подняты (`\raise`) или опущены (`\lower`).

Клей. Существует некая магическая субстанция, называемая *клеем*, которую \TeX использует для скрепления боксов друг с другом. Например, в этом руководстве между строками текста есть небольшое пространство; оно вычислено так, что базовые линии последовательных строк внутри абзаца отстоят друг от друга точно на 12 пунктов. Также есть пространство между словами; такое пространство — не “пустой” бокс; это — клей между боксами. Клей может растягиваться или сжиматься, так что правое поле каждой страницы получается ровным. Клей имеет три атрибута, а именно, естественный размер (`space`), способность *растягиваться* (`stretch`) и способность *сжиматься* (`shrink`).

Процесс распределения клея во время создания бокса из горизонтальных или вертикальных списков называется *установкой клея*. Как только клей установлен, бокс становится жестким; он более не будет ни растягиваться, ни сжиматься и оказывается, по существу, неделимым.

Клей никогда не будет сжиматься больше заданной ему сжимаемости. Но если клей имеет положительную компоненту растяжимости, ему позволено растягиваться произвольно широко.

\TeX неохотно растягивает клей больше заданной растяжимости, поэтому Вы должны решить, насколько большой сделать каждую компоненту клея, и использовать при этом следующие правила. (a) Естественный размер клея должен быть равен размеру промежутка, который выглядит наилучшим образом. (b) Растяжимость клея должна быть равна максимальной величине, которую можно добавить к естественному промежутку перед тем, как выходной документ начинает выглядеть плохо. (c) Сжимаемость клея должна быть равна максимальной величине, которую можно вычесть из естественного промежутка перед тем, как документ начинает выглядеть плохо.

Обычно \TeX задает клей так: $\langle \text{размер}_1 \rangle \text{plus} \langle \text{размер}_2 \rangle \text{minus} \langle \text{размер}_3 \rangle$, где компоненты `plus` $\langle \text{размер}_2 \rangle$ и `minus` $\langle \text{размер}_3 \rangle$ необязательны и в случае отсутствия предполагаются равными нулю; `plus` вводит величину растяжимости, `minus` — величину сжимаемости. Например, в формате `plain` `\medskip` определена как аббревиатура для `\vskip6pt plus2pt minus2pt`. Естественный размер клея должен быть всегда задан явно, даже когда он равен нулю.

Клей может быть *бесконечно* растяжимым. Если такой бесконечно растяжимый клей помещен с левого края ряда боксов, это действует, как сдвиг их в крайне правое положение, т.е., максимально прижимает к правой границе создаваемого бокса. А если Вы берете *два* участка бесконечно растяжимого клея, поставив их слева и справа, это действует, как помещение списка боксов в *центр* большого бокса. Так работает инструкция `\centerline` в `plain` \TeX ’е: она помещает бесконечно растяжимый клей на обоих концах, затем создает бокс, ширина которого равна текущему значению `\hsize`.

\TeX распознает несколько видов бесконечности, некоторые из которых “более бесконечные”, чем другие. Вы можете сказать как `\vfil`, так и `\vfill`; вторая команда

сильнее, чем первая. Другими словами, если не присутствует другой бесконечной растяжимости, `\vfil` растянется так, чтобы заполнить все оставшееся пространство; но если присутствуют как `\vfil`, так и `\vfill` одновременно, то `\vfill` мешает `\vfil` растягиваться.

Кроме `\vfil` и `\vfill`, \TeX имеет `\hfil` и `\hfill` для бесконечного растяжения в горизонтальном направлении, а также `\hss` или `\vss` для клея, который бесконечно и растягивается, и сжимается.

\TeX также широко использует *керны*. Керн похож на клей, но это не тоже самое, поскольку керн не может растягиваться или сжиматься. Более того, \TeX никогда не разрывает строку на керне, за исключением случая, когда за керном следует клей.

Моды. Так же, как человек может быть в различном настроении, так и \TeX при работе бывает в различных “модах”. Всего имеется шесть мод.

- Вертикальная мода. [Построение основного вертикального списка, из которого получаются страницы выходного документа.]
- Внутренняя вертикальная мода. [Построение вертикального списка для v-блока.]
- Горизонтальная мода. [Построение горизонтального списка для абзаца.]
- Частная горизонтальная мода. [Построение горизонтального списка для h-блока.]
- Математическая мода. [Построение математической формулы для помещения в горизонтальный список.]
- Выделенная математическая мода. [Построение математической формулы для помещения на отдельной строке со временным прерыванием текущего абзаца.]

В простых ситуациях Вам не нужно знать, в какой моде работает \TeX , поскольку он сам разбирается, что и как надо делать. Но когда Вы получаете сообщение об ошибке, в котором, например, сказано: “! Вы не можете делать то-то и то-то в частной горизонтальной моде”, знание мод помогает исправить ошибку.

Когда \TeX находится в вертикальной или внутренней вертикальной моде, первый знак нового абзаца изменяет моду на горизонтальную для продолжения абзаца. Другими словами, то, что не имеет вертикальной ориентации, указывает моде автоматически переключиться из вертикальной в горизонтальную. Вместо неявного переключения моды можно явно приказать \TeX ’у перейти в горизонтальную моду, указав `\indent` или `\noindent`. Когда обрабатываются простые рукописи, \TeX почти все время работает в горизонтальной моде (создание абзацев) с короткими путешествиями в вертикальную моду (между абзацами). Абзац завершается, когда встречается `\par` или пустая строка. Абзац также заканчивается, когда встречается что-либо, несовместимое с горизонтальной модой.

Если знак начала математической моды ($\$$) появляется в горизонтальной моде, \TeX переходит в математическую моду и обрабатывает формулу до тех пор, пока не встретит закрывающий $\$$, затем добавляет текст этой формулы к текущему абзацу и возвращается в горизонтальную моду. Если же в абзаце появляются два последовательных знака начала математической моды ($\$$ $\$$), \TeX прерывает абзац, в котором находится, добавляет прочитанную часть этого абзаца к вертикальному списку, затем обрабатывает

математическую формулу в выделенной математической моде, добавляет эту формулу к вертикальному списку и возвращается в горизонтальную моду для продолжения абзаца. (Выделенная в отдельную строку формула должна оканчиваться `$$`.)

Когда `TeX` находится в вертикальной или внутренней вертикальной моде, он игнорирует обычные пробелы и пустые строки (или команды `\par`). Принудительный же пробел (`_`) будет начинать новый абзац.

`TeX` попадает во внутреннюю вертикальную моду, когда ему надо сделать что-нибудь из вертикального списка боксов (используя `\vbox`, `\vtop`, `\vcenter`, `\valign`, `\vadjust` или `\insert`). Он попадает в частную горизонтальную моду, когда создает что-нибудь из горизонтального списка боксов. Между внутренней и обычной вертикальной, а также между частной и обычной горизонтальной модами разница очень мала, но они не абсолютно одинаковы, поскольку служат различным целям.

Всякий раз, когда `TeX` рассматривает очередной элемент входного файла, чтобы решить, что нужно делать дальше, текущая мода может повлиять на то, что означает этот элемент, т. е. команды `TeX`'а контексто-зависимы. Например, `\kern` в вертикальной моде указывает на вертикальный пропуск, а в горизонтальной моде — на горизонтальный пропуск; символ математического переключения `$` указывает на переход в математическую моду из горизонтальной моды, но когда он встречается в математической моде, он указывает на выход из этой математической моды; два последовательных знака (`$$`), появляясь в горизонтальной моде, иницируют выделенную математическую моду, а в частной горизонтальной моде они просто обозначают пустую математическую формулу. `TeX` использует тот факт, что в некоторых модах некоторые операции неуместны, для того, чтобы помочь Вам избавиться от ошибок, которые могли вкрасться в рукопись.

Обычно лучше окончить всю работу, поставив в конце входного файла `\bye`, что является сокращением для `\vfill\eject\end`. Команда `\vfill` переводит `TeX` в вертикальную моду и вставляет достаточно пробелов для того, чтобы до конца заполнить последнюю страницу; `\eject` выводит эту последнюю страницу в выходной файл, а `\end` завершает работу `TeX`'а.

1.4. Как `TeX` формирует строки и страницы

Строки. Одна из главных обязанностей `TeX`'а — это взять длинную последовательность слов и разбить ее на строки подходящего размера, сформировав тем самым абзац. `TeX` использует для выбора таких точек разбиения интересный способ, который рассматривает абзац как единое целое; слова в конце абзаца могут даже повлиять на вид первой строки. В результате пробелы между словами насколько это возможно единообразны, и можно во много раз уменьшить количество переносов слов или формул, разорванных между строками.

`TeX` присоединяет к каждой строке некоторую числовую величину, так называемую *плохость* (*badness*), чтобы оценить эстетическое восприятие пробелов между словами, а также имеет параметр `\tolerance` (допуск). Задавая различные значения параметра `\tolerance`, можно получать различные результаты. Более высокий допуск означает, что допускаются более широкие пробелы. Plain `TeX` обычно требует, чтобы ни у одной

строки плохость не превышала 200. Т_ЕX будет искать самый лучший способ напечатать каждый абзац в соответствии с принципом наименьшей плохости.

Если Вы хотите заставить Т_ЕX сделать разрыв между строками в некоторой точке в середине абзаца, надо просто поместить в этой точке `\break`. Однако, это может привести к тому, что на строке будут слишком широкие пробелы. Если Вы хотите, чтобы Т_ЕX дополнил правую часть строки пробелами перед принудительным разрывом строки, чтобы следующая строка была без отступа, введите `\hfil\break`.

Иногда бывает нужно, чтобы каждой строке на входе соответствовала строка на выходе. Plain Т_ЕX предусматривает команду `\obeylines`, которая каждый конец строки рассматривает как `\par`. После того, как Вы зададите `\obeylines`, Вы будете получать по одной строке выхода на каждую строку входа, если только входная строка не оканчивается % или если она не настолько длинна, что должна разрываться.

Грубо говоря, Т_ЕX разбивает абзац на строки следующим образом. Точки разбиения вставляются между словами или после знака переноса, так что получаются строки, плохость которых не превышает текущего значения `\tolerance`. Если не существует способа вставить такие точки, фиксируется переполненный бокс. Точки разбиения выбираются так, что абзац получается оптимальным, т. е., наилучшим из возможных в том смысле, что имеет не больше “дефектов”, чем можно было бы получить при любой другой последовательности точек разбиения. В дефекты входят плохости отдельных строк, такие вещи, как последовательные строки, которые оканчиваются переносом, а также слишком сжатые строки, которые следуют за слишком свободными.

Сначала Т_ЕX пытается разбить абзац на строки, не вставляя переносов. Первый проход будет успешным, если найденные точки разбиения не приводят к строкам, плохость которых превышает текущее значение `\pretolerance` (преддопуска). Если первый проход неудачен, делается вторая попытка, используя переносы слов и `\tolerance` вместо `\pretolerance`. Plain Т_ЕX по умолчанию устанавливает `\pretolerance=100` и `\tolerance=200`. Если Вы сделаете `\pretolerance=10000`, первый проход будет, по существу, всегда успешным, поэтому не будет попыток переноса (а распределение пробелов будет не очень хорошим). С другой стороны, если Вы сделаете `\pretolerance=-1`, Т_ЕX пропустит первый проход и немедленно приступит к расщеплению слов.

Каждой потенциальной точке разрыва приписан некоторый “штраф” (`\penalty`), который представляет собой “эстетическую цену” разрыва на этом месте. Штраф может задаваться явно или с помощью параметров Т_ЕX’а. Например, если Вы в некоторой точке абзаца зададите `\penalty 100`, эта точка будет законным местом для разрыва между строками, но разрыву в этом месте будет назначен штраф равный 100. Если Вы указываете `\penalty-100`, Вы этим сообщаете Т_ЕX’у, что эта позиция является довольно хорошей точкой для разрыва, потому что отрицательный штраф реально является “премией”. Строка, которая оканчивается премией, может даже иметь “заслуги” (отрицательную дефектность).

Т_ЕX оценивает каждую последовательность точек разбиения, суммируя *дефекты*, которые присущи каждой конкретной строке. Цель этой оценки — выбрать такие точки разбиения, которые дают наименьшую сумму дефектов. Минимизация дефектов абзаца — это, грубо говоря, то же самое, что минимизация суммы квадратов плохостей и

штрафов. Обычно это означает, что максимальная плохость каждой конкретной строки также минимизируется по всем последовательностям точек разрыва.

Страницы. Преобразовав список элементов в список строк, \TeX пытается выбрать подходящее место, чтобы разделить этот список строк на отдельные страницы. Но задача создания страниц намного труднее, чем задача разбиения на строки, потому что страницы часто имеют намного меньшую гибкость, чем строки. Если вертикальный клей на странице имеет малую растяжимость или сжимаемость, то у \TeX 'а обычно нет выбора, где начать новую страницу. Наоборот, если у клея слишком большая эластичность, результат будет выглядеть плохо, потому что страницы окажутся слишком разными. Математические документы, которые обычно содержат много выделенных формул, имеют в этом отношении преимущество, потому что клей, окружающий выделенные уравнения, обычно очень эластичный. \TeX также получает ценную возможность маневра, когда между абзацами используются `\smallskip`, `\medskip` или `\bigskip`.

Для повседневной работы Вас, вероятно, устроит автоматический метод \TeX 'а для разбиения страниц. А если окажется, что этот метод дает неприемлемый результат, можно заставить разорвать страницу на понравившемся Вам месте, напечатав `\eject`. Но будьте внимательны: `\eject` указывает \TeX 'у растянуть страницу, если это необходимо, так что верхние и нижние базовые линии будут согласовываться с такими же базовыми линиями других страниц. Если Вы хотите получить укороченную страницу, дополненную до конца пробелами, введите вместо этого `\vfill\eject`.

Чтобы помешать разрыву страницы, можно сказать `\nobreak` в вертикальной моде, так же как `\nobreak` в горизонтальной моде мешает разрыву между строками. Например, можно между заголовком и первой строкой текста раздела задать `\nobreak`. Но `\nobreak` не отменяет действие других команд типа `\eject`, которые указывают \TeX 'у сделать разрыв. Он только задерживает разрыв до ближайшего соседнего клея.

\TeX разбивает список строк на страницы, вычисляя плохости и штрафы более или менее так же, как он это делает, когда разбивает абзац на строки. Но страницы по мере формирования выводятся в выходной файл \TeX 'а — нельзя заглянуть вперед и посмотреть, как формирование одной страницы действует на следующую. Другими словами, \TeX использует метод для нахождения оптимальных точек разрыва для строк в целом абзаце, но не пытается найти оптимальные точки разрыва для страниц в целом документе, так что \TeX выбирает разбиение каждой страницы насколько может хорошо, производя “локальную”, а не “глобальную” оптимизацию.

Как и в горизонтальных списках, каждая потенциальная точка разрыва связана со штрафом, который высок для нежелательных точек разрыва и отрицателен для желательных. Штраф равен нулю для разрывов на клее и керне.

Plain \TeX позволяет вставлять на страницу иллюстрации. Простейший способ вставки иллюстраций — “плавающая верхняя вставка”. Если задать

`\topinsert <вертикальный материал>\endinsert`,

и \TeX попытается поставить вертикальный материал сверху текущей страницы.

Существует также `\midinsert <вертикальный материал>\endinsert`, которая вначале пытается поместить материал в то место, где Вы находитесь. Если там есть достаточно места, Вы получите следующее:

`\bigskip\ vbox{ <вертикальный материал> }\bigbreak.`

В противном случае `\midinsert` благополучно преобразуется в `\topinsert`. Кроме иллюстраций, которые вставляются в верхнюю часть страницы, plain TeX также умеет вставлять сноски в ее нижнюю часть. Это делает команда `\footnote`.

1.5. Набор математических формул

Как уже упоминалось, математические формулы могут находиться в текстовой строке — так называемая *математическая мода*, а также быть “выключенными” из текста или выделенными — *выделенная математическая мода*. Формула в обычной математической моде заключается в `$... $`, а в выделенной математической моде — в `$$... $$`.

Начинающий наборщик обычно не умеет правильно распределять пробелы в формулах, поэтому TeX большинство пробелов делает сам и *игнорирует* любые пробелы, которые Вы сами поставили между знаками `$`. В большинстве случаев пробелы TeX’a будут привычными для математика, но существуют команды, которыми можно подавить правила TeX’a по расстановке пробелов.

Все вводимые символы в математических формулах имеют специальную интерпретацию. Буквы TeX называют *ординарными символами*, потому что они составляют большую часть математических формул. Буквы в формулах набираются курсивом.

Plain TeX считает ординарными и 18 символов

`0 1 2 3 4 5 6 7 8 9 ! ? . | / ‘ @ ”`

т.е. он не вставляет дополнительных пробелов, когда эти символы следуют один за другим или рядом с буквами. В отличие от букв, эти 18 символов, когда появляются в формулах, остаются в прямом шрифте.

Три символа `+`, `-`, и `*` обозначают *бинарные операции*. Например, `+` — это знак плюс, который используется для обозначения суммы двух чисел, `-` — это знак минус. Звездочка в математике используется редко и тоже ведет себя, как бинарная операция. Заметим, что `-` и `*` дают математические символы, отличные от тех, которые появляются в обычном тексте. Знак дефис (`-`) становится знаком минуса (`-`), а поднятая звездочка (`*`) опускается на более низкий уровень (`*`).

TeX не считает знак `/` бинарной операцией, хотя он обозначает деление (которое в математике считается бинарной операцией). Причина в том, что наборщики традиционно ставят дополнительные пробелы вокруг символов `+`, `-` и `*` и не ставят их вокруг `/`. Если бы TeX трактовал `/` как бинарную операцию, то формула `$1/2$` получилась бы в виде `1 / 2`, что было бы некрасиво; поэтому TeX считает `/` ординарным символом. Другие бинарные операции задаются командами, а не символами.

Plain TeX трактует символы `=`, `<`, `>` и `:`, как *отношения*, потому что они выражают отношение между двумя величинами. Пробелы вокруг отношений распределяются не так, как вокруг бинарных операций. Некоторые отношения вводятся командами.

Два символа `‘,’` (запятая) и `‘;’` (точка с запятой) трактуются в формулах как знаки пунктуации. Это означает, что TeX ставит небольшой пробел после них и не ставит

до них. В математических формулах не принято ставить дополнительный пробел после ‘.’ (точки), поэтому \TeX считает точку ординарным символом. Если символ ‘:’ должен быть знаком пунктуации, а не отношением, задавайте его командой `\colon`. Если вы хотите использовать запятую как ординарный символ, поставьте ее в фигурных скобках: \TeX трактует все, что находится в фигурных скобках, как ординарный символ.

Символы ‘(’ и ‘[’ считаются *открывающими ограничителями*, а символы ‘)’ и ‘]’ — *закрывающими ограничителями*. Затем имеется символ ‘\prime’, который используется как сокращение для верхнего индекса `\prime`. Специальные символы не используются в качестве символов в математической моде, если только не было изменено значение их `\catcode`. Хотя { и } указывают группирование, команды `\{` и `\}` можно использовать, чтобы получить открывающие и закрывающие фигурные скобки.

В формулах можно использовать греческие буквы и множество математических символов. Для этих символов используются специальные команды, которые разрешены только в математических модах.

В математических модах, чтобы указать “подформулы”, т.е., простые части более сложной формулы, используются фигурные скобки. \TeX для каждой подформулы создает бокс и рассматривает этот бокс, как если бы он был одним символом. Фигурные скобки служат и для обычных целей группирования. Например, нельзя вводить x^y^z или x_y_z : \TeX будет возражать против “двойного верхнего индекса” или “двойного нижнего индекса”. Вы должны вводить $x^{\{y^z\}}$, $x^{\{yz\}}$, $x_{\{y_z\}}$ или $x_{\{yz\}}$. Если за знаком верхнего или нижнего индекса следует символ, индексом становится только этот символ, но когда за знаком индекса следует подформула, то эта подформула будет, соответственно, поднята или опущена.

Математики любят писать над буквами *акценты*. В plain \TeX е предусмотрены математические акценты разных видов. Когда в математических формулах акцентируются буквы i и j , то под акцентами должны использоваться *бесточечные символы* i и j . Эти символы задаются в plain \TeX е командами `\imath` и `\jmath`.

\TeX имеет восемь стилей, в которых может обрабатывать формулы, а именно:

выделенный стиль	(для формул на отдельной строке)
текстовый стиль	(для формул в тексте)
индексный стиль	(в верхних или нижних индексах)
стиль повторных индексов	(в повторных индексах)

и четыре других “сжатых” стиля, которые почти такие же, за исключением того, что показатель степени не так поднят. \TeX также использует для математики три различных размера. Они называются: текстовый размер, размер индексов и размер повторных индексов. Обычный способ набрать формулу с помощью \TeX а — заключить ее в знаки доллара `$...$`, это дает формулу в текстовом стиле. Или можно заключить ее в двойные знаки доллара `$$...$$`, это выделяет формулу в выделенном стиле. Подформулы этой формулы могут, конечно, быть в других стилях.

Символы типа \sum и \int (и несколько других символов типа \cup , \prod , \oint и \otimes) называются *большими операторами* и вводятся почти так же, как обычные символы или буквы.

Отличие в том, что \TeX в выделенном стиле выберет более крупный большой оператор, чем он это делает в текстовом стиле. В больших операторах часто бывают *пределы*. Пределы вводятся так же, как верхние и нижние индексы. Некоторые наборщики предпочитают ставить пределы над и под знаками \int ; это занимает больше места на странице, но приводит к лучшему внешнему виду, если подформула сложная, потому что отделяет пределы от остальной формулы. Аналогично, пределы изредка желательны в текстовом или индексном стиле, а некоторые наборщики предпочитают не ставить пределы на выделенных знаках \sum . Вы можете изменить стандартные соглашения \TeX 'а командами `\limits` или `\nolimits` непосредственно после большого оператора. Если вы введете `\nolimits\limits` (например, потому что некоторые макроккоманды, также как `\int`, задают `\nolimits`, а вам нужны пределы), преимущество будет за последним словом. Есть также команда `\displaylimits`, которую можно использовать, чтобы восстановить нормальные соглашения \TeX 'а о том, что пределы появляются только в выделенных стилях.

Поскольку математические формулы бывают очень большими, \TeX умеет создавать увеличивающиеся символы. Например, $\sqrt{1 + \sqrt{1 + x}}$.

Аналогичная вещь происходит со скобками и другими так называемыми “ограничивающими” символами. Для того, чтобы получить несколько увеличенный вариант этих символов, просто поставьте перед ним `\bigl` (для открывающего ограничителя) или `\bigr` (для закрывающего ограничителя). Можно также вводить `\Bigl` и `\Bigr`, чтобы получить ограничители, подходящие для выделенных формул. Они на 50% выше их `\big` двойников. Выделенные формулы наиболее часто используют ограничители, которые в два раза выше `\big`; такие ограничители конструируются при помощи `\biggl` и `\biggr`. И, наконец, есть варианты `\Biggl` и `\Biggr`, которые в 2.5 раза выше ограничителей `\bigl` и `\bigr`.

\TeX имеет встроенный механизм, который вычисляет высоту пары ограничителей, поэтому можно не гадать, должен ли быть ограничитель `\big`, `\bigg` или какой-нибудь еще. Надо просто ввести

$$\backslash\mathbf{left}\langle\text{ограничитель}_1\rangle\langle\text{подформула}\rangle\backslash\mathbf{right}\langle\text{ограничитель}_2\rangle$$

и \TeX наберет подформулу, вставляя слева и справа заданные ограничители нужной величины. Всякий раз, когда вы используете `\left` и `\right`, они должны быть в паре друг с другом, так же как фигурные скобки в группах.

Все символы, которые вводятся в математической моде, принадлежат к одному из шестнадцати *семейств шрифтов* с номерами от 0 до 15. Каждое из этих семейств состоит из трех шрифтов: для текстового размера, для индексного размера и для размера повторного индекса. Чтобы определить члены каждого семейства, используются команды `\textfont`, `\scriptfont` и `\scriptscriptfont`.

Каждому математическому символу соответствует идентифицирующее кодовое число от 0 до 4095, которое получается, если добавить к номеру позиции 256, умноженное на номер семейства. Это легко выражается в шестнадцатеричной записи, используя одну шестнадцатеричную цифру для семейства и две — для символа. Например, “24A” обозначает символ “A” семейства 2. Каждый символ отнесен к одному из следующих восьми классов, пронумерованных от 0 до 7.

Класс	Значение	Пример
0	Обычные	/
1	Большие операторы	\sum
2	Бинарные операторы	+
3	Отношения	=
4	Открывающие	(
5	Закрывающие)
6	Пунктуация	,
7	Переменное семейство	x

Классы с 0 до 6 указывают на “части речи”, которые принадлежат математическому языку; класс 7 — это специальный случай, который позволяет математическим символам изменять семейство. Он ведет себя точно так же, как класс 0, за исключением того, что указанное семейство заменяется на текущее значение целого параметра, называемого `\fam`, при условии, что `\fam` лежит между 0 и 15. \TeX , когда входит в математическую моду, автоматически устанавливает `\fam=-1`, поэтому классы 7 и 0 эквивалентны, если только `\fam` не присвоено новое значение. Plain \TeX изменяет `\fam` на 0, когда пользователь вводит `\rm`; это делает удобным получать в формулах буквы прямого шрифта, поскольку буквы принадлежат классу 7. Номер класса умножается на 4096 и прибавляется к номеру символа, а это то же самое, что сделать его первой цифрой четырехзначного шестнадцатиричного числа. Например, формат plain определяет, что `\sum` — это математический символ “1350”, а это означает — большой оператор (класс 1), находящийся в позиции “50” семейства 3.

Интерпретация символов в математической моде определяется таблицей 128 значений “математических кодов”. Элементы этой таблицы могут быть изменены командой `\mathcode` точно также, как коды категории изменяются командой `\catcode`. Каждый математический код указывает класс, семейство и положение символа.

\TeX связывает классы с подформулами так же, как с отдельными символами. Так, например, можно трактовать составную конструкцию, как если бы она была бинарной операцией или отношением. Для этой цели используются команды `\mathord`, `\mathop`, `\mathbin`, `\mathrel`, `\mathopen`, `\mathclose` и `\mathpunct`. За каждой из них следует либо простой символ, либо подформула, заключенная в фигурные скобки.

1.6. Макроопределения

Как уже упоминалось, пользователь может определять новые макрокоманды, используя *макроопределения*. Макроопределения имеют общий вид

```
\def<команда>{<параметры>}{<текст подстановки>}
```

где параметры не содержат фигурных скобок, а все { и } в тексте подстановки правильно вложены. Символ # имеет специальное значение: в параметрах за первым # должен следовать 1, за следующим — 2, и так далее; разрешено до девяти #. В тексте подстановки за каждым # должна следовать цифра, которая появлялась после # в параметрах. Это обозначает вставку соответствующего аргумента.

Когда \TeX встречает во входном файле макрокоманду, он ищет ее определение и заменяет макрокоманду ее *макроподстановкой*.

В \TeX е есть правило: элементу \backslash par не *позволяется быть частью аргумента*. Если \TeX встретит \backslash par в аргументе, он прервет макроопределение и сообщит, что обнаружен “убегающий аргумент”. Если же Вам нужна команда, у которой разрешены аргументы с элементами \backslash par , можно определить “длинную” макрокоманду, указав \backslash long перед \backslash def .

Когда макрокоманде предшествует \backslash outer , это означает, что соответствующую команду нельзя использовать ни в тексте подстановки макроопределения, ни в преамбуле к выравниванию, ни в условном тексте, который пропускается. Если \backslash outer все же появляется в таких местах, \TeX прекращает свои действия и сообщает либо о ситуации “убегающего аргумента”, либо о “незавершенном” условии. Конец входного файла так же рассматривается в этом смысле как \backslash outer . Например, файл не должен оканчиваться в середине макроопределения. Если Вы разрабатываете формат для того, чтобы его использовали другие пользователи, используйте \backslash outer со всеми командами, которые должны появляться только “тайно” внутри документа.

Теперь мы видели, что перед \backslash def может следовать \backslash long или \backslash outer , а им также может предшествовать \backslash global , если предполагается, что определение выходит за границы группы. Эти три приставки могут прилагаться к \backslash def в любом порядке, и даже могут появляться более одного раза. У \TeX а также есть примитив \backslash gdef , который эквивалентен $\text{\backslash global}\text{\backslash def}$.

Можно задавать макроопределения, которые меняют свое поведение в зависимости от текущих условий. Для этой цели \TeX предусматривает множество примитивных команд. Общий вид таких “условных текстов” следующий:

$\text{\backslash if}\langle\text{условие}\rangle\langle\text{текст}_1\rangle\text{\backslash else}\langle\text{текст}_2\rangle\text{\backslash fi}$

где текст_1 пропускается, если условие не выполняется, а ложный текст_2 пропускается, если условие выполняется. Если ложный текст пустой, можно опустить \backslash else . Часть “ $\text{\backslash if}\langle\text{условие}\rangle$ ” в этой конструкции начинается с команды, первые две буквы которой “if”; например,

$\text{\backslash ifodd}\text{\backslash count0}\text{\backslash rightpage}\text{\backslash else}\text{\backslash leftpage}\text{\backslash fi}$

задает условие, которое истинно, когда целый регистр \TeX а \backslash count0 нечетный.

Заметим, что все команды для сравнений начинаются с $\text{\backslash if}\dots$ и имеют парную \backslash fi . Вложение $\text{\backslash if}\dots\text{\backslash fi}$ независимо от вложения $\{\dots\}$; так вы можете начинать или оканчивать группу в середине условия и начинать или оканчивать условия в середине группы.

Можно определить макрокоманду, текст подстановки которой раскрывается в зависимости от текущих условий. Для этой цели \TeX имеет команду \backslash edef (расширенное определение) и \backslash xdef (которая эквивалентна $\text{\backslash global}\text{\backslash edef}$). Общий формат их такой же, как у \backslash def и \backslash gdef , но \TeX до конца раскрывает элементы текста замены.

\TeX имеет возможность читать текст сразу из вплоть до 16 файлов вдобавок к тем файлам, которые появляются в \backslash input . Чтобы начать читать такой вспомогательный файл, надо сказать

`\openin<число>=<имя файла>`

где число расположено между 0 и 15. К имени файла добавляется расширение `.tex`, как и в `\input`, если явно не задано другое расширение. Если файл не найден, `TeX` не дает сообщения об ошибке; он просто будет считать, что входной поток не открыт. Это условие можно проверить при помощи `\ifeof`. По окончании работы с файлом можно указать

`\closein<число>`

и файл, связанный с этим номером входного потока, закроется. Чтобы вводить из открытого файла, Вы говорите

`\read<число>to<команда>`

и команда становится определенной, как макрокоманда без параметров, текст подстановки которой — это содержание следующей строки, прочитанной из указанного файла.

И наконец, если во входном файле встретятся несколько определений одной макрокоманды, будет действовать последнее из них.

1.7. Программа вывода

`TeX` собирает материал, пока его не накопится больше, чем может поместиться на странице, затем он формирует страницу текста, основываясь на том, что, как ему кажется, является самой лучшей точкой разбиения между страницами, и отправляет ее в выходной файл, имя которого имеет расширение `.dvi`, затем он таким же способом собирает материал для следующей страницы. Номера страниц, заголовки и тому подобные вещи присоединяются при помощи специальной последовательности команд `TeX`'а, которая называется *текущей программой вывода*.

Plain `TeX` имеет программу вывода, выполняющую простые операции, которые требуются при наборе большинства документов, а также справляется и с более сложными задачами, такими, как создание вставок при помощи `\footnote` и `\topinsert`. Можно делать простые изменения в поведении программы вывода plain `TeX`'а, а также определить программу вывода, которая решала бы более сложные задачи.

Программа вывода plain `TeX`'а, используемая по умолчанию, нумерует страницы и помещает номера внизу. Каждая страница будет иметь ширину около $8\frac{1}{2}$ дюймов и высоту около 11 дюймов, включая поля в 1 дюйм со всех четырех сторон. Этот формат подходит для препринтов технических статей, но если вы используете `TeX` для других целей, его можно изменить. Например, можно изменить ширину страницы `\hsize` и вертикальный размер страницы `\vsize`.

Если же, когда выходной документ в конце концов напечатан, Вам хочется расположить его по-другому, можно сдвинуть его, задавая ненулевые значения `\hoffset` и `\voffset`.

`TeX` часто используется для набора объявлений или других бумаг, в которых не нужна нумерация страниц. Команда `\nopagenumbers` в начале рукописи отменяет вставку номеров внизу каждой страницы.

В действительности `\nopagenumbers` — это специальный случай более общего механизма, при помощи которого можно управлять верхним и нижним текущим заголовком

страницы (верхним и нижним колонтитулами). Программа вывода plain TeX'a помещает специальную строку текста, называемую *headline*, сверху каждой страницы и другую специальную строку текста, называемую *footline*, снизу. Строка верхнего текущего заголовка (*headline*) обычно бывает пустой, а в центре строки нижнего текущего заголовка (*footline*) обычно находится номер страницы. Но можно задать те строки, какие Вам хочется, используя команды `\headline` и `\footline`.

Когда Вы определяете строки верхнего и нижнего текущего заголовка, важно явно задавать имена шрифтов, поскольку программа вывода TeX'a начинает действовать в некоторое непредсказуемое время. Например, предположим, что `\footline` была установлена в `\hss\folio\hss` без указания `\tenrm`. Тогда номер страницы будет набран тем шрифтом, который будет текущим в то время, когда TeX решит выводить страницу. В таком случае может быть непредсказуемый эффект, поскольку TeX часто находится в середине страницы 101, когда выводит страницу 100.

Начинающим пользователям обычно не приходится делать слишком большие изменения программы вывода plain TeX'a, поскольку их вполне устраивают стандартные возможности.

2. Как пользоваться каталогом

В каталог включены специальные символы, примитивы и ключевые слова \TeX 'а, а также макрокоманды формата `plain`.

Сначала приводятся специальные символы и команды, имена которых состоят из одной не буквы, а затем остальные команды и ключевые слова, расположенные в соответствии с английским алфавитом без учета бэкслэша. Специальным символам и ключевым словам не предшествует бэкслэш, а примитивы помечены символом `*`.

Мы здесь не приводим строгого определения и полного описания команд. Это, скорее, просто упоминание о их существовании, иногда сопровождающееся поучительными примерами.

Как уже упоминалось, классическим описанием \TeX 'а является фундаментальная книга Дональда Кнута “Все про \TeX ” ([1]), где дается исчерпывающее объяснение всех команд и терминов \TeX 'а. Строчка чисел в конце каждой рубрики — это ссылки на страницы этой книги, на которых находится описание или примеры применения описываемой команды. Номер страницы, где дается точное определение слова или команды, подчеркнут, а если номер страницы напечатан курсивом, то там приводятся примеры применения команды.

В каталог включены и несколько макрокоманд, которые не входят в формат `plain`, но показались автору интересными. Это в описании указано явно. Если они Вам понадобятся, поместите их в свою библиотеку макроопределений.

В процессе работы можно получать информацию о командах и параметрах, используя команды `\show` (чтобы посмотреть на определение макрокоманды) и `\showthe` (чтобы узнать значение внутреннего параметра). Так, например, если Вы введете `\show\bar`, на экране появится

```
> \bar=macro
->\mathaccent "7016
\show\bar
?
```

а после `\show\hsize` — только

```
> \hsize=\hsize
\show\hsize
?
```

потому что `\hsize` — это примитив. Зато после команды `\showthe\hsize` Вы увидите, например

```
> 469.75499 pt
\showthe\hsize
?
```

то есть текущее значение примитива `\hsize`. После `?` работу можно продолжить нажатием клавиши `CR`. Имеются и другие `\show`-команды, которые помогут Вам заглянуть в глубины \TeX 'а.