

Это руководство предназначено как для тех, кто никогда не пользовался $\text{T}_\text{E}\text{X}$ ’ом раньше, так и для опытных пользователей.

Д. Кнут, Все про $\text{T}_\text{E}\text{X}$

Как мы увидим ... , сортировка помогает и при поиске, с ее помощью можно сделать выдачи ЭВМ более удобными для человеческого восприятия. В самом деле, листинг (распечатанный машинный документ), отсортированный в алфавитном порядке, зачастую выглядит весьма внушительно, даже если соответствующие числовые данные были рассчитаны неверно.

Д. Кнут, Искусство программирования для ЭВМ

ИНСТИТУТ ФИЗИКИ ВЫСОКИХ ЭНЕРГИЙ

М.В. Лисина

PLAIN T_EX
ОСНОВНЫЕ ПОНЯТИЯ
И КАТАЛОГ КОМАНД

Под редакцией С.В. Клименко

Протвино, 1995 г.

Содержание

Введение	4
1. Основные понятия $\text{\TeX}$ 'а	5
1.1. Грамматика $\text{\TeX}$ 'а	5
Символы	5
Кавычки	5
Тире	5
Пробелы	5
Команды	6
Размеры	6
Группирование	6
1.2. Шрифты $\text{\TeX}$ 'а	7
1.3. Как $\text{\TeX}$ читает входной файл	8
Боксы	8
Клей	10
Моды	11
1.4. Как $\text{\TeX}$ 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 $\text{\TeX}$ 'а	22
3. Каталог $\text{\TeX}$ 'а	23
Список литературы	156

Введение

Уважаемый читатель! Представляем вам краткое описание основных понятий и полный каталог команд системы $\text{\TeX}$ — 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 Кнудом и в настоящее время широко распространена во всем мире. Подготовить рукопись по правилам $\text{\TeX}$ '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 $\text{\TeX}$ 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 Кнудом макронадстройка из примерно 600 макрокоманд образует формат plain $\text{\TeX}$. В настоящее время образовалось довольно много различных макронадстроек, таких, как $\text{\LaTeX}$, $\text{\AMS-TeX}$, $\text{\RNZZX}$ и т. д., из которых особой популярностью пользуется $\text{\LaTeX}$, предназначенный для подготовки научно-технических документов стандартных стилей (статья, доклад, книга, препринт, ...). Если набора макрокоманд plain $\text{\TeX}$ 'а (или какого-нибудь другого формата) Вам покажется недостаточно, не составляет особого труда определить свои собственные макрокоманды.

В настоящее время $\text{\TeX}$ 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 $\text{\TeX}$ '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 $\text{\TeX}$ '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 $\text{\TeX}$ 'а.

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

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

1. Основные понятия Т_ЕX’а

1.1. Грамматика Т_ЕX’а

Входной информацией для Т_ЕX’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_ЕX 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_ЕX’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
<code>\</code>	Сигнальный символ	<code>\$\backslash\$</code>
<code>{</code>	Признак начала группы	<code>\${}</code>
<code>}</code>	Признак конца группы	<code>\$\}\$</code>
<code>\$</code>	Переключатель в математическую моду	<code>\\$</code>
<code>&</code>	Табулятор	<code>\&</code>
<code>#</code>	Признак параметра в макроопределениях	<code>\#</code>
<code>^</code>	Верхний индекс	<code>\^</code>
<code>_</code>	Нижний индекс	<code>_</code>
<code>%</code>	Символ комментария	<code>\%</code>
<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 (\bigl и \Bigl — это разные команды). Имеются также слова, которые в определенном контексте имеют особый смысл. Такие слова называются *ключевыми*. Например, в конструкции \hbox 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 T_EX'a

Когда Вы начинаете готовить входной файл для T_EX'a, Вам надо знать, какими полиграфическими возможностями Вы можете располагать. В plain T_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 T_EX может иметь до 256 символов на шрифт. Так, например, если Вы запрашиваете `\char'35` в шрифте `cmr10`, то получаете $\text{\textcircled{A}}$. Текстовые шрифты включают *лигатуры* и *акценты*.

Plain T_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>cmti10</code>	(Computer Modern Text Italic в 10 пунктах)		
<code>cmtt10</code>	(Computer Modern Typewriter type в 10 пунктах)		
<code>cmni10</code>	(Computer Modern Math Italic в 10 пунктах)		
<code>cmni7</code>	(Computer Modern Math Italic в 7 пунктах)		
<code>cmni5</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`.

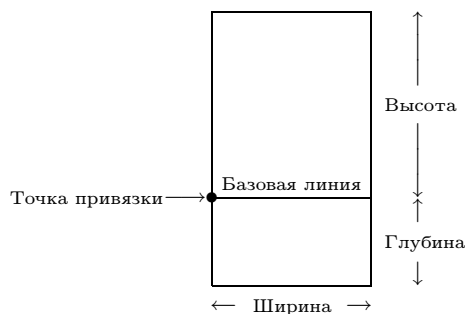
1.3. Как $\TeX$ читает входной файл

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_ЕХнические термины, как *боксы* и *клеи*.

Боксы в Т_ЕХ'е — 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_ЕХ'а 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_ЕХ 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_ЕХ'а (`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_ЕХ'ом страницы составлены из простых символьных и линейчатых боксов, соединенных в различных комбинациях. Т_ЕХ соединяет боксы двумя способами, *горизонтально* и *вертикально*. Когда Т_ЕХ 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_ЕХ 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` 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 вкратиться 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 $\text{\TeX}$ считает точку ординарным символом. Если символ ‘:’ должен быть знаком пунктуации, а не отношением, задавайте его командой $\text{\code\colon}$. Если вы хотите использовать запятую как ординарный символ, поставьте ее в фигурных скобках: $\text{\TeX}$ 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 $\text{\code\prime}$. Специальные символы не используются в качестве символов в математической моде, если только не было изменено значение их $\text{\code\catcode}$. Хотя { и } указывают группирование, команды $\text{\code\{}$ и $\text{\code\}$ 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. $\text{\TeX}$ 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 : $\text{\TeX}$ 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 $\text{\TeX}$ ’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 $\text{\TeX}$ ’е командами $\text{\code\imath}$ и $\text{\code\jmath}$.

$\text{\TeX}$ 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. $\text{\TeX}$ также использует для математики три различных размера. Они называются: текстовый размер, размер индексов и размер повторных индексов. Обычный способ набрать формулу с помощью $\text{\TeX}$ ’а — заключить ее в знаки доллара $\$ \dots \$$, это дает формулу в текстовом стиле. Или можно заключить ее в двойные знаки доллара $\$ \$ \dots \$ \$$, 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 макрокоманды, 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

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

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 “4A” 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. Т_ЕX, когда входит в математическую моду, автоматически устанавливает `\fam=-1`, поэтому классы 7 и 0 эквивалентны, если только `\fam` не присвоено новое значение. Plain Т_ЕX 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.

Т_ЕX 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 Т_ЕX встречает во входном файле макрокоманду, он ищет ее определение и заменяет макрокоманду ее *макроподстановкой*.

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

Когда макрокоманде предшествует \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 \outer все же появляется в таких местах, Т_ЕX прекращает свои действия и сообщает либо о ситуации “убегающего аргумента”, либо о “незавершенном” условии. Конец входного файла так же рассматривается в этом смысле как \outer. Например, файл не должен оканчиваться в середине макроопределения. Если Вы разрабатываете формат для того, чтобы его использовали другие пользователи, используйте \outer со всеми командами, которые должны появляться только “тайно” внутри документа.

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

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

```
\if<условие><текст1>\else<текст2>\fi
```

где текст₁ пропускается, если условие не выполняется, а ложный текст₂ пропускается, если условие выполняется. Если ложный текст пустой, можно опустить \else. Часть “\if<условие>” в этой конструкции начинается с команды, первые две буквы которой “if”; например,

```
\ifodd\count0 \rightpage \else\leftpage \fi
```

задает условие, которое истинно, когда целый регистр Т_ЕX'а \count0 нечетный.

Заметим, что все команды для сравнений начинаются с \if... и имеют парную \fi. Вложение \if...\fi независимо от вложения {...}; 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 Т_ЕX имеет команду \edef (расширенное определение) и \xdef (которая эквивалентна \global\edef). Общий формат их такой же, как у \def и \gdef, но Т_ЕX до конца раскрывает элементы текста замены.

Т_ЕX имеет возможность читать текст сразу из вплоть до 16 файлов вдобавок к тем файлам, которые появляются в \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 Т_ЕX'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 Т_ЕX'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 Т_ЕX решит выводить страницу. В таком случае может быть непредсказуемый эффект, поскольку Т_ЕX часто находится в середине страницы 101, когда выводит страницу 100.

Начинающим пользователям обычно не приходится делать слишком большие изменения программы вывода plain Т_ЕX'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.

3. Каталог TeX'a

$\backslash_{\square}^*$	(бэкслэш с последующим пробелом). Для вставки пробела. Используется после команд, поскольку обычные пробелы после них TeX удаляет. $\backslash\mathrm{TeX}$ удаляет пробелы TeX удаляет пробелы $\backslash\mathrm{TeX}\backslash$ удаляет пробелы TeX удаляет пробелы Эта команда также используется в математической моде, когда надо явно задать пробел: $\\$x>0,\backslash y\leq 1,\backslash z\geq 2\\$ $x > 0, y \leq 1, z \geq 2$ Только что приведенная строка умышленно записана в такой неуклюжей форме. Элегантнее было бы написать <math>\\$x>0\$, <math>\\$y\leq 1\$, $\\$z\geq 2\\$ (начинать и заканчивать математическую моду каждый раз, когда это необходимо, и тогда достаточно обычного пробела).</math></math> 10, 13, 25, 92, 106-107, 186, 197, 202, 335, <u>339</u>, <u>347</u>, 384, 416, 448.
#	Специальный символ, зарезервированный для записи аргументов макроккоманд. 48, 65, 138, 238-240, <u>242</u>, 243-245, 271, <u>235</u>, 236-240.
$\backslash\#$	Для записи диеза (#) в выходном документе. 48, 65, <u>421</u>.
\$	Специальный символ, зарезервированный для математической моды. Он указывает на начало и конец математической моды. Двойной доллар \$\$ указывает на начало и конец выделенной математической моды. 4, 48, 65, 68, 107-110, 114, 155, 163-164, 221-222, <u>319</u>, 335, <u>341</u>, <u>349</u>.
$\backslash\$$	Для записи доллара (\$) в выходном документе. 48, 65, 240, 369, <u>421</u>.
%	Специальный символ для записи комментариев. Когда TeX встречает этот символ, он прекращает чтение строки. После него можно вставлять комментарии к программе и замаскировать символ CR в конце строки (чтобы избежать лишнего пробела, например, в какой-либо макроккоманде). 35, 48, 49, 55, 61, 65, 137-138, 150, 295, 400, 404, <u>407</u>.
$\backslash\%$	Для записи в выходном документе знака процента (%). 48, 55-56, 65, <u>421</u>.
&	Специальный символ, зарезервированный для задания различного рода выравниваний (табуляция, таблицы). 48, 65, 211-212, 227-233, 275-295, <u>334</u>, 452-456.

<code>\&</code>	Для записи в выходном документе знака амперсанда (&). 48, 65, 68, <u>421</u> .
<code>\'</code>	Ставит знак ударения над последующим символом (каким бы он ни был): <code>\'e, \'E, \'o, \'=</code> é, É, ó, ≠ 9-11, 66-67, 365, <u>421</u> , 420.
<code>\‘</code>	Ставит знак обратного ударения над последующим символом (каким бы он ни был): <code>\‘e, \‘E, \‘u</code> è, È, ù 10, 66-68, 365, <u>422</u> .
<code>\"</code>	Ставит знак умляута над последующим символом (каким бы он ни был): <code>\"e, \"a, \"o, \"O</code> ë, ä, ö, Ö С буквами типа i в формате plain следует использовать команду <code>\"i</code> (поскольку <code>\"i</code> дает <code>ï</code>). 9, 12, 32, 33, 66-67, 68, <u>421</u> .
<code>{</code>	Специальный символ, зарезервированный в формате plain для обозначения начала группы. 18-19, 25-28, 48, 66, 238-240, <u>241-243</u> , 244-246, 256, <u>319</u> , 326-327, <u>335</u> , <u>340</u> , <u>347</u> , 393.
<code>\{</code>	Для получения в выходном документе открывающей фигурной скобки (<code>{</code>). Употребляется только в математической моде: <code>\$...\{...\\$</code> . Величина фигурной скобки может меняться с помощью <code>\big</code> и ее вариантов, а также команд <code>\left...\right</code> . 163, 176-178, 210-211, <u>426</u> .
<code>}</code>	Специальный символ, зарезервированный в формате plain для обозначения конца группы. 18-19, 25-28, 48, 66, 238-240, <u>241-243</u> , 244-246, <u>319</u> , 326-327, <u>330</u> , 360, 393.
<code>\}</code>	Для получения в выходном документе закрывающей фигурной скобки (<code>}</code>). Пояснения те же, что и для открывающей скобки. 163, 176-178, 210-211, <u>420</u> .
<code>+</code>	Бинарная операция “плюс”. Только в математической моде. 65, 161, 318.
<code>\+</code>	Начало табулируемой строки, которая обязательно должна заканчиваться командой <code>\cr</code> . Эта строка может включать в себя знак амперсанда, указывающего на позиции табуляции. Например:

<code>\+Фамилия\kern 10mm&Имя \cr</code>	Фамилия	Имя
<code>\+Лисина &Марина\cr</code>	Лисина	Марина

На строке, которая начинается командой `\+`, можно *устанавливать*, *использовать* или *подавлять* (`\cleartabs`) черточки табуляции. Внимание! Макрокоманда `\+` не может быть параметром другой команды (на самом деле она определяется как `\outer\def\+{\tabalign}`). Например, команда `\smash{...\+...\cr..}` вызовет сообщение `\runaway argument ?`. Если такое случилось, надо в начале программы напечатать `\let\+=\tabalign`.

275-279, 295, 403, 419.

– Бинарная операция “минус”. Только в математической моде.

5, 65, 115, 117, 155, 161, 318.

`\-*` Обозначение возможного места расщепления слова для переноса с одной строки на другую: `Ро\до\ден\дрон`.

Едва ли вам придется часто обращаться к этой команде, поскольку `TeX` сам знает, как следует правильно переносить английские слова (а также и русские, если установить `\language=1`). Лучше использовать команду `\hyphenation`.

117, 335, 341, 348, 530.

`\*` Обозначение возможного места расщепления формул для переноса их с одной строки на другую по знаку умножения: `$(x+y)\*(z+t)$`. Если формула поместится на строке, получится обычное $(x+y)(z+t)$. Если же позиция знака умножения оказывается подходящей для переноса строки, получится $(x+y) \times$ в конце первой строки, и $(z+t)$ в начале строки последующей. См. описание команды `\discretionary` для выяснения определения этой команды. Для переноса других формул используйте команды `\allowbreak` или `\break`: `$X_1+x_2\cdots+x_n \allowbreak +y_1+y_2+\cdots+y_m$`. Если же какая-либо формула разорвана неудачно, вставьте команду `\nobreak` в том месте, где надо избежать разрыва или заключите формулу в бокс: `\hbox{$\dots$}`. Не забудьте перейти в математическую моду внутри бокса.

208, 423.

`\/*` Курсивная поправка (вообще говоря, любая поправка, независимо от сочетания шрифтов). Речь идет о крошечном дополнительном пробеле, размеры которого зависят от сочетания букв:

```
{\it Good} buy ..... Good buy
{\it Good} buy\/* ..... Good buy
{\sl Good} buy ..... Good buy
{\sl Good\/*} buy ..... Good buy
```

18, 80, 341, 348, 366, 450, 455.

$\backslash $	(См. $\backslash\text{Vert}$) Печатает $\ $. Только в математической моде. Эта двойная вертикальная черта может менять свою величину под действием команды $\backslash\text{big}$ и ее вариантов, а также конструкции $\backslash\text{left}...\backslash\text{right}$. 176-178, <u>426</u> , 438, 507, 511.
$\backslash$	Бэкслэш. Специальный символ, обозначающий начало команды. 9, 48, 49, 50, 66, 176-178, <u>408</u> , 508.
$=$	Знак “равно”. В математической моде используется в качестве отношения. 65, 162, 248, 268, 326, 442.
$\backslash=$	Проводит черту над следующим за ней символом (в <i>текстовой моде</i>): $\backslash=\text{occam}$, $\backslash=\text{assis}$ ̄occam , ̄assis Чтобы провести черту над символом в <i>математической моде</i> , используйте команду $\backslash\text{bar}$. Например, сопряженное $\bar{z}$ комплексного числа z записывается как $\backslash\text{bar } z$. Часто можно получить лучшие результаты, используя макрокоманду $\backslash\text{overline}$: $\backslash\text{overline}\{z\}+\backslash\text{overline}\{k\}$ $\bar{z} + \bar{k}$ $\backslash\text{overline}\{\text{strut } z\}+\backslash\text{overline}\{\text{strut } k\}$ $z + k$ Обратите внимание на действие команды $\backslash\text{strut}$ (подпорка): благодаря ей черточки располагаются на одной высоте. Макрокоманда $\backslash\text{overline}$ обязательна, если надо “надчеркнуть” сразу несколько символов. $\backslash\text{overline}\{\text{mathstrut}(u+v)\}=\backslash\text{overline}\{\text{mathstrut } u\}+\backslash\text{overline}\{\text{mathstrut } v\}$ $\overline{(u+v)} = \bar{u} + \bar{v}$ 66, 67, <u>422</u> .
$>$	Отношение “больше”. Только в математической моде. 66, 67, 162, 181, 248.
$\backslash>$	Пробел средней величины (2/9 квадрата — см. $\backslash\text{quad}$), используется в математической моде. Посмотрите, как он выглядит между двумя вертикальными чертами: $ $. Встречается редко. 202, 205, <u>423</u> .
$\backslash,$	Мини-пробел (1/6 квадрата — см. $\backslash\text{quad}$), используется в математической моде. Используется чаще всего. $\backslash\int_a^b f(x)dx$, $\backslash\int_a^b f(x)\backslash, dx$ $\int_a^b f(x)dx$, $\int_a^b f(x) dx$ 6, 202-208, 365, <u>423</u> , 479-480.
$\backslash.$	Ставит точку над следующим за командой символом (в <i>текстовой моде</i>): $\backslash.x$ дает $\dot{x}$. В математической моде $\dot{x}$ получается командой $\backslash\text{dot } x$. 66, <u>422</u> .

- `\;` Широкий пробел (5/18 квадрата — см. `\quad`), используется в математической моде. Посмотрите, как он выглядит между двумя вертикальными чертами: `| |`. На практике используется редко.
202, 206, [423](#).
- `\!` *Отрицательный* мини-пробел (−1/6 квадрата — см. `\quad`), используется в математической моде. Используется очень часто! Этот мини-пробел нужен, чтобы “сдвигать” символы:

$$\int\limits_{cal D} \frac{e^{x+y}}{x+y+1} dx dy$$
202, 204, [423](#).
- `_` Специальный символ, зарезервированный для получения нижнего индекса в математической моде: `$x_i$` дает x_i .
48, 66, 156-158, 163.
- `\_` Для получения символа, который программисты называют *подчеркиванием* и который соединяет слова в пределах одного идентификатора.
`Plus\_Grand\_Commun\_Diviseur` `Plus_Grand_Commune_Diviseur`
48, 199, [422](#).
- `^` Специальный символ, зарезервированный для получения верхнего индекса в математической моде: `$x^i$` дает x^i .
48, 66, 156-158, 163, 435, 494.
- `\^` Ставит “шляпку” над следующим за командой символом (в текстовой моде):
`\^e,\^E,\^a,\^i,\^o,\^0` `ê,Ê,â,î,ô,Ô`
Обычно команда `\^i` дает $\hat{i}$. Чтобы получить $\hat{i}$, надо удалить точку над i : `\^i`.
66, [422](#).
- `~` Специальный символ, порождающий пробел, на котором запрещено разрывать строку. Например, в записи “Д.~Кнут, стр.~38--45” употребление `~` вместо пробела обязательно. Если же вы по недосмотру вставите и обычный пробел, например, Д. ~Кнут или Д. ~ Кнут, то имя может разбиться на две строки.
48, 66, 408, [418](#).
- `\~` Для получения символа `~` в выходном документе.
66, [422](#).

a

`\aa` Буква ‘a’ из скандинавских алфавитов (с маленьким кружочком над буквой): $\text{\aa}$.

421.

`\AA` Буква ‘A’ из скандинавских алфавитов (с маленьким кружочком над буквой): $\text{\AA}$. Для получения $x = 3 \text{\AA}$ введите `$x=3$ \AA`. Если после $\text{\AA}$ нужен пробел, не забудьте написать `\AA\`.

422.

`\above*` Дополнительно задает толщину дробной черты. Синтаксис аналогичен команде `\over`.

`$x+\{y+z\above 1pt v+w\}$` $x + \frac{y+z}{v+w}$
`$\displaystyle x+\{y+z\above 1pt v+w\}$` $x + \frac{y+z}{v+w}$

174, 183, 349, 517-518.

`\abovedisplayshortskip*` Пробел перед формулой в выделенной математической моде, когда предыдущая строка коротка. См. `\abovedisplayskip`.

225, 325, *413*, 486.

`\abovedisplayskip*` Пробел перед формулой в выделенной математической моде, когда предыдущая строка длинна. Пробелы после формулы регулируются командами `\belowdisplayskip` и `\belowdisplayshortskip`. В формате `plain` предлагаются следующие значения этих параметров:

`\abovedisplayskip=12pt plus 3pt minus 9pt`
`\abovedisplayshortskip=0pt plus 3pt`
`\belowdisplayskip=\abovedisplayskip`
`\belowdisplayshortskip=7pt plus 3pt minus 3pt`

Не стоит менять эти размеры для экономии места вокруг формулы. Для этого вполне достаточно задать `\vskip -1mm` или `\vskip -1mm plus .2mm minus .2mm`. Такие отклонения в обе стороны обеспечивают определенную эластичность вертикальных пробелов, что облегчает верстку текста.

225, 226, 231, 325, 347, *413*, 486.

`\abovewithdelims*` Прimitives для построения дроби с ограничителями (круглые, квадратные и другие скобки). Следует уточнять, какие именно используются ограничители — от этого зависит толщина горизонтальной черты:

`\def\toto{\abovewithdelims[.1pt]}`

Обратите внимание на синтаксис: первым ограничителем является открывающая квадратная скобка. Точка после этой скобки (в команде)

означает, что заказан *пустой закрывающий ограничитель*. Не подумайте, что речь идет о горизонтальной черте толщиной в 0.1 pt! Толщина горизонтальной черты будет 1 pt. Синтаксис так определенной команды `\toto` такой же, как команды `\over`:

`$x+{a\toto b+c}$` $x + \left[\frac{a}{b+c} \right]$
`$\displaystyle x+{a\toto b+c}$` $x + \left[\frac{a}{b+c} \right]$
183, [349](#), 517-518.

`\accent*` Примитив `\TeX'a`, с помощью которого определено большинство акцентов. Например, `\'E` эквивалентно `\accent 19 E`. Число после команды указывает номер акцентирующего символа в текущем шрифте.
12, 68, 106, 335, [341](#).

`\active` Для превращения символа в активный символ, т.е., в макрокоманду. Например, чтобы сделать символ `W` командой, надо ввести

```
catcode'\W=\active
defW{...}
```

Внимание, в `\defW` перед `W` не нужен `\`. Теперь сам символ `W` является макрокомандой. Чтобы вызвать эту макрокоманду, пишут просто `W` (без обратной косой черты!). Чтобы переопределить `W` в обычный символ, введите `\catcode'\W=11` (11 — это категория буквы). Следует использовать макрокоманду `\string`, чтобы `W` печатал символ `W`. Если задать `\defW{\string W`, то каждый `W` во входном файле будет давать `W` в конечном документе.
241, [407](#), 364, 492.

`\acute` Ставит знак ударения над буквой, следующей сразу за командой (только в математической моде):
`$\acute a$, $\acute x$, $\acute y$` $\acute{a}, \acute{x}, \acute{y}$
164, [424](#).

`\adjdemerits*` Величина дополнительного дефекта в случае визуальной несовместимости соседних строк. `\TeX` использует этот примитив для формирования абзацев. В формате `plain` `\adjdemerits=10000`.
[121](#), 323, 376, 413.

`\advance` Добавляет к некоторой величине другую, сходной природы. Используется в языке программирования `\TeX`.

```
\advance\count13 by 1 \advance\count17 by\count13
\advance\dimen8 by 3pt \advance\skip8 by lfil
```

27, 143, 259, 305, [327](#), 420.

`\advancepageno` Увеличивает абсолютное значение числа `\pageno`, используемого для порядкового номера страницы. Определение данной команды в формате `plain` вполне доступно пониманию:

```

\def\advancepageno{\ifnum\pageno<0
\global\advance\pageno by -1
\else\global\advance\pageno by 1\if}

```

Комментарий: `\pageno` является переменной целого типа, которая отрицательна, если страницы нумеруются латинскими цифрами. Префикс `\global` используется здесь, чтобы увеличение `\pageno` выходило за пределы группы, в которой оно происходит. Чтобы напечатать номер страницы, используйте команду `\folio`.

305, 306, 428, 487.

`\ae` Для получения буквы æ (для œ есть `\oe`).
`ros\ae, \oe d\‘eme .....rosæ, œedème`
Теперь вы понимаете, почему пробел, следующий непосредственно за макрокомандой, на самом деле пробелом не является?

23, 58, 67, 421.

`\AE` Для получения буквы Æ. Последовательность `\AE SOPE` дает ÆSOPE. Обратите внимание, что пробел после `\AE` игнорируется.

67, 421.

`\afterassignment*` Знак, следующий за этой командой, будет вставлен сразу после первого присваивания.

255, 331, 417, 430, 442, 470.

`\aftergroup*` Знак, следующий за этой командой, будет вставлен сразу после окончания текущей группы.

255, 331, 439, 440, 444, 445.

`\aleph` Для получения буквы ℵ. Только в математической моде.

`$2^{\aleph_0}=\aleph_1$ ..... 2ℵ0 = ℵ1`
12, 423, 507.

`\allocationnumber` Специальный счетчик, содержащий последнее число, присваиваемое командами `\newcount`, `\newdimen`, ... , `\newinsert`. Используется при написании пакетов макрокоманд.

411.

`\allowbreak` Указывает на место возможного разбиения текста для переноса с одной строки на другую (`\penalty 0`). У TeX'a есть несколько способов выполнения этой операции, например, можно попробовать `\goodbreak` и другие варианты.

209, 419, 465.

`\allowhyphens` Разрешает перенос следующего слова. Используется при написании пакетов макрокоманд.

463, 463.

<code>\alpha</code>	Как и указывает название команды, это буква α . Только в математической моде. 156, 239, <u>423</u>, 506.
<code>\amalg</code>	Бинарная операция $\amalg$. Только в математической моде: $x \amalg y \dots\dots\dots x \amalg y$ Не путайте с большим оператором <code>\coprod</code> : $x \amalg y \coprod z \displaystyle \coprod \dots\dots\dots \amalg \amalg \amalg$ 424, 508.
<code>\angle</code>	Знак $\angle$. Только в математической моде: угол $\angle AMB$ $\dots\dots\dots$ угол $\angle AMB$ Иногда угол обозначают по-другому: угол $\widehat{ABM}$ $\dots\dots\dots$ угол $\widehat{ABM}$ 423, 507.
<code>\approx</code>	Отношение $\approx$. Только в математической моде. А может быть, Вам потребуется символ <code>\simeq</code> ($\simeq$) или <code>\cong</code> ($\cong$)? 156, 508.
<code>\arccos</code>	Функция арккосинус. Только в математической моде. 196, <u>427</u>.
<code>\arcsin</code>	Функция арксинус. Только в математической моде. 196, 427.
<code>\arctan</code>	Функция арктангенс: <code>arctan</code> . Только в математической моде. Можно определить и такую макрокоманду <code>\arctg</code> : $\def\arctg{\mathop{\rm arctg}\nolimits}$ 196, 427.
<code>\arg</code>	Оператор <code>arg</code> для аргументов комплексных чисел. Только в математической моде: $\arg(zz') = \arg z + \arg z' \dots\dots\dots \arg(zz') = \arg z + \arg z'$ 196, 427.
<code>\arrowvert</code>	Ограничитель $\mid$. Только в математической моде. 181, <u>425</u>.
<code>\Arrowvert</code>	Ограничитель $\parallel$. Только в математической моде. 181, <u>425</u>.
<code>\ast</code>	Бинарная операция $*$. Только в математической моде. 508.

<code>\asymp</code>	Отношение $\asymp$. Только в математической моде.	508.
<code>at</code>	Ключевое слово, используется для увеличения размера шрифта. 21, 75, 253, 329, 478, 484, 505.	
<code>\atop*</code>	Строит дробь без горизонтальной черты. Синтаксис такой же, как и <code>\over</code>. $x + \frac{u}{v+w}$ $\displaystyle x + \frac{u}{v+w}$ $\displaystyle S = \sum_{\substack{1 \leq i \leq n \\ J, K \subset X}} A_i B_{JK}$ В последнем примере обратите внимание на <code>\scriptstyle</code> в числителе и знаменателе. Это прием, который не дает индексам стать слишком маленькими. Иначе они окажутся в стиле <code>\scriptscriptstyle</code> (5 пунктов вместо 7), поскольку <code>\atop</code> сама уже в индексе. 173, 175, 183, 213, 349, 517.	
<code>\atopwithdelims*</code>	Вариант <code>\atop</code>, строящий дробь без горизонтальной черты и окружающий ее ограничителями. Синтаксис такой же, как и у макрокоманды <code>\over</code>: $q + \left\langle \frac{x+u}{x+vw} \right\rangle$ $\displaystyle q + \left\langle \frac{x+u}{x+vw} \right\rangle$ 183, 349, 386, 360, 517.	
b		
<code>\b</code>	Проводит черту под буквой, которая следует за этой командой (в текстовой моде): $\underline{x}, \underline{o}, \underline{k}$ Сравните с командой <code>\underline</code>, которая выполняет подчеркивание в математической моде. Последняя, ко всему прочему, может подчеркнуть более одного символа: $\underline{\text{PAO}}, \underline{PAO}$ 62, 66, 67, 478.	
<code>\backslash</code>	Бэкслэш без пробелов слева и справа. Только в математической моде: $H \backslash G / K$ Не путать эту команду с командой <code>\setminus</code> (вычитание множеств), которая задает бинарную операцию:	

- множество $H \setminus G$ множество $H \setminus G$
класс $H \backslash G$ класс $H \backslash G$
Чтобы напечатать ‘\’ с текстом шрифта `\tt`, введите `\char‘\’`.
48, 177-178, 425, 507.
- `\badness*` Параметр, равный плохости распределения клея в создаваемом боксе.
254, 272, 322.
- `\bar` Математический акцент надчеркивания ($\bar{x}$). Только в математической моде. Проводит черту над буквой, следующей непосредственно за командой: например, мнимая часть комплексного числа z задается в виде `\bar z`. В математической моде также можно использовать команду `\overline`:
- `\bar z + \bar k` $\bar{z} + \bar{k}$
`\overline{z} + \overline{k}` $\bar{z} + \bar{k}$
`\overline{\mathstrut z} + \overline{\mathstrut k}` $\bar{z} + \bar{k}$
- Обратите внимание на действие команды `\mathstrut` (невидимая подпорка величиной с круглую скобку, которая позволяет размещать черточки на одной высоте). Команда `\overline` может подчеркнуть более одного символа.
164, 165.
- `\baselineskip*` Расстояние между базовыми линиями в тексте (в формате `plain` `\baselineskip=12pt`). При использовании команды внутри группы, не забудьте перед закрытием группы написать `\par`:
- `{\baselineskip=14pt...\par}`,
- иначе интервал не изменится. Некоторые команды (например, `\matrix`) вызывают команду `\normalbaselines`, которая заново задает значение `\baselineskip`. Подробности в `\normalbaselines`.
87, 97, 128, 231, 301, 302, 325, 333, 406, 414, 416-417, 479, 485-486.
- `\batchmode*` Один из примитивов, которые управляют уровнем взаимодействия во время работы $\TeX$ ’а. Говорит $\TeX$ ’у продолжать работу, несмотря на встреченные ошибки, исправляя их по своему разумению. Вывод на терминал подавляется. Сообщения об ошибках записываются в протокольный файл. Другие уровни взаимодействия задаются примитивами `scrollmode` и `nonstopmode`.
42, 329, 358, 399.
- `\begingroup*` Показывает на начало группы. Конец группы указывается командой `\endgroup`. Функционально две эти команды сходны с фигурными скобками. Имеются также `\bgroup` и `\egroup`, которые служат для определения группы. Несмотря на сходство имен, не следует вместе с `\begingroup` использовать `\egroup`, они не состыкуются. Различаются `{ ... }` и `\bgroup ... \egroup` с одной стороны и `\begingroup ...`

`\endgroup` — с другой. В первом случае $\TeX$ *запоминает полностью всю группу* перед тем как ее обработать. Во втором случае $\TeX$ рассматривает группу по мере ее чтения. Команды `\bgroup`, `\begingroup` и т.д. необходимы в том случае, если одно определение должно содержать в себе начало группы, а другое определение — ее конец. Нельзя употреблять фигурные скобки ни в одном из этих случаев! (С опытом Вы это поймете!) Рассмотрите еще и вариант с `\narrower`. Команда `\par` обязательна:

```
\def\begincitation{\begingroup\sl\leftskip=1cm\rightskip=1cm}
\def\endcitation{\par\endgroup}
```

27, 295, 312, 330, 380, 477, 490.

`\beginsection` Команда форматирования текста, синтаксис:

```
\beginsection...\par
```

(Можно заменить команду `\par` на пустую строку). Например, при

```
\beginsection Глава 2\par
\sl В предыдущей главе мы видели, что
интегралы  $J_\nu(x)$  сходятся при ...
```

Вы получите следующий результат:

Глава 2

В предыдущей главе мы видели, что интегралы $J_\nu(x)$ сходятся при $\nu > 1$. Рассмотрим теперь их поведение при x стремящимся к бесконечности.

404, 420.

`\belowdisplayshortskip*` Пробел, который $\TeX$ размещает после формулы в выделенной математической моде, если строка ниже этой формулы коротка.

225, 325, 423, 486.

`\belowdisplayskip*` Пробел, который $\TeX$ размещает после формулы в выделенной математической моде, если строка ниже этой формулы не коротка. Соответствующие переменные для пробелов перед выделенными формулами, естественно, `\abovedisplayshortskip` и `\abovedisplayskip`

225, 227, 232, 325, 347, 413, 486.

`\beta` Греческая буква β . Только в математической моде.

156, 506.

`\bf` Команда включения жирного шрифта. Команда может быть *локальной*, т.е. ограниченной одной группой.

`это {\bf замечание} интересно` `это замечание` интересно

или *глобальной*, если писать `\bf` В этом случае жирным шрифтом будет выделяться весь последующий текст до тех пор, пока не встретится какой-либо другой переключатель шрифтов (например, `\rm`, `\it` или `\sl`).

17, 198-199, 390, 416, 479, 485.

`\bffam` Глобальное задание жирного шрифта при записи математических формул. Для получения жирного шрифта в математической моде использование этой команды обязательно.

416, 485.

`\bgroup` Второе имя для открывающей фигурной скобки `{`. Команда задается следующим образом: `\let\bgroup={`. Аналогично, `\egroup` — это второе название для закрывающей скобки `}`. Не путайте эти команды с `\begingroup` и `\endgroup`. Две последние команды выполняют несколько другие функции фигурных скобок.

319, 416, 429, 449, 452, 477, 492.

`\big` Слегка увеличивает растяжимый символ. Только в математической моде. Синтаксис: `\big` растяжимый символ (круглая скобка, квадратная скобка, фигурная скобка, вертикальная черта и т.д.) Может быть в трех различных вариантах, а именно: `\bigl`, `\bigr` и `\bigm`. Команда `\bigl` используется с открывающим ограничителем, а команда `\bigr` — с закрывающим ('l' для левой скобки и 'r' для правой):

`$f\bigl(x+(y+z)\bigr)$` $f(x + (y + z))$

Эти нюансы весьма существенны. Обращайте на них внимание, иначе пробелы расставятся неправильно. Вариант команды `\bigm` слегка увеличивает ограничитель и немного промежуток вокруг него:

`$(x\in A(n)|y\in B(m))$` $(x \in A(n)|y \in B(m))$
`$(\big(x\in A(n)\big|y\in B(m)\big)$` $(x \in A(n)|y \in B(m))$
`$(\bigl(x\in A(n)\bigm|y\in B(m)\bigr)$` $(x \in A(n) \mid y \in B(m))$

Итак, рекомендации предельно просты: `\big` — для увеличения растяжимого символа без модификации окружающих его интервалов, `\bigl` для открывающего ограничителя, `\bigr` для закрывающего и `\bigm`, когда нужно увеличить интервал с обеих сторон от символа.

Чтобы еще больше увеличить растяжимые символы, имеются команды `\Big`, `\bigg` и им подобные. Полный список см. в `\Bigg`.

↑ Если Вам нужно провести большую вертикальную стрелку, попробуйте `\Bigg\uparrow`. Но возможно, и этого будет недостаточно. Для этого крайнего случая познакомьтесь с примером получения стрелки высотой 54 мм:

→ `\left\uparrow\vbbox to 27mm{}\right.`

Результат налицо, т.е. слева от данного абзаца. Пожалуй, сюда следует добавить еще и горизонтальную стрелку, чтобы напомнить, как получается *вертикально отцентрированный бокс*. Эта макрокоманда получается прямо из определения команды `\big`.

178, 206, 381, 425, 485.

`\Big` Увеличивает ограничители на 50% по сравнению с “маленькой” командой `\big`. Естественно, у Вас по-прежнему есть право на всевозможные окончания: `\Bigl`, `\Bigr`, `\Bigm`, `\Big`.

178, 206, 425.

`\bigbreak` Если этой команде предшествует вертикальный пропуск по крайней мере 12 пунктов, она заменяет этот пропуск на `\bigskip`. Она дает к тому же хорошее место для смены страницы (`\penalty -200`). Имеются другие команды, которые вызывают пропуски: это `\smallbreak` (которая соединяет `\smallskip` и `\penalty -50`) и `\medbreak` (которая соединяет `\medskip` и `\penalty -100`). Макрокоманда `\goodbreak` дает хорошее место для разрыва строки или страницы (`\penalty -500`). Ее чаще используют для разрыва страниц (для строк достаточно `\allowbreak`). Макрокоманда `\filbreak` является, возможно, более интересной (но и более капризной), чем макрокоманды, которые предлагают разбивку страницы. Имеется, наконец, `\break` (только `\penalty -1000`), которая разрывает строку или страницу.

135, 141, 353, 429.

`\bigcap` Большой оператор пересечения $\cap$ или $\bigcap$. Только в математической моде:

$$\text{\$}\displaystyle\overline{A} = \bigcap_{U \supset A} U$$

Как Вы могли сами заметить, горизонтальная черта ставится не точно над буквой ‘A’. Чтобы получить правильную запись $\overline{A}$, следует написать `\overline{\!\! A\!}`.

178, 507.

`\bigcirc` Оператор $\bigcirc$. Только в математической моде.

508.

`\bigcup` Большой оператор $\bigcup$ или $\bigcup$. Только в математической моде.

<code>\Biggm</code>	Средний ограничитель <code>\Bigg</code> . См. <code>\Bigg</code> .	177, 425 .
<code>\biggr</code>	Правый ограничитель <code>\bigg</code> . См. <code>\bigg</code> .	177, 178, 425 .
<code>\Biggr</code>	Правый ограничитель <code>\Bigg</code> . См. <code>\Bigg</code> .	177, 178, 425 .
<code>\bigl</code>	Левый ограничитель <code>\big</code> . См. <code>\big</code> .	176-181 , 187, 206-210 , 425 , 510.
<code>\Bigl</code>	Левый ограничитель <code>\Big</code> . См. <code>\Big</code> .	177, 425 , 437.
<code>\bigm</code>	Средний ограничитель <code>\big</code> . См. <code>\big</code> .	178, 206, 210 , 425 .
<code>\Bigm</code>	Средний ограничитель <code>\Big</code> . См. <code>\Big</code> .	178, 425 .
<code>\bigodot</code>	Большой оператор $\odot$ или $\bigodot$. Только в математической моде.	507.
<code>\bigoplus</code>	Большой оператор $\oplus$ или $\bigoplus$. Только в математической моде: $\text{\texttt{\$}\displaystyle E=\bigoplus_{i\in I}E_i\text{\texttt{\$}}} \dots\dots\dots E = \bigoplus_{i\in I} E_i$	507.
<code>\bigotimes</code>	Большой оператор $\otimes$ или $\bigotimes$. Только в математической моде. $\text{\texttt{\$}\bigvee\limits^n E=\bigl(\textstyle\bigotimes^n E\bigr)/S_n(E)\text{\texttt{\$}}} \dots\dots\dots \bigvee^n E = (\bigotimes^n E)/S_n(E)$	507.
<code>\bigr</code>	Правый ограничитель <code>\big</code> . См. <code>\big</code> .	177-178 , 179-181, 206, 210 , 425 .
<code>\Bigr</code>	Правый ограничитель <code>\Big</code> . См. <code>\Big</code> .	177, 425 .
<code>\bigskip</code>	Вертикальный интервал размером в 12 пунктов (возможное отклонение 4 пункта в обе стороны). См. также <code>\bigbreak</code> .	89, 133, 135, 140-141, 417 , 420 , 477 , 482 .

`\bigskipamount` Значение `\bigskip`. По умолчанию `\bigskipamount=12pt plus 4pt minus 4pt`. При изменении этого значения меняется действие всех команд, которые обращаются к `\normalbaselines`.

149, [414](#), [416-418](#), [429](#).

`\bigsqcup` Большой оператор $\sqcup$ или $\bigsqcup$. Только в математической моде.

507.

`\bigtriangledown` Бинарная операция ∇ . Только в математической моде.

508.

`\bigtriangleup` Бинарная операция $\triangle$. Только в математической моде. Хотя имеются также команды `\triangleleft` ($\triangleleft$) и `\triangleright` ($\triangleright$), эта команда не является `\big`-версией двух последних треугольников.

508.

`\biguplus` Большой оператор $\uplus$ или $\biguplus$. Только в математической моде.

507.

`\bigvee` Большой оператор $\vee$ или $\bigvee$. Только в математической моде. Примеры см. в `\bigotimes`.

507.

`\bigwedge` Большой оператор $\wedge$ или $\bigwedge$. Только в математической моде.

507.

`\binoppenalty*` Штраф за разрыв формулы после бинарной операции. В формате `plain` это значение равно 700.

124, 209, 323, 384, [413](#), [508](#).

`\bmod` Бинарная операция модуль. Только в математической моде:

`$x:= x\bmod a$` $x := x \bmod a$

Еще одна макрокоманда, относящаяся к “модулю” — это команда `\pmod`, которая печатает пробел в один квадрат, а затем пишет модуль, заключенный в круглые скобки:

`$x\equiv y\pmod n$` $x \equiv y \pmod n$

[198](#), 384, [427](#).

`\bordermatrix` Для *окаймленной* матрицы. Только в математической моде.

`$$A=\bordermatrix{\cr`
`&p &q \cr`
`p &I_p &0 \cr`
`q &0 &J_q \cr}$$`

$$A = \begin{matrix} p & q \\ I_p & O \\ O & J_q \end{matrix}$$

Матрица $n \times n$ задается как матрица $(n+1) \times (n+1)$, в которой коэффициент $(1,1)$ отсутствует. Т_ЕX сам решит, как ему разместить круглые скобки.

213, 427.

\bot Значок $\perp$. Только в математической моде:

$$(E+F)^{\bot} = E^{\bot} \cap F^{\bot} \quad \dots\dots\dots (E+F)^{\perp} = E^{\perp} \cap F^{\perp}$$
Имеются также команды `\perp`, `\vdash` и `\dashv`, рисующие, соответственно, значки $\perp$, $\vdash$, и $\dashv$.

507.

\botmark* Метка, наиболее часто размещаемая на странице. См. `\mark`.
254, 307, 308-309, 310-311, 312-313, 332.

\bowtie Отношение $\bowtie$. Только в математической моде. Если Вам интересно, обратитесь к команде `\joinrel`.

424, 508.

\box* (С целым положительным числом от 0 до 255). Вынимает содержимое бокса, хранящегося в памяти Т_ЕX'а. Разумеется, предварительно надо этот бокс чем-то наполнить, что делается командой `\setbox`:

$$\setbox2=\hbox{СТРАТЕГ}\box2 \quad \dots\dots\dots \text{СТРАТЕГ}$$
Чтобы использовать бокс несколько раз подряд и не потерять его содержимое, задайте команду `\copy`, сопровождаемую целым положительным числом в пределах от 0 до 255, например, `\copy2`. Многие макрокоманды используют боксы `\box0` или `\box1`. Поэтому Вам настоятельно не рекомендуется их использовать. Для большей безопасности используйте боксы с номерами в пределах от 10 до 250. Не трогайте боксов с номерами 250 и выше: в них хранятся Ваши страницы! Если Вам нужен ряд индивидуальных боксов, обратитесь к команде `\newbox`. Команды `\boxdef`, которая бы присваивала “маленькое имя” боксу с памятью, не существует. Можно назвать бокс с помощью команды `\chardef`.

$$\chardef\toto=30$$

Тогда Вы можете с одинаковым успехом писать `\box30` или `\box\toto`.
145, 182, 264, 329, 410, 419, 454, 455.

\boxmaxdepth* Задаёт максимально возможную глубину бокса. В формате `plain` этот примитив равен максимально возможному размеру.

100, 138, 296, 304, 325, 413.

\brace Аналогично биномиальным коэффициентам, но скобки фигурные. Используется только в математической моде. Синтаксис сходен с `\over`:

$$p + \{n \brace k\} \quad \dots\dots\dots p + \left\{ \begin{matrix} n \\ k \end{matrix} \right\}$$

	$p + \binom{n}{k}$	426.
<code>\bracedl</code>	Элемент горизонтальной фигурной скобки. Используется для автоматического формирования таких скобок.	422.
<code>\bracelu</code>	Элемент горизонтальной фигурной скобки. Используется для автоматического формирования таких скобок.	422.
<code>\bracerd</code>	Элемент горизонтальной фигурной скобки. Используется для автоматического формирования таких скобок.	422.
<code>\braceru</code>	Элемент горизонтальной фигурной скобки. Используется для автоматического формирования таких скобок.	422.
<code>\bracevert</code>	Ограничитель . Только в математической моде.	181, 425.
<code>\brack</code>	Аналогично биномиальным коэффициентам, но с квадратными скобками. Только в математической моде. Синтаксис сходен с <code>\over</code> . $k + \left[\begin{smallmatrix} n \\ k \end{smallmatrix} \right]$ $\left[\begin{smallmatrix} n \\ k \end{smallmatrix} \right] + k$	426.
<code>\break</code>	В горизонтальной моде (то есть в строке) эта команда <i>обязательно</i> (<code>\penalty-10000</code>) оборвет строку в указанном месте. Последующий текст переходит на начало следующей строки, <i>но новый абзац не начинается</i> , отступ сделан не будет. Будьте внимательны: <code>xyz \break</code> порождает один дополнительный пробел после <code>xyz</code> . Поэтому правильной будет запись <code>xyz\break</code> . Если нет каких-либо особых целей, пишите <code>\hfill\break</code> , чтобы пробелы на обрываемой строке не растягивались.	116, 120, 130, 139, 200, 230, 418.
<code>\breve</code>	Рисует над последующим символом знак “качели” ($\breve$). Только в математической моде. $\breve{a}, \breve{x}, t\breve{}, \breve{}$	164.
<code>\brokenpenalty*</code>	Один из специальных видов штрафа, используемых Т _Е X’ом при формировании абзацев. В формате <code>plain</code> он равен 100.	128, 323, 413.

`\buildrel` Для размещения какого-либо математического символа над другим символом или их последовательностью. Только в математической моде:

```
$x \buildrel def \over{=} 1$ .....  $x \stackrel{def}{=} 1$ 
$x \buildrel \hbox{\sevenrm слабо} \over{\longrightarrow} 0$ .....  $x \stackrel{\text{слабо}}{\longrightarrow} 0$ 
$T \buildrel \hbox{\sevenrm равномерно} \over{\hbox to 2cm{\rightarrowfill}} 1$ .....  $T \stackrel{\text{равномерно}}{\longrightarrow} 1$ 
```

Обратите внимание на использование команды `\hbox`, с помощью которой допускается изменение шрифтов в математической моде. Иногда после `\over` надо использовать группирование: `\buildrel...\over{...}`. Это происходит, когда после `\over` идет бокс. Команда `\buildrel` сама использует верхний индекс, поэтому и `TeX` возражает, если встречается конструкция типа `^{\hbox}`; такую конструкцию он не принимает. Возьмите за правило всегда заключать `\buildrel` в скобки, как, например:

$$\{\buildrel...\over{...}\}$$

Это придаст Вам уверенности в том, что пробелы будут проставлены правильно — с одной стороны, а с другой — надежнее будет результат. Для того чтобы разместить что-либо *под* или *над* символом, используйте команду `\mathop`. Команда `\build` расширяет возможности команды `\buildrel`.

426, 509.

`\bullet` Бинарная операция $\bullet$. Только в математической моде. Чтобы получить ‘ $\circ$ ’, используйте `\circ`. (Не забывайте указывать математическую моду!)

161, 186, 420, 508.

`by` Ключевое слово `TeX`’а, используется вместе с “арифметическими” командами типа `\advance`, Например, `\advance\count1 by 8`.

144, 328.

`\bye` Наилучший способ выйти из `TeX`’а (лучше чем `\end`).

107, 404, 423.

C

`\c` Рисует под следующим за командой символом сидиль:

```
hame\c con, FRAN\c CON, \c S ..... hameçon, FRANÇON, §
```

32, 67, 422.

`\cal` Позволяет получать каллиграфические буквы. Работает только в математической моде и только с заглавными буквами:

```
$\cal A$, $\cal H$--$\cal X$ .....  $\mathcal{A}$ ,  $\mathcal{H}$  –  $\mathcal{X}$ 
```

Не забывайте ставить фигурные скобки, иначе результаты могут оказаться непредсказуемыми:

	$\mathcal{A} * \lfloor + \rfloor * \mathcal{T} + \S / \dagger - \ddagger$ $\mathcal{A} * \lfloor + \rfloor * \mathcal{T} + \S / \dagger - \ddagger$ 199, 416 , 503.
<code>\cap</code>	Бинарная операция $\cap$. Только в математической моде: $\mathcal{A} \cap \mathcal{B} \cap \mathcal{C} = \mathcal{A} \cap (\mathcal{B} \cap \mathcal{C})$ $A \cap B \cap C = A \cap (B \cap C)$ 161, 508.
<code>\cases</code>	Соединение условий с помощью фигурной скобки. В данном случае используется несколько мод: в первой колонке — математическая, чего не скажешь про вторую колонку: $\varphi(x) = \begin{cases} 0 & \text{для } x \leq 0 \\ e^{-1/x} & \text{иначе.} \end{cases} \quad \left \quad \varphi(x) = \begin{cases} 0 & \text{для } x \leq 0, \\ e^{-1/x} & \text{иначе.} \end{cases}$ Следовательно, в первой колонке знак $\$$ не требуется. Напротив, ни в коем случае не следует забывать его при задании математических объектов второй колонки (“для $x \leq 0$ ”). Чтобы слегка раздвинуть строки, введите <code>\noalign{\smallskip}</code> непосредственно после <code>\cr</code> . Будьте внимательны: <code>\openup</code> не оказывает такого действия на <code>\cases</code> . 211, 427 .
<code>\catcode*</code>	Изменяет <i>категорию</i> символа. Можно даже преобразовать символ в макрокоманду. См. <code>\active</code> . 50, 163, 186, 254, 322 , 365, 407-408 , 447-449 , 452 , 459 , 492 , 495 .
<code>cc</code>	Цицеро — одна из единиц измерения TeX’a (1 cc = 12 dd). 71, 321.
<code>\cdot</code>	Бинарная операция $\cdot$ (точка, окруженная пробелами). Только в математической моде. Служит, например, для обозначения скалярного произведения: $x \cdot y = x_1 y_1 + \dots + x_n y_n$ Чтобы поставить точку в качестве знака препинания после горизонтальной черты в дроби, используйте команду <code>\cdotp</code> , которая следует далее. 161, 208, 381, 508.
<code>\cdotp</code>	Только в математической моде. Точка, которая ставится <i>после</i> горизонтальной черты в дроби. Судите сами, на что приятнее посмотреть: $x = \frac{1}{2} + \frac{1}{3}$ $x = \frac{1}{2} + \frac{1}{3}$ К запятой (или другому знаку препинания) применяется <code>\raise 2pt</code> , что требует использования <code>\hbox</code> :

$x=\frac{1}{2}+\frac{1}{3}$,
 $x=\frac{1}{2}+\frac{1}{3}$
 424, 510.

`\cdots` Три точки, обозначающие прерывание текста. Только в математической моде. Пример можно посмотреть в `\cdot` (без буквы ‘s’ на конце). Вставляется между знаками +, −, ×, =, >, ≥, <, ≤ и ∩.

207, 180-181, 424, 510.

`\centerline` Размещает текст посередине строки:

`\centerline{\bf Глава 2}`

Очевидно, что `\centerline` должна быть *первой командой* в начале, например, главы (иными словами, `\centerline` должна появляться после пустой строки, задаваемой `\par` или `\vskip`). Если это условие не соблюдается, Т_ЕX обижается на переполненный `\hbox`. `\centerline` помещает свой аргумент в центре с помощью `\hss` для *отрицательного вытягивания*: текст может даже симметрично выступать за пределы строки на поля.

26, 32, 42, 90, 105, 124, 142, 276, 372, 404, 418.

`\char*` Для печати символа, соответствующего восьмеричному, десятичному или шестнадцатиричному коду ASCII. Число 15 означает ‘17 в восьмеричной системе и “F — в шестнадцатиричной. Все три команды `\char’017`, `\char 15` и `\char”F` печатают символ текущего шрифта, десятичный код ASCII которого — 15. Это дает лигатуру ‘ffl’ в прямом шрифте и ‘*ffl*’ — в курсивном. В шрифте `\tt` Вы получите ‘z’.

Т_ЕX прекрасно понимает, что таблица шрифтов не всегда у Вас перед глазами, поэтому предлагает такое средство: `\char’\A` печатает символ текущего шрифта, ASCII-код которого такой же, как у A (функция ‘\’ заключается в том, чтобы выдавать нужный код). Например, если текущий шрифт — `\tt`, `\char’\{` напечатает открывающую скобку ‘{’, а `\char’\` — обратную косую черту ‘\’ (в романском шрифте нет бэкслэша!). Очевидно, эта хитрость работает только с символами, которые есть на клавиатуре (лигатуры ‘ffl’ на клавиатуре нет).

55, 95, 106, 187, 335, 341, 346, 404, 499, 527.

`\chardef*` Приписывает имя символу, который уже известен в коде ASCII. Если после команды

`\chardef\esperluete=38`

ввести `\esperluete`, получится символ 38 *текущего шрифта*. Можно использовать команды `\chardef\esperluete=’46` (восьмеричный код) и `\chardef\esperluete=”26` (шестнадцатиричный код). Вообще говоря, команда `\chardef` позволяет присоединять имя к числу, находящемуся

в пределах от 0 до 255. Можно задавать `\count\esperluete` вместо `\count38`. Другие примеры в описании `\box`. Для присваивания имени математическому символу см. `\mathchardef`.

56, 147, 187, 249, 255, 298, 322, 328, 492, 527.

`\check` “Галочка”, без которой некоторые математики не представляют себе жизни. Естественно, математическая мода.

`$\check x*\check C+\check D\check{}$` $\check x * \check C + D$

Иногда “галочка” нужна справа от буквы. Как можно заметить, “галочка” в этом случае оказывается слишком близко к ‘D’. Приведем описание макрокоманды, которая прилично справляется с этой проблемой:

`\def\tcheche#1{#1\mkern2.5mu\check{}}`

Макрокоманда `\mkern` — это керн для математической моды. Используются единицы `mu` (математические единицы). Зачем эти сложности? Затем, что имеется четыре стиля: `\displaystyle`, `\textstyle`, `\scriptstyle` и `\scriptscriptstyle`. Значение `mu` зависит от этих стилей. Вот что получается в каждом из этих стилей (синтаксис `$\tcheche A+\tcheche B$`):

$A^{\sim} + B^{\sim} + C^{\sim} + D^{\sim} + M^{\sim} + X^{\sim} + Y^{\sim} + Z^{\sim}$

$A^{\sim} + B^{\sim} + C^{\sim} + D^{\sim} + M^{\sim} + X^{\sim} + Y^{\sim} + Z^{\sim}$

$A^{\sim} + B^{\sim} + C^{\sim} + D^{\sim} + M^{\sim} + X^{\sim} + Y^{\sim} + Z^{\sim}$

$A^{\sim} + B^{\sim} + C^{\sim} + D^{\sim} + M^{\sim} + X^{\sim} + Y^{\sim} + Z^{\sim}$

164.

`\chi` Греческая буква χ . Только в математической моде.

1, 506.

`\choose` Биномиальный коэффициент. Только в математической моде. Синтаксис аналогичен команде `\over`, так как определение этой команды использует “дробь с ограничителем” `\atopwithdelims`:

`$p+\{n\choose n+k\}$` $p + \binom{n}{n+k}$

`$\displaystyle p+\{n\choose n+k\}$` $p + \binom{n}{n+k}$

169, 173, 183, 213, 425.

`\circ` Бинарная операция $\circ$. Только в математической моде:

`$(f\circ g)'=f'\circ g\times g'$` $(f \circ g)' = f' \circ g \times g'$

Если не нравится, что вокруг кружочка остается слишком много места, предлагаем два способа исправления этого недостатка:

`$u{\circ}v$, $u\mathord{\circ} v$` $u \circ v, u \circ v$

Чтобы нарисовать этот кружочек на месте показателя степени, например, так: 1°), напишите `1$\^{\circ}`).

161, 385, 508.

`\cleaders*` Проводники, в которых повторяемый элемент располагается в центре отведенного ему места. См. `\leaders`.

266, 267-268, 422, 440.

`\cleartabs` Подавляет все позиции табуляции, которые были помещены справа от текущей колонки.

279, 419.

`\closein*` За командой следует число от 0 до 15. Закрывает входной файл с соответствующим номером, когда работа с ним уже закончилась. Файл предварительно был открыт командой `\openin`.

257, 331.

`\closeout*` То же, что и предыдущая команда, но для выходных файлов.

269, 302, 332, 493.

`\clubpenalty*` Один из специальных видов штрафа, который использует $\TeX$ для формирования абзаца. В формате `plain` он равен 150.

128, 138, 323, 379, 413, 490.

`\clubsuit` Знак $\clubsuit$. Только в математической моде.

507.

`cm` Одна из единиц измерения в $\TeX$ 'е, сантиметр ($2.54\text{ cm} = 1\text{ in}$).

32, 71, 321.

`\colon` Двоеточие ($:$) внутри математической формулы:

`$x\colon y : z$, $X\colon=Y:=Z$ .....  $x:y:z$ ,  $X:=Y:=Z$`

162, 425, 510.

`\columns` Указывает, на сколько колонок одинаковой ширины желательно разбить страницу. Неотделимо от `\settabs`:

```
\settabs 3\columns
\+$x_i^2$ &\it колонка 2 &колонка 3 \cr
\+      &\bf длинная строка текста\cr
\+      &                &колонка 3 \cr
```

Эта программка выдаст следующий результат:

x_i^2	колонка 2	колонка 3
	длинная строка текста	
		колонка 3

	Внимание на синтаксис: нет <code>\cr</code> , так как строка не начинается с <code>\+</code> . 275, 419.
<code>\cong</code>	Отношение $\cong$. Только в математической моде. Имеется также команда <code>\simeq</code> , которая выдает $\simeq$. 183, 426, 508.
<code>\coprod</code>	Большой оператор $\coprod$ или $\coprod$. Только в математической моде. Не путать с бинарной операцией <code>\amalg</code> : <code>\amalg</code> , <code>\coprod</code> , <code>\displaystyle\coprod</code> $\coprod$, $\coprod$, $\coprod$ 507.
<code>\copy*</code>	(Команда задается вместе целым положительным числом в пределах от 0 до 255). Выдает содержимое бокса, не разрушая его, тогда как команда <code>\box</code> опустошает бокс: <code>\setbox2=\hbox{этот бокс не пустой.}</code> <code>\copy2</code> этот бокс не пустой. <code>\box2</code> этот бокс не пустой. <code>\box2</code> 145, 182, <u>264</u> , 390, 391, 440, 454, 477.
<code>\copyright</code>	Знак защиты авторского права ©. Для “автоматического авторского права” см. команду <code>\romannumeral</code> . 368, 403, <u>422</u> .
<code>\cos</code>	Функция косинус. Только в математической моде. <code>\cos^2x+\sin^2x=1</code> $\cos^2 x + \sin^2 x = 1$ 196, 427.
<code>\cosh</code>	Функция гиперболический косинус. Только в математической моде: <code>\cosh^2t-\sinh^2t=1</code> $\cosh^2 t - \sinh^2 t = 1$ Если <code>\cosh</code> вам не по душе, используйте другое определение: <code>\def\ch{\mathop{\rm ch}\nolimits}}</code> 196, 427.
<code>\cot</code>	Функция котангенс. Только в математической моде. Если Вы предпочитаете обозначение <code>cotg</code> , посмотрите на две строки выше, где определяется макрокоманда <code>\ch</code> . <code>\cot(x)=\cos(x)/\sin(x)</code> $\cot(x) = \cos(x)/\sin(x)$ 196, 427.
<code>\coth</code>	Функция гиперболический котангенс. Только в математической моде: <code>\coth x=\cosh x/\sinh x</code> $\coth x = \cosh x / \sinh x$

Сравните это с записью котангенса, где используются круглые скобки. Какая из них кажется Вам более читабельной?

196, 427.

\count* (Команда задается вместе целым положительным числом в пределах от 0 до 255). Т_ЕX имеет 256 ячеек памяти, называемые счетчиками и предназначенные для хранения целых чисел. Счетчики от 0 до 9 зарезервированы для внутреннего использования, их трогать нельзя! Не следует также использовать `\count250` и последующие: они содержат информацию, касающуюся текущих страниц. Например:

```
\def\resetcountCXII{\count112=0}
\def\incrementeCXII{\global\advance\count112 by 1}
```

Пояснение: команда `\resetcountCXII`, как и следует из ее названия, устанавливает счетчик 112 в 0. После каждого `\incrementeCXII` значение счетчика `\count112` увеличивается на единицу (команда `\global` существенна). Это позволяет выполнять автоматическую нумерацию глав и т.д. Начинаящие пользователи Т_ЕX'а, видимо, уже чувствуют, что здесь речь идет об уровне хороших профессионалов.

143, 246, 322, 328, 411, 446.

\countdef* Присвоить имя счетчику. Манипулировать переменной, имя которой `\count112`, не слишком удобно. Чтобы переименовать ее, напишите:

```
\countdef\toto=112
```

Отныне `\count112` и `\toto` будут эквивалентами, и, например, предыдущий пример тогда можно переписать следующим образом:

```
\def\initttoto{\toto=0}
\def\incrementetototo{\global\advance\toto by 1}
```

145, 147, 249, 255, 322, 328, 410-411.

\cr* Указывает на конец строки при выравнивании. См. `\matrix`, `\eqalign`, `\halign`, `\displaylines` и т.д.

211-214, 227, 275, 245, 294, 326, 335, 416, 417, 453, 482, 489, 491.

\crrcr* Команда, способная генерировать завершающий `\cr`, если он пропущен в конце таблицы. Используется в программировании на Т_ЕX'е.

295, 326, 335, 427-428, 453, 482, 492.

\csc Функция косеканс (csc). Только в математической моде.

196, 427.

\csname* Конструкция `\csname ... \endcsname` преобразует знаки на месте многоточия, в название команды. Например, `\csname TeX \endcsname` — это то же самое, что `\TeX`.

50, 263, 413, 441.

\cup Бинарная операция $\cup$. Только в математической моде:

$(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$

Большой символ объединения задается командой `\bigcup`:

`\cup`, `\bigcup`, `\displaystyle\bigcup` $\cup, \bigcup, \bigcup$

Макрокоманда пересечения будет, естественно, выглядеть как `\bigcap`.
161, 508

d

`\d` Ставит точку под последующим символом (в текстовой моде):
`\kr\d sna, \KR\d SNA` $\text{kr}\grave{\text{s}}\text{na}, \text{KR}\grave{\text{S}}\text{NA}$
67, 422.

`\dag` Знак †. Только в текстовой моде.
142, 422, 511.

`\dagger` Бинарная операция †. Это “математическая” версия предыдущего знака `\dag` (пишется `$...\dagger...$`). Только в математической моде.
508, 511.

`\dashv` Отношение $\dashv$. Только в математической моде. Если нужно получить зеркальное отражение ($\vdash$), дается команда `\vdash`.
508.

`\day*` Выдает текущее число месяца (по крайней мере, если вы правильно ответили компьютеру, когда он Вас запрашивал о текущей календарной дате). Перед данной командой сначала нужно указывать `\the`, иначе Вы ничего не получите. Почему? Потому что `\day` — одна из *переменных* T_EX’a, а не макрокоманда в строгом смысле слова. Чтобы написать содержание переменной, Вы пишете *write* на Паскале и *print* на Бейсике. А в T_EX’e за это все отвечает команда `\the`:

`\the\day` 12
`\the\month` 6
`\the\year` 1991
334, 413, 476.

`dd` Одна из единиц измерения в T_EX’e (1157 dd = 1238 pt).
71, 321.

`\ddag` Два маленьких крестика ‡. Только в текстовой моде.
142, 422, 511.

`\ddagger` Бинарное отношение ‡. Только в математической моде.
508, 511.

<code>\ddot</code>	Ставит две точки над следующей за командой буквой. Только в математической моде. Необходима для тех, кто использует для производных ньютоновские обозначения: $\text{\textbackslash}\ddot{x}(t)=f(x,t) \dots\dots\dots \ddot{x}(t) = f(x,t)$ 164, 424.
<code>\ddots</code>	Три точки по диагонали ($\ddots$). Только в математической моде. 213, 424.
<code>\deadccles*</code>	Внутренняя целая переменная <code>TeX</code>'а, которая приравнивается нулю после каждого <code>\shipout</code> и увеличивается на 1 перед каждым <code>\output</code>. Используется, чтобы предотвратить заикливание при выводе результатов (если <code>\deadccles</code> больше <code>\maxdeadccles</code>, дается сообщение об ошибке). 254, 303, 314, 322, 335, 470.
<code>\def*</code>	Для задания макрокоманды. Это руководство насыщено примерами использования этой команды. 56, 165, 237-248, 255, 326.
<code>\defaultshphenchar*</code>	Параметр <code>TeX</code>'а — символ переноса <code>\shphenchar</code> после загрузки шрифта (см. <code>\shphenchar</code>). В формате <code>plain</code> <code>\shphenchar</code> — это дефис. 324, 413.
<code>\defaultskewchar*</code>	Значение <code>\skewchar</code> после загрузки шрифта (см. <code>\skewchar</code>). В формате <code>plain</code> <code>\skewchar=-1</code>. 324, 413.
<code>\deg</code>	Оператор, который пишет в формулах “deg” прямым шрифтом. Только в математической моде: $\text{\textbackslash}\deg(FG)=\deg F+\deg G \dots\dots\dots \deg(FG) = \deg F + \deg G$ Если Вам больше нравятся градусы, составьте другую макрокоманду: $\text{\textbackslash}\def\degres{\text{\textasciicircum}\text{\textbackslash}\circ}$ угол в 138\degres \dots\dots\dots угол в 138° 196, 427.
<code>\delcode*</code>	Используется для задания ограничителей. Например, команда $\text{\textbackslash}\delcode{x}="123456$ означает, что если буква <code>x</code> используется как ограничитель, три первые шестнадцатичные цифры укажут на “маленький” вариант этого ограничителя (1 — номер семейства, 23 — номер позиции), а следующие три цифры — на “большой” вариант (4 — номер семейства, 56 — номер позиции). 188, 254, 322, 436, 409.

<code>\delimiter*</code>	Примитив Т _Е X'а для явного задания ограничителей. Применяется Т _Е Xпертами при программировании на Т _Е X'е.	189, 345, 425.
<code>\delimiterfactor*</code>	Параметр для вычисления размеров ограничителей. В формате <code>plain</code> он равен 901. Применяется Т _Е Xпертами.	186, 324, 413, 519.
<code>\delimitershortfall*</code>	Параметр для вычисления размеров ограничителей. В формате <code>plain</code> он равен 5 pt. Применяется Т _Е Xпертами.	184, 325, 413, 519.
<code>\delta</code>	Греческая буква δ . Только в математической моде.	156, 506.
<code>\Delta</code>	Заглавная греческая буква Δ . Только в математической моде.	205, 222, 506.
<code>depth</code>	Ключевое слово для задания “глубины” линейки (см. <code>\hrule</code> и <code>\vrule</code>).	263, 300, 334, 400.
<code>\det</code>	Оператор <code>det</code> (для сокращенного обозначения детерминанта). Только в математической моде: Если <code>\det A \ne 0</code> , матрица A^{-1} существует. Если $\det A \neq 0$, матрица A^{-1} существует. Для печати самого детерминанта матрицы используйте команду <code>\left \matrix{...}\right </code> .	196, 427.
<code>\diamond</code>	Бинарная операция $\diamond$. Только в математической моде. закон <code>\$x\diamond y=xy/(x+y)\$</code> закон $x \diamond y = xy/(x + y)$	508.
<code>\diamondsuit</code>	Знак $\diamond$. Только в математической моде.	507.
<code>\dim</code>	Оператор <code>dim</code> (для размерности). Только в математической моде: <code>\$\dim(U+V)=\dim U+\dim V-\dim U\cap V\$</code> $\dim(U + V) = \dim U + \dim V - \dim U \cap V$	196, 427.
<code>\dimen*</code>	Регистр, содержащий размер. Команда должна задаваться вместе с целым числом в пределах от 0 до 255. Например, нужно наполовину уменьшить высоту бокса <code>\box3</code> . Но язык программирования Т _Е X слишком примитивен: невозможно написать что-либо вроде <code>\ht3=0.5\ht3</code> . Тогда следует предпринять эти вычисления в промежуточной памяти:	

`\dimen1=\ht3 \divide\dimen1 by 2 \ht3=\dimen1`

143, [322](#), [324-328](#), [411](#), [414](#), [426](#), [429](#), [463](#).

\dimendef* Для присваивания имени регистру размеров (`\toto` то же самое, только более удобное, чем `\dimen117`):

`\dimendef\toto=117`

Отныне Вы всегда будете писать `\toto` на месте `\dimen117`. Не путайте с командой `\newdimen`, которая позволяет *приватизировать* регистры размеров. Если регистру `\dimen117` присвоить посредством `\dimendef` имя `\toto`, это не помешает тому, чтобы в дальнейшем с другими командами, по Вашему желанию, употреблять прежний регистр. Поэтому лучше использовать `\newdimen`, если Вы интенсивно и с каким-либо специальным значением используете регистры размеров.

145, 255, [328](#), [411](#).

\discretionary* Предположим, что Вам хочется задать такой изощренный символ “точка-тире” (`.—`). Этот знак внутри строки должен появляться в нетронutom виде. Но если `TeX` решил разбить строку между точкой и тире, точка должна будет остаться в конце предыдущей строки, а тире должно исчезнуть (чтобы не появиться в начале следующей строки). Приведем пример команды, сообщающей `TeX`у о наших требованиях:

`\def\pointir{\discretionary{.}{.}{.---}}`

Третья группа `{.---}` содержит то, что `TeX` должен написать, когда нет разрыва. Если разбиение приходится на точку с тире, первая группа `{.}` содержит то, что будет печататься с правого края строки, а вторая группа `{}` — то, что будет печататься с левого края последующей строки (здесь, поскольку эта группа пуста, ничего печататься не будет). Третья группа должна содержать оба объекта (иначе — что разделять?). Если там находится более двух объектов, перегруппируйте их в два с помощью команды `\hbox`. В нашем примере ‘---’ — один объект, поскольку `TeX` его рассматривает как лигатуру. Испытаем наш символ “точка-тире” (повторяя последовательность `\pointir aa\pointir bb\pointir cc`):

aa.—bb.—cc .—aa.—bb.—cc .—aa.—bb.—cc .—aa.—bb.—cc .—aa.—bb.
cc .—aa.—bb.—cc .—aa.—bb.—cc .—aa.—bb.—cc

118, 335, [341](#), [348](#).

\displayindent* Отступ строки для записи формулы на отдельной строке. Если Вы окружаете какую-либо формулу двойными долларами `$$`, `TeX` конструирует бокс шириной `\displaywidth`, который обеспечивает отступ `\displayindent`. Значения по умолчанию этих двух переменных, естественно, `\hsize` и `0pt`. Но по желанию можно изменять эти значения. Продемонстрируем это с помощью следующей программки:

`$$\displaylines{\longleftarrow`

```
\hfill\cos^2x+\sin^2x=1\hfill\longrightharrow\cr
}$$$
```

$$\longleftarrow \cos^2 x + \sin^2 x = 1 \longrightarrow$$

(команда `\displaylines` обязательна, поскольку она создает `\line`). В данном пособии, мы имеем `\hsize=115mm`. Исключительно в демонстрационных целях сдвинем временно левый край на два сантиметра командой `\leftskip=20mm`.

Если мы снова запросим ту же самую формулу, она не будет отцентрирована относительно текста:

$$\longleftarrow \cos^2 x + \sin^2 x = 1 \longrightarrow$$

Уменьшим теперь значение `\displaywidth` на 2 см, введя команду `$$\displaywidth = 95mm \displaylines{...}$$`, в результате получим:

$$\longleftarrow \cos^2 x + \sin^2 x = 1 \longrightarrow$$

“Ширина” формулы такая же, как и текста, однако появилось нежелательное смещение. Сдвинем опять на 2 см вправо:

```
$$\displaywidth=95mm\displayindent=20mm
\displaylines{...}$$$
```

$$\longleftarrow \cos^2 x + \sin^2 x = 1 \longrightarrow$$

Эти модификации `\displaywidth` и `\displayindent` являются *локальными*. Чтобы не заставлять Вас повторять эту команду каждый раз, запишите ее в начало своего файла:

```
\everydisplay={\displaywidth=115mm
\displayindent=20mm}
```

TeX сам размечает текст после каждого двойного доллара, открывающего выделенную математическую моду.

224, 226, 325, 347, 413.

`\displaylimits*` Восстанавливает нормальные соглашения о пределах больших операторов ($\sum$, $\int$, ...), которые могли быть Вами изменены командами `\limits` и `\nolimits`. Только в математической моде.

175, 192, 348, 516.

`\displaylines` Для центрирования нескольких формул без дополнительных усилий по вертикальному выравниванию их между собой. Только в выделенной математической моде:

```
$$\displaylines{
```

```
x_1+\cdots+x_n=y_1+\cdots+y_m,\cr
A=B+C. \cr
}$$
```

Таким образом мы получим две отцентрированные строки:

$$x_1 + \cdots + x_n = y_1 + \cdots + y_m,$$

$$A = B + C.$$

Синтаксис такой же, как в таблицах без преамбулы с одной колонкой. Не забывайте, что каждая строка заканчивается `\cr`. Довольно частая ошибка, которая приводит в смущение многих начинающих пользователей — ставить знак пунктуации после `\cr`. $\TeX$ справедливо предполагает, что начинается новая строка. Он не встречает `\cr`, который закрывает эту строку, но, поскольку существует механизм, который исправляет эту ошибку (`\crrcr`), $\TeX$ не жалуется. Единственный осязаемый результат возникает в том случае, если знак пунктуации оказывается размещенным на другой строке, в центре, как если бы это была новая формула. Грубой ошибкой, встречающейся довольно часто, считается пропуск фигурной скобки, закрывающей запись макрокоманды (если Вы недостаточно дисциплинированы), т.е. примерно так:

```
$$\displaylines{
...
...\cr}$$
```

В конце программы написан `\cr`, но закрывающая фигурная скобка отсутствует. После множества протестов $\TeX$ 'а Вы не получите ... ровным счетом ничего! Если Вам захочется разместить два знака равенства один под другим, см. `\eqalign`.

231, 233, 428.

`\displaystyle*` Задаёт выделенный стиль. Естественно, только в математической моде: пробелы по обе стороны от формулы делаются более широкими, чем в обычной математической моде (`\textstyle`). Есть разница в размерах символов и в положении индексов. См. пример с `\displaystyle` (между двумя одиночными долларами мы находимся в режиме `\textstyle`, чтобы задать режим `\displaystyle`, нужно запрашивать его явно):

```
$\displaystyle\sum_{n=0}^{\infty}\frac{x_n}{1+x_n^2}$ ..... \sum_{n=0}^{n=\infty} \frac{x_n}{1+x_n^2}
```

Вот та же самая формула в `\textstyle`:

```
$\sum_{n=0}^{\infty}\frac{x_n}{1+x_n^2}$ ..... \sum_{n=0}^{n=\infty} \frac{x_n}{1+x_n^2}
```

Разумеется, можно в любом месте задавать `\displaystyle`:

```
дробь $\{a\over b\}$ ..... дробь \frac{a}{b}
```

дробь $\displaystyle{a\over b}$		дробь $\frac{a}{b}$
дробь $\{\displaystyle a\over b\}$		дробь $\frac{a}{b}$
дробь $\{a\over\displaystyle b\}$		дробь $\frac{a}{b}$
		172, <u>348</u> , 428.
$\displaywidowpenalty^*$	Один из специальных штрафов, который Т _Е X использует при формировании абзацев. В формате plain он равен 50.	128, 323, 413.
$\displaywidth^*$	Максимальная ширина формулы в выделенной математической моде. См. команду $\displayindent$.	224, 227, 325, 413.
$\div$	Бинарная операция $\div$. Только в математической моде: $\$15\div 4=3\$$ $15 \div 4 = 3$	508.
$\divide^*$	Делит одно число на другое. Часть языка программирования Т _Е X. Команда $\divide\dimen117\ by\ 12$ заменяет содержимое $\dimen117$ на результат его деления на 12, т.е. в современном языке программирования надо было бы писать $\dimen117:=\dimen117\ div\ 12$. Другой пример см. в $\dimen$.	143, 259, <u>327</u> , 460, 466, 468, 488.
$\do$	Макрокоманда, которая, в частности, используется в определении макрокоманды $\dospecial$. Что-то очень специальное для опытных пользователей.	408, 447, 493.
$\dospecials$	Представляет множество символов, которые имеют специальный код категории. Применяется при задании “дословной печати”, когда все специальные символы выводятся так, как они набраны.	<u>408</u> , 447, 494.
$\dosupereject$	Усиленный $\eject$. Выводит все задержанные вставки и иллюстрации.	305, <u>430</u> .
$\dot$	Для изображения точки над символом, следующим за командой. Только в математической моде. Незаменима при изображении ньютоновских производных: $\$ \dot x(t)=f\bigl(x(t),t\bigr)\$$ $\dot x(t) = f(x(t), t)$ Обратите внимание на использование $\big$. Для получения двух точек над каким-либо символом используется команда $\ddot$.	164.

- `\doteq` Отношение $\doteq$. Только в математической моде. 426, 508.
- `\dotfill` Команда для заполнения части строки точками. Эту макрокоманду следует использовать внутри какого-либо `\hbox`, определенного командами `\line`, `\halign` и т.д., или вместе с `\break`. Например, программка `\line{Дональд Кнут\dotfill XX век}` выдаст Вам следующее:
Дональд Кнут.....XX век
Не используйте эту макрокоманду непосредственно в вертикальной моде и для точек в таблицах (лучше использовать `\leaders`).
290, 396, 397, 404-405, 493.
- `\dots` Для получения трех маленьких точек в тексте..., поскольку просто три последовательных точки не дадут хорошего результата (...). Не забывайте писать правильно `\dots\` (обратная косая черта с последующим пробелом), если после трех точек продолжается фраза. В математической моде есть две команды: `\ldots` (точки на строке) и `\cdots` (точки подняты). 208, 422.
- `\doublehyphendemerits*` Специальный параметр, который $\TeX$ использует при автоматическом формировании абзацев — дефект, который добавляется, если две строки подряд оканчиваются переносом. В формате `plain` он равен 10000. 121, 323, 413, 526.
- `\downarrow` Знак $\downarrow$. Только в математической моде. Его величина может быть изменена командой `\big` и ее вариантами или уже известной конструкцией `\left...\right`. Для получения вертикальной стрелки высотой 12 мм, например, задайте:

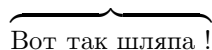
$$\left\downarrow\right.$$

И не забывайте, что стрелка, даже увеличенная, является бусиной, проткнутой посередине (следовательно, применяется `\vcenter`). Пример см. в `\big`. 177, 218, 425, 509.
- `\Downarrow` Знак $\Downarrow$. Только в математической моде. Замечания в предыдущей команде. 177, 425, 509.
- `\downbracefill` Для получения горизонтальной фигурной скобки, открывающейся вниз (в текстовой моде). Программа

```
\setbox2=\hbox{Вот так шляпа !}
\setbox3=\hbox to\wd2{\downbracefill}
```


`\vbox{\box3\box2}`

выдаст Вам


Вот так шляпа !

Запомните последовательность записи: сначала определяется `\box2`, поскольку нужно знать его ширину для того, чтобы нарисовать соответствующую скобку. В математической моде используйте `\overbrace` и `\underbrace`.

268, 394, [423](#).

`\dp*` (Команда задается вместе с целым числом в пределах от 0 до 255). Содержит глубину (`depth`) соответствующего бокса. См. `\setbox` и `\showthe`.

145, 322, 377, [456-457](#), 488.

`\dump*` Если эта команда напечатана на месте `\end` или `\bye`, она позволяет предкомпилировать формат `plain`.

[335](#), [340](#), 399, 408.

е

`\edef*` Для определения макрокоманд, текст замены которых не просто дословно копируется, а расширяется в зависимости от текущих условий.

[255-256](#), 326, 390, 413, 439.

`\egroup` Еще одно имя для закрывающей фигурной скобки. Неотделима от команды `\bgroup`. Хотя это и некрасиво, но можно в начале группы написать `\bgroup`, а в конце — `}`. Это станет понятно, когда Вы узнаете, что `\egroup` имеет определение:

`\let\egroup=}`

Макрокоманды `\begingroup` и `\endgroup` образуют другую систему фигурных скобок. Как указывают их имена, они определяют группу. Но не соединяйте их в пару с `\bgroup` и `\egroup`.

319, [416](#), 429, 449, [452](#), 477, 492.

`\eject` Вызывает окончание страницы. Если Вы хотите, чтобы $\TeX$ закончил текущую страницу и начал новую, используйте команду `\eject`, или даже лучше:

`\vfill\eject`

Команда `\vfill` прижимает текст к верху страницы, чтобы страница получилась не слишком разреженной.

Разбиение текста на страницы — это сфера действия команд `\bigbreak`, `\smallbreak`, `\medbreak`, `\goodbreak`, `\filbreak`, а также `\penalty`.

Если Вам нужно получить пустую страницу (чтобы, например, затем вставить туда рисунок), напишите следующую программку:

`\vfill\eject\null\vfill\eject.`

Ни в коем случае не забывайте команду `\null`, (что означает пустой бокс), иначе никакой пустой страницы у Вас не получится!

32-33, 129, 133, 226, 418, 488, 489.

`\ell` Буква ℓ (только в математической моде). Сравните следующие примеры и выберите то, что Вас больше устраивает:

`$l-l*x_1^2+y_1^\alpha$` $l - l * x_1^2 + y_l^\alpha$

`$_ell-ell*x_ell^2+y_ell^\alpha$` $\ell - \ell * x_\ell^2 + y_\ell^\alpha$

Две другие латинские буквы в этих формулах пишутся в несколько модифицированном виде: `\imath` (i) и `\jmath` (j). Это позволяет использовать их вместе со стрелками (или с чем-либо еще): в результате `$_vec\imath$` и `$_vec\jmath$` получается $\vec{i}$ и $\vec{j}$. В математической моде нельзя просто употреблять `\i` и `\j`, поскольку в этом случае получаются символы включения $\subset$ и $\supset$!

160, 507.

`\else*` Термин `else` из языка программирования TeX:

`\iftest действие 1 \fi`

`\iftest действие 1 \else действие 2\fi`

Будет выполнено действие 1, если `\iftest` — истинно, и действие 2 — в противном случае. Часть `\else` необязательна. Никогда не забывайте ставить в конце `\fi`!

246, 250, 252.

`em` Одна из единиц измерения TeX'a, зависящая от текущего шрифта. Приблизительно равна ширине буквы `m`.

75, 186, 201, 254, 321, 417, 485, 505.

`\emergencystretch*` При разбиении абзаца на строки уменьшает плохости при последнем проходе.

131, 325.

`\empty` В формате `plain` команда `\empty` эквивалентна `{}`.

313, 416, 444.

`\emptyset` Пустое множество $\emptyset$. Только в математической моде. Не путать со скандинавской буквой `\O` ($\mathcal{O}$), которая имеет более округлую форму.

156, 507.

`\end*` Указывает TeX'у, что его работа закончена. Любая работа должна заканчиваться командой `\end`, иначе TeX будет терпеливо ждать ваших дальнейших указаний, посылая на экран приглашающий символ. Следует отдать предпочтение команде `\bye`, нежели `\end`, поскольку первая

вынуждает $\TeX$ перед остановкой вывести все, что оставалось в памяти. Эту команду можно использовать в том случае, если вставляются рисунки (макροкоманды `\midinsert` и `\topinsert`).

31, 34, 27, 107, [314](#), [335](#), 358, 357, 399, 473.

`\endcsname*` См. команду `\csname`.

50, [253](#), 336, 412, 441.

`\endgraf*` Обычная операция $\TeX$ 'а `\par`. Используется, если значение `\par` изменено.

312, 340, 393, [416](#), 477, 486, 490.

`\endgroup*` Завершить группу, начатую командой `\begingroup`.

27, 295, 312, [330](#), 447, 477, 489, 490.

`\endinput*` Указывает на конец входного файла. $\TeX$ прекращает чтение из этого файла.

60, [254](#).

`\endinsert` Завершить часть программы, с помощью которой можно вставлять текст в начало последующей страницы, причем начало этой части выглядит как `\topinsert` или `\midinsert`.

140, [429](#).

`\endline` Эквивалент примитива `\cr`. Применяется, если значение `\cr` изменено.

[416](#).

`\endlinechar*` Символ, помещаемый с правого конца входной строки. Обычно он равен 13, но может быть изменен, как и любой параметр.

[61](#), 324, 393, [413](#), 459.

`\endtemplate` Специальная внутренняя операция, которая всегда автоматически помещается в конце шаблона таблицы.

285.

`\enskip` Пробел, равный полуквадрату (`\hskip .5em\relax`) текущего шрифта. Команда `\relax` служит для отделения `\hskip` от последующего текста в том случае, когда он начинается с “plus”, “Plus”, “minus” или “Minus”. $\TeX$ 'у в этом случае может показаться, что написано `\hskip .5em plus...`, и он будет горько жаловаться на отсутствие размера после plus.

89, [417](#).

`\enspace` Керн величиной в полуквадрат (`\kern .5em`) текущего шрифта. В противоположность `\enskip`, нельзя обрывать строку на команде `\enspace` (поскольку это `\kern`).

`\epsilon` Греческая буква ϵ . Возможно, Вы предпочтете ей букву `\varepsilon` (ε), линии которой более закруглены. Только в математической моде.

1, 156, 506.

`\eqalign` Для печати многострочных формул с вертикальным выравниванием. Эту команду можно использовать только в выделенной математической моде.

$ \begin{array}{l} A = \frac{2x}{1+x^2} \\ \leq 2x. \end{array} $	$ \begin{array}{l} A = \frac{2x}{1+x^2} \\ \leq 2x. \end{array} $
--	--

Синтаксис такой же, как для таблиц без преамбулы в две колонки. Каждая строка (вернее сказать, каждая формула), должна иметь в своем составе символ `&` и заканчиваться `\cr`. Символы, непосредственно идущие за амперсандом, будут выравниваться вертикально. Пробелы, которые отделяют символы с той или с другой стороны от амперсанда, будут расположены правильно.

К сожалению `\eqalign` нечувствительна к действию `\hfill`. Следовательно, `\hfill` не может быть средством отдаления знака амперсанда! Не нужно писать поэтому `A\hfill&=B\` или `A&\hfill=B\`, это ни к чему не приведет. Если у Вас возникли с этим проблемы, используйте матрицу (`\matrix`), `\cases` или, например, таблицу (`\halign`). Анализ возможных ошибок смотрите в `\displaylines`. Чтобы вертикально отодвинуть строку, используйте `\noalign` или `\openup`.

227, 287, 388, 428.

`\eqalignno` Для получения уравнений с вертикальным выравниванием и нумерацией справа. Для нумерации слева существует команда `\leqalignno`. Синтаксис аналогичен таблицам без преамбулы в три колонки:

левая часть & правая часть & номер \cr

Здесь, “левая часть” и “правая часть” относятся к формуле. Чтобы пронумеровать какую-либо формулу, добавляют второй амперсанд с номером уравнения: `A=B&(1)\cr`. Номер добавлять не обязательно: можно просто справа написать `A=B\cr`, если номер не нужен. Познакомьтесь с примером полностью:

$$\begin{array}{l}
 \pi^2 \over 6 \\
 = \sum_{n=1}^{\infty} \frac{1}{n^2} \\
 > \sum_{n=1}^{\infty} \frac{1}{n(n+1)} & (1) \\
 \geq \sum_{n=1}^{\infty} \frac{1}{n} - \frac{1}{n+1} = 1 & (2)
 \end{array}$$

В результате получим:

$$\frac{\pi^2}{6} = \sum_{n=1}^{\infty} \frac{1}{n^2} > \sum_{n=1}^{\infty} \frac{1}{n(n+1)} \quad (1)$$

$$\geq \sum_{n=1}^{\infty} \frac{1}{n} - \frac{1}{n+1} = 1 \quad (2)$$

Первому уравнению не приписан номер, поэтому там нет амперсанда, задающего нумерацию. Одно полезное замечание: “номер” формулы или уравнения не обязательно должен быть числом, а может быть и буквой, например (Σ) . После нумерующего амперсанда Вы всегда находитесь в математической моде. Следовательно, правильной будет запись: `...&(\Sigma)\cr`, а не `...&$(\Sigma)$\cr`. И последнее замечание, которое относится к верстке страниц: если программа включает в себя `\eqalign`, может оказаться, что $\TeX$ не может в конце страницы разделить набор формул. В этом случае просто добавьте “no” к `\eqalign`. Поскольку команда `\eqalignno` (так же как и `\leqalignno`) производит “нагромождение” `\line`, $\TeX$ теперь сможет переносить их без всяких проблем.

228, 231, 428.

`\eqno*`

Печатает номер справа от формулы, записанной на отдельной строке. Только в выделенной математической моде.

`$$e^{i\pi}=-1\eqno\hbox{\leftarrow (ЭЙЛЕР)}$$`

$$e^{i\pi} = -1 \quad (\text{ЭЙЛЕР})$$

Чтобы поставить номер слева от формулы, используется `\leqno`:

`$$e^{i\pi}=-1\leqno\hbox{\leftarrow (ЭЙЛЕР)}$$`

(ЭЙЛЕР)

$$e^{i\pi} = -1$$

Вы заметили `\hbox`, обязательный из-за того, что в математической моде происходит смена шрифтов? Внимание: хотя в последнем варианте номера печатаются слева, в команде тем не менее они должны быть записаны справа от формулы. И последняя рекомендация: Вы не можете использовать макрокоманду `\eqno` в `\displaylines`. Иначе говоря, `\eqno` позволяет нумеровать только *одну изолированную отцентрированную строку*. Чтобы пронумеровать все или несколько строк в математической моде, используйте `\eqalignno` и `\leqalignno`.

222-223, 225-231, 349, 422.

- `\equiv` Отношение $\equiv$. Только в математической моде. Противоположное отношение $\neq$ задается с помощью `\not\equiv`.
162, 508.
- `\errhelp*` Вспомогательное сообщение об ошибке (в дополнение к `\errmessage`, которое выводится на терминал, если мало краткого сообщения).
302, 326, 331, 412.
- `\errmessage*` Сообщение об ошибке, которое выводится на терминал. Может быть дополнено сообщением `\errhelp`.
257, 331, 412, 489.
- `\errorcontextline*` Максимум строк дополнительного контекста, который показывается на терминале при обнаружении ошибки. В формате `plain` равен 5.
44, 324, 413.
- `\errorstopmode*` Общепринятый стандартный уровень взаимодействия во время работы `TeX`'а, при котором после каждой ошибки работа приостанавливается, на терминал выводится сообщение об ошибке и запрос на дальнейшие действия. Ср. `\scrollstopmode`, `\nonstopmode`, `\batchmode`.
42, 329, 357.
- `\escapechar*` Значение сигнального символа. В формате `plain` сигнальным символом служит бэкслэш.
51, 253, 271, 324, 368-369, 412-413, 443.
- `\eta` Греческая буква η . Только в математической моде.
506.
- `\everycr*` Материал, который `TeX` автоматически вставляет непосредственно после каждой команды `\cr`.
326, 428.
- `\everydisplay*` Материал, который `TeX` вставляет непосредственно после каждого двойного доллара, открывающего выделенную математическую моду. Если Вы пишете `\everydisplay={\toto}`, то каждая выделенная формула `$$формула$$` будет преобразована в `$$\toto формула$$`. Пояснения к использованию см. в `\displayindent`.
216, 326, 335, 388.
- `\everyhbox*` Материал, который `TeX` автоматически вставляет в начало каждого горизонтального бокса.
326, 331.
- `\everyjob*` Материал, который `TeX` автоматически вставляет в начало работы.
326.

`\everymath*` Материал, который $\TeX$ вставляет непосредственно перед каждой математической формулой в тексте. Сравните с `\everydisplay`.

216, [326](#), [349](#).

`\everypar*` Материал, который $\TeX$ автоматически вставляет в начало каждого абзаца. И тогда программа

```
\everypar={{\bf\the\count17}\quad
\global\advance\count17 by 1\relax}
```

автоматически нумерует абзацы. С этого времени каждый абзац будет начинаться с команды

```
{\bf\the\count17}\quad\global\advance\count17 by 1\relax,
```

которая пишет жирным шрифтом содержимое `\count17`, далее пробел величиной в квадрат, а затем счетчик увеличивается на единицу. Поскольку результат вычислений должен быть определенным образом запомнен, нужно перед `\advance` поставить `\global`. Иначе нумерация осуществится неудачно. Не забывайте инициализировать счетчик — это делается командой `\count17=1` в начале текста.

Обратите внимание: макрокоманда `\everypar` выполняет весьма деликатную функцию — в нее нельзя вставлять какую-нибудь макрокоманду, которая содержит `\par`, или другую команду, создающую начало нового абзаца (например, `\vskip`). Что же произойдет, если написать `\everypar={\smallskip}`? Предположим, что $\TeX$ начинает новый абзац с некоторого символа. Он вставляет содержимое `\everypar`, т.е. `\smallskip`, перед этим символом. Затем, сделав этот маленький промежуток, он опять сталкивается с этим символом, и весь сценарий прогоняется заново. Вы поняли? Программа заикливается.

Для автоматического разделения всех абзацев командой `\smallskip` введите:

```
\parskip=\vskip 2pt plus 1pt minus 1 pt.
```

129, 256, 253, [302](#), 311, [326](#), [335](#), [396](#), [448](#), [477](#), [492](#).

`\everyvbox*` Материал, который $\TeX$ автоматически вставляет в начало каждого вертикального бокса.

326, 331.

`ex` Одна из единиц измерения $\TeX$ 'а, зависящая от текущего шрифта. Приблизительно равна высоте буквы `x`.

75, 186, [321](#), [422](#), 505.

`\exhyphenpenalty*` Специальный штраф, который используется при автоматическом формировании абзацев. В формате `plain` он равен 50.

[119](#), [311](#), [323](#), [413](#).

`\exists` Математический знак $\exists$. Только в математической моде. Для логиков, а также для любителей “современной математики”.

	Функция f непрерывна по x, если $\forall \varepsilon \in \mathbf{R}_*^+ \exists \nu \in \mathbf{R}_*^+$ такое, что $\forall h \in \mathbf{R} \ h < \nu, h < \nu$ выполняется $f(x+h) - f(x) < \varepsilon$	507.
<code>\exp</code>	Задание экспоненциальной функции. Только в математической моде: $\exp(x+y) = \exp x \exp y \dots \dots \dots \exp(x+y) = \exp x \exp y$	196, 727.
<code>\expandafter*</code>	Команда, которая прочитывает следующий за ней элемент (скажем, t), не делая его расширения, затем делает расширение элемента, следующего за t, а затем возвращает t перед этим расширением.	51, <u>253</u> , 255, 310, 368, 393, 412, 440.
f		
<code>\fam*</code>	(Сразу после командного слова должно быть указано целое число). Номер текущего шрифта в математической моде.	186-189, 324, <u>346</u> , 411-412, 416, 423, 485.
<code>\fi*</code>	Завершает условный оператор в языке программирования Т_ЕX: $\text{\if...\fi или \if...\else...\fi}$	246, 249-250, <u>252</u> .
<code>fil</code>	Единица измерения бесконечного клея первого порядка. См. <code>fill</code> и <code>filll</code>.	91, 153-154, <u>321</u> , 412, 463.
<code>\filbreak</code>	Завершает страницу, заполняя ее до конца пробелами. Если Т_ЕX встречает <code>\filbreak</code> и если внизу страницы мало места, он завершает страницу прямо в этом месте, добавляя команду <code>\vfill</code>, устраняющую излишние строки. Если же остается дополнительное место, Т_ЕX продолжает работу и ищет другое место для переноса, чуть подальше. Эта макрокоманда имеет такую же возможность разбивать текст (<code>\penalty -200</code>), как и команда <code>\bigbreak</code>, которая к тому же привлекает еще и <code>\bigskip</code>. Не стоит использовать <code>\filbreak</code>, если абзац слишком длинный, поскольку Т_ЕX прекрасно понимает, что страница, наполненная наполовину, менее “некрасива”, чем страница переполненная! Напротив, макрокоманда <code>\filbreak</code> идеальна для текста, составленного из маленьких абзацев (например, библиография). Завершая каждый абзац командой <code>\filbreak</code>, можно быть уверенным, что никакая из ссылок не будет разбита на две части. И последнее: никаких <code>\everypar{\filbreak}</code>, иначе Т_ЕX заикливается (см. <code>\everypar</code>, чтобы понять, почему).	135, <u>418</u> .

<code>fill</code>	Единица измерения бесконечного клея второго порядка, более “бесконечного”, чем <code>fil</code> .	91, 153-154, 321 , 394 .
<code>filll</code>	Единица измерения бесконечного клея третьего порядка, еще более “бесконечного”, чем <code>fill</code> .	91, 153, 321 , 394 .
<code>\finalhyphendemerits*</code>	Специальный параметр, который используется при автоматическом формировании абзацев. В формате <code>plain</code> он равен 5000.	121, 130 , 323, 413 , 526 .
<code>\firstmark*</code>	Первое вхождение метки на странице, которую Вы собираетесь обрабатывать. См. <code>\mark</code> .	254, 307 , 308-309, 332.
<code>\fivebf</code>	Включение жирного шрифта размером в 5 пунктов. См. <code>\fiverm</code> .	
<code>\fiverm</code>	Включение прямого шрифта размером в 5 пунктов. Используется так же, как жирный шрифт или курсив: <code>\fiverm...</code> для глобального вызова и <code>{\fiverm...}</code> — для локального.	184, 414 , 415 , 484-485 .
<code>\flat</code>	Знак бемоль <code>b</code> . Только в математической моде.	479 , 507.
<code>\floatingpenalty*</code>	Специальный параметр, который используется при автоматическом формировании страниц со вставками.	149-151, 323, 332, 429 .
<code>\folio</code>	Печатает номера текущих страниц. Напомним, что номер первой страницы (скажем, 12) задается командой <code>\pageno=12</code> (знак = необязателен). Если специально ничего не оговаривается, номер первой страницы устанавливается в 1. Для нумерации римскими цифрами используйте отрицательные числа <code>\pageno=-12</code> (первая страница будет иметь номер <code>xii</code>). Макрокоманда <code>\folio</code> доделает все остальное. Для нумерации большими римскими цифрами см. <code>\romannumeral</code> . Если нумерация страниц не нужна, задайте в начале файла <code>\nopagenumbers</code> (обратите внимание: командное слово — во множественном числе).	300, 428 , 476 , 486 .
<code>\font*</code>	Название шрифтов. Предположим, Вы хотите обратиться к имеющемуся у Вас шрифту <code>cmss10</code> . Прежде всего его надо назвать: <code>\font\toto=cmss10</code> Чтобы использовать этот шрифт, введите <code>\toto</code> (для глобального изменения) или <code>{\toto...}</code> (для локального). Вы можете во время вызова	

(и только в это время) изменить размер этого шрифта. Первое решение следующее:

```
\def\toto=cmss5 at 10 pt
```

(Вы указываете размер, который должен приобрести увеличивающийся шрифт). Также можно указать коэффициент увеличения:

```
\def\toto=cmss5 scaled 2000
```

(если 1000 — коэффициент увеличения равен 1). В каждом из этих двух случаев результат один и тот же: шрифт теперь имеет размер в два раза больший первоначального. Шрифт поддерживает увеличение на 20%. Поэтому можно заменить размер в 10 пунктов на размер в 12 пунктов. К тому же Т_ЕX предоставляет Вам готовый коэффициент для этой операции:

```
\font\toto=cmr10 scaled\magstep1
```

Заметьте, что перед `scaled` нет обратной косой черты, а в команде `\magstep1` — есть. Следует представлять шрифт как тройку:

(имя, номинальная величина, коэффициент увеличения)

Важное замечание: смена шрифтов в математической моде осуществляется или в `\hbox`, или с помощью семейства (`\fam`). Иначе ничего не получится!

20, 60, 249, 253, 254-255, 322, 327.

\fontdimen* Этой командой знатоки могут менять параметры любого шрифта. Не хотим провоцировать Вас к использованию этой команды без достаточного опыта, поэтому ограничимся только ее назначением.

95, 189, *215*, 254, 322, 329, *421*, *441*, *458*, 505, 513.

\fontname* Дает внешнее имя шрифта. Например, `\fontdimen\tenrm` даст `cmr10`.
253, 255.

\footline Строка внизу страницы, которая содержит (возможно) номер страницы, заголовок, черту и т.д.
300, 305, *404*, 428.

\footins Один из номеров классов вставок, которые используются plain Т_ЕX'ом.
305, 429, 465-468, *486*.

\footnote Макрокоманда, которая позволяет вставлять примечание внизу страницы. Эта команда имеет два аргумента. Первый указывает на вызов примечания, а второй представляет собой собственно примечание. Программа

```
\sl Никола БУРБАКИ \footnote{(*)}{Мифический автор, оказавший  
огромное влияние на всю современную математику.}  
существует, я его встречал.
```

выдаст Вам следующий текст: “*Никола БУРБАКИ (*) существует, я его встречал.*” и примечание в виде сноски внизу страницы. Обратите внимание, что избранный шрифт (наклонный `\sl`) используется и в сноске внизу страницы. Несколько важных замечаний о синтаксисе. Всегда используйте две группы:

`\footnote{...}{...}`

Ничего не вставляйте между двумя этими группами, даже пробела, иначе $\TeX$ выдаст `runaway argument ?` и текст примечания будет игнорирован. Следовательно, синтаксис вида:

`\footnote{...} {...}`

ни к чему хорошему не приведет. А вот еще пример частой ошибки. Правильный синтаксис справа: вся разница заключается в знаке `%`, но эта разница весьма существенна!

<pre>\footnote{.....} {.....}</pre>		<pre>\footnote{.....}% {.....}</pre>
неправильно		правильно

Этот самый знак `%` не дает $\TeX$ ’у увидеть CR (перевод строки), который завершает строку, и решить, что между двумя группами имеется еще что-то.

102, 141, 299, 305, 404, 429, 449, 486.

\footnoterule Как и указано в имени команды, горизонтальная черта, которая отделяет текст примечаний от нижнего края основного текста страницы. Ее определение в plain $\TeX$ выглядит следующим образом:

```
\def\footnoterule{\kern -3pt
\hrule width 2truein \kern 2.4pt}
```

Ширину (`width`) этой черты можно легко изменять.

305, 430.

\forall Математический знак $\forall$. См. `\exists`.

507.

\frenchspacing Равномерно распределяет пробелы по всей строке. Чтобы понять эту макрокоманду, нужно знать, что в англо-американской полиграфии после точки, вопросительного и восклицательного знаков ставят двойной пробел. С другой стороны, после запятой и точки с запятой они вообще не используют пробелов. Французы, напротив, предпочитают, чтобы все пробелы были примерно одинаковыми. Следовательно, Вы должны выбирать (`\frenchspacing` или `\nonfrenchspacing`?) в зависимости от того, на каком языке вы пишете. В русском языке обычно используется `\nonfrenchspacing` (к тому же принятый в формате `plain` по умолчанию), а пробелы после запятых добавляются вручную.

93, 404, 414, 448, 470.

(*) *Мифический автор, оказавший огромное влияние на всю современную математику.*

<code>\frown</code>	Отношение $(\frown)$. Только в математической моде. Имеется также команда <code>\smile</code> : $(\smile)$.	508.
<code>\futurelet*</code>	Если сказать <code>\futurelet\a\b</code> в конце текста замены макроопределения, $\TeX$ установит <code>\a</code> равным элементу, следующему за макроопределением, затем раскроет <code>\b</code> .	<u>246</u> , 255, 311, 328, 429, 441, 494.
g		
<code>\gamma</code>	Греческая буква γ . Только в математической моде.	156, 506.
<code>\Gamma</code>	Заглавная греческая буква Γ . Только в математической моде.	156, 206, <u>423</u> , 506.
<code>\gcd</code>	(<i>greatest common divisor</i>) Для записи наибольшего общего делителя. Только в математической моде.	196, 229, 427.
<code>\gdef*</code>	Обычно определение, правило, задающее содержимое группы, действует только внутри группы. Если Вам надо, чтобы определение сохранялось, т.е., чтобы оно было известно на другом <i>глобальном</i> уровне, пишите <code>\gdef\toto{...}</code> . Эта макрокоманда является сокращенной записью для <code>\global\def</code> . Прекрасный пример в определении <code>\obeylines</code> .	<u>245</u> , 255, 326, 477.
<code>\ge</code>	(<i>greater or equal</i>) Для получения $\geq$. Только в математической моде. Можно также использовать <code>\geq</code> . Отношение $\leq$ задается командами <code>\le</code> (<i>lower or equal</i> — меньше либо равно) или <code>\leq</code> . Знаки строго неравенства ' $<$ ' и ' $>$ ' есть на клавиатуре.	12, 57, 211, 380, <u>427</u> , 510.
<code>\geq</code>	См. <code>\ge</code> .	380, 508.
<code>\gets</code>	Отношение $\leftarrow$. Оно имеет и другое название — <code>\leftarrow</code> . Последнее писать дольше, но запомнить гораздо легче! Только в математической моде.	<u>427</u> , 510.
<code>\gg</code>	Отношение $\gg$. Только в математической моде. Противоположное отношение $\ll$ задается <code>\ll</code> . Не путать с угловыми скобками <code>\angle</code> и <code>\rangle</code> (более “открытыми”): <code>\ll\kern 1cm\gg</code> $\ll$ $\gg$	

`\langle\!\langle\kern 1cm\rangle\!\rangle` $\langle \rangle$
508.

`\global*` Это префикс, который, будучи помещенным перед командой TeX'a, позволяет этой команде сохраниться за пределами группы, в которой она фигурирует. Пример Вы уже видели с командой `\everypar`.

27, 144, 215, 245, 259, 275, 305, 326, 368, 382, 410.

`\globaldefs*` Параметр для программирования на TeX'e, не равен нулю, если не находится под действием `\global`.

284, 324, 326.

`\goodbreak` Внутри текста команда `\goodbreak` указывает на очень хорошее место (`\penalty -500`) для окончания строки или страницы. Абзац, в котором встречается `\goodbreak`, не заканчивается, следовательно, отступа сделано не будет. Не путайте эту макрокоманду с `\bigbreak`, которая содержит более слабый штраф (`\penalty -200`) и вертикальное перемещение по крайней мере на `\bigskip` (от чего и происходит прерывание текущего абзаца).

141, 418.

`\grave` Ставит знак обратного удара над последующим символом. Только в математической моде:

`\grave a$, \grave +$, \grave X$` $\grave{a}, \grave{+}, \grave{X}$
164.

h

`\H` Ставит знак акцента над следующим символом. Например, `\H o` выдаст $\ddot{o}$.

66, 67, 422, 491.

`\halign` Команда для задания таблиц. Таблица с тремя колонками задается следующим образом:

```
\halign{
.....#.....&.....#.....&.....#.....\cr
colonne 1 &colonne 2 & colonne 3 \cr
.....&.....&.....
colonne 1 &colonne 2 & colonne 3 \cr
}
```

Переменные здесь представлены диyezами #. Амперсанды & указывают на *позиции* табуляции, разграничивающие колонки. Кроме амперсанда, в колонке должен обязательно присутствовать диyez. На месте маленьких точек можно ставить все, что захочется, там содержатся макрокоманды, которые используют диyezы в качестве переменных. Например,

`\smash{#}`, а также `\raise 2pt{\hbox{#}}`. Каждая колонка непроницаема: деятельность в одной колонке (математическая мода, переключение на курсив, и т.д.) не влияет на содержимое остальных колонок, то есть в группу не могут входить две колонки. Не забывайте, что в конце каждой строки должно стоять `\cr`, и, в частности, о том, что после последнего `\cr` не должно быть никаких знаков препинания. Не забудьте завершить команду `\halign` закрывающей фигурной скобкой. Это наиболее частая ошибка, ее последствия могут быть катастрофическими.

См. также `\openup` или `\noalign` (если строки не нужны), `\multispan` (чтобы слить несколько колонок) и, наконец, `\tabskip`, если нужно получить таблицу *определенной ширины*.

142, 227, 230, 231, 275-296, 334, 340, 347, 361, 388, 417, *427-428*, 454, 460.

`\hang`

Вызывает сдвиг всего текущего абзаца вправо *за исключением первой строки*. Такой выступ имеет ширину `\parindent`. Первая строка не сдвигается, если используется `\noindent`, иначе она печатается с тем же абзачным отступом. Хороший пример использования этой команды — это макрокоманда `\item (\par\hang\textindent)`.

*В белом плаще с кровавым подбоем, шаркающей кавалерийской походкой, ранним утром четырнадцатого числа весеннего месяца нисана в крытую колоннаду между двумя крыльями дворца Ирода Великого вышел прокуратор Иудеи Понтий Пилат.*¹

`{\parindent=1cm\noindent\hang\sl...\par}`

Обратите внимание на синтаксис: в демонстрационных целях мы изменили значение `\parindent`. Поскольку это изменение должно быть *локальным*, заключаем этот абзац в группу `{...}`. В таких случаях перед тем, как завершить группу, важно указать Т_ЕX'у командой `\par` или с помощью пустой строки, что абзац закончен. Если же вы забыли написать `\par`, макрокоманда не будет действовать. Команда `\hang` может быть помещена в любое место нужного абзаца! Этот “феномен” обязан способу, которым Т_ЕX работает: перед тем, как разбить абзац на строки, Т_ЕX прочитывает его целиком.

420, 490.

`\hangafter*` Команда задается вместе с целым числом, указывающим на продолжительность (в строках) выступления абзаца. Например, `\hangafter=3` (знак = необязателен). Отступ задается в параметре `\hangindent` (представляющим собой размер). Эта макрокоманда обобщает `\hang`. Она модифицирует верстку абзаца, в котором находится, и может быть помещена в любом его месте.

¹ М. Булгаков, *Мастер и Маргарита*.

Если `\hangafter=3`, строки 1, 2, 3 не будут сдвинуты, отступ начнется только со строки 4 (разумеется, мы предполагаем, что этот абзац содержит, по крайней мере, четыре строки). Если же `\hangafter=-3`, то именно эти строки 1, 2, 3 будут напечатаны с отступом, а строка 4 и все последующие — без отступа.

И последнее: если `\hangindent` положителен, выступ оказывается слева от текста. Если `\hangindent` отрицателен, выступ появляется справа.

125, 324, 413, 490.

`\hangindent*` Размер, указывающий величину “подвешенного отступа”, используется с макрокомандой `\hangafter`. Синтаксис: `\hangindent=5mm` или `\hangindent=-20pt` (любой допустимый в $\TeX$ ’е размер).

125, 311, 325, 413, 477.

`\hat` Ставит над следующим за командой символом знак “шляпка” (в математической моде). Чтобы нарисовать “шляпку” не в математической моде, используйте команду `\^`:

`$\hat x$, $\hat +$, $\hat q$` $\hat{x}$, $\hat{+}$, $\hat{q}$

48, 66, 156-158, 164, 435, 494.

`\hbadness*` $\TeX$, чтобы “оценить” качество сформированной страницы, пользуется понятием плохости (badness). Если плохость `\hbox` превышает `\hbadness`, то $\TeX$ об этом предупреждает. В формате `plain` задан порог `\hbadness=1000`. Можно изменить это значение, если оно часто мешает что-либо напечатать. Для этого в начале файла задайте, например, `\hbadness=5000`.

38, 323, 361, 413, 456, 470.

`\hbar` Перечеркнутое h ($\hbar$), употребляемое физиками. Только в математической моде.

204, 423, 507.

`\hbox*` Укладывает материал в бокс. Говоря по-простому, любой объект $\TeX$ ’а можно представить в виде бусины с горизонтальным отверстием. Это отверстие может быть сдвинуто *вниз* (случай `\vbox`), находиться в *центре* (случай `\vcenter`) или *наверху* (случай `\vtop`). Разместить несколько боксов в одном `\hbox` означает пропустить через все отверстия одну нить. См. `spread` и `to`.

81, 96, 106, 115, 182, 192, 197, 215, 221-222,
263, 264, 330, 334, 458.

`\headline` Материал, который в общем случае содержит заголовок или номер страницы, размещаемые сверху печатного листа.

300, 304, 428, 476.

<code>\heartsuit</code>	Знак ♡. Только в математической моде.	507.
<code>height</code>	Ключевое слово для задания высоты линейки (см. <code>\hrule</code> и <code>\vrule</code>).	263, <u>334</u> , 400.
<code>\hfil*</code>	(С одной ‘l’) Слабый бесконечно растяжимый клей в горизонтальной моде. Например, <code>\line{\hfil...}</code> эквивалентно <code>\rightline{...}</code> . Не пользуйтесь этим слабым клеем при создании новых макрокоманд (прочтите следующую статью, чтобы понять, почему).	90, 231, 278-282, 335, 339, 346, 467.
<code>\hfill*</code>	(С двумя ‘l’) Бесконечно растяжимый клей в горизонтальной моде, более мощный, чем предыдущий. Иными словами, <code>\line{\hfil...\hfill}</code> эквивалентно <code>\line{...\hfill}</code> и <code>\leftline{...}</code> . Этому клею требуется место для растяжения, но он не может его создавать! Следовательно, <code>\hfill</code> в <code>\hbox{\hfil...}</code> и <code>{...\hfill}</code> бесполезен, поскольку это эквивалентно <code>\hbox{...}</code> и <code>{...}</code> . Все макрокоманды, которые <i>центрируют</i> аргументы (например, <code>\matrix</code> , <code>\over</code> , <code>\displaylines</code> , <code>\centerline</code>) используют слабо бесконечный клей <code>\hfil</code> . Из-за этого можно всегда модифицировать их поведение, используя несколько более мощный клей <code>\hfill</code> : <pre> \def\toto#1{\hbox to 3cm{\${\alpha\hfil#1\hfil\omega}\$}} \toto{\mu\hfill} [$\alpha\mu$] ω] \toto{\mu} [α μ ω] \toto{\hfill\mu} [α $\mu\omega$] </pre>	90, 173, 212, 231, 278, 335, 339, 346.
<code>\hfilneg*</code>	Примитив, который отменяет растяжимость <code>\hfil</code> .	90, 123, 278, 335, 339, 346, 467.
<code>\hfuzz*</code>	Если Вы в начале файла печатаете <code>\hfuzz=5pt</code> , T _E X будет выдавать сообщение об <code>overfull \hbox</code> только для тех боксов, размер которых превышает <code>\hsize</code> больше, чем на 5 пунктов. Также имеется и вертикальная версия этой команды — <code>\vfuzz</code> , но она используется реже.	39, 325, <u>361</u> , <i>413</i> , <i>456</i> .
<code>\hglue</code>	Горизонтальный пробел, не исчезающий при переносе строки.	<u>417</u> .
<code>\hideskip</code>	Специальный клей, используемый в определениях “табличных” макрокоманд.	<u>411</u> , 412, 419.
<code>\hidewidth</code>	Как и команда выше, специальный клей, используемый в таблицах:	


```
...& \hidewidth текст &...\cr
...& текст \hidewidth &...\cr
```

В первом случае текст в колонке можно сдвинуть вправо, и при этом никак не нарушить всю остальную таблицу. Во втором случае — влево.

288, 291, 294, 387, [419](#).

\hoffset* Позволяет горизонтально перемещать весь текст по листу бумаги. Команда необходима для размещения документа на странице при выдаче его на печать. Синтаксис: `\hoffset=7mm` (или любой другой допустимый размер). Знак `=` необязателен. Если размер положителен, текст смещается вправо. Если размер отрицателен, текст сдвигается влево. Вертикальная версия, как Вы уже, вероятно, догадались, `\voffset`.

[299](#), 325, [406](#).

\holdinginserts* Параметр, который положителен, если при выводе еще остались “спящие” вставки.

152, 324, 469

\hom Функция `hom`. Только в математической моде.

196, 427.

\hookleftarrow Отношение $\leftarrow$. Только в математической моде.

424, 509.

\hookrightarrow Отношение $\rightarrow$. Только в математической моде.

424, 502, 509.

\hphantom Когда Вы вводите `\hphantom{...}`, $\TeX$ создает горизонтальную невидимую черту, длина которой равна ширине содержимого фигурных скобок. Очень часто используется в некоторых видах выравниваний.

214, 250, [426](#).

\hrule* Команда

```
\hrule height 2pt depth 1 pt width 3cm
```

проводит горизонтальную черту высотой 2 pt, глубиной 1 pt и длиной 3 см. Когда $\TeX$ встречает эту команду, он переходит в вертикальную моду (если уже там не находится). Если он этого сделать не может (например, из-за того, что Вы находитесь в `\hbox` или в `\halign`), он жалуется на то, что Ваша просьба невыполнима. В результате Вы можете применить `\hrule` только в `\vbox` или на странице (которая является частным случаем `\vbox`). Нельзя воспользоваться этой командой внутри абзаца, поскольку там Вы находитесь в горизонтальной моде). Черту следует рассматривать как бокс, закрашенный черной краской. Если высота плюс глубина этого бокса превосходит ширину, получается “вертикальная” черта:

`\vbox{\hrule height 12 pt depth 3 pt width 3pt}` **|**

Параметры `height`, `depth` и `width` должны быть расположены именно в таком порядке. Внимание, не используйте знак `=`! Часть этих параметров может отсутствовать. Если ширина `width` отсутствует, `TeX` считает, что `width` равняется ширине самого большого бокса, помещенного в `\vbox`, в котором находится эта черта (например, черта идет от одного поля страницы до другого). Если `depth` отсутствует, `TeX` присваивает ему значение 0 pt. Если отсутствует `height`, он ему дает значение 0.4 pt. Команда `\hrule` сама по себе и без параметров в тексте заканчивает текущий абзац и рисует черту толщиной 0.4 pt, идущую от одного поля страницы до другого. Помните также, что `TeX` не добавляет вертикальных пробелов ни до, ни после `\hrule`.

Чтобы провести черту в горизонтальной моде, используйте `\vrule`.

32, 80, 105, 263, 292, 333-334, 340, 422, 491, 492

.

`\hrulefill` Для проведения горизонтальной черты в `\hbox` или в горизонтальной моде (ограниченной или нет). Макрокоманда `\hrulefill` — это клей, который “рисует линейки”. Она не занимает места, следовательно, `\hbox{\hrulefill}` и `{\hrulefill}` ничего не делают. Пустоты следует создавать другими командами:

`\hbox to 2.5cm{вот\ \hrulefill\ строка}` вот — строка
290, 300, 422, 482.

`\hsize*` Ширина печатаемого текста. Синтаксис: `\hsize=12cm`. Более обще, управляет шириной строки `\line` (а следовательно, и шириной `\vbox`). Знак `=`, как всегда, необязателен:

`\vbox{\hsize=8cm...}` или `\vbox{\hsize 8cm...}`
35, 74-75, 125, 224, 282, 299, 306, 325, 404, 413, 455, 476, 477,
483, 486, 488.

`\hskip*` Горизонтальный пропуск переменной величины. Синтаксис: `\hskip 3mm` или также `\hskip 5pt plus 2pt minus 3pt`. Так созданный пробел является хорошим местом для переноса строк. При переносе `\hskip` исчезает. Чтобы получить принудительный пробел, используйте `\kern`. А для получения пробела, который не может исчезнуть во время разрыва строки, используйте `\hglue`. Каждый раз, когда это возможно, старайтесь обеспечить команде `\hskip` некоторую степень свободы, задавая необязательные параметры `plus` и `minus`. Перед этими параметрами не ставится обратная косая черта, а также после них не ставится знак `=`; `plus` должен всегда идти раньше, чем `minus`. И наконец, любой из этих двух параметров может быть опущен. Вы значительно облегчите себе верстку страницы, если зададите эластичные пробелы.

89, 106, 203, 335, 339, 346, 375.

- \hss*** Горизонтальный клей, который может выполнить и отрицательное растяжение. Сначала это трудно поддается интуитивному восприятию, но потом будет Вам необычайно полезно. Новички, воздержитесь! Подождите, пока не приобретете достаточной закалки. Но если Вы сгораете от любопытства, вот вам пример:
- `\hbox to 0pt{...\hss}` или также `\hbox to 0pt{\hss ...}`
- В результате получится бокс нулевой ширины, но содержащий материал, который может быть *нормально напечатан*. На листе бумаги создается впечатление, что текст выходит за пределы бокса в сторону `\hss`. Эту хитрость используют макрокоманды `\llap` и `\rlap`.
- 90, 103, 278, 335, 339, 346, 514.
- \ht*** После команды должно следовать целое число, расположенное в пределах от 0 до 255. Этот параметр содержит высоту бокса `\box`, номер которого равен этому числу. Позволяет массу различных фокусов (например, `\smash`), но это уже для особо опытных Т_ЕXпертов.
- 145, 322, 456-457, 488.
- \hyphenation*** Позволяет добавлять новое слово и способ его разбиения на слоги в специальную таблицу. Если Вы юрист и печатаете какой-нибудь труд по конституционному праву, Вы захотите научить Т_ЕX соответствующим переносам. Поместите в начале своего файла команду:
- `\hyphenation{кон-сти-ту-ци-он-ный кон-сти-ту-ция}`
- Таким способом Вы указываете, как правильно разбивать эти слова для переноса с одной строки на другую. Разумеется, этот список новых слов можно увеличивать.
- 329, 499, 527-528, 530.
- \hyphenchar*** Символ переноса. Каждый шрифт имеет свой связанный с ним символ переноса. В шрифтах формата `plain` это дефис.
- 118, 254, 322, 324, 329, 341, 415, 464, 484, 529.
- \hyphenpenalty*** Специальный штраф, который использует Т_ЕX при автоматическом формировании абзацев, в формате `plain` равен 50.
- 119, 124, 323, 413, 526.
- ı**
- \i** Символ *i* без точки сверху. Это позволяет ставить над ним другие знаки (`\^i`, `\"i`). В математической моде следует использовать `\imath`, иначе получится знак включения $\subset$, а не *i*!
- 67, 421.
- \ialign** Общее название всех видов выравнивания: предназначено для больших спецов и изощренных пользователей.
- 419.

<code>\if*</code>	Зарезервированное слово <i>if</i> в языке программирования Т_ЕX. Обычно после <code>\if</code> стоит имя. В этом случае <code>\if-имя</code> является булевской переменной. Синтаксис (внимание: в Т_ЕX'е нет никакого <code>\then</code>): <code>\if...\fi</code> или <code>\if...\else...\fi</code> Проверка отсутствует, или, вернее, она включена в <code>\if</code>, так как есть множество команд <code>\ifcase</code>, <code>\ifcat</code>, <code>\ifdim</code>, <code>\ifmmode</code> и т.д. Как язык программирования, и это следует признать, Т_ЕX достаточно примитивен. <u>249</u>, 250-251, 368, 433, <i>446</i>.
<code>\ifcase*</code>	Т_ЕХовская версия обычного <i>case</i>. <u>250</u>, <i>414</i>, <i>439</i>, <i>459</i>, <i>476</i>.
<code>\ifcat*</code>	Проверить равенство номеров категорий. <u>249</u>, 250, 368, <i>443</i>.
<code>\ifdim*</code>	Проверить, являются ли два размера одинаковыми. <u>248</u>, <i>418</i>, <i>456</i>, <i>488</i>.
<code>\ifeof*</code>	Проверка на конец файла. <u>249</u>, 257.
<code>\iff</code>	Отношение $\iff$ (добавлен пробел <code>\</code>; с обеих сторон от знака эквивалентности). Только в математической моде. 198, <u>427</u>, 511.
<code>\iffalse*</code>	Условие, которое всегда ложно. <u>249</u>, 251, <i>310</i>, <i>412</i>, <i>454</i>.
<code>\ifhbox*</code>	Проверка регистра бокса. Истинно, если соответствующий регистр пуст или содержит горизонтальный бокс. <u>249</u>, <i>461</i>, <i>469</i>.
<code>\ifhmode*</code>	Проверить, находится ли Т_ЕX в горизонтальной моде (обычной или ограниченной). <u>248</u>, <i>429</i>.
<code>\ifinner*</code>	Проверить, находится ли Т_ЕX во внутренней вертикальной, ограниченной горизонтальной или математической моде. <u>249</u>.
<code>\ifmmode*</code>	Проверить, является ли текущая мода математической (выделенной или нет). Приведем пример очень простого и в то же время очень нужного ее использования. Чтобы написать α, нужно задать <code>\$_\alpha\$</code>. Это долго. Поможет какое-нибудь сокращение, например: <code>\def\alpha\$_\alpha\$</code>. К

сожалению, использование этой макрокоманды в какой-нибудь формуле, скажем, в $\$a+\beta\$$, вызовет ужасные последствия: Т_ЕX не любит математической моды внутри бокса. Действительно, когда он раскрывает в предыдущей программке макрокоманду `\a`, он получает текст $\alpha+\beta$ — остальное додумайте сами. Решение заключается в том, чтобы сделать правильный выбор между математической и нематематической модами:

```
\def\a{\ifmmode\alpha\else$\alpha$\fi}
```

Тогда никаких проблем не возникнет! Другим решение, менее элегантным, будет

```
\def\a{\hbox{$\alpha$}}
```

248, 255, 286, 419, 422, 426, 494.

<code>\ifnum*</code>	Сравнить два целых числа.	247, <u>248</u> , 259-260.
<code>\ifodd*</code>	Проверить, является ли целое число нечетным.	246, <u>248</u> , 487.
<code>\iftrue*</code>	Условие, которое всегда истинно.	<u>249</u> , 251, 310, 412.
<code>\ifvbox*</code>	Проверка указанного регистра бокса. Истинно, если регистр пуст или содержит вертикальный бокс.	<u>249</u> .
<code>\ifvmode*</code>	Проверить, находится ли Т _Е X в вертикальной или внутренней вертикальной моде.	<u>248</u> .
<code>\ifvoid*</code>	Проверить, пустой ли соответствующий регистр боксов.	<u>249</u> , 305.
<code>\ifx*</code>	Проверка согласования элементов.	<u>249</u> , 255, 368, 441-443, <u>452</u> , 489.
<code>\ignorespaces*</code>	Т _Е X читает текст и ничего не делает, пока не встретит не-пробел. Можно использовать в макрокомандах для уничтожения нежелательных пробелов.	<u>331</u> , 396, 420, 495.
<code>\Im</code>	Мнимая часть комплексного числа $\Im$. Только в математической моде. Правда, эта готическая буква $\Im$ часто приводит к недоумению (кто в наше время знаком с готическим шрифтом?). Можно помочь зрительному восприятию этой $\Im$, написав после нее букву 'm': $\Im z$, $\Im m$, z $\Im z$, $\Im m z$	

Обратите внимание на использование математического мини-пробела `\,`. Если Вы часто используете эту конструкцию, лучше сделать из нее одну макрокоманду. Пробелы будут автоматически вставляться при верстке страницы:

```
\def\IM{\mathop{\Im m}\nolimits}
$\IM z$, $\IM(z+z')$ .....  $\Im z$ ,  $\Im(z + z')$ 
```

Действительная часть задается с помощью `$\Re$`, при этом получается готическая $\Re$. И в этом случае $\Re$ производит лучшее впечатление: теперь вам ничего другого не остается, как определить макрокоманду `\RE`. Обратите внимание на употребление (обязательное) заглавных букв, чтобы не смешивать новые макрокоманды с макрокомандами `\Im` и `\Re` формата plain.

507.

`\imath` Символ ‘i’ без точки сверху в математической моде: i . Это позволяет писать другие надстрочные символы: `$\vec{\imath}` дает $\vec{i}$. В текстовой моде следует использовать `\i`. Имеются еще два латинских символа, специально предназначенные для математической моды: `\jmath` и `\ell`.
165, 507.

`\immediate*` Если эта команда стоит перед командами вывода `\write`, `\openout` и `\closeout`, эти операции будут выполнены немедленно. Например,

```
\immediate\write16{Большой привет!}
```

сразу же передаст Вам на терминале **Большой привет!** Без `\immediate` Вы получили бы этот большой привет только после вывода всего текущего списка, а могли и вообще его не получить, если список был занесен в бокс, который копировался.

269-270, [331](#), [493](#), [494](#).

`in` Одна из единиц измерения в TeX’e, дюйм (1 in = 2.54 cm).

[32](#), [71](#), [321](#).

`\in` Знак принадлежности $\in$. Постарайтесь использовать эту команду как можно реже. Взгляните свежим взглядом на типографии времен наших дедов, которые в этом понимали намного больше нас! Если Вы предпочитаете более современные книги, то сосчитайте, сколько раз знак $\in$ встречается у Бурбаки.

[156](#), [178](#), [508](#).

`\indent*` Переводит TeX в горизонтальную моду и сдвигает начавшуюся строку на величину `\parindent`. Можно также использовать `\indent` в любом месте абзаца для получения горизонтального пробела величиной `\parindent`. А команды `\indent\indent` в начале абзаца вызывают двойной сдвиг.

106, 116, 124, 312, [335](#), [340](#), [347](#), [420](#).

<code>\inf</code>	Оператор <code>inf</code>. Только в математической моде: $m = \inf_{t \in T} t$ $m = \inf_{t \in T} t$ Можно изменять поведение индексов в <code>\textstyle</code> и <code>\displaystyle</code>, задавая команды <code>\inf\limits_{t \in T}</code> или <code>\inf\nolimits_{t \in T}</code>. 196, 427.
<code>\infty</code>	Для задания знака бесконечности ∞. Только в математической моде. 12, 380, 507.
<code>\input*</code>	Указывает $\TeX$'у читать файл, имя которого указано после команды. 9, 12, 43-45, 60, 238, 254, 257, 447, 450, 473, 493
<code>\inputlineno*</code>	Одна из многих внутренних величин $\TeX$'а, с помощью которой разработчик формата может влиять на поведение $\TeX$'а. Конечно же, только для $\TeX$пертов. 254, 322.
<code>\insert*</code>	За командой следует целое число от 0 до 254. Задаёт класс вставки. Используется при автоматическом формировании страниц со вставками (сносками, иллюстрациями, ...). 117, 148, 308, 322, 429, 496, 529.
<code>\insertpenalties*</code>	Один из специальных штрафов, который $\TeX$ использует при автоматическом формировании страницы со вставками, равен сумме всех штрафов за расщепление вставок на странице. 136, 139, 149-152, 254, 303, 305, 322.
<code>\int</code>	Для задания знака интеграла $\int$ и $\int$. Только в математической моде. Знаки <code>_</code> (индекс) и <code>^</code> (показатель степени) служат для обозначения пределов. $\Gamma(z) = \int_0^\infty e^{-t} t^{z-1} dt$ Если Вам тесно на строке, используйте конструкцию <code>\int\limits</code>, которая позволяет “индексам” размещаться над и под знаком интеграла: $\Gamma(z) = \int_0^\infty e^{-t} t^{z-1} dt$ 174, 204-205, 229, 424, 507.

`\interdisplaylinepenalty` Междустрочный штраф в `\displaylines`, равен 100.
230, 414, 428.

`\interfootnotelinepenalty` Междустрочный штраф в сносках, равен 100.
414, 429.

`\interlinepenalty*` Один из специальных штрафов, используемых при автоматическом формировании абзацев.
128, 323, 363, 476, 490.

`\iota` Греческая буква ι . Только в математической моде.
387, 506.

`\it` Вызывает курсивный шрифт. Используется или в виде `\it...` для изменений глобального порядка, или внутри фигурных скобок `{\it...}` — для локальных изменений.
17, 199, 276, 395, 416, 479, 485, 489, 500.

`\item` Создает форму абзаца. Если Вы составляете письмо или, например, свод инструкций, макрокоманда `\item` будет очень полезна. Если Ваш абзац начинается как-то так:

`\item{$\bullet$}` ...текст...

макрокоманда `\item` временно увеличивает левое поле `\parindent` и пишет `\bullet` уже на новом поле. В следующем абзаце изменения не делаются (в примере ниже `\parindent` присвоено значение 1 см):

- Я подозреваю, что читателей этой книги интересуют подробности космического путешествия: перегрузки, звездные пейзажи, столкновения с метеоритами, встречи и сражения с представителями иных цивилизаций.

Увы, ничего подобного в нашем путешествии не было. Кому такие вещи интересны, пусть читают научно-фантастические романы, к которым лично я никакого отношения не имею. Я описываю только то, что было, и ничего лишнего.²

`\item{$\bullet$}` Я подозреваю, что ... цивилизаций.\par
Увы, ничего подобного в нашем путешествии ...\par

Не забывайте, что `\item` является макрокомандой с одной переменной: если опустить окружающие фигурные скобки, $\text{\TeX}$ не сможет уловить тот символ, который надо поместить на полях. Если не нужно ничего помещать на полях, задайте `\item{}`. В этом случае $\text{\TeX}$ будет знать, что переменная, которую он ищет, является пустой.

Автоматический отступ — это хорошая идея для литературных текстов, но это ужасно, когда нужно работать со сложными сборками боксов и текстов. Когда набираются математические выражения, результат представляется не слишком радостным, поскольку абзацы, в общем

² В. Войнович, Москва 2042.

случае, оказываются короткими. Приходим к парадоксальному результату, когда текст тщательно выровнен по правому краю и изрезан слева отступами! Поэтому часто работают со значением `\parindent=0mm`. Для использования `\item` необходимо *локально* модифицировать значение `\parindent`:

```
{\parindent=1cm
\item{(iii)} ... абзац ...
\par}
```

Команда `\par` перед фигурной скобкой, которая закрывает группу, обязательна. (Можно заменить `\par` пустой строкой, которая для Т_ЕX'a имеет в точности то же значение). Если Вы хотите знать, какова роль этой `\par`, посмотрите, что получается, если ее удалить:

- *А то, что было на самом деле, даже не очень удобно рассказывать. Некоторые детали я охотно бы опустил, только моя исключительная правдивость не позволяет мне ни на шаг отступить от правды фактов.*³

```
{\parindent 1cm\item{${\bullet}$}\sl A то, что было ...}
```

Поле не переместилось! Поэтому существенно завершать абзац командой `\par` или пустой строкой *до того*, как будет закрыта группа. Если друг за другом следуют несколько абзацев, к которым хотелось бы применить команду `\item`, нет необходимости предпринимать какие-либо дополнительные действия *между* абзацами. Сама команда `\item` и завершает предыдущий абзац (ее действие начинается с `\par`).

125, 142, 404-406, 420, 486, 490.

`\itemitem` Оказывает точно такое же действие, как и макрокоманда `\item`, за тем исключением, что пробельная полоса слева равна двойному `\parindent`. Рекомендации по использованию те же самые, что для `\item`. Вот пример двух абзацев, причем первый — это `\item{P}` В двадцать втором году ..., а второй — `\itemitem{${\bullet}$}` Но и в таком виде

¶ *В двадцать втором году, когда я вернулся в Москву, я нашел ее опустевшею, полуразрушенной. Такою она вышла из испытаний первых лет революции, такою осталась и по сей день. Население в ней поредело, новых домов не строят, старых не подновляют.*

- *Но и в таком виде она остается большим современным городом, единственным вдохновителем воистину современного нового искусства.*⁴

125, 406, 420.

`\itfam` Такая команда смены шрифта, что `\it` должна обязательно обращаться к какому-либо семейству в математической моде. Иначе ничего не происходит.

416, 485.

³ В. Войнович, *Москва 2042*.

⁴ Б. Пастернак, *Доктор Живаго*.

j

`\j` Буква j без точки сверху: j. В математической моде следует писать `\jmath`, иначе получится знак включения $\supset$!

67, [421](#).

`\jmath` См. предыдущую макрокоманду. Только в математической моде.

165, 507.

`\jobname*` Раскрывается в имя, которое Т_ЕX выбрал для данной работы.

[253](#), 255, 399.

`\joinrel` Отрицательный керн, используемый для составления длинных знаков. Применяется для скрепления двух знаков отношений. Например, конструкция $\circ\!\!\rightarrow$ задается такой программой:

```
$\mathrel{\joinrel\mathrel{\circ}\joinrel\mathrel{\rightarrow}}$

```

Пришлось иметь дело с отношениями, чем и объясняется нагромождение `\mathrel`. Именно таким образом составляются все длинные знаки, такие как `\longrightarrow`, `\Longleftarrow` и т.д.

[424](#).

k

`\kappa` Греческая буква κ. Только в математической моде.

156, 506.

`\ker` Оператор для задания $\ker$ в математике. Только в математической моде.

для $\$x\in\ker u\$$ справедливо $\$u(x)=0\$$
..... для $x \in \ker u$ справедливо $u(x) = 0$

196, 427.

`\kern*` Керн, т.е. неизменяемый пробел, который нельзя ни сжать, ни растянуть. Синтаксис: `\kern 2cm`. Писать `\kern 2cm plus 1mm minus 2mm` — это полнейший абсурд, который Т_ЕX прочтет как `{\kern 2cm} plus 1mm minus 2mm`. Догадывайтесь, что Вы получите в ответ? Эта макрокоманда обладает еще одной особенностью: `\kern 2cm` задает горизонтальное перемещение (т.е. пробел), если Т_ЕX находится в горизонтальной моде, и вертикальное перемещение, если Т_ЕX — в вертикальной моде. Т_ЕX может разрывать строку на команде `\hskip` или страницу — на `\vskip`, но не может этого делать на `\kern`. В математической моде имеется команда `\mkern`, более приспособленная к особенностям этой моды. Поэтому `\mkern` вызывает только горизонтальные перемещения.

13, 50, 82, 94, 108, 203, 305, 313, [332](#), 366, 458, 463, 486, 496, 529.

1

<code>\l</code>	Перечеркнутое l.	67, 421 .
<code>\L</code>	Перечеркнутое L.	67, 422 .
<code>\lambda</code>	Греческая буква λ. Только в математической моде.	212, 506.
<code>\Lambda</code>	Заглавная греческая буква Λ. Только в математической моде.	385, 506.
<code>\land</code>	Бинарный оператор ∧. Только в математической моде. Для задания векторного произведения: $\begin{aligned} &\text{\def\#1{\overrightarrow{\#1}} \quad \$(\text{\u\land}\text{\v})\land\text{\w}=} \\ &(\text{\v}\text{\cdot}\text{\w})\text{\,},\text{\u}-(\text{\u}\text{\cdot}\text{\w})\text{\,},\text{\v}\$} \\ &(\vec{u} \wedge \vec{v}) \wedge \vec{w} = (\vec{v} \cdot \vec{w}) \vec{u} - (\vec{u} \cdot \vec{w}) \vec{v} \end{aligned}$	161, 427 , 510.
<code>\langle</code>	Открывающая угловая скобка ⟨. Только в математической моде. Закрывающая угловая скобка задается командой <code>\rangle</code> . Чтобы поставить по две скобки рядом, пишем <code>\langle\!\!\langle</code> и <code>\rangle\!\!\rangle</code> :	
	$ \langle\langle A, B \rangle\rangle ^2 \leq \langle\langle A, A^+ \rangle\rangle \times \langle\langle B^+, B \rangle\rangle $ Не путайте угловую скобку ⟨ с более “острым” знаком неравенства ‘<’.	176, 181, 189, 425 , 510.
<code>\language*</code>	TeX может запомнить до 256 различных наборов правил переноса. Конкретный набор правил задается целым параметром <code>\language</code> . Например, <code>\language=1</code> задает русский перенос.	324, 411, 530.
<code>\lastbox*</code>	Один из видов боксов: если последний элемент горизонтального или вертикального списка — бокс, он убирается из списка и становится <code>\lastbox</code> , иначе <code>\lastbox</code> пустой.	264-265, 329 , 420 , 461 , 467 .
<code>\lastkern*</code>	<code>\the\lastkern</code> дает величину керна в последнем элементе текущего списка.	255, 322.
<code>\lastpenalty*</code>	<code>\the\lastpenalty</code> дает величину штрафа в последнем элементе текущего списка.	255, 322.

`\lastskip*` `\the\lastskip` дает величину клея или математического клея в последнем элементе текущего списка.

255, 265, 322, 461.

`\lbrace` Известно, что фигурные скобки ‘{’ и ‘}’ зарезервированы Т_EX’ом для специальных целей. Для получения открывающей скобки в тексте, задайте `$_lbrace$`, или более просто, `$_{ $`. Закрывающая скобка получается аналогично: `$_rbrace$` или `$_}$`. Нельзя задавать фигурную скобку вне математической моды. Величина фигурной скобки может меняться с помощью команд серии `\big` или конструкции `\left...\right`. Например, в результате `$_E=\bigl\{x\mid P(x)\bigr\}$` получится:

$$E = \{x \mid P(x)\}$$

163, 176, 425, 510.

`\lbrack` Открывающая квадратная скобка [. Это точно такой же символ, какой имеется на любой клавиатуре. Только в математической моде. Что касается величины скобки, см. комментарии выше.

176, 416, 436, 510.

`\lccode*` Все 128 возможных символа имеют два связанных значения, `\uccode` и `\lccode` (верхний и нижний регистры), которые, как и `\catcode`, можно изменять. Переход к верхнему или нижнему регистру делают команды `\uppercase` и `\lowercase`.

51, 254, 322, 409, 527.

`\lceil` Символ $\lceil$. Только в математической моде. Величина может меняться, как и у `\lbrace`:

`$_\lceil x\rceil$` --- это наименьшее целое `$n\geq x$`
..... $\lceil x \rceil$ — это наименьшее целое $n \geq x$
176, 424, 510.

`\ldotp` Точка на строке в математических выражениях с пробелом до и после нее. Конечно же, только в математической моде! Обратите внимание на то, где расположены приведенные ниже точки и как выглядят окружающие их пробелы:

`$_\cdot x\ldotp x$` $x \cdot x \cdot x \cdot x$
424, 510.

`\ldots` Три точки, обозначающие пропущенный текст между двумя запятыми. Только в математической моде.

`$_{(x_1,\ldots,x_n)}$` $x = (x_1, \dots, x_n)$
`$_{(x_1,\ldots,x_n)}$` $x = (x_1, \dots, x_n)$

Разница заметна, не так ли? Если нужны *отцентрированные* точки (т.е., на высоте знака ‘=’ или ‘+’, как, например, в формуле $S_n = x_1 + \dots + x_n$), используйте `$_\cdots$`.

92, 207, 213, 216-217, 237-238, 424, 510.

`\le` (*lower or equal*) Отношение $\leq$. Только в математической моде. Его можно задавать и командой `$$\leq$`. Противоположное отношение $\geq$ задается командами `$$\geq$` и `$$\geq$`.

12, 57, 162, 196, 380, 427, 510.

`\leaders*` Отточия. Команда идеальна для задания таблицы содержания книги:

```
\def\toto{\leaders\hbox to 5mm{\hfil.\hfil}\hfill}
\line{Глава 1 \toto 3}
\line{Глава 2 \toto 15}
```

```
Глава 1 . . . . . 3
Глава 2 . . . . . 15
```

Заметьте, что точки оказываются выровненными вертикально; команда `\dotfill` осуществить такое не в состоянии. Не пытайтесь разбираться в том, как эта макрокоманда работает! Запомните только, что Вы можете заменять `\hbox` в определении `\toto` на другой `\hbox` с тем содержимым, которое Вам требуется: `\hbox to 3mm{\hfil$*\hfil}`, `\hbox to 10mm{${\hfil\circ\hfil}`, `\hbox{\ \TeX\ }`, и т.д. В формате `plain` есть также команды `\cleaders` и `\xleaders`, которые отличаются тонкостями в расположении повторяемых элементов.

117, 134, 266, 225, 422, 460-462.

`\leavevmode` (в названии команды две буквы 'v'.) Обеспечивает переход Т_ЭX'a из вертикальной моды в горизонтальную или, если Вам так больше нравится, указывает на начало абзаца. Предположим, Вы попытаетесь начать какой-либо абзац с бокса, задав `\hbox to 1cm{${\bullet}\hfill}` **И тра-та-та, и тра-та-та ...**. Тогда в результате получите (здесь значение `\parindent` равно 1 см):

•

И тра-та-та, и тра-та-та
Пояснение: когда Т_ЭX встречается `\hbox`, он находится в вертикальной моде. Он входит внутрь бокса, делает там что-то, выходит и оказывается снова в вертикальной моде. Текст, который следует дальше, в свою очередь, преобразуется в строки ниже `\hbox`. Эта стопка строк выравнивается слева, что и объясняет тот факт, что знак • находится на краю левого поля. Чтобы избежать образования стопки строк, следует перевести Т_ЭX в горизонтальную моду до того, как он окажется в `\hbox`. Программка

```
\leavevmode\hbox to 1cm{${\bullet}\hfill} И тра-та-та ...
```

выдаст нам искомый результат:

• И тра-та-та, и тра-та-та

Будем опять следовать этой логике. Пусть Вы хотите сдвинуть абзац с помощью керна `\kern 2cm`. Если Вы напишете `\kern 2cm`, команда `\kern` вызовет вертикальный разрыв в 2 см. Следует сначала перевести Т_ЭX в горизонтальную моду и задать `\leavevmode\kern 2 cm....`

Но проще написать `\hskip 2cm...`. Есть и другие команды, которые вызывают переход в горизонтальную моду, например:

`\noindent\hbox to 1 cm{\$ \bullet \$\hfill}` И тра-та-та, ...

- И тра-та-та, и тра-та-та 374, 396, 421, 478, 491.

\left* Создает левые ограничители нужной по контексту величины. Неотделима от команды `\right`. Команды `\left` и `\right` обязательно должны сопровождаться каким-либо ограничителем или точкой (точкой обозначается *пустой ограничитель*, то есть, отсутствие ограничителя). Только в математической моде. Система уравнений задается командами `$$\left\{\matrix{...}\right.$$, а определитель матрицы — командами $$\left|\matrix{...}\right|$$.`

Действие пары `\left...\right` легко понять: $\TeX$ вычисляет высоту и ширину объекта, помещенного внутри `\left` и `\right`, и соответственно на это реагирует, модифицируя размеры ограничителей, которые следуют за `\left` и `\right`. Именно по этой причине эти две макроккоманды неразделимы. Пара `\left...\right` не заменяет команду `\big` и ее варианты. Что Вам больше по душе:

$$\left(\sum_{N=0}^{N=x^k} x_n\right) \quad \text{или} \quad \left(\sum_{N=0}^{N=x^k} x_n\right)?$$

Слева мы видим результат действия конструкции `\left...\right`, в то время как справа были использованы `\biggl(...\biggr)`. Любопытно, что команды `\left...\right` и их разновидности являются примером применения `\biggl(...\biggr)`. Можно посмотреть определения этих команд в формате plain.

178, 187-188, 206, 233, 348, 510.

\leftarrow Отношение $\leftarrow$. Только в математической моде. Еще более длинную стрелку можно получить с помощью команды `\longleftarrow`. Правый вариант получается через `\rightarrow` и `\longrightarrow`. Вертикальные стрелки получаются командами `\uparrow` и `\downarrow`. Можно изменять их размер с помощью команд `\big` или `\left...\right`.

268, 509.

\Leftarrow Отношение $\Leftarrow$. Только в математической моде. Более длинная стрелка получается с помощью `\Longleftarrow`. Правые варианты этих скобок получаются ... впрочем, сообразите сами!

268, 509.

\leftarrowfill Для получения стрелки, направленной влево и имеющей произвольную длину. Это бесконечный клей, который пишет стрелку. Используется внутри `\hbox`, в таблице или следует за командой `\break`, если Вы находитесь в `\vbox`:

- `\hbox to 2.5cm{\leftarrowfill}`  422.
- `\leftharpoonowdown` Отношение $\leftharpoonowdown$. Только в математической моде. Имеется также и команда `\rightharpoonowdown`, которая рисует $\rightharpoonowdown$. 509.
- `\leftharpoonoup` Отношение $\leftharpoonoup$. Только в математической моде. Имеется также `\rightharpoonoup`, которая рисует $\rightharpoonoup$. 509.
- `\lefthyphenmin*` Задаёт длину в символах наименьшего фрагмента начала переносимого слова. При изменении `\language` меняется и `\lefthyphenmin`. 324, 430, 529, 530.
- `\leftline` Представляет собой команду `\line(\hbox to \hsize)`, в которой присутствует клей `\hss` справа. Вы должны находиться в начале абзаца, иными словами, непосредственно за командой `\vskip` или пустой строкой, а то попадете прямо в переполненный `\hbox`! 124, 305, 309, 388, 418.
- `\leftrightharpoonow` Отношение $\leftrightharpoonow$. Только в математической моде. Более длинная двойная стрелка получается с помощью `\longleftrightharpoonow`: $\longleftrightarrow$. 509.
- `\Leftrightharpoonow` Отношение $\Leftrightharpoonow$. Только в математической моде. Более длинная двойная стрелка получается с помощью `\Longleftrightharpoonow`: $\longleftrightarrow$. 509.
- `\leftskip*` Пробел, который вставляется слева от каждой строки какого-либо абзаца. Команда `\leftskip=1cm` (знак = необязателен) создает иллюзию, что левое поле смещается вправо на 1 см, а команда `\leftskip=-5mm` — влево на 5 мм. Можно соединять `\leftskip=1cm` и `\parindent=-1cm`, например, при создании библиографического списка. Аналогичная команда для правого поля, естественно, `\rightskip`. Если надо, чтобы смещение поля осуществлялось только в пределах одного абзаца, задайте *локальный* порядок использования этой команды, написав:
- `{\leftskip 20pt ... \par}`.
- Очень важно перед тем, как закрыть группу, написать `\par`. Если Вы забудете это сделать, никакого смещения поля не произойдет.
- Чтобы задать только правое выравнивание, введите *эластичный* пробел в начале каждой строки (и аннулируйте пробел, который $\TeX$ автоматически ставит в конце абзаца):
- `\leftskip=0mm plus 5mm \parfillskip=0mm`

Разумеется, Вы можете изменять максимальное растяжение этого пробела.

123, 325, 379, 477, 489.

`\leq` (*lower or equal*) Отношение $\leq$. Только в математической моде. Допускается и более короткая запись `\le`.

380, 424, 508.

`\leqalignno` (После ‘q’ отсутствует ‘u’, двойное ‘n’). Для вертикального выравнивания при большом скоплении математических формул, с нумерацией слева. Подробнее см. описание `\eqalignno`.

229, 231, 428.

`\legno*` Проставляет слева от единичной формулы ее номер. Формула должна быть в выделенной математической моде: `$$...\legno...$$`. Следовательно, не действует при `\displaylines`. Подробнее см. `\eqno`.

223, 349, 422.

`\let*` Вводит еще одно имя для уже существующей макрокоманды, причем предыдущее имя этой макрокоманды остается в силе. Очень и очень часто используемая команда. Имена макрокоманд в $\TeX$ ’е выбраны так, чтобы их было легче запомнить. Например, `\longleftarrow` всем своим видом информирует о предполагаемом действии команды, и после непродолжительной тренировки можно запросто предвидеть результат той или иной команды. Но каким же длинным является это имя и как трудно его напечатать без ошибки! Макрокоманда `\let` позволяет избавиться от этого недостатка. Если, например, Вы предпочитаете говорить по-русски, то задайте:

`\let\длс=\longleftarrow`

(длс — “длинная левая стрелка”), и, начиная с этого момента, у Вас будет две макрокоманды, выполняющие одну работу: общепринятая и Ваша собственная. Обратите внимание на синтаксис: `\let`, затем Ваше сокращенное название, знак `=` и имя макрокоманды, которую Вы собираетесь переименовывать — и это все! Не путайте эту команду с `\def`, которая имеет в своем составе фигурные скобки.

245, 255, 328, 368, 369, 417, 442.

`\lfloor` Ограничитель $\lfloor$. Только в математической моде. Примеры:
`$$\lfloor x \rfloor = E(x)`(целая часть)
..... $\lfloor x \rfloor = E(x)$ (целая часть)

176, 425, 510.

`\lg` Функция логарифм (lg). Только в математической моде.

196, 427.

`\lgroup` Один из левых математических ограничителей. Только в математической моде. Соответствующий правый ограничитель — `\rgroup`:

	$\$\left\lgroup E(x)\right\rgroup$$ $(E(x))$ 181, 212, 425, 510.
<code>\lhook</code>	Отношение “левый крючок”: ($\hookleftarrow$). Только в математической моде. Имеется также команда <code>\rhook</code>, которая выдает ($\hookrightarrow$). Эти два отношения служат для составления (с помощью <code>\joinrel</code>) более сложных конструкций. 424.
<code>\lim</code>	Математический предел. Только в математической моде, само собой разумеется. Результат зависит от стиля (<code>\displaystyle</code> — это выделенный математический стиль): $\$\lim_{x\rightarrow 0}\{\sin x\over x\}=1$ \lim_{x\rightarrow 0} \frac{\sin x}{x} = 1$
	$\$\displaystyle \lim_{x\rightarrow 0}\{\sin x\over x\}=1$ \lim_{x\rightarrow 0} \frac{\sin x}{x} = 1 $
	Можно изменять поведение верхнего и нижнего индексов, используя варианты <code>\limits</code> или <code>\nolimits</code>. 196, 198, 427.
<code>\liminf</code>	Оператор <code>\liminf</code>. Только в математической моде: $\$\liminf_{x\rightarrow 0} f(x)$ \liminf_{x\rightarrow 0} f(x)$
	Некоторые предпочитают оператор <code>\lim</code> с таким определением: $\def\liminf{\mathop{\underline{\rm lim}}}$ $\$\liminf_{x\rightarrow 0} f(x)$ \lim_{x\rightarrow 0} f(x)$
	$\displaystyle\liminf_{x\rightarrow 0} f(x)$ \lim_{x\rightarrow 0} f(x) $
	Определение <code>\limsup</code> аналогично. Достаточно заменить <code>\underline</code> на <code>\overline</code> в предыдущем определении <code>\liminf</code>: $\def\limsup{\mathop{\overline{\rm lim}}}$ $\$\limsup_{x\rightarrow 0} f(x)$ \overline{\lim}_{x\rightarrow 0} f(x)$
	$\displaystyle\limsup_{x\rightarrow 0} f(x)$ \overline{\lim}_{x\rightarrow 0} f(x) $
	196, 198, 214, 427.
<code>\limits*</code>	Меняет поведение верхних и нижних индексов у операторов. Должна сопровождаться оператором (макрокомандой, составленной с помощью <code>\mathop</code>). Только в математической моде. Введем определение: $\def\Som{\mathop{\bf S}}$ Формула <code>\Som_{i=0}^n a_i</code> выглядит по-разному в зависимости от того, печатается она в тексте (между двумя простыми долларами, то есть в <code>\textstyle</code>), или на отдельной строке (между двумя двойными долларами, в <code>\displaystyle</code>): $\$\Som_{i=0}^n a_i$ \mathbf{S}_{i=0}^{i=n} a_i$
	$\$\displaystyle\Som_{i=0}^n a_i$ \mathbf{S}_{i=0}^{i=n} a_i $

Если макрокоманда `\limits` следует за оператором, индексы всегда размещаются над и под этим оператором, как в `\displaystyle`:

$\$ \backslash \mathrm{Tom}_{\{i=0\}^{\sim \{i=n\}} a_i \$$	$\prod_{i=0}^{i=n} \mathbf{T} a_i$
$\$ \backslash \mathrm{displaystyle} \backslash \mathrm{Tom}_{\{i=0\}^{\sim \{i=n\}} a_i \$$	$\prod_{i=0}^{i=n} \mathbf{T} a_i$

$$\text{\textbackslash Tom\nolimits_}\{i=0\}^{\{i=n\}}a_i \quad \dots \quad \mathbf{T}_{i=0}^{i=n} a_i$$
$$\displaystyle\prod_{i=0}^na_i \dots\dots\dots \prod_{i=0}^na_i$$

175, 192, 348, 425, 516.

`\line` Создает `\hbox to \hsize`, т.е., одну строку. Если эта команда не первая в абзаце, `TeX` жалуется на `overfull \hboxes!` Абзац начинается после пустой строки или после команд `\par` или `\vskip`. Не забудьте в фигурных скобках после `\line` поставить бесконечно растяжимый клей (`\hfill`), иначе материал в скобках будет стремиться занять всю ширину строки и пробелы будут слишком растянутыми. Вот что получится, если ввести `\line{Это неудачная кодировка!}`:

Добавим теперь с обеих сторон бесконечный клей:

Теперь получим:

Обратили внимание, что пробелы на строке распределены в отношении 1 : 2 (один клей против двух)? Такой же прием применяется и с вертикальным клеем `\vfill`. Не забывайте в этом случае `\eject` (завершающую страницу), а то останетесь с пустой страницей.

121, 323, 376, 377, 413.

`\lineskip*` Пробел, который $\TeX$ ставит между двумя боксами, находясь в вертикальной моде, когда он не может соблюдать интерлиньяж. См. следующие макрокоманды.

97, 128, 231, 325, 333, 414, 416-417.

`\lineskiplimit*` Когда $\TeX$ вертикально настраивает друг на друга боксы, он пробует сначала расположить их так, чтобы базовые линии находились на расстоянии `\baselineskip`. Затем он проверяет пробел, разделяющий два бокса. Если этот пробел слишком маленький (т.е., меньше `\lineskiplimit`), он аннулирует предыдущую операцию и раздвигает боксы, вставляя пробелы, высота которых равна `\lineskip`. Примеры см. в `\normalbaselines`.

97, 128, 231, 325, 333, 414, 416-417.

`\ll` Отношение $\ll$. Правый вариант ($\gg$) называется, естественно, `\gg`. Только в математической моде.

503, 508.

`\llap` Макрокоманда, которая записывает свой аргумент на предыдущем тексте. Иными словами, эта команда позволяет вносить запись слева от курсора, не переходя на это новое место. Если набрать что-нибудь вроде `000\llap{$| | | |$}` МММ, $\TeX$ сначала изобразит “ООО МММ”, а затем “| | | |” *поверх* “ООО”, а в результате получится “ОО| | | МММ”. Это используется очень часто. Аналогичная команда, позволяющая вносить запись справа от курсора, не передвигаясь на новое место, называется `\rlap`. Две этих макрокоманды имеют по одной переменной, следовательно, сразу после командного слова должна следовать группа. Для примера попробуем изобразить следующее отношение:

```
\def\toto{\mathrel{\vbox{\hsize=9pt\hrule\kern1pt
\centerline{$\circ$}\kern.6pt\hrule}}}
$P\toto Q$ .....  $P \overline{\phantom{Q}}$ 
```

Можно получить и противоположное отношение, перечеркнув `\toto` косой чертой. Для этого нужно собрать вместе `$\toto$\llap{/}\kern2pt`, превратив их в одно отношение. Некоторые усовершенствования позволят получить окончательный вариант макрокоманды:

```
\def\nototo{\setbox1=\hbox{$\toto$\llap{/}\raise1pt\hbox{\big/}
\kern 2pt}} \mathrel{\box1}}
$P\nototo Q$ .....  $P \not\overline{\phantom{Q}}$ 
```

Для записи на левом поле, как здесь, задайте:

```
\llap{$\diamond$\quad}Начало строки...
```

- ◇ Начало строки
Для записи на правом поле требуется больше усилий, так как неизвестно заранее, где будет курсор в конце абзаца. Переводим курсор в конец строки с помощью `\hfill\break`, затем пишем справа от курсора, не

передвигаясь туда. Поэтому введите `бrrrrr`, `бrrrrr` ... конец строки.
`\hfill\break\rlap{\quad$\diamond$}`, и получите:

`бrrrrr`, `бrrrrr` конец строки. $\diamond$

Используя `\llap`, Вы находитесь в `\hbox`, так что не забывайте, если надо, переходить в математическую моду: `\llap{$(\Sigma)$}`.

102, 226, 404, 418, 355, 448, 486-487, 493.

`\lmoustache` Один из левых математических ограничителей. Только в математической моде. Соответствующий правый ограничитель — `\rmoustache`:

`$\left\lmoustache` формула `\right\rmoustache$` $\int$ формула $\int$
 181, 425, 510.

`\ln` Оператор `ln`. Для задания натуральных логарифмов. Только в математической моде.

196, 186, 222.

`\not` Оператор $\neg$. Только в математической моде.

427, 510.

`\log` Оператор `log`. Только в математической моде.

196, 204, 427.

`\long*` Префикс, который надо ставить перед `\def` при определении макрокоманды, длина аргументов которой более одного абзаца. Например:

```
\long\def\ctitre#1{\vbox{\leftskip=0pt plus 1fil
\rightskip=0pt plus 1fil \parfillskip=0pt#1}}
```

позволяет получать заголовок из нескольких строк, причем все строки будут отцентрированы. Итак, пишем следующую программку:

```
\ctitre{М. В. Лисина\par\medskip
{PLAIN \TeX\par\smallskip ОСНОВНЫЕ КОМАНДЫ\par
И КАТАЛОГ КОМАНД}}
```

которая выдает заголовок:

М. В. Лисина

PLAIN $\TeX$

ОСНОВНЫЕ КОМАНДЫ
 И КАТАЛОГ КОМАНД

Без префикса `\long` $\TeX$ не воспринял бы `#1`, содержащий `\par`. Ограничения, накладываемые на параметры какой-либо макрокоманды (не более одного абзаца), необходимы в целях предосторожности: если макрокоманда написана неправильно, убытки понесет только текущий абзац. Действительно, если $\TeX$ встретит конец абзаца или `\par`, он пошлет Вам сообщение “runaway argument?” и прекратит верстку неправильного абзаца. Однако любое правило существует для того, чтобы его

нарушали. Будьте благоразумны: печатая `\long\def\toto#1`, Вы создаете чудовище, способное за один раз проглотить 250 страниц книги ..., поэтому используйте префикс `\long`, только если нет никакой возможности сделать это другим способом. В целях безопасности имеется еще одна команда, которая позволяет (или не позволяет) какой-либо макрокоманде иметь аргументами другие макрокоманды: это `\outer`.

244, 249, 326, 393, 441, 444, 449.

`\longleftarrow` Отношение $\longleftarrow$. Только в математической моде. Правый вариант называется `\longrightarrow`.

424, 509.

`\longlefttrightarrow` Отношение $\longleftrightarrow$. Только в математической моде.

424, 509.

`\Longlefttrightarrow` Отношение $\Leftrightarrow$. Только в математической моде.

424, 509.

`\longmapsto` Отношение $\longmapsto$. Только в математической моде.

424, 509.

`\longrightarrow` Отношение $\longrightarrow$. Только в математической моде. Левый вариант называется `\longleftarrow`.

424, 509.

`\Longrightarrow` Отношение $\Longrightarrow$. Только в математической моде. А как задается знак $\Longleftarrow$?

424, 509.

`\loop` Команда цикла. В формате `plain` имеется такая конструкция `\loop ... \repeat`, которая работает следующим образом. Вы говорите `\loop  $\alpha$  \if ... $\beta$  \repeat`, где α и β — любые последовательности команд, а `\if` — любой вид условия (без `\fi`). `TeX` выполняет α , затем, если условие выполняется, выполняет β и повторяет процесс, начиная с α . Если же условие не выполняется, цикл заканчивается.

258, 417, 439, 455, 488.

`\looseness*` Дает возможность изменять число строк в абзаце. Если поместить в абзаце команду `\looseness=1`, `TeX` попытается сформировать из него на одну строку больше, чем при обычной верстке (учитывая всевозможные допуски для каждой строки). Можно легко догадаться, что представляют собой команды `\looseness=2` или `\looseness=-1`. Естественно, подобная команда эффективна в том случае, если абзац достаточно насыщен. Команду можно ставить в любом месте абзаца, и это понятно, если Вы вспомните, что `TeX` прочитывает абзац целиком до того, как

разбить его на строки. К другим абзацам это мероприятие не будет иметь никакого отношения.

127, 133, 324, 406, 413.

`\lor` Бинарное отношение $\vee$. Только в математической моде. Например:

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$$

161, 427, 510.

`\lower*` Опускает бокс на указанную величину. Можно обойтись и без этой макрокоманды, поскольку то же самое действие осуществляется с помощью `\raise`: команда `\raise -2pt\hbox{...}` делает то же самое, что и `\lower 2pt\hbox{...}`. Понятно, что опускать можно любой бокс, а не только `\hbox`. Макрокоманды `\raise` и `\lower` заново “продырявливают” боксы, если эти боксы рассматривать как бусины.

82, 100, 182, 215, 339, 347.

`\lowercase*` Переводит следующие за командой буквы в строчные. Команда предназначена для супер-TeXпертов. Действительно, она не нужна, если Вы печатаете свой собственный текст: Вы и так знаете, что и как нужно делать. Поэтому есть смысл ее использовать только для внесения изменений в результаты действия других макрокоманд. Но это уже требует знания многих тонкостей (например, команды `\expandafter`, даже комментарии к которой здесь совершенно излишни):

```
\def\molieret{\sl MARКИЗА, Ваши ПРЕКРАСНЫЕ глаза СУЛЯТ
мне СМЕРТЬ от любви}
\molieret: МАРКИЗА, Ваши ПРЕКРАСНЫЕ глаза СУЛЯТ мне
СМЕРТЬ от любви
\lowercase{\molieret}: МАРКИЗА, Ваши ПРЕКРАСНЫЕ гла-
за СУЛЯТ мне СМЕРТЬ от любви
\lowercase\expandafter{\molieret}: маркиза, ваши прекрасные
глаза сулят мне смерть от любви
```

Обратите внимание на фигурные скобки, которые окружают команду `\mol`. Если Вы их забудете, у Вас начнутся неприятности. См. также команду `\lccode`.

51, 255, 331, 368, 409.

`\lq` (*left quote*) Открывающая кавычка. Вам эта команда, можно сказать, не нужна, если только, конечно, клавиша “открывающая кавычка” на Вашей клавиатуре не перестала работать.

5, 62, 416, 436, 464.

m

`\mag*` Примитив, с помощью которого TeX управляет увеличением выходного документа, целый параметр. См. `\magnification`.

75, 321, 324, 413.

\magnification Для увеличения документа. Команда `\magnification=1200` в начале документа увеличивает его на 20%, а 1000 даст соотношение 1:1. Выполнять `\magnification` можно только один раз для одного документа, и ни в коем случае не менять его во время работы. Иногда можно случайно изменить коэффициент увеличения, не догадываясь об этом. Если Вы напишете:

```
\hsize==11truecm \vsize=18truecm
...
\magnification
```

(размеры, после которых стоит `true`, не модифицируются), $\TeX$, увидев `true` и не увидев `\magnification`, выдаст в этом месте его значение по умолчанию, т.е. 1000. Конфликт обеспечен! Не пишите команд вроде `\hsize=12cm \magnification=1200 \hsize=10cm`, поскольку $\TeX$ отказывается подчиняться таким командам. Следовательно, $\TeX$ должен встретить команду `\magnification` *до первого указания на размер*. Напомним также, что если вы используете `true`, окончательная верстка документа зависит от выбранного значения `\magnification`.

22, 73-74, [430](#), [473-474](#).

\magstep (С целым числом от 0 до 6). Синтаксис: `\magnification=\magstep1`. Имеются варианты `\magstep0=1000`, `\magstep1=1200`, `\magstep2=1440`, и т.д. Правило простое: увеличение целого числа на 1 добавляет 20% к предыдущему `\magstep`.

22, 74, [414](#), [473](#).

\magstephalf По умолчанию `\magstephalf=1095`. Две последовательные команды `\magnification=\magstephalf` эквивалентны `\magstep1`.

22, [414](#), [473](#).

\makefootline Одна из команд программ вывода. Помещает `\footline` в соответствующем месте.

305-306, [430](#).

\makeheadline Одна из команд программ вывода. Помещает `\headline` в соответствующем месте.

304-306, [430](#).

\mapsto Отношение $\mapsto$. Только в математической моде. Имеется также команда `\longmapsto` ($\longmapsto$), используемая в выделенной математической моде.

156, [424](#), 509.

\mapstochar Маленькая вертикальная черточка, необходимая для построения отношения `\mapsto`: ($\mapsto$). Ее используют для получения с помощью `\joinrel` и других знаков отношений.

424.

\mark* Для “маркировки” части текста, на которую нужно будет в дальнейшем сделать ссылку. Например, в нашем тексте можно использовать макрокоманду — назовем ее просто `\macro` — которая печатает имя макрокоманды на левом поле. Вот, в общих чертах, как она выглядит:

```
\def\macro#1{\setbox3=\hbox{\tt\char'\#1}
\vskip 3mm plus 1mm minus 1mm \hskip -20mm
\ifdim\wd3<17mm\hbox to 20mm{\unhbox3\hfill}
\else\unhbox3 : \fi\mark{#1}}
```

Параметры отделяются от командного слова знаком `#`. Не пытайтесь маркировать текст, например, так: `\mark{Paskal}`. Это не работает. Рубрики следует начинать так: `\macro{mark}` Для “маркировки”.... Все имена рубрик промаркированы (последняя строка макрокоманды после `\fi`). Когда $\TeX$ составляет страницу, промаркированные имена (или участки текста) можно получить с помощью трех следующих команд:

- `\firstmark`, первый маркер на текущей странице;
- `\botmark`, последний маркер на текущей странице;
- `\topmark`, значение `\botmark` предыдущей страницы.

Предположим, например, что Ваш текст содержит четыре метки и что метка `\mark{\alpha}` появится на странице 2, метки `\mark{\beta}` и `\mark{\gamma}` появятся на странице 4, а метка `\mark{\delta}` — на странице 5:

page	<code>\firstmark</code>	<code>\botmark</code>	<code>\topmark</code>
1	null	null	null
2	α	α	null
3	α	α	α
4	β	γ	α
5	δ	δ	γ
6	δ	δ	δ

Внимание: совсем не все равно, где в определении помещать `\mark`. Если написать `\def\macro#1{\mark{#1}...}`: значения `\firstmark` и `\botmark` окажутся неправильными. Надо, чтобы текст, который нужно промаркировать, уже был занесен в файл до того, как делается маркировка.

117, 190, 257, 307-313, [332](#), 487, 529.

\mathaccent* Прimitив низшего уровня, с помощью которого определяются акценты в математике. Например, `\def\widehat{\mathaccent"362}`: За командой следует математический код, так что $\TeX$ знает номер семейства и позиции акцентирующего символа.

189, 205, [348](#), [425](#), [516](#).

\mathbin* Создание бинарного оператора. Только в математической моде. Предположим, что Вы используете бинарную операцию τ :


```

\def\loi{\mathbin{\tau}}
$(a,b)\loi(a',b')=
(a+\rho(b)a',b+b')$. . . . . $(a,b)\tau(a',b') = (a + \rho(b)a', b + b')
$(a,b)\tau(a',b')=
(a+\rho(b)a',b+b')$. . . . . $(a,b)\tau(a',b') = (a + \rho(b)a', b + b')

```

Сделав τ бинарным оператором, можно быть уверенным, что этот оператор окружен правильными пробелами.

187, 348, 427.

\mathchar* Эта команда позволяет вызывать математический символ по его номеру. Номер состоит из номера класса, номера семейства и номера позиции. Команда является математическим эквивалентом команды `\char`. Например, команда `\mathchar"1ABC` задает символ класса 1 (большой оператор) семейства 10 ("A") позиции "BC". Специальные математические символы $\TeX$ 'а определены так: `\def\sum{\mathchar"1350}`. Но лучше использовать команду `\mathchardef`, которая относится к `\mathchar`, как `\chardef` относится к `\char` (`\mathchardef\sum="1350` — формат `plain` содержит десятки подобных определений).

Замечание: классы описаны в $\TeX$ book: пробел, который разделяет два математических символа, зависит от классов этих двух символов.

187, 345.

\mathchardef* Выполняет ту же работу, что и предыдущая макрокоманда.

187, 237, 254, 255, 322, 328, 345, 398, 463.

\mathchoice* Для создания макрокоманды, действие которой зависит от стиля (`\displaystyle`, `\textstyle`, `\scriptstyle`, `\scriptscriptstyle`):

```

\def\toto{\mathchoice{1}{2}{3}{4}}
$x+\toto$ . . . . . $x + 2$
$\displaystyle\toto +x_{\toto}^{\a_{\toto}}$ . . . . . $1 + x_3^{a_4}$

```

Не злоупотребляйте этой макрокомандой, поскольку она переполняет память и замедляет работу $\TeX$ 'а, так как ведет параллельно вычисления всех четырех опций и выбирает нужную только в последний момент. Четыре `\mathchoice` в одной формуле, следовательно, требуют вычисления $4 \times 4 \times 4 \times 4$ опций ...

182, 190, 348.

\mathclose* $\TeX$ связывает понятие "класс" (см. `\mathchar`) не только с отдельными символами, но и с целыми подформулами. Класс символам или подформулам можно задать командами `\mathord`, `\mathbin`, `\mathopen`, `\matclose`, `\mathrel` и `\mathpunct`. Команда `\mathclose` используется для создания закрывающего ограничителя.

187, 347, 384, 425.

\mathcode* Каждый математический символ описывается математическим кодом. При шестнадцатиричной записи этого кода первая цифра представляет

собой номер класса (от 0 до 7), вторая цифра — номер семейства, а третья и четвертая — номер позиции. Так, например, в формате `plain` команда `\sum` определена как математический символ "1350, то есть, большой оператор (класс 1), находящийся в позиции "50 семейства 3.

163, 186-188, 254, 322, 345, 381, 388, 408-409.

`\mathinner*` Восьмой математический класс, который обычно не используется для отдельных символов; дроби и конструкции `\left ... \right` рассматриваются как “внутренние подформулы”, что означает, что при некоторых обстоятельствах они будут окружены дополнительными пробелами.

187, 206, 237, 348, 425.

`\mathop*` См. `\mathclose`. Для создания *оператора*: пробелы до и после этого оператора адаптируются к контексту, верхний и нижний индексы печатаются сбоку в `\textstyle` и “над-под” в `\displaystyle`. Например:

```
\def\min{\mathop{\rm min}}
```

Для модификации поведения индексов используются команды `\limits` и `\nolimits`. Для создания нового имени функции, пишется, например:

```
\def\airy{\mathop{\rm Airy}\nolimits}
```

187, 214, 348, 386, 427.

`\mathopen*` См. `\mathclose`. Для создания открывающих ограничителей.

187, 348, 384, 425.

`\mathord*` См. `\mathclose`. Для создания ординарного символа (*класс 1*). Например, `\circ` является отношением. Если написать `$u\circ v$`, это дает $u \circ v$. Некоторые считают, что вокруг кружка слишком широкие пробелы. Это можно исправить, превратив отношение в ординарный символ: `$\mathord{\circ} v$` дает $u \circ v$. Этот же результат можно получить проще: `$\circ v$`.

109-110, 187, 348.

`\mathpalette` Эта макрокоманда полезна для конструкций `\mathchoice`. Например, команда `\mathpalette\alpha{xyz}` раскрывается в следующую последовательность команд:

```
\mathchoice{\alpha\displaystyle{xyz}}
...
{\alpha\scriptscriptstyle{xyz}}
```

См. описание `\mathchoice`.

183, 426.

`\mathpunct*` См. `\mathclose`. Для создания знаков пунктуации в формулах.

187, 348.

`\mathrel*` См. `\mathclose`. Для конструирования *отношения*. Эта макрокоманда добавляет пробелы с обеих сторон символа (или символов), который объявлен отношением:

`$a?b$, $a\mathrel{?}b$ $a?b$, $a ? b$
$X\alpha\beta Y$, $X\mathrel{\{\alpha\beta\}}Y$
..... $X\alpha\beta Y$, $X \alpha\beta Y$`

Макрокоманда `\bigm`, которая добавляет пробелы вокруг ограничителя, использует макрокоманду `\mathrel`.

187, 348, 425-427.

`\mathstrut` Вертикальная невидимая черта (`width=0pt`), имеющая высоту и глубину круглой скобки. Хотя она и называется `\math...`, ею можно пользоваться и за пределами математики. Очень часто используется для измерения переполнения (см. `\overline`). Не смешивать `\mathstrut` с другой невидимой чертой, `\strut`. Эта последняя черта больше. Покажем их видимыми (`\mathstrut` окружена круглыми скобками):

`(\ \hbox{\mathstrut\vrule}\ ) \hbox{\strut\vrule} ..... (|)|`
159, 216, 426.

`\mathsurround*` Параметр, равный размеру пробелов, который $\TeX$ вставляет до и после каждой математической формулы. В формате `plain` его значение равно 0 pt, так что, если Вы хотите увеличить пробелы вокруг формул, измените этот параметр сами.

120, 196, 325, 375, 384, 419, 465, 520.

`\matrix` Конструирует матрицу, не ограниченную круглыми скобками. Только в математической моде. Очень легко использовать:

<code>\$\$A=\matrix{</code>	$A = \begin{matrix} a & b_1 + \dots + b_n \\ \alpha & \beta \end{matrix}$
<code>a \quad &b_1+\cdots+b_n \cr</code>	
<code>\alpha &\beta \cr }\$\$</code>	

- Синтаксис, как в `\halign`, но без преамбулы. Знак амперсанда служит для разделения колонок. Не забывайте завершать строки матрицы (в том числе и последнюю) командой `\cr`.

- Внутри матрицы включена математическая мода.

- Посмотрите на β по отношению к $b_1 + \dots + b_n$: она автоматически отцентрирована. Чтобы передвинуть элемент в колонке, пользуйтесь клеем `\hfill`. Для сдвига элемента β влево используется команда `\alpha&\beta\hfill\cr`, а вправо — команда `\alpha&\hfill\beta\cr`. Сама матрица также является центрированным боксом (то есть бусиной, просверленной посередине). Именно поэтому `A=` находится точно между двумя строками матрицы.

- Чтобы вертикально раздвинуть строки матрицы, введите между ними (т.е., сразу после `\cr`) `\noalign{\vskip 2mm}`. Так можно изменять величину раздвигания или вставлять то, что Вам хочется, например, текст. Обратите внимание: команда `\noalign{...текст...}`

“просверлит” матрицу напротив левого поля, в то время как команда `\noalign{\hbox{...}}` вставит текст в отцентрированную матрицу.

- Чтобы раздвинуть строки матрицы, вставьте `\noalign` между двумя строками... Не пытайтесь использовать `\openup` (которая модифицирует `\baselineskip`), поскольку это не работает. В описании команды `\matrix` указано, что она вызывает `\normalbaselines`, которая восстанавливает нормальный интерлиньяж:

`\baselineskip=\normalbaselineskip.`

Следовательно, изменять интерлиньяж следует так:

`$$\normalbaselineskip=20pt\matrix{...}$$`

Но это слишком сложно для простого наборщика, который не знает наизусть все 900 макрокоманд формата plain.

- Матрица в круглых скобках задается командой `\pmatrix` (‘p’ обозначает *parentheses*). Вообще говоря, конструкция `\left ... \right` позволяет окружать матрицу любыми ограничителями. Детерминант матрицы печатается так:

`$$\left|\matrix{...}\right|$$`

Внимание, макрокоманда `\det` пишет только “det” романским шрифтом в тексте. Система уравнений кодируется так:

`$$\left\{\matrix{...}\right.\legno(\Sigma)$$`

Обратите внимание на *нулевой ограничитель* (`\right.`) и “нумерацию слева” `\legno(\Sigma)`, которая дает (Σ) .

212, 218, 387, 427.

`\max`

Оператор max (строится с помощью `\mathop`).

`$M=\max_{i\in I}x_i$` $M = \max_{i \in I} x_i$
`$\displaystyle M=\max_{i\in I}x_i$` $M = \max_{i \in I} x_i$

Для модификации поведения индексов см. `\limits`.

196, 205, 427.

`\maxdeadczcles*` Целый параметр, который в формате plain равен 25. См. команду `\deadczcles`.

303, 324, 413.

`\maxdepth*` Параметр, указывающий Т_ЕX’у поднять нижний бокс на странице, если этот бокс имеет слишком большую глубину, чтобы глубина создаваемой страницы не превысила указанное значение (см. `\boxmaxdepth`).

137-138, 148-150, 152, 304, 312, 313, 325, 413, 470, 486.

`\maxdimen` Максимальный разрешенный размер в Т_ЕX’е. В формате plain равен 16383.99999 pt.

73, 224, 312-313, 412.

- `\meaning*` Выводит на Ваш терминал информацию о символе, следующем за командой. Например, обычно `\meaning A` выдает `'the letter A'`, а после `'\def\A#1B{\C}'` команда `\meaning A` выдаст `'macro:1B->\C'`.
253-255, 398, 449.
- `\medbreak` Комбинирует вертикальный пропуск `\medskip` и `\penalty -100` (чтобы облегчить верстку страниц). В формате `plain` имеются также команды `\bigbreak` и `\smallbreak`.
135, 138, 418, 421, 490, 493.
- `\medmuskip*` Математический клей средней величины:
$$\backslash\medmuskip=4\mu\text{ plus }2\mu\text{ minus }4\mu$$
203, 325, 413, 520.
- `\medskip` Средний вертикальный пропуск. См. выше. Имеется также `\smallskip` и `\bigskip`.
89, 99, 126, 133, 135, 417, 480-482.
- `\medskipamount` Регистр, содержащий значение вертикального пропуска `\medskip`. В формате `plain` `\medskipamount` равен `6 pt plus 2 pt minus 2pt`.
414, 417-418, 421, 477.
- `\message*` Если Вы хотите знать, что делает `TeX`, попросите его послать сообщение. Например, сделайте
$$\backslash\def\toto#1{\message{\#1}...}$$
и на Ваш экран будет посылаться аргумент `#1` каждый раз, когда будет срабатывать команда `\toto`.
256, 257-258, 271, 331, 368, 390, 408, 421, 489.
- `\mid` Вертикальная черта, рассматриваемая как отношение (с пробелом до и после). Только в математической моде.
$$\$I=\{t \mid 0 < t \leq 1\}\$ \dots\dots\dots I = \{t \mid 0 < t \leq 1\}$$
$$\$I=\{t \mid 0 < t \leq 1\}\$ \dots\dots\dots I = \{t \mid 0 < t \leq 1\}$$
210, 508.
- `\midinsert` Позволяет вставлять материал наилучшим образом. Синтаксис:
$$\backslash\midinsert\dots\backslash\endinsert$$
Если хватает места, вставка текста между `\midinsert` и `\endinsert` выполняется в предусмотренном месте. Иначе это делается на следующей странице. “Пропавшее” место заполняется текстом, который следует за командой `\endinsert`. На месте многоточия Вы размещаете то, что хотите. Эта макрокоманда идеальна для вставки рисунка:
$$\backslash\midinsert$$
$$\backslash\glue 6cm\centerline{\it P\!r\!i\!c.} 5\backslash\skip 5mm$$
$$\backslash\endinsert$$

Команда `\vglue` обязательна, поскольку `\vskip` на самом верху страницы исчезает. Если Вы хотите использовать `\vskip`, пишите

`\midinsert\null\vskip...\endinsert.`

В силу концепции Т_ЕX’a, эта макрокоманда действует только на уровне страницы: ею нельзя пользоваться внутри бокса. Если надо, чтобы рисунок в любом случае находился в начале следующей страницы, используйте `\topinsert...\endinsert.`

141, 404, 429.

`\min` Оператор `min`. Только в математической моде. Более подробно см. в `\max` `\mathop`.

196, 206, 427.

`minus` Ключевое слово, которое вместе с ключевым словом `plus` используется при задании эластичного клея.

89, 321, 400.

`\mit` Задание математического курсивного шрифта (только в математической моде).

199, 416, 502, 507.

`\mkern*` Горизонтальный керн в математической моде. В математической моде можно использовать и `\kern` или `\hskip`. Но при создании макрокоманды, которая работает с математическим объектом, лучше использовать `\mkern`, поскольку такие пробелы автоматически адаптируются к изменениям стиля. Пример можно посмотреть в команде `\check`. `\mkern` выражается в единицах `mu` (*math unit* — математические единицы). Пробел (`\space` или `\`) равен примерно `6mu`, а самый маленький пробел, определенный в математической моде — это мини-пробел, равный `3mu`. Нам его определение послужит примером синтаксиса:

`\def\,{\mkern 3mu}`

`\mkern` делает горизонтальный пробел, вертикального аналога не имеется.

203, 322, 515.

`mm` Миллиметр, одна из единиц измерения в Т_ЕX’e.

71, 321.

`\models` Отношение $\models$. Только в математической моде.

424, 508.

`\month*` Порядковый номер текущего месяца года. Синтаксис: `\the\month` или `\number\month`. См. макрокоманду `\date`.

324, 413, 476.

`\moveleft*` Переместить бокс влево. Для этого надо находиться в вертикальной моде:

<code>\moveleft 5mm\ vbox{...}</code>		100, 334 .
<code>\moveright*</code>	Переместить бокс вправо. См. выше.	100, 264 , 334 .
<code>\mp</code>	Бинарное отношение $\mp$. Только в математической моде.	161 , 508.
<code>\mskip*</code>	Пробел в математической моде. Используйте только единицы <code>mu</code> (математические): <code>\mskip 3mu plus 1mu minus 2mu</code> Эта команда используется, когда пишут макроккоманды, относящиеся к математическим объектам, поскольку значение единицы <code>mu</code> (без обратной косой черты) зависит от действующего стиля.	203, 346 , 515.
<code>mu</code>	Математическая единица измерения в $\TeX$ 'е, зависит от выбранного стиля.	203, 321, 514.
<code>\mu</code>	Греческая буква μ . Только в математической моде. Не путайте с единицей математической длины <code>mu</code> , которая рассматривалась выше.	197 , 506.
<code>\multiply*</code>	Умножение на целое число. Примеры: <code>\multiply\count123 by 2, \multiply\dimen7 by 6.</code> Для сложения (или вычитания) используется <code>\advance</code> , а для деления — <code>\divide</code> .	143 , 259 , 327 , 414, 460 , 468 .
<code>\multispan</code>	(Должна сопровождаться целым числом, большим или равным 1). Позволяет объединять несколько колонок <code>\halign</code> . В терминах журналистики <code>\multispan5</code> означает “пять колонок за одну”. Целое представляет число колонок для объединения. Эта команда должна обязательно фигурировать в начале колонки: <code>\multispan2...&...&\multispan6...&...</code> Эта <code>\multispan n</code> заменила $(n - 1)$ амперсандов <code>&</code> . <code>\multispan1</code> является синонимом <code>\omit</code> . Инструкция <code>\multispan3\hrulefill</code> проводит горизонтальную черту в трех колонках. Не ставьте пробел между командами <code>\multispan3</code> и <code>\hrulefill</code> , поскольку это введет паразитный пробел.	289, 293, 397 , 419 .
<code>\muskip*</code>	Регистры <code>\muskip0 ... \muskip255</code> используются для хранения математического клея.	143, 203, 322 , 328 .

`\mskipdef*` Эффективно создает новые параметры математического клея.
145, 255, [328](#).

n

`\nabla` Знак ∇ . Только в математической моде.
507.

`\narrower` Перемещает левое и правое поля, увеличивая `\leftskip` и `\rightskip` на `\parindent`. Используется для длинных цитат:

Переехавши границу, русский культурный человек становится необыкновенно деятельным. Всю жизнь он слыл фатюем, фетиком, фалалеем; теперь он во что бы то ни стало хочет доказать, что он совсем не фатюй, и если являлся таковым в своем отечестве, то или потому только, что его “заела среда”, или потому, что это было согласно с видами начальства.⁵

`{\parindent=1cm\narrower\sl...\par}`

Если цитата длинная, действие команды `\narrower` ограничено группой `{\narrower...\par}` или `\begingroup\narrower...\par\endgroup`. Перед закрытием группы необходимо писать `\par`, иначе поля не переместятся! Эти рекомендации действительны для всех манипуляций с полями (`\leftskip`, `\hangindent`, `\item`, `\itemitem`). Понятно, что можно ввести `\par` пустой строкой (так сказать, двумя последовательными CR).

124, [406](#), [420](#).

`\natural` Знак бекар $\natural$. Только в математической моде.
507.

`\ne` Отношение $\neq$. Только в математической моде.
12, 57, [162](#), 380, [427](#), 510.

`\nearrow` (*north east arrow*). Отношение $\nearrow$. Только в математической моде.
509.

`\neg` Математический символ $\neg$. Конечно, только в математической моде.
507.

`\negthinspace` Отрицательный тонкий пробел (кern размером -0.16667 em).
395, [417](#).

`\neq` Еще одно имя для $\neq$. Только в математической моде.
380, [424](#), 510.

⁵ М. Салтыков-Щедрин, *За рубежом*.

- \newbox** Т_EX предлагает 256 боксов, `\box0, ..., \box256`. Чтобы приватизировать бокс, т.е., чтобы получить такой бокс, который зарезервирован только для Вашего использования и который никакая другая макрокоманда не сможет использовать без Вашего разрешения, введите `\newbox\toto`, а чтобы поместить что-либо в бокс `\toto`, который Вы хотите приватизировать, введите `\setbox` вместе с `\toto`:
- `\setbox\toto=\hbox{...}`
- Чтобы написать содержимое бокса `\toto` на страницу, следует писать `\box\toto`, или `\copy\toto`, если необходимо сохранить содержимое этого бокса, или `\unhbox\toto`, `\unvbox\toto` и так далее.
- 147, 411, [412](#), [419](#), [462](#), [487](#).
- \newcount** Для приватизации счетчика.
- 147, [259](#), 411, [412](#), [414](#), [489](#).
- \newdimen** Команда `\newdimen\toto` приватизирует регистр размера из 256 регистров `\dimen`, которыми располагает Т_EX. Отныне “номер” приватизированного регистра будет `\toto`.
- 147, 411, [412](#), [414](#), [486](#).
- \newfam** Все команды шрифтов в математической моде обязательно используют понятие семейства. Подробно см. в `\mathchardef`. Команда `\newfam`, аналогично команде `\newbox`, в математической моде присваивает символическое имя новому семейству шрифтов.
- 147, 190, 411, [412](#), [416](#).
- \newhelp** Команда для создания самодельных пояснений к сообщениям об ошибках. Можно, например, сказать
- `\newhelp\helpout{Мое сообщение.},`
- а затем перед `\errmessage` дать команду `\errhelp=\helpout`. Этот метод экономит память, так как в этом случае сообщение занимает меньше места.
- 411, [412](#).
- \newif** Для создания новых тестов. Например, если Вы скажете `\newif\ifabc`, будут определены три команды, `\ifabc`, `\abctrue` и `\abcfalse`.
- 251, [259](#), [412](#), [419](#), [441](#), [486](#), [494](#).
- \newinsert** Команда для задания новых классов вставок.
- 147, 148, 411, [412](#), [429](#), [486](#).
- \newlanguage*** Для задания “нового языка” (см. `\language`).
- 411.
- \newlinechar*** Символ, который начинает новую строку.
- [271](#), 324, [413](#).

- `\newmuskip` Для приватизации `\muskip`-регистра. 147, 411, [412](#).
- `\newread` Команда, аналогичная команде `\newbox`, назначает номер от 0 до 15 одному из входных файлов. 147, 257, 411, [412](#).
- `\newskip` Для приватизации `\skip`-регистра. 147, 411, [412](#), [414](#), [462](#), [484](#).
- `\newtoks` Для создания новых элементов (“цепочек символов”). См. `\toks`. 147, 251, [312](#), 411, [412](#), [470](#).
- `\newwrite` Команда, аналогичная команде `\newbox`, назначает номер от 0 до 15 одному из выходных файлов. 147, 269, 411, [412](#), [494](#).
- `\ni` Перевернутое отношение `\in`: $\ni$. Только в математической моде. Зерзервировано для адептов *машинного языка*, которые могут написать (и прочитав) вот это:

$$\mathbf{R}^n \ni x \xrightarrow{f} x + \phi(x)y \in \mathbf{R}^m$$

508.

- `\ninepoint` Задаёт текущий размер шрифтов, равный 9 pt. Может использоваться как локально, так и глобально. 20, [485](#), [490](#).

- `\noalign*` Позволяет вставлять бокс в стопку боксов, которую производит команда `\halign`. Помещайте макрокоманду `\noalign{...}` сразу после `\cr` (другими словами, между двумя строками таблицы). Например, инструкция `\noalign{\smallskip}` позволяет локально раздвигать две строки таблицы. При вставке текста на краю страницы имеется интересный эффект:

```


$$\begin{array}{l}
\displaylines{
\noalign{\sl Формулы для синуса суммы : }
\sin(a+b) = \sin a \cos b + \cos a \sin b \cr
\noalign{\sl и для косинуса суммы : }
\cos(a+b) = \cos a \cos b - \sin a \sin b. \cr
}
\end{array}$$

```

Эта программка даст следующий результат:

Формулы для синуса суммы :

$$\sin(a + b) = \sin a \cos b + \cos a \sin b$$

и для косинуса суммы :

$$\cos(a + b) = \cos a \cos b - \sin a \sin b.$$

Заменяем `\displaylines` на `\eqalign` (ставя амперсанды прямо перед знаком равенства). Все сдвигается влево!

Формулы для синуса суммы :

$$\sin(a + b) = \sin a \cos b + \cos a \sin b$$

и для косинуса суммы :

$$\cos(a + b) = \cos a \cos b - \sin a \sin b.$$

Пояснение: текст каждой `\noalign` создает строку `\line`, вставляемую внутрь боксов, которые делает таблица.

- Команда `\displaylines` сама тоже создает `\line`, в которых центрирует свои формулы (как с `\centerline`) и которые настраивает друг на друга. `\noalign` вводит `\line` в эту кучу `\line`: это желательное место на странице.

- Команда `\eqalign` создает `\hbox`, который содержит формулы. Введение `\line` в `\hbox` “притягивает” их к левому полю, именно поэтому формулы больше не центрируются.

Введем теперь `\noalign{\hbox{...}}`. С `\displaylines` результат будет такой же, как и раньше. Напротив, с `\eqalign`, получается вот что:

Формулы для синуса суммы :

$$\sin(a + b) = \sin a \cos b + \cos a \sin b$$

и для косинуса суммы :

$$\cos(a + b) = \cos a \cos b - \sin a \sin b.$$

Здесь команда `\noalign` создает боксы, которые вставляются между боксами, генерируемыми командой `\eqalign`. Поскольку ширина этих боксов меньше `\hsize`, центрирование теперь заметно.

211, 228, 230, 282, 292, 296, 334, 340.

`\noboundary*` Если нет `\noboundary`, то на лигатуры и керны могут влиять невидимые “граничные” символы слева и справа.

336, 341, 346.

`\nobreak` Препятствует разрыву строки или страницы в этом месте (`\penalty 10000`). По крайней мере, в теории! Чтобы это было более эффективно, испытайте `TeX`, предложив ему `\allowbreak` (обозначающую `\penalty 0`) или `\break` (обозначающую `\penalty -10000`) для разрыва строки и `\goodbreak` (что означает `\par\penalty -500`) для разрыва страницы.

120, 134, 209, 398, 418, 463, 477.

`\noexpand*` Команда указывает, что ее аргумент не должен подвергаться макрорасширению.

249, 254, 255, 256, 413, 443-444, 485.

`\noindent*` Помещенная в начале абзаца, эта команда запрещает автоматический отступ, который равен `\parindent`.

106, 224, 312, 335, 340, 347, 404, 420-421, 490.

`\nointerlineskip` Не дает $\TeX$ у вставлять вертикальные пробелы, когда он настраивает боксы друг на друга:

`\vbox{\hbox{...}\nointerlineskip\vbox{...}...}`.

Чтобы избавиться сразу от всех межблочных пробелов, лучше использовать не `\nointerlineskip`, а `\vbox{\offinterlineskip...}`.

99, 304, 393, 417, 458.

`\nolimits*` См. `\mathop` и `\limits`.

175, 192, 348, 427.

`\nonfrenchspacing` См. `\frenchspacing`.

93, 416.

`\nonscript*` Если в математической моде перед клеем или керном стоит команда `\nonscript`, $\TeX$ не будет использовать этот клей или керн в индексах.

215, 346, 515, 520.

`\nonstopmode*` Уровень взаимодействия во время работы $\TeX$ 'а, при котором он не останавливается при обнаружении ошибок. См. также `\errorstopmode`, `\scrollmode` и `\batchmode`.

42, 329, 358.

`\nopagenumbers` Используется, когда не нужна нумерация страниц.

300, 428, 476, 479.

`\normalbaselines` Эта макрокоманда восстанавливает значения трех параметров:

`\baselineskip=\normalbaselineskip`
`\lineskip=\normallineskip`
`\lineskiplimit=\normallineskiplimit`.

Другими словами, она восстанавливает нормальный интерлиньяж и межблочный пробел. В формате plain значения указанных трех переменных равны 12 pt, 1 pt и 0 pt.

Макрокоманды `\matrix` и `\case` вызывают `\normalbaselines`. Следовательно, не имеет смысла перед этими командами изменять величину междустрочных пробелов (прямо, написав `\baselineskip=15pt`, или косвенно, написав `\openup 3pt`), потому что `\normalbaselines` после этого все равно восстановит обычное расположение строк.

387, 414, 416, 484-485.

`\normalbaselineskip` См. выше. Для шрифтов в 10 пунктов равен 12 pt.

414, 484-485.

<code>\normalbottom</code>	Отменяет действие <code>\raggedbottom</code> для получения страниц одинаковой высоты. В большом тексте лучше использовать <code>\raggedbottom</code> .	<u>428</u> .
<code>\normallineskip</code>	См. <code>\normalbaselines</code> .	<u>414</u> , <u>416</u> .
<code>\normallineskiplimit</code>	См. <code>\normalbaselines</code> .	<u>414</u> , <u>416</u> , <u>428</u> .
<code>\not</code>	Ставит косую черту на бинарном отношении, которое следует далее. Только в математической моде. $\$x\not=y$, \$U\not\subset V\$ x \neq y, U \not\subset VСм. также \llap, чтобы получать такие отношения “вручную”.$	<u>162</u> , <u>424</u> , <u>508-509</u> .
<code>\notin</code>	Отрицание $\in$: $\notin$. Результат более читабелен, чем <code>\not\in</code> : $\notin$.	<u>426</u> , <u>509</u> .
<code>\nu</code>	Греческая буква ν . Только в математической моде.	<u>156</u> , <u>197</u> , <u>506</u> .
<code>\null</code>	Пустой бокс, который эквивалентен <code>\hbox{}</code> . Очень часто используется в разных конструкциях. Не дает, в частности, <code>\vskip</code> исчезнуть вверх страницы (см. <code>\midinsert</code>).	<u>372</u> , <u>394</u> , <u>416</u> .
<code>\nulldelimiterspace*</code>	Ширина “нулевого ограничителя”. В формате <code>plain</code> его значение равно 1.2 pt.	<u>181</u> , <u>325</u> , <u>413</u> , <u>514</u> .
<code>\nullfont*</code>	Обозначает шрифт, который не содержит символов. Этот шрифт всегда присутствует, если Вы не указали других шрифтов.	<u>19</u> , <u>185</u> , <u>322</u> , <u>506</u> .
<code>\number*</code>	Дает десятичную запись числа, которое за ней следует: <code>\number1988</code> , <code>\number-01515</code> 1988, -1515 Не ставьте фигурные скобки! <code>TeX</code> ожидает найти после командного слова только число или знак и очень удивляется, когда встречает что-либо другое. В действительности, эта макрокоманда служит для записи в тексте содержания регистра целого типа. В Паскале есть <i>write</i> , в Бейсике — <i>print</i> , а в <code>TeX</code> е — <code>\number</code> . Если написать <code>\count213=1789</code> , то <code>\number\count213</code> напечатает 1789. Есть также <code>\the</code> , которая выдает содержание регистра целого типа: <code>\the\count213</code> и <code>\number\count213</code>	

являются эквивалентными. Напротив, `\the1988` или `\the{1988}` не допускаются, так как за `\the` следует не регистр.

51, [253](#), 254, 300, 476, 495.

`\nwarrow` (*north west arrow*) Отношение $\nwarrow$. Только в математической моде. Команды `\nwarrow`, `\swarrow`, `\nearrow` и `\searrow` указывают на четыре стороны света. Жаль, но размер стрелок нельзя менять, поэтому не пытайтесь увеличивать их командами `\big` или `\left...\right`.

509.

O

`\o` Скандинавская буква ø.

[421](#).

`\O` Скандинавская буква Ø. Не путайте с символом пустого множества `\emptyset` ($\emptyset$), который применяется только в математической моде.

[421](#).

`\oalign` Позволяет писать что-либо точно под символом:

```
$\displaystyle\mathop{A}_x+$  
\oalign{$A$\cr\hfill$\scriptstyle x$\hfill\cr}$ .....  $\frac{A}{x} + \frac{A}{x}$ 
```

Именно эта макрокоманда используется в формате `plain` для помещения *акцентов* под буквой.

[422](#).

`\obeylines` (Переходит на новую строку в выходном документе в то же самое время, что и Вы во входном файле.) Обычно переход на новую строку во входном файле интерпретируется Т_ЕX'ом как пробел. Т_ЕX не переходит на следующую строку, пока его строка им заполнена. Чтобы заставить его перейти на новую строку одновременно с Вами, пишите:

```
{\obeylines --- Скажи-ка, дядя, ведь не даром  
Москва, спаленная пожаром, ...}
```

Отметьте фигурные скобки: они необходимы, чтобы действие команды `\obeylines` было *локальным*. Так, в приведенном примере эта команда прекращает функционировать, как только пройдена закрывающая фигурная скобка. Понятно, что если Вы все время печатаете стихи, фигурные скобки не нужны. Подумайте об использовании `\begingroup` и `\endgroup`, если текст в фигурных скобках слишком длинный.

116, 295, 312, 406, [417](#), 447-449, 477, 489.

`\obeyspaces` То же самое, что и `\obeylines`, но для промежутков.

303, 406, [417](#), 447, 463, 492.

<code>\odot</code>	Бинарное отношение $\odot$. Только в математической моде. Если Вы хотите сделать его более крупным, пишите <code>\bigodot</code> : $\bigodot$. 508.
<code>\oe</code>	‘o’ и ‘e’ переплетенные: <code>\oe uf</code> , <code>b\oe uf</code> , <code>v\oe ux</code> ... œuf, bœuf, vœux. Аналогом для пары ‘a-e’ является <code>\ae</code> : <code>n\ae vus</code> дает nævus. 67, 421 .
<code>\OE</code>	То же, что и выше, но заглавными буквами: L’\OE IL DE CUIVRE L’ÆIL DE CUIVRE 67, 68, 421 .
<code>\offinterlineskip</code>	Уничтожает все межблочные пробелы во время построения <code>\vbox</code> : $\vbox{\offinterlinelimeskip...}$ Эта макрокоманда обязательна для построения вертикальных линеек в таблице: $\vbox{\offinterlineskip\halign{...}}$ Чтобы устранить пробел только между двумя блоками, надо поместить между ними команду <code>\nointerlineskip</code> : $\vbox{\box1\offinterlineskip\box2\box3}$ 291, 372, 417 , 487 .
<code>\oint</code>	Большой оператор $\oint$ и $\int$. Только в математической моде. 423 , 507.
<code>\oldstyle</code>	Команда для получения шрифта “древних” цифр. Делает удивительные даты: 12 июня {\oldstyle 1991} года 12 июня 1991 года Производит большее впечатление, чем 12 июня 1991 года. Полный список “древних” чисел — 1234567890. Если Вам приходится часто использовать этот стиль, лучше использовать <code>\let\old=\oldstyle</code> и после этого печатать просто 12 июня {\old 1991}года. Если Вы забудете фигурные скобки, макрокоманда <code>\oldstyle</code> обратится не только к числам... Результат может быть достаточно странным: <code>\oldstyle 12 учеников кончили занятия 12учениковкончилизанятия</code> 416 .
<code>\omega</code>	Греческая буква ω . Только в математической моде. 385, 423 , 506.
<code>\Omega</code>	Заглавная греческая буква Ω . Только в математической моде. 423 , 506.

<code>\ominus</code>	Бинарное отношение $\ominus$. Только в математической моде.	508.
<code>\omit*</code>	Аббревиатура <code>\multispan1</code> .	285, 289, 335.
<code>\openin*</code>	См. <code>\closein</code> .	249 257, 331.
<code>\openout*</code>	См. <code>\closeout</code> .	269, 302, 332, 493, 494.
<code>\openup</code>	Макрокоманда, которая изменяет пробел между строками таблицы (увеличивает <code>\baselineskip</code>). Синтаксис: <code>\openup 3mm\halign{...}</code> . <i>Существенно</i> писать <code>\openup</code> <i>перед</i> командой <code>\halign</code> . Если поменять их местами (<code>\halign{\openup 3mm...}</code>), команда <code>\openup</code> не работает: $\frac{1}{1+x^2} \quad \frac{y}{1+x^2+y^2}$ $\sum_{i=0}^{i=n} a_i b_{n-i} \quad \int_x^{x^2} f(x) dx$ $\frac{1}{1+x^2} \quad \frac{y}{1+x^2+y^2}$ $\sum_{i=0}^{i=n} a_i b_{n-i} \quad \int_x^{x^2} f(x) dx$ <code>\halign{\openup 3mm...}</code> <code>\openup 3mm\halign{...}</code> Эффект от команд <code>\openup</code> суммируется. Внимание: <code>\openup</code> не функционирует с макрокомандами <code>\matrix</code> и <code>\cases</code>. См. <code>\matrix</code>, чтобы понять почему.	231, 282, 287, 428.
<code>\oplus</code>	Бинарное отношение $\oplus$. Только в математической моде: $\dim(U \oplus V) = \dim U + \dim V$ $\dim(U \oplus V) = \dim U + \dim V$ Для большого оператора $\bigoplus$ имеется макрокоманда <code>\bigoplus</code> . $\displaystyle E = \bigoplus_{i \in I} E_i$ $E = \bigoplus_{i \in I} E_i$	12, 186, 508.
<code>\or*</code>	Используется в языке программирования TeX. См. <code>\cases</code> .	249-250, 252, 476.
<code>\oslash</code>	Бинарное отношение $\oslash$. Только в математической моде.	508.
<code>\otimes</code>	Бинарное отношение $\otimes$. Только в математической моде. $(E \otimes F) \otimes G = E \otimes (F \otimes G)$ $\otimes(F \otimes G)$ $(E \otimes F) \otimes G = E \otimes (F \otimes G)$ Для большого оператора $\bigotimes$ используйте макрокоманду <code>\bigotimes</code> :	

$$\text{\textbackslash displaystyle } E = \bigotimes_{i \in I} E_i \quad \dots \quad E = \bigotimes_{i \in I} E_i$$

12, 508.

\outer* Если ввести `\outer\def\toto{...}`, то такая макрокоманда не сможет быть аргументом или параметром другой макрокоманды. Если Вы попытаетесь сделать это, $\TeX$ пошлет Вам `runaway argument ?`. Используйте эту макрокоманду для повышения надежности. В $\TeX$ е не рекомендуется слишком глубоко вкладывать макрокоманды, так как результат быстро станет неконтролируемым. Это хорошее средство призвать неразумных пользователей к порядку.

245, 249, 326, 420, 423, 489-490, 493.

\output* Определяет программу вывода, которая будет использоваться. Если не указана программа вывода или если Вы скажете `\output={}`, $\TeX$ будет использовать свою программу, которая, по существу, эквивалентна `\output={\shipout\box255}` и выводит страницы без нумерации и без нижнего и верхнего колонтитулов. Формат `plain` автоматически вызывает программу вывода `\plainoutput`.

151, 302, 326, 430, 437, 487.

\outputpenalty* Один из специальных штрафов, который используется при автоматическом формировании страниц.

1515, 303, 323, 413, 469, 488.

\over* Строит дробь с горизонтальной чертой. Только в математической моде. Синтаксис:

`{числитель \over знаменатель}`

$\TeX$ автоматически центрирует числитель и знаменатель с помощью слабого клея `\hfil`. Вы можете помешать этому сильным клеем `\hfill` (с двумя 'l'). Не следует использовать дроби с горизонтальной чертой в тексте (в `\textstyle`, а не в `\displaystyle`). Результат (например, $\frac{1+x^2}{1+a^2+b^2}$) является не самым удачным. Не забывайте о читателях с плохим зрением! Лучше используйте косую черту и круглые скобки: $(1+x^2)/(1+a^2+b^2)$. Напротив, "маленькие" дроби типа $\frac{1}{2}x$ и $\frac{3}{4}y$ в тексте выглядят красиво.

Существуют многочисленные варианты команд для дробей, работающие по такому же принципу. Это `\overwithdedims`, `\atopwithdelims`, `\atop` и `\abovewithdelims`. Макрокоманда `\overwithdelims` создает дробь так же, как и `\over`, но окружает ее ограничителями (об этом говорит имя макрокоманды). Вторая команда, `\atop`, дает дробь без горизонтальной черты, третья, `\atopwithdelims`, строит дробь без горизонтальной черты, но окруженную ограничителями (например, биномиальные коэффициенты). Что касается четвертой, то она обобщает две предыдущих, давая возможность задавать толщину горизонтальной черты. Все эти варианты имеют такой же синтаксис, что и `\over`.

Черта дроби, проведенная командой `\over`, может показаться слишком толстой. Если Вы согласны с этим, замените `\over` на вновь определенную команду `\sur`: `\def\sur{\above.2pt}`. См. `\sur`.

169, 183, 349, 509, 517.

\overbrace Макрокоманда, которая помещает фигурную скобку над своим аргументом. Только в математической моде:

`$e=\bigl[2,\overbrace{1,2,1}^{k=1},\overbrace{1,4,1}^{k=2},\overbrace{1,6,1}^{k=3},\ldots,\overbrace{1,2k,1}^{k=3}\bigr]$`

$$e = [2, \overbrace{1, 2, 1}^{k=1}, \overbrace{1, 4, 1}^{k=2}, \overbrace{1, 6, 1}^{k=3}, \dots, \overbrace{1, 2k, 1}^{k=3}, \dots]$$

Попробуйте поместить подпорку в команду `\overbrace`, написав так: `\overbrace{\strut...}`. Это немного поднимет фигурную скобку и улучшит разборчивость:

$$e = [2, \overbrace{1, 2, 1}^{k=1}, \overbrace{1, 4, 1}^{k=2}, \overbrace{1, 6, 1}^{k=3}, \dots, \overbrace{1, 2k, 1}^{k=3}, \dots]$$

Команда `\overbrace` действует только в математическом контексте. Она “прозрачна” в том смысле, что внутри фигурных скобок сохраняется математическая мода. Следовательно, не надо `\overbrace{\$...\$}`.

Команда, помещающая горизонтальную фигурную скобку под своим аргументом, называется `\underbrace`. Она работает в математической моде. Вне математической моды имеется команда `\downbracefill` (которая соответствует `\overbrace`) и команда `\upbracefill` (которая соответствует `\underbrace`).

211, 268, 425.

\overfullrule* Задаёт толщину черной вертикальной черты, которая печатается на полях в случае “`overfull \hbox`”. Чтобы от нее избавиться, или, более точно, сделать эту черту невидимой, введите в начале входного файла `\overfullrule=0mm`. Чтобы Вас не слишком часто беспокоило сообщение об `overfull \hboxes`, задайте, например, в начале своего файла `\hfuzz=3pt`.

325, 367, 413.

\overleftarrow Печатает над аргументом левую стрелку (это макрокоманда с параметром). Только в математической моде:

`$\overleftarrow{A}=\overleftarrow{f(x,y)}$` $\overleftarrow{A} = \overleftarrow{f(x,y)}$

425.

\overline* Проводит черточку над своим аргументом. Только в математической моде. Таким образом, `$\overline{uvw}$` даёт $\overline{uvw}$. Для подчеркивания следует использовать команду `\underline`. См. `\bar` для сравнения и

некоторых советов. Если Вы используете макрокоманду `\overline` для черты над заглавными буквами, результат будет не совсем удовлетворительный: проведенная черта окажется слегка сдвинутой влево. Чтобы исправить эту ошибку, поместите после `\overline` “отрицательный” мини-пробел `\!` и сравните две строки:

$$\overline{A} + \overline{B} + \overline{C} + \overline{M} + \overline{X} + \overline{Y} + \overline{Z}$$

и

$$\overline{A} + \overline{B} + \overline{C} + \overline{M} + \overline{X} + \overline{Y} + \overline{Z} :$$

В первой строке используется команда `\overline{...}`, а во второй — конструкция `\overline{\!...}`.

158, 165, 172, 205, 515.

`\overrightarrow` Чертит над своим аргументом правую стрелку. Только в математической моде. Идеальная команда для вектора:

```


$$\overrightarrow{AB^2} = \overrightarrow{AC^2} - 2\overrightarrow{AB} \cdot \overrightarrow{AB}$$


```

$$AB^2 = AC^2 - 2\overrightarrow{AB} \cdot \overrightarrow{AB}$$

Коррекция `(\kern-2pt#1\kern 2pt)` дает более хороший результат:

```


$$\overrightarrow{AB} \dots \overrightarrow{AB}$$


$$\overrightarrow{\kern-2pt AB \kern 2pt} \dots \overrightarrow{AB}$$


```

Не используйте в формате `plain` макрокоманду `\vec`: `\vec{AB}` дает $\vec{AB}$. `\vec` — это *акцент* для строчных букв: $(\vec{i}, \vec{j}, \vec{k})$.

268, 425.

`\overwithdelims*` См. `\abovewithdelims` и `\over`.

183, 349, 517.

`\owns` Разновидность `\ni`: ∋. Разумеется, только в математической моде.

427, 510.

Р

`\P` Знак “параграф” (по-русски — “абзац”) ¶.

68, 142, 422, 511.

`\pagebody` Эта команда определена специально для внутренних целей программы вывода формата `plain` `\plainoutput`.

304-306, 430.

`\pagecontents` См. `\pagebody`.

304, 430.

- `\pagedepth*` Специальный параметр, который $\TeX$ использует при формировании страниц.
139, 149, 254, 322.
- `\pagefilllstretch*` Специальный параметр, который $\TeX$ использует при формировании страниц.
139, 254, 322.
- `\pagefillstretch*` Специальный параметр, который $\TeX$ использует при формировании страниц.
139, 254, 322.
- `\pagefilstretch*` Специальный параметр, который $\TeX$ использует при формировании страниц.
139, 254, 322.
- `\pagegoal*` Специальный параметр, который $\TeX$ использует при формировании страниц.
139, 149, 254, 322.
- `\pageinsert` Команда, аналогичная `\midinsert` и `\topinsert`. Используется в конструкции `\pageinsert ... \endinsert` и только в вертикальной моде. Вставляет материал на отдельной странице.
141, 429.
- `\pageno` Счетчик, который содержит номер страницы. См. `\folio`.
300, 305, 404, 428, 476.
- `\pageshrink*` Специальный параметр, который $\TeX$ использует при формировании страниц.
139, 149, 254, 322.
- `\pagestretch*` Специальный параметр, который $\TeX$ использует при формировании страниц.
139, 254, 322.
- `\pagetotal*` Специальный параметр, который $\TeX$ использует при формировании страниц.
139, 149, 254, 322.
- `\par*` Заставляет $\TeX$ переходить в вертикальную моду. Эта команда завершает текущий абзац. Сама `\par` не начинает новый абзац. Новый абзац начинается тогда, когда $\TeX$ встречает буквы или команды `\indent`, `\leavevmode...` и им подобные.
60, 106-107, 123, 240, 244, 246, 295, 312, 335, 340-341, 404, 447.
- `\parallel` Отношение $\parallel$. Только в математической моде. Не путайте с результатом команды `\l`, вокруг которого не добавляются специальные пробелы

(так как это не отношение). Отличие такое же, как между `\mid` и вертикальной чертой `|` клавиатуры.

508.

`\parfillskip*` Пробел, который `TeX` добавляет к концу последней строки абзаца (чтобы завершить эту строку). Речь идет о слабо бесконечном клее: `\parfillskip=0pt plus 1fil` (хотя это и слабо бесконечный клей, Кнут пишет его с двумя ‘l’). Этот клей при необходимости изменяется от 0 пунктов до бесконечности. Он не дает пробелам между словами на последней строке растягиваться слишком сильно.

Для этой команды есть забавное применение: если Ваш абзац достаточно длинный, попробуйте задать `\parfillskip=0mm`. Так как `TeX` больше не имеет права добавлять пробел в конце абзаца, последняя строка, если она достаточно длинная, *должна* будет окончиться на правом поле:

*На краю мертвого марсианского моря раскинулся безмолвный городок. Он был пуст. Ни малейшего движения на улицах. Днем и ночью в универсамах одиноко горели огни. Двери лавок открыты настежь, словно люди обратились в бегство, забыв о ключах. На проволочных рейках у входа в немые закусовые нечитанные, порывшие от солнца, шелестели журналы, доставленные месяц назад ракетой с Земли.*⁶

`{\parfillskip=0mm\sl...\par}`

Но это работает не во всех случаях! Обратите внимание, что `TeX` здесь раздвигает пробелы между словами немного больше, чем обычно. Команда `\par`, которая расположена перед закрывающей фигурной скобкой группы и делает локальной команду `\parfillskip=0mm`, является обязательной.

123, 224, 325, 340, 367, 394, 413, 463, 490.

`\parindent*` Значение отступа в начале абзаца. Синтаксис: `\parindent=10mm` или `\parindent 10mm` (знак ‘=’ необязательный). Если Вы новичок или печатаете не самый простой текст, Вам лучше поставить `\parindent=0mm` и создать специальную макрокоманду `\ind`, которая делает отступ и переход в горизонтальную моду:

`\def\ind{\hskip 1 cm\relax}`

106, 124, 129, 312, 325, 335, 340, 347, 406, 413, 420, 463, 476, 486.

`\parshape*` Эффектная макрокоманда, которая позволяет манипулировать формой абзаца. Синтаксис очень простой. Начните абзац так:

`\parshape=5 2cm 8cm 2.5cm 7.5cm 3cm 7cm 0cm 7cm 0cm 8cm`

`\parshape=5` (знак ‘=’, как всегда, необязателен) указывает `TeX`у что первые пять строк формируются специальным способом. Первая строка

⁶ Р.Бредбери, *Марсианские хроники*.

должна отступить влево на 2 см и иметь длину 8 см, вторая — отступить на 2,5 см и иметь длину 5 см и т.д., а последняя строка должна отступить на 0 см (следовательно, нет отступа) и быть длиной 8 см. Если в абзаце число строк больше 5, Т_ЕX повторяет последнюю спецификацию для линий 6, 7 и всех следующих. Команда `\parshape` модифицирует абзац, который ее содержит. Она может быть помещена в любом месте абзаца, в начале, в середине или в конце. Следующие абзацы изменяться не будут. Это справедливо также и для макрокоманд `\hangindent` и `\hangafter`. Для формирования прямоугольных выемок в абзаце см. команды `\hangindent` и `\hangafter`.

125, 254, 322, 328, 335, 376, 413, 440.

`\parskip`* Вертикальный пробел *перед* абзацем. Если, например, написать:

`\parskip=2mm plus .5mm minus .5mm`

Т_ЕX добавит по 2 мм перед каждым абзацем (с некоторым допуском).

99, 128-129, 312, 325, 335, 406, 413, 420, 476, 488.

`\partial` Запись частных производных ∂ . Только в математической моде:

`\let\=\partial`

$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial x \partial y}$

Вы заметили хитрость `\let\=\partial`, которая сделала кодировку более приятной?

147, 507.

`\patterns`* Эту команду plain Т_ЕX использует, когда задает словарь образцов переноса слов для какого-либо языка (см. `\language`).

329, 528, 530.

`\pausing`* Целочисленный параметр. Если Вы установите `\pausing=1`, Т_ЕX будет останавливаться после каждой строки, прочитанной из входного файла, и Вы сможете редактировать эту строку, вставлять команды `\show` и другие, не меняя содержимое входного файла. Это удобный способ отладки.

324, 362.

pc Одна из единиц измерения: 1 pc = 12 pt.

71, 321, 486.

`\penalty`* Параметр “штраф”. Используется при формировании абзацев и страниц. Синтаксис: `\penalty`, затем целое число, положительное или отрицательное, абсолютное значение которого не превышает 10000. (На практике не превышает 500). Не ставьте после `\penalty` знак ‘=’, иначе Т_ЕX не “увидит” число, которое потом стоит. Чтобы создать строку, абзац или страницу, Т_ЕX оценивает их “плохость” (*badness*). Из плохостей складывается дефектность страницы. Т_ЕX не удовлетворяется

одной пробой. Он пробует несколько вариантов и выбирает решение с наименьшей дефектностью (см. `\tolerance`). Для разрыва строки или страницы $\TeX$ любит использовать пробел (`\hspace` или `\vspace`), а так же место команды `\penalty`. Можно повлиять на этот выбор, поместив в свой текст соответствующий штраф. Если имеется разрыв на штрафе, то он добавляется к дефектности. Например, `\def\nobreak{\penalty 10000}` (10000 есть наибольшее значение, которое может принимать `badness`). Понятно теперь, что такая макрокоманда запрещает разрыв на месте, где она помещена. Использовать штрафы очень просто: достаточно напечатать `\penalty 100` там, где Вы не хотели бы, чтобы производился разрыв (строки или страницы). Если этого приказа недостаточно, добавьте штраф. Наоборот, `\penalty -200` побуждает $\TeX$ попробовать сделать разрыв в этом месте, если $\TeX$ у нужно сделать разрыв вблизи этого места.

99, 119, 135-136, 209, 332, 418.

`\perp` Отношение $\perp$. Только в математической моде. Обратное отношение `\top` дает $\top$.

423, 508.

`\phantom` Создает невидимый бокс, ширина, высота и глубина которого являются такими же, как у “формулы”-аргумента (речь идет о макрокоманде с параметром). Слово “формула” в кавычках, поскольку можно взять фантом все равно от чего (и не в математической моде). Эта макрокоманда иногда очень полезна при вертикальном выравнивании. Для “визуализации” фантомов, окружим их рамкой:

```
\setbox2=\hbox{\$ \quad \alpha_p + \beta^q \quad \$}
\setbox3=\hbox{\phantom{\copy2}}
\hrule\copy2\hrule ..... | \alpha_p + \beta^q |
\hrule\copy3\hrule ..... | \alpha_p + \beta^q |
\hrule\vbox{\hrule\box2\hrule}\hrule ..... | \alpha_p + \beta^q |
\hrule\vbox{\hrule\box3\hrule}\hrule ..... | \alpha_p + \beta^q |
```

Есть два вида фантомов: `\hphantom`, который создает горизонтальную невидимую черту, имеющую такую же ширину, что и формула, и `\vphantom`, который создает вертикальную невидимую черту, имеющую такую же высоту и глубину, что и формула. Обратите также внимание на `\smash` — эту гениальную макрокоманду, которая аннулирует высоту и глубину бокса, нормально печатая его содержимое.

159, 214, 250, 426, 482.

`\phi` Греческая буква ϕ . Возможно Вам больше понравится круглая версия `\varphi`: φ . Только в математической моде.

156, 179, 506.

`\Phi` Заглавная греческая буква Φ . Только в математической моде.

385, 506.

<code>\pi</code>	Греческая буква π . Имеется также <code>\varpi</code> : ϖ . Только в математической моде.	12, 107, 167, 179, 506.
<code>\Pi</code>	Заглавная греческая буква Π . Только в математической моде. Не путайте с оператором <code>\prod</code> : <code>\$\$\Pi, \prod, \displaystyle\prod\$</code> $\Pi, \Pi, \prod$	12, 385, 506, 507.
<code>\plainoutput</code>	Программа вывода, которую использует формат <code>plain</code> . См. описание команды <code>\output</code> .	304, 430.
<code>plus</code>	Ключевое слово, которое вместе с <code>minus</code> используется при задании эластичности клея.	89, 321, 400.
<code>\pm</code>	Бинарный оператор $\pm$. Только в математической моде. Имеется также <code>\mp</code> , которая выдает $\mp$.	161, 423, 508.
<code>\pmatrix</code>	Матрица, окруженная круглыми скобками. Только в математической моде. Матрица может состоять из любых элементов и даже из других матриц: $J = \begin{pmatrix} \begin{pmatrix} \lambda & 1 \\ 0 & \lambda \end{pmatrix} & 0 \\ 0 & \begin{pmatrix} \mu & 0 \\ 0 & \mu \end{pmatrix} \end{pmatrix}$ Эта эффектная матрица кодируется очень просто: <code>\$\$J=\pmatrix{\pmatrix{\lambda&1\cr0&\lambda}\cr}& 0\cr0 & \pmatrix{\mu&0\cr0&\mu}\cr}</code> Конечно, $\TeX$ имеет недостатки, но его сильная сторона — это отличная кодировка формул: они читаются за километр! См. <code>\matrix</code> для синтаксиса и дополнительных рекомендаций.	212, 384, 427.
<code>\pmod</code>	Для получения “по модулю такому-то” в круглых скобках и отделенного от предшествующего текста пробелом приблизительно в квадрат. Только в математической моде. Не путайте с <code>\equiv</code> (которая дает $\equiv$) или <code>\mod</code> (которая не существует!): <code>\$\$x^p\equiv x\pmod p\$</code> $x^p \equiv x \pmod p$	198, 384, 427.

- `\postdisplaypenalty*` Один из параметров, который автоматически используется для удачного расположения выделенных формул на странице.
226, 323.
- `\prec` Отношение $\prec$. Только в математической моде.
508.
- `\preceq` Отношение $\preceq$. Только в математической моде. Противоположное отношение ($\succeq$) кодируется `\succeq`.
508.
- `\predisplaypenalty*` Один из параметров, который автоматически используется для удачного расположения выделенных формул на странице.
225-226, 323, 413.
- `\predisplaysize*` Один из параметров, который автоматически используется для удачного расположения выделенных формул на странице.
224, 227, 325, 413.
- `\pretolerance*` Вы, наверное, помните, что $\TeX$ делает несколько попыток, прежде чем разорвать строку (см. команду `\penalty`). Он сначала пытается разорвать строку между словами. Если плохость (badness) полученной строки не превышает `\tolerance`, попытка считается удачной. Если нет, $\TeX$ пытается сделать перенос слова. В этом случае он сравнивает плохость со значениями, заданными в `\pretolerance`. Если ни одно из решений нельзя считать удачным, он выберет наименее плохое (“дефектность” которого самая маленькая) и, глубоко огорченный, pošлет Вам сообщение “`overfull \hbox`”.
119, 323, 378, 413, 430, 462, 526.
- `\prevdepth*` Примитив, который обычно содержит глубину самого свежего в вертикальном списке бокса.
99-101, 322, 323, 334.
- `\prevgraf*` Внутренняя целая переменная, содержащая число строк самого последнего абзаца, завершенного или нет. Можно использовать `\prevgraf` как число, а также присвоить ей неотрицательное значение, чтобы убедить $\TeX$, что он находится в некоторой конкретной части абзаца.
127, 224, 227, 254, 322.
- `\prime` Математический символ $\prime$. Только в математической моде. Предназначен для употребления в индексах, поэтому обычно мы встречаем его в уменьшенном размере: `$y_1^\prime$` дает y_1' .
158, 187, 423, 507.
- `\proclaim` Команда, предлагаемая в формате *plain* для композиции теорем. Она имеет два параметра: `\def\proclaim#1.#2\par{...}`. Первая встречающаяся точка указывает на конец первого параметра, а первая `\par` (или первая пустая строка) — на конец второго. Кодировка:

`\proclaim Теорема [Ферма]`. Если p --- простое число,
то для всех целых x конгруенция $x^p \equiv x \pmod p$.
{\rm Другой вариант этой теоремы : $x^{p-1} \equiv 1 \pmod p$ для всех целых $x \not\equiv 0 \pmod p$.}

выдаст следующее:

Теорема [Ферма]. Если p — простое число, то для всех целых x конгруенция $x^p \equiv x \pmod p$.

Другой вариант этой теоремы : $x^{p-1} \equiv 1 \pmod p$ для всех целых $x \not\equiv 0 \pmod p$.

Обратите внимание, что #1 (заголовок теоремы) автоматически печатается жирным шрифтом, а #2 (изложение теоремы) — наклонным шрифтом.

241, 245, 404, 421.

`\prod` Большой оператор $\prod$ и $\prod$ (его размер зависит от стиля). Только в математической моде:

$$\mathcal{C}(n;a)=e^{-a}\prod_{i=1}^m(-a)^{n_i}\sum_{k\geq 0}\prod_{i=1}^m C_{n_i}^{(a)}(k)\frac{a^k}{k!}$$

$$C(n;a)=e^{-a}\prod_{i=1}^m(-a)^{n_i}\sum_{k\geq 0}\prod_{i=1}^m C_{n_i}^{(a)}(k)\frac{a^k}{k!}$$

216-217, 506.

`\propto` Отношение $\propto$. Только в математической моде. Не путайте с греческой буквой `\alpha`, которую оно напоминает:

`\propto`, `\alpha` $\propto, \alpha$
508.

`\psi` Греческая буква ψ . Только в математической моде.

387, 506.

`\Psi` Заглавная греческая буква Ψ . Только в математической моде.

506.

`pt` Пункт, единица измерения в $\TeX$. Базовые линии в этом руководстве располагаются друг от друга на расстоянии 12 pt.

32, 71-72, 318-319, 321.

q

`\quad` Горизонтальный пробел размером в два *квадрата* (`\hskip 2em`).

200, 221, 417.

`\quad` Горизонтальный пробел размером в один *квадрат* (`\hskip 1em`), т.е. чуть более 10 pt для десятипунктовых шрифтов. Один квадрат приблизительно равен трем нормальным пробелам.
116, 200-201, 221, 277-278, 417.

Г

`\radical*` Примитив низшего уровня. В формате `plain` знак квадратного корня определяется следующим образом:

```
\def\sqrt{\radical}270370
```

189, 348, 515.

`\raggedbottom` Обычно страница начинается, отступив x cm от верхнего края бумаги и завершается в y cm от нижнего края. Но это условие не всегда можно соблюсти, потому что материал, который надо разместить на странице, иногда слишком не соответствует этой норме. Чтобы в таких случаях помочь $\TeX$ у в компоновке страниц, разумно придать ему дополнительную гибкость, позволив делать неровный нижний край страницы. Для этого в начале входного файла надо написать `\raggedbottom`. Эта команда является вертикальным аналогом команды `\raggedright`. Действие `\raggedbottom` прекращается командой `\normalbottom`.

136, 301, 305, 428, 476.

`\raggedright` (*ragged*: рваный, выщербленный). Так как при печати текста $\TeX$ старается выравнивать его справа, то при слишком узкой странице или боксе Вам полетят сообщения `overfull \hbox`. В таких случаях иногда удобно отказаться от двустороннего выравнивания текста, выравнивая его только слева. Это делает команда `\raggedright`, помещенная в начале страницы или бокса:

```
\vbox{\hsize 5cm\sevenrm\raggedright ... текст ...}
```

Макрокоманда `\raggedright` делает эластичным тот пробел, который $\TeX$ систематически помещает в конец каждой строки:

```
\def\raggedright{\righskip=0pt plus 2em
\spaceskip=.3333em \xspaceskip=.5em}
```

Это определение показывает, что $\TeX$ имеет право изменять пробел (`\righskip`), расположенный в конце каждой строки, от нуля до 2 квадратов. Обычно пробелам между словами позволяется некоторая эластичность (команды `\spaceskip` и `\xspaceskip` имеют допуски `plus` и `minus`). Здесь же эти атрибуты отсутствуют: пробелы между словами имеют одну и ту же ширину (треть квадрата). Если Вы к тому же переключаете шрифт, сделайте *сначала* вызов `\raggedright`. Потому что, если написать

```
{\raggedright\ninerm...},
```

макрокоманда `\raggedright` будет работать неправильно, так как команда `\ninerm` снова модифицирует `\spaceskip` и `\xspaceskip`.

29-30, 95, 124, 131, 136, 310-311, 421, 461, 477.

`\raise*`

Бокс можно приподнять командой `\raise 2mm\ vbox{...}` и опустить командой `\raise -3mm\ hbox{...}`. Так перемещать можно любой тип боксов. Внимание: `\raise` функционирует только в *горизонтальной* моде, т.е., на строке, в `\hbox` или в таблице. За ней должен обязательно следовать бокс: `\raise 3pt{...}` вызывает сообщение об ошибке. Посмотрите на интересную формулу: она расположена в квадратных скобках, которые спускаются очень низко. Если по наивности написать `\left[дробь\right]`, получатся квадратные скобки, которые спущены, как надо, но к тому же еще и слишком высоко подняты (что нас не устраивает):

$$\sin z = z \lim_{p \rightarrow \infty} \prod_{k=1}^{p-1} \left[1 - \frac{z^2}{4p^2 \tan^2 \frac{k\pi}{2p}} \right]$$

Нам же надо создать квадратные скобки нужной высоты, а потом их опустить:

```
\def\sur{\above .2pt}
\def\crochet#1{\raise -5pt
\hbox{$\left#1\ vbox to 22pt{}\right.$}}
$$\sin z=z\lim_{p\rightarrow\infty}\prod_{k=1}^{p-1}
\crochet{[]\1-{z^2\sur\displaystyle 4p^2\tan^2{k\pi\over 2p}}
\crochet{[]}}$
```

Поиск хороших параметров потребовал нескольких попыток, но результат получился прекрасный:

$$\sin z = z \lim_{p \rightarrow \infty} \prod_{k=1}^{p-1} \left[1 - \frac{z^2}{4p^2 \tan^2 \frac{k\pi}{2p}} \right]$$

Чтобы не искать на ощупь размеры квадратных скобок, следует поступать вот как. Поместите сначала дробь в `\hbox`, затем попросите `TeX` сообщить Вам высоту и глубину этого бокса:

```
\setbox1=\hbox{$\displaystyle 1-{z^2 \dots 2p}$}
\showthe\ht1 \showthe\dp1
```

Вы получите 14,9 pt и 21,4 pt. Значит, дробь имеет высоту 36,3 pt. Квадратные скобки должны быть немного больше дроби, скажем, на 3 pt вверх и вниз. Эксперимент показывает, что хорошо смотрятся квадратные скобки в 44 pt. Их печатают командами `\left...\ vbox to 22pt{}\right...` по аналогии с примером из команды `\big`. Высота части квадратной скобки выше базовой линии равна 22 pt, а высота числителя приблизительно равна 15 pt. Следовательно, квадратную скобку надо опустить на $22 - (15 + 3) = 4$ pt. Наибольший же визуальный эффект достигается с 5 pt. Эта команда имеет горизонтальные аналоги `\moveleft` и `\moveright`, которые работают только в вертикальной моде.

Напомним еще раз, что `\raise` очень удобно использовать в таблицах. См. также команду `\smash`, потому что `\raise` и `\smash` очень часто работают вместе. В приведенной ниже таблице использовалась макрокоманда `\tvi` — невидимая черта высотой 12 pt и глубиной 5pt (`\def\tvi{\vrule height12pt depth5pt width 0pt}`), которая не входит в формат `plain`.

функция	предел в бесконечности	<code>\setbox1=\vbox{\haling{#\hfill\cr предел \cr в бесконечности\cr}} \haling{\tvi#\hfill&\quad#\hfill \cr\raise 6pt\hbox{функция}&% \smash{\box1}\cr\exp&\${\infty}\$ \cr \${\log x/x}\$&0\cr}</code>
$\exp$	$+\infty$	
$\log x/x$	0	

82, 100, 182, 215, 230, 339, 347, 478.

`\rangle` Закрывающая угловая скобка $\rangle$. Не смешивать со знаком неравенства $>$, который более острый. Только в математической моде. Открывающая угловая скобка $\langle$, конечно же, называется `\langle`.

117, 181, 425, 510.

`\rbrace` Закрывающая фигурная скобка $\}$. См. `\lbrace`.

163, 176, 425, 510.

`\rbrack` Закрывающая квадратная скобка $\}$. См. `\lbrack`.

176-177, 416, 436, 510.

`\rceil` Символ $\lceil$. См. `\lceil`.

176-177, 425, 510.

`\Re` Символ “действительная часть”: $\Re$. См. макрокоманду `\Im`.

507.

`\read*` Читает входной файл с указанным номером. Файл должен быть открыт командой `\openin`. Номер файлу назначается командой `\newread`, которая аналогична `\newbox`. Если номер файла не в пределах от 0 до 15 или нет открытого файла с таким номером, ввод производится с терминала. Например, если Вы введете `\read16 to \myname`, команда `\read` напишет на терминале ‘`\myname=`’ и будет ждать Вашего ответа. Этот ответ и станет значением `\myname`, в чем можно убедиться с помощью команды `\message{Привет, \myname!}`.

255, 257-258, 327, 410.

`\relax*` Говорит $\TeX$ ’у ничего не делать! Эта макрокоманда иногда обязательна, так как она позволяет “разорвать” последовательность макрокоманд, которые могут провозимодействовать друг с другом или с текстом. В описании команды `\parindent` предлагалась макрокоманда `\def\ind{\hskip 1cm\relax}`. Попробуйте в этой макрокоманде опустить `\relax` и написать что-нибудь вроде:

`\ind Plus pr\'ecis\'emment, on a ...`

Для $\TeX$ 'а это будет значить `{\hskip 1cm plus} pr\'ecis\'emment on a ...` (фигурные скобки наши). Не обращая внимания на заглавную букву P и прочитав `\hskip 1cm plus`, $\TeX$ пожалуется на отсутствие размера! Или другой неприятный пример. Если вы напишете

`\righskip=0mm plus 1fil Les silences sont de natures...`,

то к своему большому изумлению получите следующее:

`es silences sont de natures ...`

$\TeX$ 'у показалось, что вы написали `fill` с двумя 'l': `\rightskip=0mm plus 1fill` (заглавная 'L' его совсем не беспокоит). С правильно поставленной командой `\relax` порядок восстанавливается.

31, 34, 89, 286, 327, 339, 367, 419.

`\relbar` Знак минуса, превращенный в отношение ($-$). Только в математической моде. Используется для удлинения размера стрелки:

`\def\verylongrightarrow{\relbar\joinrel\longrightarrow}`
`$\rightarrow\quad\longrightarrow\quad\verylongrightarrow$`

$\rightarrow \longrightarrow \Longrightarrow$

Если нужна более длинная стрелка, используйте растяжимую стрелку:

`\hbox to 8mm{\rightarrowfill}` $\longrightarrow$
424.

`\Relbar` Знак равенства, превращенный в отношение. Только в математической моде. Такое же использование, как и выше, но с двойной стрелкой:

`\def\Verylongrightarrow{\Relbar\joinrel\Longrightarrow}`
`$\Rightarrow\quad\Longrightarrow\quad\Verylongrightarrow$`

$\Rightarrow \Longrightarrow \implies$

424.

`\relpenalty*` Один из видов штрафа, используемый при формировании абзацев. В формате `plain` `\relpenalty=500`.

124, 209, 323, 384, 413, 520.

`\removelastskip` Специальная команда, которая в формате `plain` используется при автоматическом формировании абзацев и страниц.

418.

`\repeat` См. команду `\loop`.

258, 417.

`\rfloor` Ограничитель $\rfloor$. См. `\lfloor`.

176-177, 425, 510.

<code>\rgroup</code>	См. команду <code>\lgroup</code> .	181, 212, <u>425</u> , 510.
<code>\rho</code>	Греческая буква ρ . Но, возможно, Вам больше понравится версия ϱ , которая кодируется <code>\varrho</code> . Только в математической моде.	165, 387, 506.
<code>\rhook</code>	Отношение $\succ$. Только в математической моде. Употребляется вместе с другими отношениями для создания новых символов: <code>\def\longhookleftarrow{\leftarrow</code> <code>\joinrel\relbar\joinrel\rhook}</code> <code>\hookleftarrow\quad\longhookleftarrow\$</code> $\hookleftarrow$ $\longhookleftarrow$ <u>424.</u>	
<code>\right*</code>	Неотделимо от своей пары <code>\left</code> . См. <code>\left</code> . 178-150, 187-188, 206, 233, <u>348</u> , 510.	
<code>\rightarrow</code>	Отношение $\rightarrow$. Только в математической моде.	269, 509.
<code>\Rightarrow</code>	Отношение $\Rightarrow$. Только в математической моде.	269, 509.
<code>\rightarrowfill</code>	Правая стрелка, которая заполняет пробел, <i>созданный другими командами</i> . Следовательно, может работать только внутри <code>\hbox</code> , таблицы и т.д. Примеры: <code>\hbox to 5cm{длинная \rightarrowfill\ стрелка}</code> длинная $\longrightarrow$ стрелка 269, <u>422.</u>	
<code>\rightharpoondown</code>	Отношение $\searrow$. Только в математической моде.	509.
<code>\rightharpoonup</code>	Отношение $\nearrow$. Только в математической моде.	509.
<code>\righthyphenmin*</code>	Задаёт длину в символах наименьшего фрагмента конца переносимого слова. При изменении <code>\language</code> меняется и <code>\righthyphenmin</code> .	324, 430, 529, 530.
<code>\rightleftharpoons</code>	Отношение $\rightleftharpoons$. Только в математической моде.	<u>426</u> , 509.
<code>\rightline</code>	Макрокоманда <code>\line</code> плюс бесконечно растяжимый клей <code>\hss</code> , который сдвигает материал вправо. Может располагаться только в начале абзаца, иначе возникает <code>overfull hbox!</code> Используется, например, в письмах, чтобы поставить дату:	

Протвино, 12 июня 1991 года.

124, 379, 404, 418.

`\rightskip*` Пробел, который TeX систематически ставит в конце каждой строки. Обычно он равен нулю (полное выравнивание). См. описание команд `\leftskip` и `\raggedright`.

123, 325, 379, 421, 462, 492.

`\rlap` Позволяет писать справа от курсора, *не передвигаясь туда*, или, что то же самое, без того, чтобы TeX брал в расчет в своих вычислениях ширину того, что он напечатает! Синтаксис `\rlap{...}`. Внимание: в `\rlap` Вы находитесь в текстовой моде, поэтому не забывайте \$, когда это необходимо: `\rlap{$(\Sigma)$}`. Для примеров см. `\llap`.

102, 226, 294, 381, 418, 389, 487.

`\rm` Команда для перехода на романский шрифт. Команду можно использовать глобально `\rm...` или локально `{\rm...}`. Второй вариант часто используется в математической моде.

17, 186, 197, 382, 416, 430, 479, 484-485, 490, 499.

`\rmoustache` См. команду `\lmoustach`.

181, 425.

`\romannumeral*` Команда `{\romannumeral 1991}` печатает mcmxci. Не забывайте фигурные скобки, иначе 1991 будет склеена с последующим словом. Чтобы получить MCMXCI, пишите:

`\uppercase\expandafter{\romannumeral1991}`

(не пытайтесь это понять). Для “автоматического” copyright, пишите:

`\copyright\ \uppercase\expandafter{\romannumeral\year}`,

что Вам даст © MCMXCIX.

51, 253, 255, 300.

`\root` Для печати радикалов: `$_\root 5\of{1+x^2}$` дает $\sqrt[5]{1+x^2}$ в стиле `\textstyle` и $\sqrt[5]{1+x^2}$ в `\displaystyle` (посвободнее). Если Вам нужен квадратный корень, то используйте команду `\sqrt{...}`:

`$$\root\scriptstyle 3\of{-{q\over 2}+\sqrt{{q^2\over 4}}}{p^3\over 27}}+\root\scriptstyle 3\of{-{q\over 2}-\sqrt{{q^2\over 4}}}{p^3\over 27}}$$`

$$\sqrt[3]{-\frac{q}{2} + \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}}$$

Заметьте использование `\scriptstyle` для некоторого увеличения показателя корня ‘3’ в формуле Кардана. Без этой предосторожности 3 окажется в `\scriptscriptstyle` (и будет слишком маленькой).
159, 215, 387, 426.

`\rq` Правая кавычка, точно такая же, как на Вашей клавиатуре. См. `\lq`.
5, 62, 416, 436, 464.

S

`\S` Символ §. Вместо команды `\S~4`, которая дает § 4, лучше использовать `\S\kern.15em 4`, что дает §4. В этом случае пробел смотрится лучше, и строка между 4 и § не разрывается.
68, 142, 422, 511.

`\sb` Команда нижнего индекса для тех, кто почему-либо не может использовать ‘_’. Только в математической моде.
164, 423, 436.

`scaled` Ключевое слово для увеличения размера шрифта. См. `\font`.
21-22, 75, 329, 415, 505.

`\scriptfont*` Задаёт шрифт, используемый для индексов в `\scriptstyle`. За командой следует целое число, указывающее номер семейства. Синтаксис: `\scriptfont0=\sevenrm`. Это очень важно для того, чтобы общие команды типа `\tm`, `\rm`, `\it` и т.д. работали в математической моде.
184, 203, 253, 322, 383, 416, 485, 513.

`\scriptscriptfont*` Задаёт шрифт, который используется для повторных индексов (`\scriptscriptstyle`). См. выше и ниже.
184, 203, 253, 322, 416, 485, 513.

`\scriptscriptstyle*` В математической моде $\TeX$ работает в четырёх *стилях*, `\displaystyle`, `\textstyle`, `\scriptstyle` и `\scriptscriptstyle`. От стиля зависит используемый шрифт и распределение пробелов (горизонтальных и вертикальных). В формате `plain` используется шрифт в 10 пунктов. Стили `\displaystyle` и `\textstyle` также используют шрифт в 10 пунктов, стиль `\scriptstyle` — шрифт в 7 пунктов, а стиль `\scriptscriptstyle` — шрифт в 5 пунктов. Обычно стиль выбирается автоматически. Но можно и явно задавать стиль по его имени:

`\displaystyle N=2^{\{2^{\{2^2\}}\}}` $N = 2^{2^{2^2}}$

Первый показатель степени печатается в `\scriptstyle`, а все последующие — в `\scriptscriptstyle`. Если Вам не нравятся шрифты в 10, 7 и 5 пунктов, которые использует формат `plain`, можете их заменить командами `\textfont`, `\scriptfont` и `\scriptscriptfont`.

172, 215, 348.

`\scriptspace*` Параметр, равный дополнительному пробелу после индекса. В формате `plain` равен 0.5 pt.

325, 413, 518.

`\scriptstyle*` Стиль печати индексов и показателей степени. Только в математической моде:

`$$\{1\over\Gamma(z)}=ze^{Cz}\prod_1^{\infty}\Bigl(1+\{z\over n\}\over\Bigr)e^{\scriptstyle z\over\scriptstyle n}$$`

$$\frac{1}{\Gamma(z)} = ze^{Cz} \prod_1^{\infty} \left(1 + \frac{z}{n}\right) e^{-\frac{z}{n}}$$

(Мы применили команду `\sur`, см. `\over`.) Команда `\scriptstyle` используется крайне редко. Не просто найти интересный пример (см. `\atop`). В формуле выше дробь `z\over n` является показателем степени, следовательно, печатается в `\scriptstyle`. Если бы не было `\scriptstyle` внутри дроби, то ‘z’ и ‘n’ были бы в `\scriptscriptstyle`, т. е., печатались бы шрифтом в 5 пунктов, а не 7 пунктов. Разборчивость и уравновешенность формулы нарушились бы:

`e^{z\over n}\quad e^{\scriptstyle z\over\scriptstyle n}` $e^{\frac{z}{n}}$ $e^{\frac{z}{n}}$
172, 175, 215, 348.

`\scrollmode` Режим работы, при котором `TeX` выводит обнаруженные ошибки на терминал, не приостанавливая работу. См. команды `\batchmode` и `\scrollmode`.

42, 329.

`\searrow` (*south east arrow*). Отношение $\searrow$. Только в математической моде.

509.

`\sec` Обозначение функции секанс. Только в математической моде.

196, 427.

`\setbox*` (За командой должно следовать целое число от 0 до 255). Позволяет формировать бокс. Но для этого надо иметь некоторый опыт работы. Не используйте `\box0` и `\box1`, так как их уже используют многие макрокоманды. Если Вы вдруг получаете неожиданный результат, измените сначала номер бокса: используйте, скажем, `\box100` вместо `\box0`. Также не используйте боксы с номерами выше 250: эти боксы содержат страницу, которую Вы формируете. Чтобы быть спокойными и ничего не повредить, приватизируйте несколько боксов с помощью команд `\newbox` и используйте эти боксы. Очень полезно запоминать боксы при сложном наборе или при выправке слишком усложненной макрокоманды, которая уже превратилась в настоящий винегрет. Помещайте промежуточные результаты в боксы:

```

\setbox2=\vbox{...}
\setbox2=\vtop{...\box2...}
\setbox2=\hbox{...\box2...\box3...}

```

После этого можно попросить Т_ЕX напечатать результаты (используйте `\copy`, чтобы не потерять содержимое боксов!), или посмотреть на размерность боксов (см. `\showthe` и `\showbox`).

Как запомнить содержание `\vcenter`, т.е., вертикально центрированного бокса? В Т_ЕX'е запрещены конструкции `\setbox2=\vcenter{...}` и `$\setbox2=\vcenter{...}$`. Правильное решение заключается в помещении `\vcenter` во временный бокс:

```
\setbox2=\hbox{$\vcenter{...}$}
```

Надо использовать именно `\hbox`. Если Вы используете `\vbox`, то получите `\box2` шириной `\hsize`, что впоследствии приводит к `overfull \hbox`. Не беспокойтесь об “отверстии”: когда бокс “просверлен”, отверстие не изменяется без специальных манипуляций. Так что для центрированных боксов создайте следующую макрокоманду:

```
\def\cbox#1{\hbox{$\vcenter{#1}$}}
```

После этого можно спокойно писать `\setbox2=\cbox{\hsize=...}`. Не забывайте писать `\hsize` внутри `\cbox`, который содержит текст (даже одну букву, помещенную в `\line`). Ничего этого делать не надо, если в боксе запоминается таблица: `\setbox2=\cbox{\halign{...}}`.

Заключительный совет. Лучше чаще пользоваться промежуточными боксами, чем вкладывать команды друг в друга. Это избавит Вас от больших неприятностей. И еще одно преимущество помещения в бокс: можно менять его размеры! Например, `\setbox2=\vbox{...}\wd2=0pt` говорит Т_ЕX'у, что ширина `\box2` равна нулю: именно так действует команда `\vphantom`. Макрокоманда `\smash` использует тот же принцип.

83, 97, 101, 145, 327, *454-460*.

`\setlanguage*` Эта команда меняет набор правил переноса, т. е., “текущий язык”, но не меняет значение `\language`.

341, 530.

`\setminus` Обратная косая черта (`\`) с пробелами вокруг (бинарное отношение). Только в математической моде.

508.

`\settabs` См. `\cleartabs` и `\columns`.

275, 419, 420.

`\sevenrm` Включает прямой шрифт в 7 пунктов. См. `\rm`.

19, 184, 414, 415, *484-485*.

`\sharp` Знак диэз $\sharp$. Только в математической моде.

479, 507.

- `\shipout*` Примитив для вывода страницы. Используется в макрокомандах.
270, 302, 331, 359, 360.
- `\show*` Для вывода на терминал значения команды во время работы `TeX`'а. Например, результатом `\show\input` будет `> \input=\input.`, потому что `\input` — примитив, а `\show\thinspace` покажет
- ```
> \thinspace=macro:
->\kern .16667em
?
```
- Продолжить работу можно нажатием клавиши CR (“возврат каретки”).  
13, 255, 330, 357.
- `\showbox*` С помощью этой команды можно посмотреть содержание любого регистра бокса. Результат появляется на терминале. Работа продолжается клавишей CR. Например, если в `\box0` была запомнена логограмма `TeX`, то после `\showbox0` на терминале появится следующая информация:
- ```
\hbox(6.83331+2.15277)x18.6108
.\tenrm T
.\kern -1.66702
.\hbox(6.83331+0.0)x6.80557, shifted 2.15277
..\tenrm E
.\kern -1.25
.\tenrm X
```
- Первая строка говорит, что `\box0` — это горизонтальный бокс и показывает его высоту, глубину и ширину. Далее описывается содержимое этого бокса.
83, 146, 279, 330.
- `\showboxbreadth*` Когда `TeX` показывает бокс при диагностике, количество данных управляется параметрами `\showboxbreadth` и `\showboxdepth`. Первый из них, который в формате `plain` равен 5, задает максимальное число элементов, показанных на одном уровне, второй, равный 3, задает самый глубокий уровень.
324, [361](#), 362, 413.
- `\showboxdepth*` См. `\showboxbreadth`.
99, 324, [361](#), 362, 413.
- `\showlists*` Еще одна удобная команда при диагностике, которую можно использовать, чтобы разобраться в “закулисной” деятельности `TeX`'а. Она просит показать списки, которые обрабатывались в текущей и во всех внешних модах, когда была прервана работа.
109, 117, 136, 152, 191, 330, 349.
- `\showthe*` Если Вы добавите в свой текст команду `\showthe\wd2`, `\showthe\ht2` или `\showthe\dp2`, `TeX` остановится и покажет на экране ширину, высо-

ту и глубину `\box2`. Таким способом можно запрашивать информацию почти обо всем. Работа продолжается клавишей CR.

146, 255, 330.

`\sigma` Греческая буква σ . Имеется и другая версия этой буквы — ς , которая кодируется `\varsigma`. Только в математической моде.

232-233, 506.

`\Sigma` Греческая буква Σ . Только в математической моде. Не путайте с символом суммы (который является большим оператором).

`\Sigma`, `\sum`, `\displaystyle\sum` Σ , $\sum$, $\sum$
199, 506.

`\signed` Эта интересная команда не является частью формата plain, поэтому приведем ее определение (*TeXbook*, стр. 106):

```
\def\signed#1 (#2){\unskip\nobreak\hfill\penalty 50
\hskip 2em\null\nobreak\hfil\sl#1/\rm(#2)
\parfillskip=0pt\finalhyphendemerits=0\par}}
```

Эта тонкая макрокоманда характерна для искусства Кнута. Что она делает? Посмотрите:

```
{\sl В тех случаях, когда последняя строка текста
короткая, подпись печатается на этой же строке.
\signed В. Павлова (Москва)}
```

В тех случаях, когда последняя строка текста достаточно короткая, подпись печатается на этой же строке. *В. ПАВЛОВА (Москва)*

```
{\sl И наоборот, если последняя строка настолько длинная,
что подпись на ней уже не помещается,
эта подпись печатается на отдельной строке.
\signed М. ЛИСИНА (Протвино)}
```

И наоборот, если последняя строка настолько длинная, что подпись на ней уже не помещается, эта подпись печатается на отдельной строке.

М. ЛИСИНА (Протвино)

130.

`\sim` Отношение $\sim$. Только в математической моде.

162, 508.

`\simeq` Отношение $\simeq$. Только в математической моде. Также имеются отношения `\cong` и `\approx`, которые дают $\cong$ и $\approx$.

162, 508.

`\sin` Функция синус. Только в математической моде.

`{1\over\sin^2z}=\sum_{-\infty}^{+\infty}{1\over(z-n\pi)^2}`

$$\frac{1}{\sin^2 z} = \sum_{-\infty}^{+\infty} \frac{1}{(z - n\pi)^2}$$

196, 427.

`\sinh` Функция гиперболический синус. Только в математической моде. См. также `\cosh`. Если Вы предпочитаете обозначение “sh”, а не “sinh”, используйте определение

`\def\sh{\mathop{\rm sh}\nolimits}`

196, 427.

`\skew` Иногда требуется поставить акцент над акцентом, чтобы получить символ вроде $\hat{\hat{A}}$. Если написать просто `$(\hat{\hat{A}})$`, то получится $\hat{\hat{A}}$. Правильный результат, т.е., смещение верхнего акцента, получается так: `$(\skew6\hat{\hat{A}})$`. Число 6 выбрано экспериментально: `$(\skew7\hat{\hat{A}})$` давало $\hat{\hat{A}}$, а `$(\skew5\hat{\hat{A}})$` — $\hat{\hat{A}}$.

165, 425.

`\skewchar*` Правильно размещает акценты в математической моде (точно над символами).

254, 322, 324, 328, 415, 484, 502, 503, 516.

`\skip*` Наряду с `\count`-регистрами, `\box`-регистрами, `\dimen`-регистрами и другими, TeX имеет 256 `\skip`-регистров, содержащих клей. Этим регистрам можно присваивать новые значения и использовать их как клей.

143-144, 322, 328, 411-412, 414, 417, 429, 462-463.

`\skipdef*` Аналогично `\countdef` присваивает имя регистру клея.

145, 255, 328, 410-411.

`\sl` Включает наклонный шрифт (*slanted*). Если Вы хотите напечатать курсивом целый абзац, используйте лучше наклонный шрифт, это намного разборчивей.

17, 199, 416, 479, 484-485, 490.

`\slash` Косая черта или слэш: `/(текстовая или математическая мода)`. Можно не запоминать эту макрокоманду, так как есть соответствующая клавиша на клавиатуре. После этого символа разрешено разрывать строку или страницу. В математической моде может увеличиваться с помощью команды `\bigm` и ее вариантов:

`$(\displaystyle(a/b)\!\!\bigm/\!(x/y))$` $(a/b)/(x/y)$

116, 418.

`\smallbreak` Указание на возможное место для разрыва страницы (`\penalty 50`) в сочетании с маленьким вертикальным пробелом `\smallskip`. См. `\bigbreak`.

135, [418](#), 492.

`\smallint` Символ $\int$, очень часто используемый на практике. Только в математической моде.

`$_\smallint$`, `$_\int$`, `$_{\displaystyle\int}$`..... $\int$, $\int$, $\int$
[423](#), 508.

`\smallskip` Маленький вертикальный пробел величиной в 3 pt (приблизительно 1 мм) с допуском плюс-минус один пункт.

89, 97, [134](#), 133, 135, [217](#), [404](#), [417](#), [420](#), [480-482](#).

`\smallskipamount` Содержит маленький вертикальный пробел `\smallskip`.

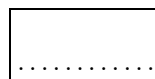
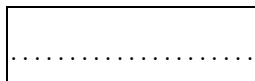
[414](#), [417](#).

`\smash` Легендарная макрокоманда, которая во многих случаях снимет Ваши затруднения. Ее так просто использовать! Запомните хорошенько синтаксис (фигурные скобки обязательны):

`\smash{...}`

Материал, который находится внутри `\smash` печатается нормальным образом, но $\TeX$ при этом считает, что имеет дело с горизонтальной невидимой чертой, у которой высота и глубина равны нулю, и именно эти значения высоты и глубины использует для своих вычислений по формированию страницы.

Вот одно из применений: если ввести `$_{\overrightarrow{AB}}$`, строка, которая содержит вектор $\overrightarrow{AB}$, немного опускается. Результат неприятный. Лучше написать `$_{\smash{\overrightarrow{AB}}}$`, что дает $\overrightarrow{AB}$. На этот раз строка не отодвигается от предыдущей. Еще один пример. Конструкция:

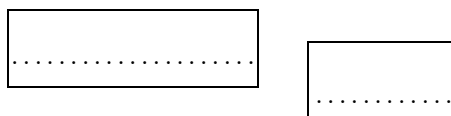


`\vbox{\hsize=60mm\leftline{\box1}\rightline{\box2}}`

занимает много места и может сделать непонятной длинную формулу, разбитую на две части. Чтобы не делать больших изменений в программе, (чтобы ввести, например, туда `\vskip-5pt`), можно лишь немного изменить первоначальную кодировку:

`\vbox{\hsize=60mm\leftline{\box1}\rightline{\smash{\box2}}}`

что даст следующий результат:



Если глубина бокса, к которому применена команда `\smash`, достаточно большая, не забудьте сопроводить эту конструкцию приличным `\vskip`, иначе текст будет слишком близко от бокса, и даже залезет во внутрь.

159, 214, 389, 426.

`\smile` Отношение $\smile$. Только в математической моде. 508.

`sp` Очень маленькая единица измерения, используется во внутренних вычислениях: $65536\text{ sp} = 1\text{ pt}$. 71, 144, 321, 468, 470.

`\sp` Команда верхнего индекса для тех, кто почему-либо не хочет для этой цели использовать клавишу $\wedge$. См. `\sb`. 164, 423, 436.

`\space` Одиночный пробел. Более простая кодировка — это `\_`, т. е., обратная косая черта, а после нее пробел. Работает и в математической моде. 303, 366, 384, 416, 446, 476.

`\spacefactor*` Коэффициент пробела. Используется $\text{\TeX}$ ’ом при распределении клея между словами. 95, 322, 339, 341, 505.

`\spaceskip*` Параметр, который управляет пробелами между словами в данном шрифте. Неотделим от `\xspaceskip`, который занимается пробелами после знаков пунктуации. Один пример см. в `\raggedright`. Приведем еще примеры: в левом столбце значение `\spaceskip` последовательно равно 1 pt, 2 pt, ... 5 pt. В правом столбце эти значения принимает `\xspaceskip`.

Заяц и охотник.	Пиф! Паф! Ой-ой-ой! Умирает зайчик мой.
Заяц и охотник.	Пиф! Паф! Ой-ой-ой! Умирает зайчик мой.
Заяц и охотник.	Пиф! Паф! Ой-ой-ой! Умирает зайчик мой.
Заяц и охотник.	Пиф! Паф! Ой-ой-ой! Умирает зайчик мой.
Заяц и охотник.	Пиф! Паф! Ой-ой-ой! Умирает зайчик мой.

Команды `\frenchspacing` и `\nonfrenchspacing` изменяют эти значения.

95, 325, 378, 421, 501.

`\spadesuit` Символ $\spadesuit$. Только в математической моде. 423, 507.

<code>\span*</code>	Употребляется в преамбуле выравнивания и говорит, что элемент, который следует за <code>\span</code> , должен быть расширен.	255, 284, 289 , 290, 291 , 294, 335, 453.
<code>\special*</code>	Эта макрокоманда позволяет $\TeX$ 'у принимать инструкции, которые прямо к нему не относятся (например, команды печати для Postscript). См. документацию к Вашей системе.	257, 269, 271 , 332.
<code>\splitbotmark*</code>	Когда команда <code>\vsplit</code> разрывает вертикальный список, устанавливаются значения двух величин, <code>\splitfirstmark</code> и <code>\splitbotmark</code> , содержащие тексты первой и последней меток в этом списке. Обе величины нулевые, если не было таких меток.	254, 308 , 332.
<code>\splitfirstmark*</code>	См. <code>\splitbotmark</code> .	254, 308 , 332.
<code>\splitmaxdepth*</code>	Максимальная глубина боксов на расщепляемых страницах.	150, 325, 332, 413 , 429 , 488 .
<code>\splittopskip*</code>	Вертикальный пробел, который помещает $\TeX$ в вершину бокса, когда разрезает этот бокс на два с помощью <code>\vsplit</code> .	151, 325, 332, 413 , 429 , 466 , 488 .
<code>spread</code>	Ключевое слово для некоторого увеличения размера бокса. Например, ширина бокса, полученного командой <code>\hbox spread 5pt{Тра-та-та}</code> , будет на 5 pt больше естественной ширины бокса <code>\hbox{Тра-та-та}</code> .	96, 264, 283, 330.
<code>\sqcap</code>	Отношение $\sqcap$. Только в математической моде.	161 , 508.
<code>\sqcup</code>	Отношение $\sqcup$. Только в математической моде.	161 , 508.
<code>\sqrt</code>	Знак квадратного корня. Синтаксис <code>\sqrt{...}</code> . Примеры см. в <code>\root</code> . Если надо получить несколько радикалов одинаковой высоты, используйте команду <code>\strut</code> или <code>\mathstrut</code> : <code>\sqrt{\strut...}</code> .	158 , 172 , 176 , 189, 204 , 426 , 515 .
<code>\sqsubseteq</code>	Отношение $\sqsubseteq$. Только в математической моде.	508.
<code>\sqsupseteq</code>	Отношение $\sqsupseteq$. Только в математической моде.	508.

<code>\ss</code>	Немецкая буква ß. Например, <code>Straßburg</code> кодируется <code>Stra\ss burg</code> . 67, 421 .
<code>\star</code>	Отношение $\star$. Только в математической моде. 508.
<code>\string*</code>	Преобразует любой символ в символ типа Ord (ординарный). С ее помощью можно превратить имя команды в последовательность символов. Так, например, <code>\string\TeX</code> дает четыре символа категории 12: <code>\</code> , <code>T</code> , <code>e</code> , <code>X</code> . Макрокоманда также используется, когда преобразуют символ в макрокоманду (см. <code>\active</code>). 50-51, 253-254 , 255, 412 , 443 .
<code>\strut</code>	Вертикальная невидимая черта (ширина ее равна нулю), высота которой равна 8.5 pt, а глубина — 3.5 pt. Не путайте <code>\strut</code> с <code>\mathstrut</code> (высота и глубина которой такая же, как у круглой скобки). <code>\mathstrut</code> меньше, чем <code>\strut</code> и посмотреть на нее можно в ее описании. Невидимая черта очень полезна для единообразия некоторых кодировок (см. <code>\bar</code> и <code>\root</code>) и для правильных промежутков между строками таблиц. 102, 172 , 214, 286, 292-293 , 377 , 396 , 419 , 465 , 470 , 492 .
<code>\strutbox</code>	Горизонтальный бокс, содержащий <code>\strut</code> . 377 , 419 , 465 , 485 .
<code>\subset</code>	Отношение $\subset$. Только в математической моде. 162 , 508.
<code>\subseteq</code>	Отношение $\subseteq$. Только в математической моде. 162 , 508.
<code>\succ</code>	Отношение $\succ$. Только в математической моде. 508.
<code>\succeq</code>	Отношение $\succeq$. Только в математической моде. 508.
<code>\sum</code>	Большой оператор суммы $\sum$ и Σ . Только в математической моде. Не путать с буквой Σ : <code>\Sigma</code> , <code>\sum</code> , <code>\displaystyle\sum</code> Σ , $\sum$, $\sum$ 169 , 174 , 180 , 424 , 507 .
<code>\sup</code>	Оператор, который в математических формулах пишет sup прямым шрифтом. Только в математической моде. См. <code>\inf</code> . 196 , 427 .
<code>\supereject</code>	Выводит весь материал, включая и отложенные вставки. 141 , 303-305 , 418 , 477 .

<code>\supset</code>	Отношение $\supset$. Только в математической моде.	508.
<code>\supseteq</code>	Отношение $\supseteq$. Только в математической моде.	508.
<code>\sur</code>	Эта удобная команда не является частью формата <code>plain</code>, тем не менее заслуживает упоминания. Черта дроби, проводимая макрокомандой <code>\over</code> на Макинтоше получается слишком толстой. Макрокоманда <code>\sur</code> это исправляет: $\def\sur{\above .2pt}$ Употребление: пишите в формулах <code>\sur</code> вместо <code>\over</code>. Если Вам надо напечатать текст, который содержит <code>\over</code>, добавьте в начало файла определение <code>\sur</code> вместе с <code>\let\over=\sur</code>, и весь входной файл можно не изменять.	
<code>\surd</code>	Символ $\sqrt{}$. Это часть квадратного корня, но без горизонтальной черты. Только в математической моде: $\text{\$}\surd a+\surd(b+1)\text{\$} \dots\dots\dots \sqrt{a} + \sqrt{b+1}$ См. также <code>\root</code> и <code>\sqrt</code>.	423, 507.
<code>\swarrow</code>	(south west arrow). Отношение $\swarrow$. Только в математической моде.	509.
<code>\t</code>	“Шапочка” над двумя буквами : <code>T\t uut</code> дает $T\hat{u}t$.	66, 422.
<code>\tabalign</code>	Не- <code>\outer</code> версия команды <code>\+</code> . Заменяет <code>\+</code> везде, но не после <code>\settabs</code> .	420, 420.
<code>\tabs</code>	Табуляция поддерживается командами <code>\+</code> и <code>\settabs</code> с помощью регистра <code>\box\tabs</code> — бокса, заполненного пустыми боксами, ширина которых равна ширине колонок в обратном порядке. Можно проверить текущую табуляцию, сказав <code>\showbox\tabs</code>. Это записывает ширины колонок в протокольный файл справа налево.	279, 419-420.
<code>\tabskip*</code>	Пробел, который $\TeX$ располагает в таблице слева, справа и между колонками. Позволяет создавать таблицы заданной ширины.	256, 282-283, 289, 293, 325, 334, 340, 419.
<code>\tan</code>	Функция тангенс: $\tan$. Только в математической моде. Те, кому нравятся обозначение <code>tg</code> , могут использовать определение:	

<code>\def\tg{\mathop{\rm tg}\holimits}</code>	196, 427.
<code>\tanh</code>	Функция гиперболический тангенс $\tanh$. Конечно же, только в математической моде. Если же Вам нравится th , см. выше. 196, 427 .
<code>\tau</code>	Греческая буква τ . Только в математической моде. 506.
<code>\tenbf</code>	Имя жирного шрифта в 10 пунктов. Синтаксис: <code>{\tenbf ...}</code> для локального и <code>\tenbf ...</code> для глобального изменения. Удобнее, однако, пользоваться командой <code>\bf</code> .
<code>\tenit</code>	Имя курсивного шрифта в 10 пунктов. См. выше, заменяя <code>\bf</code> на <code>\it</code> .
<code>\tenrm</code>	Имя прямого шрифта в 10 пунктов. См. <code>\tenbf</code> , заменяя <code>\bf</code> на <code>\rm</code> . 19, 36, 57, 184, 300, 397 , 414 , 416 , 484 .
<code>\tensl</code>	Имя наклонного шрифта в 10 пунктов. См. <code>\tenbf</code> , заменяя <code>\bf</code> на <code>\sl</code> . 19, 414 , 416 , 484 .
<code>\tentt</code>	Имя “машинописного шрифта” (typewriter) в 10 пунктов. См. <code>\tenbf</code> .
<code>\TeX</code>	Логограмма $\mathrm{T\!E\!X}$. Эта эмблема “приклеивается” к следующему слову, поэтому пробел после нее получается так: <code>{\TeX}</code> , <code>\TeX{}</code> или <code>\TeX\ .</code> 11-10, 25, 83, 243, 268, 404-405 , 422 , 489 .
<code>\textfont*</code>	Уточняет шрифты, которые должны вызываться общими командами типа <code>\rm</code> , <code>\it</code> и т.д., в <code>\textstyle</code> или <code>\displaystyle</code> . 184, 203, 225, 253, 322, 416 , 484-485 , 513-514.
<code>\textindent</code>	Симпатичная макрокоманда для начала абзаца. Альтернатива для приверженцев <code>\item</code>! Кодировка этого абзаца начинается так: <code>\textindent{\${\bullet}\$}</code> Альтернатива (В целях демонстрации значение <code>\parindent</code> временно заменено на 1 cm). Обратите внимание — первая строка отступает на <code>\parindent</code>, а большая точка <code>\${\bullet}\$</code> помещена справа от абзацного отступа. Следующие строки печатаются без отступа. 142, 420 .
<code>\textstyle*</code>	Стиль, который использует $\mathrm{T\!E\!X}$, когда печатает математические формулы в текущем тексте. Другой стиль печати математики, при котором формулы выделяются на отдельных строках — это <code>\displaystyle</code> . В <code>\textstyle</code> $\mathrm{T\!E\!X}$ старается не слишком раздвигать строки, т.е., соблюдать нормальное расстояние между ними. При внимательном рассмотрении формул, напечатанных в этом стиле, вы увидите, что индексы и показатели степеней не такие высокие, как в <code>\displaystyle</code> , что

числитель и знаменатель ближе к дробной черте и набраны меньшим шрифтом, и т.д. В этом руководстве можно найти примеры формул в обоих стилях.

172, 348, 388.

`\the*` Это аналог `\TeX`'а для *print* в BASIC и *write* в PASCAL. Если число запомнено в регистре `\count25`, `\the\count25` напишет содержимое этого регистра, не уничтожая его. См. также макрокоманду `\number`.
254-256, 257, 440, 441, 493.

`\theta` Греческая буква θ . Вариант ϑ кодируется `\vartheta`. Только в математической моде.
156, 196, 506.

`\Theta` Заглавная греческая буква Θ . Только в математической моде.
506.

`\thickmuskip*` Большой пробел в математической моде. Он может кодироваться `\;` и приблизительно равен 5 му.
203, 325, 413, 520.

`\thinmuskip*` Маленький пробел в математической моде (“мини-пробел”). Он может кодироваться `\,` и приблизительно равен 3 му. “Отрицательный мини-пробел”, записывается `\!`.
203, 325, 413, 520.

`\thinspace` Пробел, равный $\frac{1}{6}$ квадрата. Это нематематический эквивалент математического “мини-пробела” `\`:
`x\thinspace x` `$x\,x$` `xx xx`
5, 13, 365, 372, 417, 479.

`\tilde` Акцент тильда. Только в математической моде :
`$\tilde x+\tilde y$` $\tilde x + \tilde y$
Увеличенная `\widetilde` вытягивается вплоть до трех символов:
`$\widetilde{xyz}=\widetilde{xy}*\tilde z$` $\widetilde{xyz} = \widetilde{xy} * \tilde z$
164, 198.

`\time*` Время в минутах от полуночи. Значение запоминается в момент запуска `\TeX`'а и далее не модифицируется. Поэтому, к сожалению, этим нельзя пользоваться для определения времени работы команд. Синтаксис: `\the\time` если речь идет о переменной, а не о макрокоманде:
`\the\time` 778
324, 413.

`\times` Бинарная операция $\times$. Только в математической моде.
161, 508.

<code>to</code>	Ключевое слово, используется в конструкциях <code>\hbox to ...</code> и <code>\vbox to ...</code> для задания горизонтального бокса заданной ширины или вертикального бокса заданной глубины.	96, 257, 265, 283, <u>327</u> , <u>330</u> .
<code>\to</code>	Отношение $\rightarrow$. Это еще одно определение <code>\rightarrow</code> .	<u>163</u> , <u>427</u> , 510.
<code>\toks*</code>	В Т _Е X'е есть 256 “регистров списков элементов”, от <code>\toks0</code> до <code>\toks255</code> , таких, что списки элементов могут перетасовываться без передачи через аппарат чтения Т _Е X'а. Есть также такая команда <code>\toksdef</code> , что <code>\toksdef\catch =22</code> делает <code>\catch</code> эквивалентной <code>\toks22</code> , а также команда <code>\newtoks</code> , которая назначает новый регистр списка элементов.	251, 256, 312, 328.
<code>\toksdef*</code>	См. <code>\toks</code> .	251, 255, 328, <u>411</u> , <u>444</u> .
<code>\tolerance*</code>	Порог допустимой плохости при формировании строк и страниц. Синтаксис: <code>\tolerance=200</code> или <code>\tolerance 200</code> (знак ‘=’, как всегда, необязателен). Когда Т _Е X пытается разбить абзац на строки, он вычисляет числовое значение, определяющее качество каждой из попыток, и выбирает ту, плохость (<i>badness</i>) которой не превышает <code>\tolerance</code> (или, если это невозможно, <code>\pretolerance</code>). Это переменная целого типа от 0 (в этом случае все блокируется!) до 10000 (в этом случае все идет). В формате <code>plain</code> <code>\tolerance=200</code> . В русских текстах значение <code>\tolerance</code> следует увеличить, скажем, до 1000, потому что русские слова в среднем немного длиннее английских. Если текст помещается в узкий <code>\vbox</code> , допуск также надо увеличить, иначе у Т _Е X'а будут трудности с формированием страницы: $\vbox{\hsize 5cm\tolerance 5000....}$ Здесь изменение допуска является <i>локальным</i> и ограничено пределами <code>\vbox</code> . См. также <code>\pretolerance</code> .	38, 113, 116, <u>119</u> , 131, 323, <i>378</i> , <i>396</i> , <i>406</i> , <i>413</i> , <i>526</i> .
<code>\top</code>	Символ $\top$. Только в математической моде. Также имеются <code>\perp</code> , <code>\dashv</code> и <code>\vdash</code> , которые дают $\perp$, $\dashv$ и $\vdash$.	507.
<code>\topglue</code>	Вертикальный клей, используемый для начала страницы (ее верха).	<u>404</u> , 418.
<code>\topins</code>	См. <code>\footins</code> .	305, <u>429</u> , <i>430</i> .
<code>\topinsert</code>	Применяется, чтобы оставить место для иллюстрации. Синтаксис: <code>\topinsert\vglue 5cm\centerline{{\it Fig} 7}\bigskip\endinsert</code>	

Когда $\TeX$ встречает эту команду, он пропускает текст от `\topinsert` до `\endinsert` и печатает текст после `\endinsert`. Когда страница заканчивается, он начинает следующую страницу с материала, расположенного между `\topinsert` и `\endinsert`. Поэтому вместо `\vskip` необходимо использовать `\vglue` (другое решение — это `\null\vskip`). Макрокоманда `\midinsert` делает почти то же самое, но $\TeX$ сам выбирает место для вставки. Команда `\topinsert` не дает выбора.

140, 299, 429.

`\topmark*` См. `\mark`.

254, 307, 332.

`\topskip*` Клей, вставляемый сверху страницы перед самым первым боксом.

138-139, 151, 305, 325, 413.

`\tracingall` Если с горя начать с `\tracingall`, $\TeX$ покажет все, что он делает.

146, 362, 430.

`\tracingcommands*` Позволяет записывать в протокольный файл `log` описание всех макрокоманд, которые использует $\TeX$. Применяется в запутанной ситуации, чтобы разобраться в сообщениях $\TeX$ 'а. Чтобы команда заработала, введите `\tracingcommands=1`. При `\tracingcommands=2` подробностей будет еще больше. Остановить трассировку можно, установив значение в ноль.

108-109, 252, 324, 357.

`\tracinglostchars*` Если положительно, указывает $\TeX$ 'у записать символ, который пропускается, поскольку не должен появляться в текущем шрифте.

324, 360, 413, 470.

`\tracingmacros*` Если положительно, $\TeX$ отмечает в протокольном файле, когда раскрывает макрокоманду и когда читает аргументы макрокоманды.

243-244, 252, 324, 392.

`\tracingonline*` Обычно диагностика записывается в протокольный файл. Чтобы она выводилась на терминал, задайте `\tracingonline=1`.

146, 252, 324, 362.

`\tracingoutput*` Если положительно, показываются все боксы, отправляемые в `dvi`-файл.

302, 324, 360.

`\tracingpages*` Если положительно, показывается итог вычислений $\TeX$ 'а по формированию страницы.

137, 150, 324, 362.

`\tracingparagraphs*` Если положительно, показывается итог вычислений $\TeX$ 'а по формированию абзаца.

121, 324, 362.

- `\tracingrestores*` Если положительно, показываются все присваивания, восстанавливаемые после окончания группы.
324, [360](#), [362](#).
- `\tracingstats*` Если положительно, показывается статистика памяти.
324, [359](#), [362](#), [451](#).
- `\triangle` Символ Δ . Только в математической моде. Не путайте с греческой большой буквой `\Delta`:
 `$\Delta$, $\triangle$ .....` Δ , $\triangle$
и с бинарным отношением `\bigtriangleup`, которое похоже на символ `\triangle`, но окружено пробелами.
507.
- `\triangleleft` Бинарная операция $\triangleleft$. Только в математической моде.
508.
- `\triangleright` Бинарная операция $\triangleright$. Только в математической моде.
508.
- `true` Ключевое слово, которое, если стоит перед единицей измерения в размере, не дает этому размеру увеличиваться при увеличении выходного документа командой `\magnification`. Например, `\hsize=6.5 true in`.
74, [321](#), [477-478](#).
- `\tt` Вызывает машинописный шрифт. Синтаксис: `\tt ...` для глобального и `{\tt ...}` для локального использования.
17, 67, 138, 199, [404](#), [416](#), [447](#), [484-485](#), [501](#).
- `\ttraggedright` Для облегчения верстки страницы, набранной шрифтом `\tt`. Синтаксис: начинайте текст с `\ttraggedright` (команда `\tt` уже не нужна).
[421](#).
- ## u
- `\u` Акцент коротких гласных : `\u a`, `\u e`, `\u i`, `\u o` дают á , é , í и ó .
66, [422](#).
- `\uccode*` См. `\lccode`.
51, 254, 322, 409, [412](#), [443](#), [463](#).
- `\uchyph*` Параметр, который положителен, если переносимое слово начинается с заглавной буквы.
324, [413](#), [529](#).
- `\underbar` Макрокоманда, которая помещает свой аргумент в `\hbox` и подчеркивает его:
`\underbar{Красота} ...` Красота спасет мир!

В математической моде есть макрокоманда `\underline`:

`$\underline{u+v}_{\,t}$` $\underline{u+v}_t$
 290, 385, 418.

`\underbrace` Рисует под формулой горизонтальную фигурную скобку. Только в математической моде. Например:

`$\displaystyle\underbrace{a-2b}_{>0}`
`+\underbrace{y-z+t}_{>0}>0$` $\underbrace{a-2b}_{>0} + \underbrace{y-z+t}_{>0} > 0$

Нематематический аналог этой команды — `\downbracefill`. Горизонтальная фигурная скобка сверху называется `\overbrace` в математической моде и `\upbracefill` в текстовой.

211, 268, 425.

`\underline*` Команда, которая подчеркивает свой аргумент. Только в математической моде:

`$\underline{b+q}=\underline{b}+\underline{q}$` $\underline{b+q} = \underline{b} + \underline{q}$

Чтобы подчеркивание было на одном уровне, используйте `\strut` или другие определенные Вами невидимые вертикальные черты:

`\def\strit{\vrule depth 1.5pt width 0pt}`
`$\underline{\strit b+q}=\underline{\strit b}+\underline{\strit q}$` $\underline{b+q} = \underline{b} + \underline{q}$
 158, 172, 347, 515.

`\unhbox*` (С целым числом между 0 и 255). Позволяет “вынимать” материал, содержащийся в `\hbox`. Т_ЕX может растягивать или сжимать пробелы, использовать бесконечную растяжимость и выбирать хорошие разбиения абзацев на строки, но при этом он никогда не *перекomпoнoвывaет* боксы (если Вы пишете `\it\unhbox1{...}`, команда `\it` не влияет на содержимое бокса). Работает только с регистрами типа `\box`.

145, 335, 339, 420, 422, 427, 469.

`\unhcopy*` Если написать `\unhbox5`, содержимое `\box5` теряется. Если Вы этого не хотите, используйте `\unhcopy`.

146, 335, 339, 419.

`\unkern*` Примитив для “внутреннего употребления” при создании макрокоманд. Убирает из текущего списка керн, если он последний.

332.

`\unpenalty*` То же, что и `\unkern`, но относительно штрафа.

332.

`\unskip*` Уничтожает последний элемент списка элементов(tokens), если это пробел (горизонтальный или вертикальный). Мало кто правильно обращается с пробелами до и после знаков пунктуации. Но можно помочь рассеянными и невежественным:

`\catcode'\;=\active \def;\{\unskip\kern .2em\string;\ }`

После этой команды, что бы Вы ни делали, пробелы будут всегда правильными. Докажем это: если напечатать `aaa;bbb;ccc;ddd` (заметьте отсутствие пробелов!), результат все равно окажется правильным:

`aaa;bbb;ccc;ddd`

Команда `\unskip` устраняет все пробелы (правильные или нет), которые встречаются перед точкой с запятой. Затем перед точкой с запятой помещается пробел в 2 pt (`\kern 2pt`), а после точки с запятой — обычный пробел. Пробел перед точкой с запятой должен быть равен $\frac{2}{3}$ обычного пробела (который, в свою очередь, для шрифтов в 10 пунктов приблизительно равен 3 pt). Т_ЕX не разрывает строки на керне, поэтому точка с запятой всегда будет “приклеена” к слову, за которым она стоит. Наконец, команда `\string` указывает, что речь идет о символе точка с запятой, а не о макрокоманде “;” (без этого Т_ЕX “укусил бы себя за хвост”).

265, [332](#), 340, [374](#), [461](#), [490](#).

`\unvbox*` Аналогично команде `\unhbox`, но для вертикальных боксов. Если Вы написали `\setbox5=\vbox{...}`, команда `\unvbox5` поместит в тексте содержание *уже* созданного `\box5`, *ничего не изменяя во внутреннем расположении строк* `\vbox`, за исключением пробелов и бесконечно растяжимого клея. Пример использования см. в `\vsplit`. См. также `\unhbox` для более детального понимания механизма.

146, 303, [334](#), 340, [419](#), [427](#), [429](#), [430](#), [460](#), [469](#), [487](#).

`\unvcopy*` Аналогично `\unhcopy`, но для `\vbox`.

146, [334](#), 340, [427](#).

`\uparrow` Отношение $\uparrow$. Только в математической моде. Конечно же имеется и отношение `\downarrow`($\downarrow$). Эта стрелка может увеличиваться с помощью `\big` (или ее вариантов), а также конструкции `\left...\right`. Не забудьте, что вертикальная стрелка является “бусиной”, просверленной в середине (для примера см. `\big`).

177, [182](#), [425](#), 509.

`\Uparrow` Отношение $\Uparrow$. Только в математической моде. См. предыдущую команду.

177, [425](#), 509.

`\upbracefill` Рисует большую фигурную скобку, повернутую вверх. Используется в текстовой моде (в математике пользуются командой `\underbrace`):

```
\setbox2=\hbox{0, more! 0, воздух!}
\setbox3=\hbox to\wd2{\upbracefill}
\vbox{\box2\nointerlineskip\box3}
```

После этих команд получается:

О, море! О, воздух!

Обратите внимание на последовательность действий: сначала создается бокс 2, так как бокс 3 должен знать ширину бокса 2. Фигурная скобка сверху рисуется командой `\downbracefill`. Ее математический аналог называется `\overbrace`.

268-269, [422](#).

`\updownarrow` Отношение $\updownarrow$. Только в математической моде. Может увеличиваться командами `\big` или `\left...\right`.

177, [425](#), 509.

`\Updownarrow` Отношение $\Updownarrow$. Только в математической моде. См. замечание выше.

177, [425](#), 509.

`\uplus` Бинарная операция $\uplus$. Только в математической моде:

`\uplus`, `\biguplus`, `\displaystyle\biguplus` $\uplus$, $\biguplus$, $\biguplus$
508.

`\uppercase*` Текст, который следует за этой макрокомандой, печатается большими буквами. Чтобы ограничить поле деятельности этой макрокоманды, применяйте ее *локально*, используя группу `{\uppercase...}`. См. в `\romannumeral` пример автоматической печати знака авторского права. Аналогичная команда для маленьких букв — это, очевидно, `\lowercase`.

51, 255, 258, [331](#), 368, 409, [412](#), [441](#), [443](#), [463](#).

`\upsilon` Греческая буква υ . Только в математической моде. Она очень похожа на букву ‘v’, когда та печатается в математической моде: `$v$` и `$\upsilon$` дают v и υ .

506, 575.

`\Upsilon` Заглавная греческая буква Υ . Только в математической моде. Жаль, что такие забавные символы редко используются математиками.

506.

V

`\v` Ставит акцент-“птичку” над следующей за ней буквой: `\v Seby\v sev` дает $\check{\text{Seby}}\check{\text{sev}}$. Это нематематическая версия макрокоманды `\check`.

66, [422](#).

`\vadjust*` Вставляет вертикальный список сразу за строкой, которая содержит эту команду. Например, `\vadjust{\vskip 2mm}` отодвигает следующую строку на 2 мм.

Врач, исцелился сам: и ты исцелишь также и своего больного. Было бы лучшей помощью для него, чтобы увидел он глазами того, кто сам себя исцеляет.⁷

Если надо прервать страницу сразу после строки, которая содержит некоторое слово, поставьте после этого слова `\vadjust{\goodbreak}` или `\vadjust{\vfill\eject}`. У этой команды есть интересное применение: как вставить примечание на полях? Пусть левое поле равно 2 см. Определим такую макрокоманду:

```
\def\leftnote#1{\vadjust{\setbox1=\vtop{\hsize 20mm
\parindent=0pt\sevenrm\baselineskip=9pt
\rightskip=4mm plus 4mm#1}
\hbox{\kern-2cm\smash{\box1}}}
```

Параметром здесь будет текст вставки, а саму вставку делает команда `\leftnote{...}`. Текст запоминается в `\box1`, боксе, ширина которого равна 20 мм, и состоит из нескольких строк (без отступов, прямой шрифт в 7 пунктов, расстояние между базовыми линиями 9 пунктов). Обратите внимание на пробел `\rightskip`, который $\TeX$ добавляет в конце каждой строки: он должен быть не меньше 4 мм. Так как ширина бокса равна 20 мм, строка текста никогда не превышает $20 - 4 = 16$ мм. Как только бокс запомнен, он “сплюсчивается” командой `\smash`. Иначе говоря, $\TeX$ думает, что имеет дело (с точки зрения размеров) с горизонтальной чертой нулевой толщины. Команда `\vadjust\llap{...}` вставляет такой фантом между двумя строками на полях слева от текста.

Один совет: сама по себе эта макрокоманда работает разумно. Но нельзя гарантировать, что Вам удастся заставить эту сложную макрокоманду работать в боксе. Если Вам это все-таки нужно, делайте это в два этапа: сначала отдельно создайте бокс, а затем отправляйтесь туда из своего `\vadjust{...}`. В $\TeX$ е есть золотое правило: если не хотите неприятностей, не злоупотребляйте вложенностью макрокоманд.

В человеке все должно быть прекрасно ...

Осталась последняя деталь: первая строка примечания не на одном уровне с соответствующей строкой текста. Она располагается между строками. Чтобы правильно расположить примечание, надо заменить `\smash{\box1}` на `\smash{\raise .5ex\box1}`: результат рядом.

А как вставлять примечания на правом поле? Сначала тактика такая же. Но когда примечание скомпоновано, курсор посылают направо `\line{\hfil...}` и пишут на правом поле с помощью `\rlap`. Изобразим это подробно:

```
\def\rightnote#1{\vadjust{\setbox1=\vtop{\parindent=0pt
\hsize 17mm\sevenrm\baselineskip=9pt
\leftskip=4mm \rightskip=0mm plus 4mm#1}
\line{\hfill\rlap{\smash{\raise .5ex\box1}}}}
```

Каждая строка начинается пропуском величиной 4 мм (`\leftskip=4mm`): это зрительно отделяет примечание от текста. Как и для предыду-

⁷ Ф. Ницше, *Так говорил Заратустра*

	шей макрокоманды, вставка примечания на полях делается записью <code>\rightnote{...}</code> .	117, 128, 133, 134, 143, 308, <u>333</u> , 378, 462, 467, 529.
<code>\valign*</code>	Аналогично <code>\halign</code> , но для вертикального выравнивания. Синтаксис такой же. Применяется очень редко, так как это требует редкого качества — уметь поворачивать свой мозг на 90°.	295, <u>335</u> , 340, 391, <i>397</i> , <i>467</i> .
<code>\varepsilon</code>	Греческая буква ε . Только в математической моде. Есть также команда <code>\epsilon</code> , которая дает ϵ .	156, 506.
<code>\varphi</code>	Греческая буква φ . Только в математической моде. Имеется также команда <code>\phi</code> : ϕ .	156, <i>178</i> , 506.
<code>\varpi</code>	Греческая буква ϖ . Только в математической моде. Можно также использовать команду <code>\pi</code> : π .	506.
<code>\varrho</code>	Греческая буква ϱ . Только в математической моде. Альтернатива — команда <code>\rho</code> , которая дает ρ .	156, 506.
<code>\varsigma</code>	Греческая буква ς . Только в математической моде. Имеется также <code>\sigma</code> , которая дает σ .	506, 575.
<code>\vartheta</code>	Греческая буква ϑ . Только в математической моде. Имеется также <code>\theta</code> , которая дает θ .	156, 506.
<code>\vbadness*</code>	Порог плохости, свыше которого вертикальные боксы считаются переполненными.	<i>323</i> , <i>413</i> , <i>467</i> , <i>488</i> .
<code>\vbox*</code>	Боксы, которые находятся в <code>\vbox</code> , накладываются друг на друга вертикально. Если в <code>\vbox</code> находится какой-нибудь текст (единственный символ — это тоже текст!), он сначала преобразуется в строки <code>\line</code> (<code>\hbox</code> шириной <code>\hsize</code>), а затем эти строки располагаются одна под другой. Ширина <code>\vbox</code> равна наибольшей ширине входящих в него боксов. Следовательно, ширина бокса <code>\vbox</code> , который содержит по крайней мере один символ, равна <code>\hsize</code> . См. также <code>to</code> и <code>spread</code> .	81, 100, 126, 182, 230, 263-264, 330, 456-457.
<code>\vcenter*</code>	Делает вертикальный бокс, “продырявленный” в середине, а не внизу (<code>\vbox</code>) или вверху (<code>\vtop</code>). Жаль, но эта конструкция работает только в математической моде: <code>\$...\vcenter{...}...\$</code> .	

Запомнить `\vcenter` в регистре боксов, можно, например, командой:

`\setbox5=\hbox{$\vcenter{...}$}`

См. также `to` и `spread`.

182, 192, 205, 230, 265, 287, 347, 427, 515.

`\vdash` Отношение $\vdash$. Только в математической моде. Имеется также отношение `\dashv` : $\dashv$.

508.

`\vdots` Для получения трех точек, расположенных вертикально ($\vdots$). Только в математической моде.

213, 424.

`\vec` Помещает маленькую стрелку над следующим символом:

`$(\vec\imath,\vec\jmath,\vec k)$` $(\vec i, \vec j, \vec k)$

Имеется и макрокоманда `\overrightarrow`, которая автоматически вычисляет длину своей стрелки:

`$(\overrightarrow{AB}, \vec{AB})$` $\overrightarrow{AB}, \vec{AB}$
164.

`\vee` Отношение $\vee$. Только в математической моде. Обратное отношение ($\wedge$) — это `\land`.

161, 508.

`\vert` Ограничитель $|$. То же самое, что и вертикальная черта на клавиатуре, только в математической моде. Может увеличиваться командами `\big` или `\left...\right`:

`|||u|||=\sup_{x\neq 0}\left\{\frac{\|u(x)\|}{\|x\|}\right\}` $|||u||| = \sup_{x \neq 0} \left\{ \frac{\|u(x)\|}{\|x\|} \right\}$
177, 181, 222, 425, 510.

`\Vert` Ограничитель $\|$. Более простая кодировка — `\|`. Только в математической моде. Может увеличиваться под действием `\big` или уже известной конструкции `\left...\right`:

`$$u\cdot v=\left\|\frac{u+v}{2}\right\|^2+\left\|\frac{u-v}{2}\right\|^2$$`

$$u \cdot v = \left\| \frac{u+v}{2} \right\|^2 + \left\| \frac{u-v}{2} \right\|^2$$

142, 177, 181, 425, 510.

`\vfil*` Слабо бесконечно растяжимый вертикальный клей.

90, 91, 135-136, 305, 333, 340, 488.

- `\vfill*` Вертикальный клей, бесконечно более растяжимый, чем предыдущий. Когда они конкурируют в одном и том же боксе, `\vfill` полностью сжимает все `\vfil`. Для окончания одной страницы и начала новой, напечатайте `\vfill\eject`.
32, 33, 90, 91, 305, 333, 340.
- `\vfillneg*` Примитив, который отменяет растяжимость `\vfill`. См. `\hfillneg`.
90, 91, 136, 333, 340.
- `\vfootnote` Заменяет команду `\footnote`, когда делается сноска с плавающим текстом, т.е., текстом, окруженным командами `\midinsert... \endinsert` или `\topinsert... \endinsert`.
142, 429.
- `\vfuzz*` Команда `\vfuzz=1mm` разрешает $\TeX$ 'у не сигнализировать об `overfull \vbox`, которые превышают `\vsize` не более чем на один миллиметр. Используется редко. Напротив, ее горизонтальный аналог `\hfuzz` прекрасно избавляет нас от лишних сообщений.
325, 413.
- `\vglue` Пробел, который не исчезает при разрыве страницы. Например, `\vglue 3cm` в самом верху страницы будет оставлен, в то время как `\vskip 3cm` и `\kern 3cm` исчезнут. Обратите внимание на синтаксис: не пишите `\vglue=3cm`, знак равенства здесь не нужен! Команда `\vglue` необходима, когда применяется `\topinsert` или `\midinsert`.
404, 417.
- `\voffset*` Для вертикального смещения текста на листке бумаги. См. `\hoffset`.
299, 301, 325, 406, 476.
- `\vphantom` Макрокоманда с аргументом, рисующая невидимую черту (`width=0pt`) с высотой и глубиной, как у аргумента макрокоманды. Синтаксис: `\vphantom{X}x^{2^K}` или еще `\vphantom{Моя душа}`.
214-215, 250, 382, 426.
- `\vrule*` Рисует вертикальную черту. Работает в горизонтальной моде, следовательно, при использовании этой команды надо находиться внутри абзаца, в `\hbox` или внутри таблицы. Можно задавать размеры черты:
`\vrule height 12pt depth 5pt width 1pt`
Внимание: перед `height`, `depth` и `width` нет обратной косой черты. Если `height` отсутствует, $\TeX$ рисует черту до потолка бокса, который ее содержит. Аналогично, если отсутствует `depth`, черта опускается до нижней границы включающего ее бокса. Отсутствие `width` — это особый случай: по умолчанию значение ширины равно 0.4 pt. Чтобы получить невидимую черту, надо задать ей ширину 0 pt. И последнее

замечание: черта в вертикальном боксе не отделяется от боксов сверху и снизу никакими пробелами.

80, 106, 182, 263, 291, 333, 335, 422, 460, 491.

\vsize* Это высота страницы. Не используйте `\vsize` для получения бокса заданной высоты. Пишите `\vbox to 5cm{...}`, `\vtop to 45pt{...}` или `\vcenter to\dimen3{...}`. Страница получается командой `\vbox to\vsize`. Если Вы не зададите высоту “вертикального” бокса (лучше сказать, бокса, построенного в вертикальной моде), $\TeX$ сам молча определит ее.

137, 299, 301, 325, 404, 413, 470, 476, 483, 486, 418.

\vskip* Команда вертикального пробела. Синтаксис:

`\vskip 5mm plus 1mm minus 2mm`.

Части `plus` и `minus` необязательны. Внимание: кодировка `\vskip=5mm` неправильна (не нужен знак “=”). Части `plus` и `minus` придают вертикальным пробелам эластичность: $\TeX$ сможет сжимать или расширять пробелы, что очень помогает при формировании страницы. Можно в необязательную часть поместить и бесконечно растяжимый клей, написав, например, `plus 1.4fil` или `minus 3fil`. Имеется также `\vfil`, который эквивалентен `\vskip 0mm plus 1fil`, и специальный бесконечный клей для “положительного или отрицательного вытягивания” `\vss`, который эквивалентен команде `\vskip 0mm plus 1fil minus 1fil`.

32, 89, 105, 228, 333, 340.

.

\vsplit* Для расщепления бокса на две части.

150, 264, 308, 329, 466, 487.

\vss* Это вертикальный бесконечный клей, растяжение которого может быть отрицательным! См. `\vskip` и `\hss`.

90, 304, 333, 340.

\vtop* Создает вертикальный бокс, “продырявленный” сверху. Предположим, Вы хотите поместить подпись в конце страницы, как здесь:

Подпись: А. А. Логунов,
директор Института физики
высоких энергий

Для этого сначала соберите в `\vtop` строки, предварительно отправленные в `\hbox` (ширина этого `\vtop` будет равна ширине наиболее длинной строки, а не `\hsize`!):

```
\setbox1=\vtop{
  \hbox{А. А. Логунов,}
  \hbox{директор Института физики}
  \hbox{высоких энергий}}
```


Другое решение — это использовать табуляцию, кодируя:

```
\setbox1=\vtop{
  \+ А. А. Логунов,\cr
  \+ директор Института Физики\cr
  \+ Высоких Энергий\cr}
```

Проденьте затем через этот бокс натянутую нитку и прижмите его к правому полю бесконечным клеем:

```
\line{Подпись:\hfill\box1\qqquad}
```

Если Вы вместо `\vtop` будете использовать `\vbox`, получится вот что:

А. А. Логунов,
директор Института Физики
Высоких Энергий

Подпись:

См. также `to` и `spread`.

101, 182, 264, 330, 396.

W

`\wd*` (С числом между 0 и 255) Ширина бокса памяти. См. `\box` и `\setbox`.
145, 322, 456-457, 391, 488.

`\wedge` Бинарный оператор $\wedge$. Только в математической моде. Такое же действие, как у `\land`:
$$\text{\$}\text{\textit{u}}\text{\wedge} v, \text{\textit{u}}\text{\land} v\text{\$} \dots\dots\dots u \wedge v, u \wedge v$$

161, 508.

`\widehat` Большой акцент сиркомфлекс, который может располагаться не больше чем над тремя символами. Только в математической моде. Используется для записи углов:
$$\text{\$}\text{\hat{x}}, \text{\widehat{X}}, \text{\widehat{AM!B}}\text{\$} \dots\dots\dots \hat{x}, \hat{X}, \widehat{AMB}$$

165, 425, 505.

`\widetilde` Большая тильда. Также может располагаться максимум над тремя символами. Только в математической моде. См. `\tilde`:
$$\text{\$}\text{\tilde{x}}, \text{\widetilde{X}},$$

$$\text{\$}\text{\widetilde{XY}}\text{\$}, \text{\widetilde{XYZ}}\text{\$} \dots\dots\dots \tilde{x}, \widetilde{X}, \widetilde{XY}, \widetilde{XYZ}$$

165, 425.

`\widowpenalty*` Один из штрафов, которые $\text{\TeX}$ использует при формировании строк. В формате `plain` равен 150.
128, 138, 323, 413.

`width` Ключевое слово для задания ширины линейки. См. `\hrule` и `\vrule`.
263, 334, 400.

`\wp` Функция $\wp$ Вейерштрасса. Только в математической моде:
$$\text{\$}\text{\wp}(z)=\{1\over z^2\}+\sum_{\omega\in \Omega^*}$$

`\biggl[{1\over(z-\omega)^2}-{1\over\omega^2}\biggr]$$`

$$\wp(z) = \frac{1}{z^2} + \sum_{\omega \in \Omega^*} \left[\frac{1}{(z-\omega)^2} - \frac{1}{\omega^2} \right]$$

507.

`\wr` Бинарная операция $\wr$. Только в математической моде. Применяется для записи “косого произведения” или чтобы показать, что вертикальная стрелка обозначает изоморфизм в диаграмме:

```


\matrix{
A&\buildrel u\over{\hbox to 12mm{\rightarrowfill}}&B \cr
\wr\bigg\downarrow\phantom{\wr}&&\bigg\downarrow\wr \cr
A'\&\buildrel u'\over{\hbox to 12mm{\rightarrowfill}}&B' \cr}


```

$$\begin{array}{ccc} A & \xrightarrow{u} & B \\ \wr \downarrow & & \downarrow \wr \\ A' & \xrightarrow{u'} & B' \end{array}$$

Эта диаграмма является примером недоработанной кодировки: плохо распределены вертикальные пробелы, не центрируются вертикальные стрелки. Конечно же, без особых усилий ее можно сильно улучшить.

508.

`\write*` Команда записи строки в выходной файл с указанным номером, который предварительно должен быть открыт командой `\openout`. См. команду `\read`.

256, 257, 269, 302, 332, 410, 444, 493, 495.

X

`\xdef*` Команда эквивалентна `\global\edef` (см. `\edef`).

255-256, 326, 439, 489, 495.

`\xi` Греческая буква ξ . Только в математической моде.

506, 520.

`\Xi` Заглавная греческая буква Ξ . Только в математической моде.

506.

`\xleaders*` См. `\leaders`.

266.

`\xspaceskip*` Управляет пробелами после символов пунктуации в данном шрифте. См. команду `\spaceskip`, которая управляет пробелами между словами. Если Вы используете команды `\frenchspacing`, `\nonfrenchspacing` или `\raggedright`, значения этих величин могут измениться.

95, 325, 378, 421, 501, 505.

У

`\year*` Переменная, которая содержит текущий год:
`\the\year` 1999
 (`\the` играет роль *write* в Паскале и *print* в Бейсике. Чтобы ярче выделить год, пишите `\oldstyle\the\year`, что дает 1999 (не забывайте скобки). См. также `\day` для других указаний.
 52, 324, 413, 476.

Ζ

`\zeta` Греческая буква ζ. Только в математической моде. Например, известная функция Римана

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$$

и формула Эйлера

$$\zeta(s) = \frac{1}{\prod_{i=1}^{\infty} \left(1 - \frac{1}{p_i^s}\right)},$$

где $(p_i)_{i \geq 1}$ обозначает последовательность простых чисел. Приведем кодировку второй формулы:

$$\zeta(s) = \frac{1}{\prod_{i=1}^{\infty} \left(1 - \frac{1}{p_i^s}\right)}$$

Обратите внимание на команду `\displaystyle` в знаменателе и на конструкцию `\raise2pt\hbox{,}`, из-за которой запятая находится на правильном уровне относительно дробной черты.

506.

Список литературы

- [1] Кнут Д. Е. Все про Т_EX. Перевод с англ., Протвино, 1993.
- [2] Raymond Seroul Le petit Livre de Т_EX. Inter Edition, Paris, 1984.
- [3] Бажанова Л. С. Введение в Т_EX. Препринт ИФВЭ 90–116, Протвино, 1990.
- [4] Беляевская Л. В., Грицаенко И. А. Малое описание Т_EXa. Протвино, 1991.
- [5] Грицаенко И. А., Клименко С. В., Малышев В. К., Самарин А. В. Т_EX в ИФВЭ.
1. Общая характеристика. Препринт ИФВЭ 91–54, Протвино, 1991.
- [6] Грицаенко И. А., Малышев В. К., Самарин А. В. Т_EX в ИФВЭ.
2. Т_EX и L^AT_EX на VAX'ах. Препринт ИФВЭ 91–55, Протвино, 1991.
- [7] Глонти Н. Л., Грицаенко И. А. и др. Т_EX в ИФВЭ. 5. Шрифты и работа с ними. Препринт ИФВЭ 92–127, Протвино, 1991.

Рукопись поступила 17 марта 1995 г.