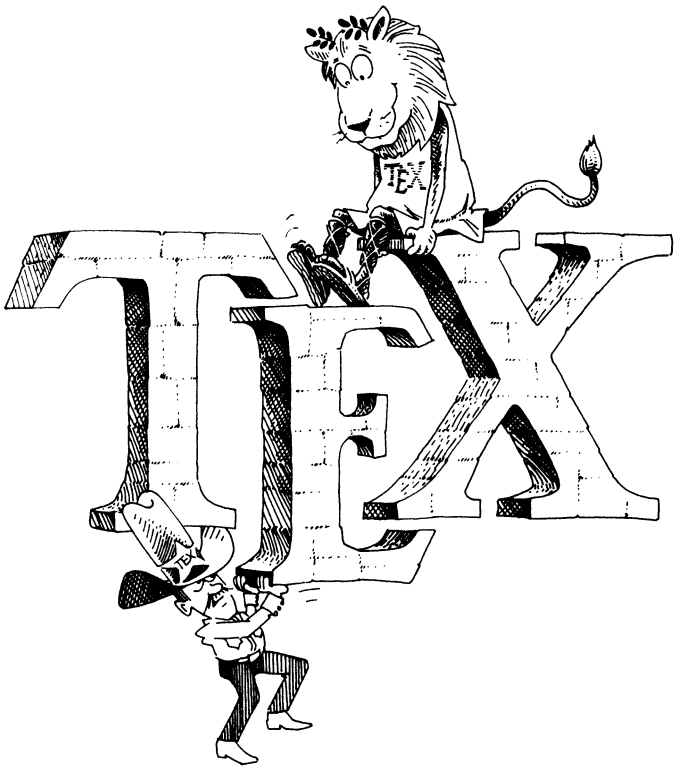
1

Название игры



Английское слово 'technology' происходит от древнегреческого корня $\tau\epsilon\chi\ldots$ и имеет два значения — искусство и технология. Отсюда и название TEX, которое является формой $\tau\epsilon\chi\ldots$, записанной прописными буквами. Поэтому следует произносить χ в слове TEX, как греческое 'chi' (т. е. русское 'х'), а не английское 'х', таким образом TEX рифмуется со словом *грех*. Звук 'х' (ch) в шотландском языке подобен звуку в конце слова *loch*, а в немецком — *ach*, это напоминает испанское 'j' и русское 'х'. Если вы правильно произнесете этот звук, обращаясь к вашему компьютеру, он прослезится.

Такой экскурс в правила произношения напомнит вам, что система TEX соприкасается с техническими текстами высокого уровня: ее сила в искусстве и технологии, как и у греческого слова, от которого происходит ее название. Если вы хотите изготовить сносный документ — что-нибудь приемлемое и в основном читаемое, но не шедевр, — достаточно любой простой системы. С помощью системы TEX вы получите документ высочайшего качества. Это потребует большего внимания к деталям, но, получив прекрасный результат, вы забудете о затраченных сравнительно небольших усилиях. С другой стороны, важно отметить следующее: в названии TEX буква 'E' стоит ниже строки. Такое расположение 'E' напоминает, что TEX имеет отношение к типографскому набору. На самом деле TEX (произносится '*tecks*') является великолепным *Text EXecutive*-процессором, разработанным фирмой Honeywell Information Systems. Так как произношение этих названий различно, то должно отличаться и написание. Правильно было бы упоминать TEX в компьютерном файле либо на других носителях информации, которые не позволяют записывать 'E' ниже строки, в виде 'TeX'. Тогда исчезнет путаница в похожих названиях, и их всегда будут произносить правильно.

▸ **УПРАЖНЕНИЕ 1.1**

Как вас будут называть после того, как вы усвоите материал этой книги: TEXэкспертом или TEXником?

Они дают болезням
новые странные названия.

— ПЛАТОН (PLATO)
Государство, книга 3 (375 до н. э.)

Техника! Подобно слишком это слово
На оскорбленного искусства визг.
Названье идиотское для всех потуг того,
Кто слишком слаб или устал, иль глуп,
чтобы вести игру.

— ЛЕОНАРД БЕКОН (LEONARD BACON)
София Трентон (1920)

2

Издание книг и печатание текстов

Впервые начав использовать печатающее устройство компьютера, вы, вероятно, заметили различие между цифрой ‘1’ и строчной буквой ‘l’. Сделав следующий шаг в понимании того, как готовить книгу к печати, вы заметите гораздо больше подобных вещей. Ваши глаза и пальцы научатся находить и другие отличия.

В первую очередь заметим, что в книгах используют два вида кавычек, в то время как в обычной пишущей машинке — только один вид кавычек. Даже клавиатура вашего компьютера с большим количеством символов, чем у обычной пишущей машинки, вероятно, имеет только неориентированные двойные кавычки ("), поскольку стандартный код ASCII разрабатывался для компьютера без учета того, что его можно использовать для печатания книг. Однако ваша клавиатура, вероятно, имеет две одинарные кавычки, а именно ‘ и ’. Вторая используется также как апостроф. Американские клавиатуры содержат обычно символ левых кавычек, который выглядит как `, и апостроф или правые кавычки, которые выглядят как ' или ´.

Чтобы в TEX получить двойные кавычки, печатают просто две одинарные кавычки соответствующего вида. Например, чтобы получить фразу

“Я понимаю.”

(включая кавычки), вы печатаете на компьютере

```
``Я понимаю.''
```

В этом учебнике используется стиль пишущей машинки для обозначения тех конструкций TEXa, которые нужно набрать на клавиатуре. Так что символы, которые для этого использованы, легко отличить от тех, что производит TEX на выходе, и от комментариев автора. Вот символы, которые будут использованы в примерах:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789"#$%&@*+-=,.:;?!
()<>[]{}‘’\|/_^~
```

Если на клавиатуре вашего компьютера нет каких-либо из этих символов, не стоит из-за этого расстраиваться — TEX может обойтись и без них. Дополнительный символ

```
␣
```

используется для обозначения *пустого места* в тех случаях, когда важно подчеркнуть, что в определенном месте при печатании должно быть пустое место (пробел). Поэтому приведенный выше пример *на самом деле* следует печатать

```
``Я␣понимаю.''
```

Без таких символов могут возникнуть трудности при воспроизведении невидимых частей определенных конструкций. Но здесь не слишком часто будет использоваться символ ‘␣’, поскольку пробелы в большинстве случаев и так хорошо видны.

Печатание книг значительно отличается от печатания обычных текстов в отношении использования символов для тире, дефиса и знака минус. В хороших математических книгах все эти символы различны. В действительности используется четыре разных символа:

дефис (-);
en-тире (короткое тире) (–);
em-тире (длинное тире) (—);
знак минус (−).

Дефисы используются в словах типа 'dauter-in-law' и 'X-rated' или 'какой-нибудь' и 'хост-система', 'n-мерный'. N-тире используется для обозначения диапазона значений (например, 'страницы 13–34'), а также в выражениях 'упражнение 1.2.6–52'. M-тире используется для пунктуации в предложениях — т. е. это то, что мы часто называем просто тире. И, наконец, знак минус используется в формулах. Добросовестный пользователь TEXa пожелает узнать, каким образом различать тире, и здесь показано как это делается:

для дефиса печатать дефис (`-`);
для en-тире печатать два дефиса (`--`);
для em-тире печатать три дефиса (`---`);
для знака минус печатать дефис в математической форме (`$-$`).

(Математическая форма записи подразумевает заключение выражения между двумя долларами; она обсуждается позже.)

▸ **УПРАЖНЕНИЕ 2.1**
Объясните как в системе TEX записать следующее выражение: Алиса сказала: "Я всегда пользуюсь en-тире вместо дефиса, когда указываю в библиографии номера страниц, как-то '480–491'."

▸ **УПРАЖНЕНИЕ 2.2**
Как вы думаете, что произойдет, если вы напишете подряд четыре дефиса?

Внимательно присмотревшись к хорошо изданным книгам, вы обнаружите, что определенные комбинации букв рассматриваются как одна. Например, в слове 'find' это наблюдается с сочетанием 'f' и 'i'. Такие комбинации в полиграфии называются *лигатурами*, и профессиональных наборщиков обычно обучают улавливать такие комбинации, как `ff`, `fi`, `fl`, `ffi` и `ffl`. (Это объясняется тем, что слова типа 'find' не очень хорошо выглядят в большинстве стилей набора, если соседствующие буквы не заменены лигатурой. Традиционные лигатуры часто появляются в английском языке, в других языках важны свойственные им комбинации.)

▸ **УПРАЖНЕНИЕ 2.3**
Придумайте английское слово, содержащее две лигатуры.

Хорошая новость — это то, что вы *не* будете заниматься лигатурами: TEX позаботится о них самостоятельно, используя такие же преобразования, как и для перевода '`--`' в '–'. TEX также рассматривает комбинации

примыкающих букв (например, ‘A’, соседствующее с ‘V’), которые следует сдвинуть, чтобы они лучше смотрелись; в полиграфии это называется *кернинг*, т. е. установка или изменение межзнаковых интервалов.

Подытожим: если вы используете TEX для обычного набора книги или статьи, то это сводится к обычной процедуре печатания на машинке, за исключением разве что числа 1, выделения цитат и различных вариантов дефис/тире. TEX автоматически позаботится о таких тонкостях, как лигатуры и кернинг.

(Вы уверены, что вам следует читать этот раздел? Знак “опасный поворот” предупреждает о том материале, который можно пропустить при первом чтении. А, возможно, и при втором. Разделы, требующие дополнительных усилий от читателя, иногда оперируют понятиями, которые объясняются позже.)

Если на вашей клавиатуре нет левой кавычки, то можно печатать `\lq`, оставляя пробел, если далее следует буква, или вставляя `\`, если далее следует пробел. Аналогично, `\rq` используют вместо правой кавычки. Понятно?

```
\lq\lq␣Я␣понимаю.\rq\rq\␣
```

Когда необходимо напечатать кавычки в кавычках, например простая кавычка следует за двойной кавычкой, нельзя просто напечатать `’’’`, так как TEX поймет это как ”’ (за двойной кавычкой следует одинарная). Если вы заглянули уже в раздел 5, то, возможно, полагаете, что решение в том, чтобы использовать группировку, т. е. записать таким образом: `{’}’’`. Но оказывается, что это не приводит к желаемому результату, поскольку обычно меньший пробел следует за простой правой кавычкой, а не за двойной правой кавычкой: вы получите ’”, что в действительности является двойной кавычкой, за которой следует простая кавычка (если вы достаточно внимательно присмотритесь), но смотрится все это как три простые кавычки, расположенные на одинаковом расстоянии друг от друга. С другой стороны, вы, конечно, не захотите записать `’␣’’`, потому что такой пробел уж слишком велик — такой же величины, как и пробел между словами, и системе TEX, возможно, даже придется начать новую строчку, как в случае начала абзаца. Решением является запись `’\thinspace’’`, которая и приведет к желаемому результату ’”.

▸ **УПРАЖНЕНИЕ 2.4**
Прекрасно, сейчас вы знаете, как записать ”’ и ’”. А как записать “‘ и ‘“?

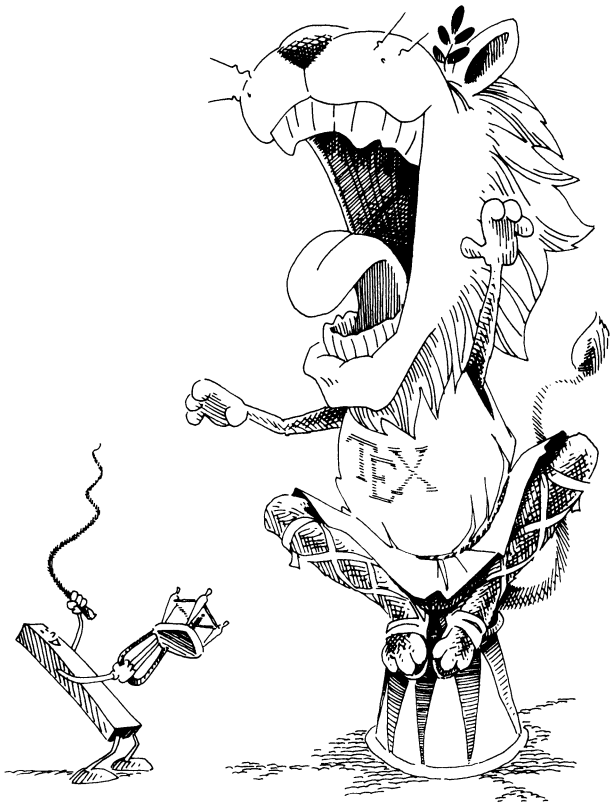
▸ **УПРАЖНЕНИЕ 2.5**
Как вы думаете, почему автор вводит управляющее слово `\thinspace` для того, чтобы записать примыкающие кавычки, а не рекомендует трюк `’$\,$’’` (который также работает)?

Сейчас всю напечатанную чушь принято выделять
многоточиями——и тире—
— ДЖОНАТАН СВИФТ (JONATHAN SWIFT)
Из поэзии: Рапсодия (1733)

Некоторые наборщики отказываются
работать в конторах, где установлены
машины для типографского набора
— ВИЛЬЯМ СТЕНЛИ ДЖЕВОНС (WILLIAM STANLEY JEVONS),
Политическая экономия (1878)

3

Управление системой TEX



Ваша клавиатура имеет гораздо меньшее количество клавиш, чем то количество символов, которые вы, возможно, захотите определить. Для того чтобы превратить ограниченный набор клавиш в достаточно гибкую систему, существует символ, который можно напечатать с помощью клавиатуры и который предназначен для специального использования. Он называется *управляющий символ* (*escape character*). Всякий раз, когда вы захотите изменить что-либо в формате вашего текста или что-либо, для чего нельзя использовать клавиатуру в обычном режиме, вам следует напечатать управляющий символ, сопровождая его указанием на то, что вы хотите сделать.

Замечание: Некоторые клавиатуры имеют ключ 'ESC', но это *не* управляющий символ! Это код, который передает специальный сигнал операционной системе компьютера, так что не путайте его с указателем под названием 'управляющий' (escape).

TEX допускает использование в качестве управляющего любого символа, однако для этих целей обычно выбирают символ "обратная косая черта" ('\'), поскольку обратная косая черта удобна для печати и редко используется в обычном тексте. Самое важное для успешной работы различных пользователей TEXа, чтобы они все делали одинаково, поэтому мы будем управлять через обратную косую черту во всех примерах этого руководства.

Немедленно после напечатания '\' (т. е. немедленно после управляющего символа) вы печатаете команду, сообщающую системе TEX, что вы имеете в виду. Такие команды называются *управляющими последовательностями*. Например, можно напечатать

```
\input MS
```

что (как увидим позже) заставит TEX начать чтение файла с именем 'MS.tex'; строка символов '\input' является управляющей последовательностью. Еще один пример:

```
George P\'olya and Gabor Szeg\"o.
```

TEX преобразует эту запись в 'George Pólya and Gabor Szegö'. Здесь две управляющие последовательности — \' и \". Они используются, чтобы поставить над некоторыми буквами диакритический знак.

Управляющие последовательности имеют два вида. Первый, например \input, называется *управляющим словом*; он содержит управляющий символ, за которым следует одна или несколько *букв*, затем следует пробел или что-нибудь другое, кроме букв. (TEX должна "знать", где конец управляющей последовательности, поэтому следует ставить пробел после управляющего слова, когда следующий символ — буква. Например, если напечатать '\inputMS', то TEX, как и следовало ожидать, воспримет это в качестве управляющего слова из семи букв.) Если вас интересует, что же такое 'буква', то знайте, что TEX воспринимает 52 символа A...Z и a...z как буквы. (*Прим. перев.* — в русифицированной версии TEXа буквами считаются все буквы русского алфавита А...Я и а...я.) Цифры 0...9 *не* рассматриваются как буквы, так что они не появляются в управляющей последовательности первого вида.

Управляющая последовательность второго вида, подобная `\'`, называется *управляющим символом*; она состоит из управляющего символа, за которым стоит единичный знак, *не являющийся буквой*. В этом случае нет необходимости ставить пробел, чтобы отделить управляющую последовательность от следующей за ней буквы, так как управляющая последовательность второго вида после управляющего символа всегда содержит только один символ.

▸ **УПРАЖНЕНИЕ 3.1**
Что является управляющими последовательностями в следующей записи: `\I'm \exercise3.1\\!'`?

▸ **УПРАЖНЕНИЕ 3.2**
Оказывается, если напечатать `P\'olya`, то получим 'Pólya'. Что, по-вашему, следует напечатать, чтобы получить французские слова 'mathématique' и 'centimètre'?

Когда пробел следует за управляющим словом (состоящая полностью из букв управляющая последовательность), то TEX не обращает на него внимания, т. е. она не рассматривает такой пробел как действительно существующий в печатаемом тексте. Но когда пробел стоит после управляющего символа, то он — реальный пробел.

Отсюда возникает вопрос: что нужно сделать, если вы на самом деле *хотите*, чтобы после управляющего слова появился пробел? Позже вы увидите, что TEX относится к двум или более последовательным пробелам как к одному пробелу, поэтому *не* старайтесь печатать два пробела. Точный ответ: напечатайте "управляющий пробел", а именно

`\␣`

(за управляющим символом следует пустое пространство). TEX будет рассматривать его как пробел, который не нужно игнорировать. Отметим, что `\␣` является управляющей последовательностью второго рода, т. е. управляющим символом, так как существует единственная не-буква (␣), которая следует за управляющим символом. Два следующих друг за другом пробела воспринимаются как эквивалент единичного пробела, так что добавочный пробел, непосредственно следующий за `\␣`, будет игнорироваться. Однако если вы хотите получить последовательно три пробела в тексте, то можно напечатать '`\␣\␣\␣`'. Между прочим, наборщиков часто обучают ставить два пробела в конце предложения, но, как увидим позже, у системы TEX для таких случаев есть собственный способ сделать дополнительный пробел. Поэтому нет необходимости думать о том, сколько символов пробелов следует печатать.

Управляющие символы, подобные ⟨return⟩ (они не выпечатываются), могут следовать за управляющим символом, и это приводит к определенным управляющим последовательностям в соответствии с правилами. TEX воспринимает `\`⟨return⟩ и `\`⟨tab⟩ точно так же, как и `\␣` (управляющий пробел). Эти определенные управляющие последовательности лучше не переопределять, потому что в файле нельзя увидеть разницу между ними.

Обычно нет необходимости использовать "управляющий пробел", так как управляющая последовательность нечасто нужна в конце слов. Однако

вот пример: это руководство было набрано в TEXe, и при наборе довольно часто появляется логотип 'TEX', который требует сдвига буквы E вниз и влево. Существует специальное управляющее слово

```
\TeX
```

которое приводит в действие полдюжины команд, которые так необходимы при наборе логотипа 'TEX'. Когда требуется напечатать выражение, подобное 'TEX игнорирует пробелы после управляющих слов.', то в руководстве оно изображается в следующем виде:

```
\TeX\ игнорирует пробелы после управляющих слов.
```

Отметим, что за выражением `\TeX` следует дополнительная косая черта (\), которая задает управляющий пробел. Это необходимо, поскольку TEX игнорирует пробелы после управляющих слов. Без дополнительного пробела получим:

TEXигнорирует пробелы после управляющих слов.

С другой стороны, нельзя во всех ситуациях ставить косую черту (\) после `\TeX`. Рассмотрим, например, фразу:

```
логотип `\TeX'
```

В этом случае дополнительная косая черта (\) не действует, как обычно. Действительно, получается любопытный результат, если напечатать:

```
логотип `\TeX\'
```

Можете предположить, что случиться? Ответ: `\'` — это управляющая последовательность, означающая диакритический знак, как в приведенном выше примере `P\'olya`; в результате получим диакритический знак над следующим непустым символом. Другими словами, будет получен диакритический знак над следующим символом:

логотип 'TEX′

Компьютер правильно следует инструкциям, но он не читает ваши мысли.

TEX понимает около 900 управляющих последовательностей, которые являются частью его встроенного словаря, и все они объяснены в разных местах данного руководства. Однако не пугайтесь такого количества новых терминов — вам не понадобятся многие из них, если не столкнетесь с чрезвычайно сложным текстом. А те, что вам понадобятся, относятся к определенным категориям, усвоить которые не составит большого труда. Например, многие из управляющих последовательностей являются просто названиями определенных символов, используемых в математических формулах: печатаете '`\pi`', чтобы получить 'π'; '`\Pi`', чтобы получить 'Π'; '`\aleph`', чтобы получить '$\aleph$'; '`\infty`', чтобы получить '∞'; '`\le`', чтобы получить '$\le$'; '`\ge`', чтобы получить '$\ge$'; '`\ne`', чтобы получить '$\ne$'; '`\oplus`', чтобы получить '$\oplus$'; '`\otimes`', чтобы получить '$\otimes$'. В приложении F содержится несколько таблиц таких символов.

Не существует встроенного отождествления прописных и строчных букв в названиях управляющих последовательностей. Например, '`\pi`' и '`\Pi`' и '`\PI`' и '`\pI`' — четыре различных управляющих слова.

Около 900 управляющих последовательностей, о которых только что упоминалось, — не окончательная цифра, поскольку легко их определить больше. Например, если вы захотите определить математические символы собственными любимыми именами, которые вам проще, то вы вправе сделать это. В главе 20 объяснено, как это делается.

Приблизительно 300 управляющих последовательностей называются *примитивными*: это элементарные операции низкого уровня, которые неразложимы на более простые функции. Все остальные управляющие последовательности определяются, в конечном счете, в терминах примитивных команд. Например, `\input` — примитивная команда, однако `\'` и `\"` не относятся к таковым — они определяются в терминах примитивной команды `\accent`.

Тяжело даже использовать примитивные управляющие последовательности системы TEX, потому что они ... ну очень ... *примитивны*. Вы напишете массу команд, если захотите, чтобы TEX выполнила операцию низкого уровня, что заберет время и добавит ошибок. Обычно проще использовать управляющие последовательности высокого уровня, которые определяют требуемые операции, вместо того, чтобы каждый раз выискивать способ осуществить операцию. Управляющие последовательности высокого уровня необходимо только однажды определить в терминах примитивных команд. Например, `\TeX` является управляющей последовательностью, которая означает: "напечатать логотип TEX"; `\'` — управляющяя последовательность, которая означает: "поставить диакритический знак над следующим символом". И обе управляющие последовательности могут потребовать различных комбинаций примитивных функций в зависимости от изменения стиля печати. Чтобы изменить логотип TEX, автору следует просто изменить одно определение, и изменения автоматически будут появляться, как только в них возникнет необходимость. Напротив, понадобится громадный объем работы, чтобы изменить логотип, если его определять каждый раз как последовательность примитивных операций.

К высокому уровню относятся управляющие последовательности, которые определяют все форматы документа. Например, в этой книге автор печатает '`\exercise`' непосредственно перед каждым упражнением. Управляющая последовательность `\exercise` была запрограммирована для того, чтобы TEX выполняла следующие действия:

- вычисляла номер упражнения (т. е., '3.2' для второго упражнения в главе 3);
- выпечатывала '▸**УПРАЖНЕНИЕ 3.2**' соответствующим шрифтом с треугольником слева;
- оставляла небольшое дополнительное расстояние между строками или, если потребуется, начинала новую страницу с этой строки;
- запретила отступ на следующей строке.

Это очевидно выгоднее, чем каждый раз вводить все перечисленные команды. Так как руководство полностью описано в терминах управляющих после-

довательностей высокого уровня, то его можно печатать в другом формате, изменив около дюжины определений.

Как отличить примитивную функцию TEXа от управляющей последовательности? Существует два способа: (1) в предметный указатель руководства внесены все обсуждаемые управляющие последовательности, и каждая примитивная функция помечена звездочкой; (2) вы можете вывести значение управляющей последовательности во время работы TEXа. Если напечатано '`\show\cs`', где `\cs` — управляющая последовательность, то TEX предъявит ее текущее значение. Например, запись '`\show\input`', дает '`> \input=\input.`', потому что `\input` является примитивной функцией. С другой стороны, '`\show\thinspace`' приводит к

```
> \thinspace=macro:
->\kern .16667em .
```

Это означает, что `\thinspace` было определено как аббревиатура для '`\kern .16667em`'. Чтобы убедиться, что `\kern` является примитивной функцией, напечатайте '`\show \kern`'. Результаты операции `\show` появятся на экране дисплея и в log-файле, где и можно найти их после окончания работы TEX.

▸ **УПРАЖНЕНИЕ 3.3**

Какая из управляющих последовательностей \␣ и \⟨return⟩ является примитивной?

В следующей главе будет часто обсуждаться формат "простая система TEX" ("plain TEX"), которая содержит около 600 базовых управляющих последовательностей, определенных в приложении B. Когда TEX начинает работу с текстом, ей доступны эти управляющие последовательности вместе с приблизительно 300 примитивными функциями. Это и является причиной того, что TEX при старте претендует на знание приблизительно 900 управляющих последовательностей. Мы увидим, как простую систему TEX можно использовать для создания документов в гибких форматах, которые находят применение в большинстве случаев, как их получить с помощью только тех шрифтов, которые включены в систему TEX. Тем не менее запомните, что простая система TEX — только одна из бесчисленного множества форматов, которые могут быть созданы на базе примитивных функций TEX. Если вам нужен другой формат, то можно преобразовать TEX таким образом, чтобы получить все, что вы хотите. Лучший способ научиться — начать с простой системы TEX, изменяя мало-помалу ее определения по мере приобретения опыта.

Приложение E содержит примеры форматов, которые можно присоединить к приложению B для применения в особых случаях. Например, существует множество определений, подходящих для деловой корреспонденции. В приложении E также дана полная инструкция по форматам, которые использовались для набора данного руководства. Таким образом, если вашей целью является изучить, как конструировать форматы в системе TEX, то вы, возможно, захотите изучить приложение E одновременно с приложением B. После этого вы научитесь определять управляющие последовательности. Возможно, вы разработаете форматы, которые захотят использовать другие; тогда вам следует написать приложение к данному руководству, чтобы объяснить правила вашего стиля.

Основная задача этих замечаний — заинтересовать новичков в использовании системы TEX, чтобы действительно появилась возможность опреде-

лять нестандартные управляющие последовательности системы TEX. Когда в руководстве говорится, что нечто является частью системы TEX, то это не означает требования поступать именно таким способом; любой может изменять правила, изменяя одно или более определений в приложении B. Но вы можете полагаться на управляющие последовательности системы TEX, пока не станете опытным TEXником-наборщиком.

▸ **УПРАЖНЕНИЕ 3.4**

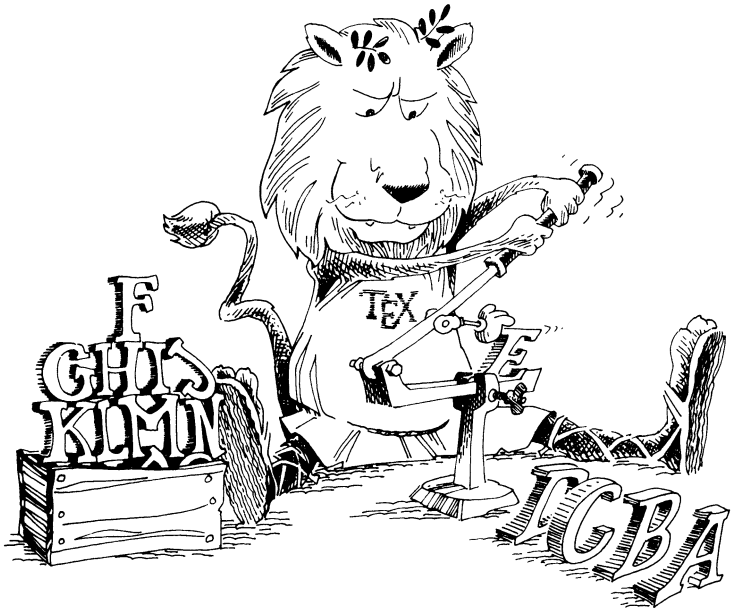
Сколько существует (включая управляющий символ) управляющих последовательностей длины 2? А сколько длины 3?

Слоги управляют словами.
— ДЖОН СЕЛДЕН (JOHN SELDEN),
Застольные беседы (1689)

Я не претендую на то, что управлял событиями,
но откровенно признаю, что события управляли мною.
— АВРААМ ЛИНКОЛЬН (ABRAXAM LINCOLN) (1864)

4

Выбор шрифта



Возможно, вы захотите перейти от одного шрифта к другому, например использовать **жирный шрифт** или *выделить* что-либо. TEX оперирует с массивами, содержащими свыше 256 символов, называемых "шрифтами", и использует управляющие последовательности для выбора определенного шрифта. Например, используя форматы простой системы TEX из приложения B, можно задать последние несколько слов в первом предложении в следующем виде:

```
использовать \bf жирный шрифт \rm или \sl выделить
\rm что-либо.
```

Простая система TEX для выбора шрифтов предусматривает следующие управляющие последовательности:

`\rm` переключает на обычный латинский шрифт:	Roman
`\sl` переключает на наклонный латинский шрифт:	*Slanted*
`\it` переключает на курсивный шрифт:	*Italic*
`\tt` — на шрифт, подобный шрифту пишущей машинки:	`Typewriter`
`\bf` переключает на стиль жирного шрифта:	**Bold**

В начале работы устанавливается латинский шрифт (`\rm`), если только вы не установите иной.

Отметим, что в двух из этих шрифтов буквы имеют наклонное положение для выразительности: *Наклонный шрифт является в сущности таким же, как и латинский, но буквы слегка скошены, тогда как буквы курсива имеют начертание в другом стиле.* (Возможно, вы сможете лучше ощутить различие между латинским стилем и курсивом, изучая буквы напечатанные не наклонным курсивным шрифтом.) Соглашения печатников находятся сейчас в переходном состоянии, потому что новые технологии дают возможность делать то, что раньше было слишком дорого. Люди пытаются эффективнее использовать обретенную свободу. Наклонный латинский шрифт был введен в 1930-х годах, но он впервые стал широко использоваться как альтернатива общепринятому курсиву в конце 1970-х. Он может быть полезен в математических текстах, поскольку наклонные буквы отличимы от курсива в математических формулах. Двойное использование курсива (курсивный шрифт) для двух различных целей — например, утверждения теорем выделены курсивом точно так же, как и переменные в этих теоремах — приводит к некоторой путанице, которой можно избежать благодаря наклонному шрифту. Обычно люди расходятся во взглядах при сравнении наклонного шрифта с курсивом, однако наклонный шрифт все чаще становится излюбленным для названий книг и журналов в библиографиях.

Особые шрифты полезны для выразительности, но не для длительного чтения, ваши глаза устанут, если большие части этого руководства будут полностью напечатаны жирным шрифтом, наклонным или курсивом. Поэтому латинский шрифт считается для основной массы печатной продукции наилучшим видом шрифта. Однако неудобно печатать '`\rm`' каждый раз, когда вы хотите возвратиться к латинскому шрифту, поэтому TEX обеспечивает более легкий способ — использование символов "фигурные скобки": можно переключать шрифты внутри специальных символов `{` и `}`, не затрагивая шрифты

вне фигурных скобок. Например, напечатанная в начале этой главы фраза записана так:

```
использовать {\bf жирный шрифт} или {\sl выделить} что-либо.
```

Этот отдельный случай общей идеи "группирования" будет обсуждаться в следующих главах. Лучше забудем о первом способе изменения шрифтов и вместо него используем группирование, тогда ваш текст напечатанный в системе TEX будет выглядеть более естественно и вам, возможно, никогда* не придется печатать '`\rm`'.

▸**УПРАЖНЕНИЕ 4.1**
Объясните, как напечатать ссылку на литературу 'Ulrich Dieter, *Journal für die reine und angewandte Mathematik* **201** (1959), 37–70'. [Воспользуйтесь группированием.]

В предшествующей дискуссии мы затронули аспект качества. Обратите внимание, например, на выделенные и наклонные слова в предложении. Поскольку курсив и наклонный шрифты наклонены вправо, то они "приклеиваются" к пробелу, который отделяет эти слова от следующих за ними слов, напечатанных латинским шрифтом, в результате пробелы кажутся слишком узкими, несмотря на то, что они в основном правильны по отношению к размерам букв. Чтобы уравновесить действительное белое пространство между словами, TEX дает возможность поставить специальную управляющую последовательность '`\/`', прежде чем переключиться снова на прямые буквы. Когда вы печатаете

```
{\it выделенные курсивом\/} и {\sl наклоном\/} слова,
```

то получаете *выделенные курсивом* и *наклоном* слова, которые лучше смотрятся. Управляющая последовательность '`\/`' указывает системе TEX, что нужно добавить к предыдущей букве *поправку на курсив*, рассчитанную на эту букву. В обычном курсиве эта поправка примерно в четыре раза больше для буквы '*f*', чем для '*c*'.

Иногда поправка на курсив нежелательна, потому что другие факторы визуально ослабляют ее. Стандартное правило 'останови проезжающий автомобиль, подняв большой палец' гласит: используйте `\/` только перед переходом от наклонного или курсивного шрифта к латинскому или жирному, если следующим символом не является точка или запятая. Напечатаем, например,

```
{\it курсив\/} для {\sl выделения}.
```

Старые руководства по стилям говорят, что для пунктуации после слова следует использовать *такой* же шрифт, как и для *слова;* но курсивная точка с запятой часто выглядит некрасиво, так что это соглашение отменяется. Когда за выделенным словом появляется точка с запятой, то автор рекомендует печатать '`{\it слово\/};`'.

---

* Хорошо, но едва ли ...

▸**УПРАЖНЕНИЕ 4.2**

Объясните, как напечатать латинским шрифтом *слово в середине выделенного курсивом предложения.*

Буквы всех шрифтов имеют поправку на курсив, которую можно ввести, напечатав `\/`. Для прямых шрифтов эта поправка обычно равна нулю, но существуют исключения: чтобы напечатать жирное '**f**' в кавычках, следует сказать `жирное '{\bf f\/}'`, иначе получите жирные букву и кавычку '**f'**.

▸**УПРАЖНЕНИЕ 4.3**

Определите управляющую последовательность `\ic` таким образом, чтобы команда '`\ic c`' помещала поправку на курсив символа *c* в регистр системы TEX `\dimen0`.

Примитивная управляющая последовательность `\nullfont` означает шрифт, который не содержит символов. Этот шрифт присутствует всегда, если не определены другие.

Шрифты изменяются по размеру так же, как и по форме. Например, шрифт, который вы сейчас читаете, называется шрифтом в "10 пунктов", потому что все детали его элементов имеют величину 10 пунктов, если измерять в печатных единицах. (Мы будем изучать систему пунктов позже, сейчас достаточно указать, что круглые скобки, в которые заключено это предложение, имеют точно 10 пунктов в высоту и em-тире — точно 10 пунктов по ширине.) Разделы "опасный поворот" руководства выполнены шрифтом в 9 пунктов, для сносок выбран шрифт в 8 пунктов, нижние индексы — в 7 или 6 пунктов, а для индексов нижних индексов выбран шрифт в 5 пунктов. Каждому шрифту, который используется в тексте TEXa, соответствует управляющая последовательность; например шрифт в 10 пунктов в этом параграфе называется `\tenrm`, а шрифт, соответствующий 9 пунктам, — `\ninerm`. Наклонный шрифт, соответствующий шрифтам `\tenrm` и `\ninerm`, называется `\tensl` и `\ninesl`. Эти управляющие последовательности не встроены в TEX, но они являются фактическими названиями шрифтов, предполагается, что пользователи системы TEX составят подходящее название, всякий раз, когда новый шрифт вводится в употребление в тексте. Такие управляющие последовательности используются, чтобы изменить вид шрифта.

Когда одновременно используются шрифты различных размеров, то TEX выстраивает буквы вдоль соответствующей им "основной линии". Например, если напечатать:

```
\tenrm меньше \ninerm и меньше
\eightrm и меньше \sevenrm и меньше
\sixrm и меньше \fiverm и меньше \tenrm,
```

то в результате получим меньше и меньше и меньше и меньше и меньше. Конечно, существуют вещи, к которым не привыкли ни автор, ни читатель, потому что с традиционной свинцовой печатью наборщики не могли делать такого. Возможно, поэты, которые захотят говорить столь слабым голосом, дадут повод выпускать в будущем книги с использованием часто изменяемых шрифтов, но на сегодняшний день это только случайное чудачество (как у автора книги), которому

нравятся подобные эксперименты. Не следует слишком увлекаться переключением шрифтов, если на это нет веских оснований.

Бдительный читатель может запутаться в этих пунктах, так как в начале главы '`\rm`' объявлялся командой, с помощью которой переходят на латинский шрифт, однако позже было сказано, что '`\tenrm`' — это метод сделать то же самое. Истина в том, что работают оба метода. Но стало уже привычным то, что `\rm` означает "переключение на латинский шрифт текущего размера", тогда как `\tenrm` — "переключение на латинский шрифт размером 10 пунктов". Формат простой системы TEX никаким другим, кроме шрифта в 10 пунктов не обеспечивает, поэтому команда `\rm` всегда дает `\tenrm`, однако в более сложных форматах значение `\rm` будет изменяться в различных частях руководства. Например, в формате, который автор использовал для печатания этой книги, существует управляющая последовательность '`\tenpoint`', благодаря которой `\rm` будет означать `\tenrm`; `\sl` — `\tensl` и т. д., до тех пор, пока '`\ninepoint`' не изменит определение таким образом, что `\rm` будет означать `\ninerm` и т. д. Существуют другие управляющие последовательности, которые используются, чтобы напечатать цитаты в конце каждой главы. Когда печатаются цитаты, то `\rm` и `\sl` временно означают прямой без засечек шрифт в 8 пунктов и наклонный без засечек шрифт в 8 пунктов, соответственно. Это изобретение постоянного переобозначения за кулисами аббревиатур `\rm` и `\sl` освобождает наборщика от необходимости запоминать, какой размер или вид шрифта используется в данный момент.

▸ **УПРАЖНЕНИЕ 4.4**

Как вы думаете, почему автор выбрал названия '`\tenpoint`' и '`\tenrm`' и т. д. вместо '`\10point`' и '`\10rm`'?

▸ **УПРАЖНЕНИЕ 4.5**

Предположим, что вы печатаете текст, используя для выразительности наклонный шрифт, но ваш издатель неожиданно предложил вам заменить весь наклонный шрифт на курсивный. Как легче всего сделать это?

Каждый шрифт имеет внешнее название, которое определяет его по отношению ко всем остальным шрифтам в особой библиотеке. Например, для набора текстов обычно используется шрифт '`cmr9`' — эта аббревиатура расшифровывается как "Computer Modern Roman 9 point". Для того чтобы подготовить TEX к использованию этого шрифта, нужно записать команду `\font\ninerm=cmr9`, приведенную в приложении E. Чтобы загрузить информацию об определенном шрифте в память системы TEX, обычно говорят '`\font\cs=`⟨внешнее название шрифта⟩'. После этого управляющая последовательность `\cs` будет выбирать этот шрифт для печати. В простой системе TEX имеются только 16 шрифтов, однако, вы можете использовать операцию `\font`, чтобы получить доступ к любому шрифту из системной библиотеки шрифтов вашего компьютера.

Иногда есть возможность часто использовать шрифты различных размеров, увеличивая или сокращая изображение символа. Каждый шрифт имеет так называемый расчетный размер, который отражает его обычный размер по умолчанию (например, расчетный размер шрифта `cmr9` равен 9-и пунктам). Однако во многих системах существует также диапазон размеров шрифтов, в котором можно использовать определенный шрифт, уменьшая или увеличивая его размер. Чтобы загрузить в память TEXа масштабированный шрифт, укажите просто

‘`\font\cs=`⟨внешнее название шрифта⟩ `at` ⟨требуемый размер⟩’. Например, команда

```
\font\magnifiedfiverm=cmr5 at 10pt
```

заставит увеличить шрифт Computer Modern Roman в 5 пунктов вдвое по сравнению с его обычным размером. (Предупреждение: прежде чем использовать свойство ‘`at`’, следует убедиться, что ваше печатающее устройство поддерживает шрифт запрашиваемого размера. TEX допускает любой положительный и меньше 2048-и пунктов ⟨требуемый размер⟩, но окончательный результат будет неверный, если задаваемый масштаб шрифта не поддерживается печатающим устройством вашего компьютера.)

Каково различие между `cmr5 at 10pt` и обычным шрифтом в 10 пунктов, `cmr10`? Множество. Изображение хорошо сконструированного шрифта в разных пунктах размера будет различаться, и буквы часто будут иметь различную относительную высоту и ширину, чтобы улучшить читабельность.

> Шрифт в 10 пунктов отличается от **увеличенного шрифта в 5 пунктов.**

Обычно лучше только немного изменять масштаб шрифта относительно его разработанных размеров, если вы не станете фотографически уменьшать текст по окончании работы TEXа или не попытаетесь получить необычный эффект.

Другим путем для увеличения шрифта является масштабный коэффициент, т. е. связанный с разработанным размером. Например, команда

```
\font\magnifiedfiverm=cmr5 scaled 2000
```

является другим способом увеличения вдвое шрифта `cmr5`. Масштабный коэффициент определяется как целое число, которое представляет увеличение, пропорциональное 1000. Таким образом, масштабный множитель 1200 определяет увеличение в 1.2 раза и т. д.

▸ **УПРАЖНЕНИЕ 4.6**
Назовите два способа загрузить в память системы TEX шрифт `cmr10` в половину его нормального размера.

Для многих компьютерных центров удобно обладать шрифтами с увеличением в геометрической прогрессии — некоторое подобие хорошо темперированного клавира. Благодаря этому доступны все шрифты как истинного размера, так и увеличенного в 1.2 и 1.44 (что равно 1.2×1.2), возможно также увеличение в 1.728 ($= 1.2 \times 1.2 \times 1.2$) и даже больше. Тогда можно увеличить полностью текст в 1.2 или 1.44 раза и при этом оставаться в множестве доступных шрифтов. В простой системе TEX предусмотрены аббревиатуры `\magstep0` для масштабного коэффициента 1000, `\magstep1` для масштабного коэффициента 1200, `\magstep2` для масштабного коэффициента 1440 и также до `\magstep5`. Например, пишем

```
\font\bigtenrm=cmr10 scaled\magstep2,
```

чтобы загрузить шрифт `cmr10`, увеличенный в 1.2×1.2 раз.

> “Это нормальный размер шрифта `cmr10` (`\magstep0`)”.
> “Это шрифт `cmr10`, увеличенный только в 1.2 раза (`\magstep1`)”.
> “Это шрифт `cmr10`, дважды увеличенный в 1.2 раза (`\magstep2`)”.

(Отметим, что небольшим увеличением достигаем многого.) Существует также команда `\magstephalf`, которая дает увеличение в $\sqrt{1.2}$, т. е. где-то между 0 и 1.

В главе 10 объясняется, как увеличить документ дополнительно к увеличению, которое задавалось когда шрифты загружались. Например, если загружен шрифт, который можно увеличить, используя `\magstep1`, и если вы также определили `\magnification=\magstep2`, то фактически шрифт, используемый для печати, будет увеличен в `\magstep3`. Аналогично, если загружен шрифт, увеличенный в `\magstephalf` и если вы также запишете `\magnification=\magstephalf`, то в результате для печати шрифт увеличится в `\magstep1`.

Шрифты подобны лицам — имеют индивидуальные особенности, свидетельствующие об их характерах.

— МАРШАЛЛ ЛИ (MARSHALL LEE), *Книгопечатание* (1965)

Это был самый благородный римлянин.

— ВИЛЬЯМ ШЕКСПИР (WILLIAM SHAKESPEARE), *Юлий Цезарь* (1599)

5

Группирование

Время от времени возникает необходимость рассматривать часть текста как одно целое, тогда нужно как-нибудь отметить начало и конец этой части. Для этих целей в системе TEX существует определенная интерпретация двух "символов группирования", которые (подобно управляющим символам) обрабатываются иначе, чем обычные символы, которые вы печатаете. В данном руководстве предполагается, что `{` и `}` являются группирующими символами, поскольку только их использует простая система TEX.

В предыдущей главе приведены примеры группирования, в которых упоминается, что изменение шрифта внутри группы не влияет на шрифт вне группы. Такой же принцип применяется, как увидим позже, почти ко всему, что определено внутри группы. Например, если определить в группе управляющую последовательность, то определение потеряет силу, когда группа кончится. Таким способом удобно давать указания системе TEX, чтобы она сделала нечто необычное, временно изменяя его нормальное состояние внутри группы, поскольку эти указания неразличимы вне группы. Не следует беспокоиться, что это приведет к беспорядку в остальном тексте, если вы забудете восстановить обычное состояние, когда редкая конструкция закончится. У компьютерщиков существует название для такого вида группирования (так как это вообще важный аспект языков программирования), они называют его "блочной конструкцией", а определения, которые имеют силу только внутри группы, — "локальными" для этой группы.

Вы, возможно, захотите использовать группирование, даже если вас не интересует блочная конструкция, — просто, чтобы лучше управлять шпациями (*прим. перев.* шпация — это пробел шириной от одного пункта до числа пунктов равного кеглю данного слова). Например, еще раз рассмотрим управляющую последовательность `\TeX`, с помощью которой получен логотип 'TEX' в данном руководстве. В главе 3 отмечалось, что пробел после этой управляющей последовательности съедается, если не напечатать '`\TeX\ `', кроме того, запись '`\TeX\`' приведет к ошибке, если следующий символ не является пробелом. Во *всех* этих случаях было бы правильным определить простую группу

```
{\TeX}
```

независимо от того, будет или нет следующий символ пробелом, поскольку '`}`' предотвратит процесс поглощения необязательного пробела в `\TeX`. Это может быть кстати, если использовать текстовый редактор (например, когда заменять все определенные слова во всем тексте с помощью управляющей последовательности). По-другому то же самое можно сделать, напечатав

```
\TeX{}
```

т. е. используя *пустую группу* в тех же целях. '`{}`' — это группа, которая не содержит символов, не дает никакого результата, а лишь останавливает TEX от съедания пустого места.

▸ **УПРАЖНЕНИЕ 5.1**

Иногда встречаются редкие слова, подобные 'shelfful', которое лучше смотрится в виде 'shelfful' без лигатуры 'ff'. Как обмануть TEX, чтобы она не поняла, что в этом слове последовательно стоят два 'f'?

▸ **УПРАЖНЕНИЕ 5.2**
Объясните, как получить три пробела подряд, не используя '\␣'.

TEX также использует группирование для совершенно различных целей, а именно, чтобы определить, насколько можно управлять текстом с помощью определенных управляющих последовательностей. Например, если вы хотите поместить что-либо в центре строки, то можете напечатать

```
\centerline{Эта информация должна быть в центре.}
```

используя управляющую последовательность `\centerline`, определенную в формате простой системы TEX.

Группирование используется системой TEX во многих более сложных командах, и возможно создание группы внутри группы, как можно увидеть, заглянув в приложение B. Однако в обычных текстах, как правило, нет необходимости в сложном группировании, так что не стоит об этом беспокоиться. Только не забывайте закрыть каждую группу, которую вы открыли, так как забытая '}' может причинить неприятности.

Приведем пример двух групп вложенных одна в другую:

```
\centerline{Эта информация должна быть в {\it центре}.}
```

Как и предполагалось, TEX расположила предложение, последнее слово которого записано курсивом, посередине строки

Эта информация должна быть в *центре*.

Но присмотритесь более внимательно к примеру: '`\centerline`' оказывается стоит перед фигурными скобками, в то время как '`\it`' оказалось внутри. В чем различие? И как может запомнить новичок, как их различать? Ответ: `\centerline` — это управляющая последовательность, которую применяют к следующему непосредственно за ней тексту. Таким образом, требуется заключить в фигурные скобки текст, который будет расположен посередине строки (если только это не простой символ или управляющая последовательность). Например, для того, чтобы записать логотип TEX посередине строки, достаточно записать '`\centerline\TeX`', но для фразы 'TEX имеет группы' необходимы фигурные скобки: '`\centerline{\TeX\ имеет группы}`'. С другой стороны, `\it` — это управляющая последовательность, которая означает "замените текущий шрифт". Она действует, не просматривая последующий текст, так что воздействует на *все*, что за ней следует, по крайней мере потенциально. Управляющую последовательность `\it` заключают в фигурные скобки для того, чтобы ограничить изменение шрифта в локальной области.

Другими словами, две группы фигурных скобок в этом примере в действительности имеют различные функции: одна служит тому, чтобы рассматривать несколько слов текста как единый объект, в то время как другая дает локальную блочную структуру.

▸ **УПРАЖНЕНИЕ 5.3**
Что по-вашему произойдет, если напечатать:

```
\centerline{Эта информация должна быть в {центре}.}
\centerline Так должно быть.
```

▸ **УПРАЖНЕНИЕ 5.4**

А что вы думаете об этом?

```
\centerline{Эта информация должна быть в \it центре.}
```

▸ **УПРАЖНЕНИЕ 5.5**

Определите управляющую последовательность `\ital` так, чтобы пользователь мог печатать '`\ital{текст}`' вместо '`{\it текст\/}`'. Приведите доводы за и против `\ital` в сравнении с `\it`.

В последующих главах описывается много примитивных операций TEXa, для которых важно локальное группирование. Например, если один из внутренних параметров TEXa изменен внутри группы, то предыдущее значение параметра восстанавливается, когда группа закрывается. Тем не менее иногда желательно сделать определения, действие которых выходит за пределы текущей группы. Это достигается тем, что перед определением ставят оператор '`\global`'. Например, TEX хранит номер текущей страницы в регистре `\count0`, и программа вывода номеров страниц захочет увеличить номер страницы. Программы вывода всегда защищают заключением в группы, чтобы их команды не повлияли на оставшуюся часть текста на TEXe. Однако `\count0` исчезнет, если его поместить в локальную группу. Команда

```
\global\advance\count0 by 1
```

решает эту проблему — она увеличивает `\count0` и сохраняет это значение до конца программы вывода. Вообще, `\global` выполняется немедленно после определения и относится ко всем существующим группам, а не только к той, которая лежит внутри нее.

▸ **УПРАЖНЕНИЕ 5.6**

Если вам понятны локальные и глобальные определения, то вы, несомненно, выполните небольшое задание: предположим, что `\c` определено как '`\count1=`', `\g` — как '`\global\count1=`' и `\c` — как '`\showthe\count1`'. Какие значения показаны в следующей записи?

```
{\c1\s\g2{\s\c3\s\g4\s\c5\s}\s\c6\s}\s
```

Для того чтобы другим способом получить в системе TEX блочную структуру, применяются примитивные функции `\begingroup` и `\endgroup`. Этими управляющими последовательностями легко открыть группу внутри одной управляющей последовательности и закрыть ее в другой. Текст, который TEX в действительности выполняет (он должен быть подробно изложен после управляющих последовательностей), должен иметь правильно вложенные группы, т. е. группы не должны перекрываться. Вот пример неправильной управляющей последовательности

```
{ \begingroup } \endgroup.
```

▸ **УПРАЖНЕНИЕ 5.7**

Определите управляющие последовательности `\beginthe`⟨название блока⟩ и `\endthe`⟨название блока⟩, которые дают блочную структуру "с заданным вами названием". Другими словами,

```
\beginthe{бегуэн}\beginthe{вальс}\endthe{вальс}\endthe{бегуэн}
```

— допустимая управляющая последовательность, но не

```
\beginthe{бегуэн}\beginthe{вальс}\endthe{бегуэн}\endthe{вальс}.
```

Я могу обратиться к множеству шрифтов и фигурных скобок.

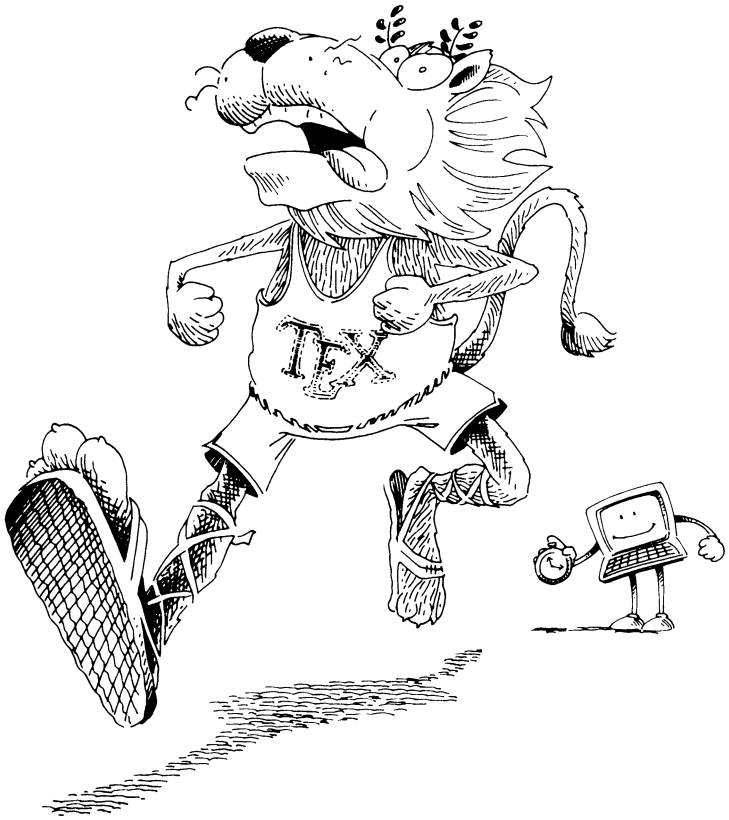
— ДЖЕЙМС МЮРХЕД (JAMES MU|RHEAD),
Общества Gaius (1880)

Конфликтная группа — это собрание на несколько часов или дней, человек двенадцать или восемнадцать ответственных, уверенных, достоверно нормальных и временно дурно пахнущих людей.

— ДЖЕЙН ГОВАРД (HOWARD JANE),
Приятное прикосновение (1970)

6

Работаем с системой TEX



Лучший способ изучить систему TEX — использовать ее. Поэтому садитесь за компьютер и пообщайтесь в диалоговом режиме с системой TEX, попытайтесь что-нибудь написать и посмотрите, что получится. Предложим для первого знакомства несколько маленьких, но законченных примеров.

Предупреждаю: глава достаточно длинная. Может, лучше прекратить чтение и возобновить завтра на свежую голову?

Замечательно, предполагаю, что вы отдохнули и взволнованы перед пробным запуском системы TEX. Шаг за шагом инструкции для пользователя появятся в этой главе. Вначале сделаем так: пройдите в лабораторию, где установлено устройство вывода, поскольку вам захочется вывести на печать все, что вы получите. Не думаю, что приятно запускать TEX на расстоянии, когда нельзя подержать в руках полученные документы. Затем введите пароль и запустите TEX. (Возможно, вас заинтересует, что делать на собственном компьютере. Обычно операционная система выводит приглашение, после которого вы должны ввести '`tex`' или '`run tex`' или что-нибудь в этом роде.)

Если вы успешно справитесь с этим, TEX пригласит вас таким сообщением:

```
This is TeX, Version 3.141 (preloaded format=plain 89.7.15)
**
```

'`**`' — таким образом TEX спрашивает название вводимого файла.

Затем печатаете '`\relax`' (включая обратную косую черту) и ⟨return⟩ (или используйте что-либо, что означает "конец строки", например "end-of-line"). TEX полностью приведена в действие, готова читать длинный текст, но вы скажете, что все это очень легко, потому что к выполнению предлагаются действительно простые действия. В самом деле, `\relax` — управляющая последовательность, которая означает "ничего не делать".

Компьютер напечатает для вас другую звездочку. На этот раз напечатайте нечто вроде "`Привет!`" и ждите появления следующей звездочки. Наконец, напечатайте '`\end`' и посмотрите, что получилось.

TEX должна ответить — '`[1]`' (означающее, что окончена страница 1 введенного вами текста), затем программа остановится, возможно, сообщив, что создан файл под названием '`texput.dvi`'. (Система TEX использует `texput` как название для окончательного результата, если в первой строке приглашения не определено другое; '`dvi`' означает — "независимо от устройства", так как `texput.dvi` допускает печать на почти любом типографском устройстве для печати.)

Сейчас вам понадобится помощь ваших друзей, местных хакеров. Они подскажут вам, как получить распечатку файла `texput.dvi`. И тогда вы увидите распечатку — о, восхитительный день! — и великолепное "Привет!", и внизу номер страницы '1'. Поздравляем вас с первым напечатанным шедевром.

Суть в том, что теперь понятно, как пройти весь цикл. Теперь остается проделать то же самое с более длинным текстом. Итак, следующий эксперимент — работа с файлом вместо того, чтобы печатать текст по строчке.

Воспользуемся вашим любимым текстовым редактором, чтобы создать файл под названием `story.tex`, который содержит 18 строк:

```
1  \hrule
2  \vskip 1in
3  \centerline{\bf A SHORT STORY}
4  \vskip 6pt
5  \centerline{\sl by A. U. Thor}
6  \vskip .5cm
7  Once upon a time, in a distant
8    galaxy called \"O\"o\c c,
9  there lived a computer
10 named R.~J. Drofnats.
11
12 Mr.~Drofnats---or ``R. J.,'' as
13 he preferred to be called---
14 was happiest when he was at work
15 typesetting beautiful documents.
16 \vskip 1in
17 \hrule
18 \vfill\eject
```

(Конечно, не следует печатать слева номера строк, они стоят только для ссылок.) Этот пример несколько глуповат, но он несложен для такого наборщика, как вы, и даст вам некоторый интересный опыт. Так что печатайте этот текст. Для вашего же блага. И думайте, как и что вы печатаете. В примере приводится несколько важных свойств системы TEX, которые можно изучить в процессе работы над файлом.

А сейчас дадим краткое объяснение того, что сейчас напечатано: строки 1 и 17 проведут горизонтальную черту (тонкую линию) поперек страницы. Строки 2 и 16 пропускают 1 дюйм (≈ 2.5 см), '`\vskip`' означает "вертикальный пробел" и этот дополнительный промежуток будет отделять горизонтальную черту от остального текста.

Строки 3 и 5 расположат в центре название рассказа и имя автора, напечатанные жирным и наклонным шрифтами. Строки 4 и 6 создадут дополнительное расстояние между строками (строкой 3 и 5, 5 и 7). (Единицы измерения, подобные '`6pt`' и '`.5cm`', обсуждаются в главе 10.)

Рассказ занимает строки 7–15 и состоит из двух абзацев. В действительности строка 11 является пробелом, информирующим TEX, что строка 10 — конец первого абзаца, а '`\vskip`' на строке 16 означает, что второй абзац заканчивается на строке 15, потому что вертикальные пробелы не появляются внутри абзацев. В данном случае пример, видимо, всецело заполнен указаниями системе TEX, но в этом отношении он нетипичен, поскольку короток и приведен только для обучения. Беспорядочные конструкции, подобные `\vskip` и `\centerline`, допустимы только в первых набранных текстах, если еще не используется готовый формат, но это продлится недолго. Вскоре вы обнаружите, что правильно печатаете текст с относительно небольшим количеством управляющих последовательностей.

И хорошие новости для тех, кто прежде не использовал для набора текста компьютер: вам не придется заботится о том, где прерывать строку