

PLAIN.TW

Włodek Bzyl

matwb@halina.univ.gda.pl

Contents

Introduction	2
The layout of the format	3
Text fonts	5
Font encoding	6
Math fonts	8
Mathematical spacing	9
Registers allocation	13
Parameters	15
Macros for text	20
Macros for math	26
Macros for output	36
Hyphenation	39
Initialization	39
Programming support	40
Appendices	41
Efficiency and memory-space considerations	41
Extensible delimiters	42
Font dimensions	42
Font tables	44

Introduction

However, we will not include a verbatim description, because some parts of that file are too boring, and because the actual macros have been “optimized” with respect to memory space and running time.

D.E. Knuth about plain.tex format.

What follows is devoted to the details of the plain \TeX format. This file serves two purposes:

- (1) As a documentation of the `plain.tex` format. Weaving and texing this document should produce a handy reference.
- (2) The division of this web source into ‘chunks’ should ease creation of other formats tailored to particular applications. Chunks could be easily modified, removed, added, or replaced.

The change file mechanism is not needed in case of \TeX language. Change files are used to incorporate system dependent code into source file, but \TeX code is already system independent. \TeX code could be only ‘format dependent’ and here change files could be used. Another feature of format file is that it evolves with time, yet some intermediate versions are used for preparation of books, articles etc. All these versions and configurations must be kept well organized, otherwise you are lost. The Revision Control System is the tool that assists with these tasks. With the RCS it is possible, with small overhead, to preserve *all revisions* which evolved from given text document, merge changes made by others, compare different versions, keep log of changes.

This document consists mainly of excerpts from the *\TeX Book*, but it is organised around the macros as they appear in the `plain.tex` rather than around the topics as in a user manual. Therefore this document is not a *user manual*, although many definitions are contained here.

The layout of the format

```

< * >≡
  < Establish standard category code values >
1  \catcode'@ = 11
  < Define commonly used constants >
  < Provide programming constructs >
  < Allocate registers >
  < Assign initial values to parameters >
  < Set up text fonts >
  < Set up math fonts >
  < Provide macros for text formatting >
  < Provide macros for math formatting >
  < Prepare page for output >
  < Read hyphenation patterns >
  < Initialize the layout >
2  \catcode'@ = 12
  < Identify the format >

```

There are 256 characters that T_EX might encounter at each step, in a file or in a line of text typed directly on your terminal. These 256 characters are classified into 16 categories numbered 0 to 15:

<i>Category</i>	<i>Meaning</i>	<i>Default</i>
0	Escape character	\
1	Beginning of group	{
2	End of group	}
3	Math shift	\$
4	Alignment tab	&
5	End of line	<return>
6	Parameter	#
7	Superscript	^
8	Subscript	_
9	Ignored character	<null>
10	Space	␣
11	Letter	A, ..., Z and a, ..., z
12	Other character	none of the above or below
13	Active character	~
14	Comment character	%
15	Invalid character	<delete>

When INITEX begins, category 12 (other) has been assigned to all 256 possible characters, except that the 52 letters A...Z and a...z are category 11 (letter), and assignments equivalent to the following have been made:

```

\catcode '\ = 0
\catcode '\^M = 5
\catcode '\^@ = 9
\catcode '\ = 10
\catcode '\% = 14
\catcode '\^^? = 15

```

Thus ‘\’ is already an escape character, ‘\char"20’ is a space, and ‘%’ is available for comments on the first line of the file; ASCII ⟨null⟩ is ignored, ASCII ⟨return⟩ is an end-of-line character, and ASCII ⟨delete⟩ is invalid.

Furthermore ⟨tab⟩ is given category space, ⟨formfeed⟩ becomes an active character that will detect runaways on files that have been divided into “file pages” by ⟨formfeed⟩ characters. Finally the control sequence \active is defined to yield the constant 13.

To re-catcode these special characters—not counting ASCII

```

    ⟨null⟩      ^^@
    ⟨tab⟩       ^^I
    ⟨linefeed⟩  ^^J
    ⟨formfeed⟩  ^^L
    ⟨return⟩    ^^M
    ⟨delete⟩    ^^?

```

—use the control sequence \dospecials that lists all the characters whose catcodes should probably be changed to 12 (other) when copying things verbatim. Each symbol in the list is preceded by \do, which can be defined if you want to do something to every item in the list.

⟨ *Establish standard category code values* ⟩≡

```

3      \catcode'\{=1
4      \catcode'\}=2
5      \catcode'\$=3
6      \catcode'\&=4
7      \catcode'\#=6
8      \catcode'\^=7 \catcode'\^K=7 % uparrow is for superscripts
9      \catcode'\_ =8 \catcode'\^A=8 % downarrow are for subscripts
10     \catcode'\^I=10
11
12     \chardef\active=13 \catcode'\~=\active % tilde is active
13     \catcode'\^L=\active \outer\def^L{\par} % ascii form-feed is "\outer\par"
14
15     \def\dospecials{\do\ \do\\\do\{\do\}\do\$\do\&%
16     \do#\do\^ \do\^K\do\_ \do\^A\do\%\do\~}

```

To make the plain macros more efficient in time and space, several constant values are declared as control sequences. *If they were changed, anything could happen.* So be careful!

⟨ *Define commonly used constants* ⟩≡

```

17     \chardef\@ne=1
18     \chardef\tw@=2
19     \chardef\thr@@=3
20     \chardef\sixt@@n=16
21     \chardef\@cclv=255
22     \mathchardef\@cclvi=256
23     \mathchardef\@m=1000
24     \mathchardef\@M=10000
25     \mathchardef\@MM=20000

```

Text fonts

⟨ Set up text fonts ⟩≡

⟨ Provide support for font scaling ⟩
⟨ Define text fonts ⟩
⟨ Encode special characters, and characters not available on the keyboard ⟩
⟨ Provide support for accented characters ⟩
⟨ Assign uppercase and lowercase code values ⟩
⟨ Assign space factor codes ⟩

Fonts assigned to `\preloaded` are not part of the format, but they are preloaded so that other format packages can use them. For example, if another set of macros says `\font\ninerm=cmr9`, T_EX will not have to reload the font metric information for `cmr9`.

⟨ Define text fonts ⟩≡

```

26     \font\tenrm=cmr10 % roman text
27     \font\preloaded=cmr9
28     \font\preloaded=cmr8
29     \font\sevenrm=cmr7
30     \font\preloaded=cmr6
31     \font\fiverm=cmr5
32
33     \font\preloaded=cmss10 % sans serif
34     \font\preloaded=cmssq8
35
36     \font\preloaded=cmssi10 % sans serif italic
37     \font\preloaded=cmssqi8
38
39     \font\tenbf=cmbx10 % boldface extended
40     \font\preloaded=cmbx9
41     \font\preloaded=cmbx8
42     \font\sevenbf=cmbx7
43     \font\preloaded=cmbx6
44     \font\fivebf=cmbx5
45
46     \font\tentt=cmtt10 % typewriter
47     \font\preloaded=cmtt9
48     \font\preloaded=cmtt8
49
50     \font\preloaded=cmsl10 % slanted typewriter
51
52     \font\ten10=cmsl10 % slanted roman
53     \font\preloaded=cmsl9
54     \font\preloaded=cmsl8
55
56     \font\tenit=cmti10 % text italic
57     \font\preloaded=cmti9
58     \font\preloaded=cmti8
59     \font\preloaded=cmti7
60
61     \font\preloaded=cmu10 % unslanted text italic
62

```

```

63     \font\preloaded=cmcsc10 % caps and small caps
64
65     \font\preloaded=cmssbx10 % sans serif bold extended
66
67     \font\preloaded=cmdunh10 % Dunhill style
68
69     \font\preloaded=cmr7 scaled \magstep4 % for titles
70     \font\preloaded=cmtt10 scaled \magstep2
71     \font\preloaded=cmssbx10 scaled \magstep2
72
73     \font\preloaded=manfnt % METAFONT logo and dragon curve and special symbols

```

Additional `\preloaded` fonts can be specified here. (And those that were `\preloaded` above can be eliminated.)

⟨ Define text fonts ⟩≡

```

74     \let\preloaded=\undefined % preloaded fonts must be declared anew later.

```

⟨ Provide support for font scaling ⟩≡

```

75     \def\magstephalf{1095 }
76     \def\magstep#1{\ifcase#1 \@m\or 1200\or 1440\or 1728\or 2074\or 2488\fi\relax}
77     \def\magnification{\afterassignment\m@g\count@}
78     \def\m@g{\mag\count@}
79     \hsize6.5truein\vsiz8.9truein\dimen\footins8truein}

```

Font encoding

We usually think of text files as containing characters. It doesn't cause any problems most of the time when we use plain ASCII characters—letters A–Z, a–z, the numerals 0–9 and some of punctuation characters. This illusion is broken down when we start using characters that do not belong to this limited set, for example, accented characters / mathematical symbols. Then what we see on screen may not match what we key in. What gets printed may not match what we see on screen. Moreover, what gets shown on screen and what gets printed depends on what machine we are on and how the fonts that we are using are set up. In reality text files contain just numeric codes (in range 0–255) stored in 8-bit bytes, and the mapping between ‘character’ and numeric code is quite arbitrary. This is because there are very many more characters than the 256 numeric codes possible with 8-bits. Consequently, there will be a need for more than one possible mapping or ‘font encoding’, or in other words, there would not be a ‘standard’ encoding that suits all purposes.

When a symbol is built up by forming a box, the `\leavevmode` macro is called first; this starts a new paragraph, if \TeX is in vertical mode, but does nothing if \TeX is in horizontal mode or math mode. `\chardef` positions are taken from the fonts `cmr10` and `cmsy10`.

⟨ Encode special characters, and characters not available on the keyboard ⟩≡

```

80     \chardef\%= '\%
81     \chardef\&= '\&
82     \chardef\#= '\#
83     \chardef\$= '\$
84     \chardef\ss="19
85     \chardef\ae="1A
86     \chardef\oe="1B
87     \chardef\o="1C

```

```

88     \chardef\AE="1D
89     \chardef\OE="1E
90     \chardef\O="1F
91     \chardef\i="10 \chardef\j="11 % dotless letters
92     \def\aa{\accent23a}
93     \def\l{\char321}
94     \def\L{\leavevmode\setbox0\hbox{L}\hbox to\wd0{\hss\char32L}}
95
96     \def\leavevmode{\unhbox\voidb@x} % begins a paragraph, if necessary
97     \def\_ {\leavevmode \kern.06em \vbox{\hrule width.3em}}
98     \def\AA{\leavevmode\setbox0\hbox{h}\dimen@ht0\advance\dimen@-1ex%
99       \rlap{\raise.67\dimen@\hbox{\char'27}}A}
100
101     \def\mathhexbox#1#2#3{\leavevmode
102       \hbox{\m@th \mathchar"#1#2#3}}
103     \def\dag{\mathhexbox279}
104     \def\ddag{\mathhexbox27A}
105     \def\S{\mathhexbox278}
106     \def\P{\mathhexbox27B}

```

The accent positions are taken from Computer Modern font family. We are about to ‘hard-wire’ CM accent encoding into the format. Different encoding will be necessary if other styles of type are used.

Three alternative control-symbol accents are defined, suitable for keyboards with extended character sets: `\let\^^_=\v`, `\let\^^S=\u`, `\let\^^D=\^`.

⟨ *Provide support for accented characters* ⟩≡

```

107     \def\oalign#1{\leavevmode\vtop{\baselineskip\z@skip \lineskip.25ex%
108       \ialign{##\crrc#1\crrc}}}\def\o@lign{\lineskiplimit\z@ \oalign}
109     \def\ooalign{\lineskiplimit-\maxdimen \oalign} % chars over each other
110     \def\sh@ft#1{\dimen\z@.00#1ex\multiply\dimen\z@\fontdimen1\font
111       \kern-.0156\dimen\z@} % compensate for slant in lowered accents
112     \def\d#1{\o@lign{\relax#1\crrc\hidewidth\sh@ft{10}.\hidewidth}}
113     \def\b#1{\o@lign{\relax#1\crrc\hidewidth\sh@ft{29}%
114       \vbox to.2ex{\hbox{\char22}\vss}\hidewidth}}
115     \def\c#1{\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent24 #1%
116       \else{\oalign{\unhbox\z@\crrc\hidewidth\char24\hidewidth}}\fi}
117     \def\copyright{\oalign{\hfil\raise.07ex\hbox{c}\hfil\crrc\mathhexbox20D}}
118
119     \def\dots{\relax\ifmmode\ldots\else$\m@th\ldots$, $\fi}
120     \def\TeX{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX}
121
122     \def\`#1{\accent18 #1}
123     \def\'#1{\accent19 #1}
124     \def\v#1{\accent20 #1} \let\^^_=\v
125     \def\u#1{\accent21 #1} \let\^^S=\u
126     \def\=#1{\accent22 #1}
127     \def\^#1{\accent94 #1} \let\^^D=\^
128     \def\.#1{\accent95 #1}
129     \def\H#1{\accent"7D #1}
130     \def\~#1{\accent"7E #1}
131     \def\"#1{\accent"7F #1}
132     \def\t#1{\edef\next{\the\font}\the\textfont1\accent"7F\next#1}

```

INITEX sets `\uccode‘x = ‘X` and `\uccode‘X = ‘X` for all letters x , and `\lccode‘x = ‘x`, `\lccode‘X = ‘x`; all other values are zero.

⟨ Assign uppercase and lowercase code values ⟩≡

```
133      %% Thats all for English language.
```

Space factor code affects setting of interword glue. The space factor is normally 1000, which means that the interword glue should not be modified. If the space factor f is different from 1000, the interword glue is computed as follows: Take the normal space glue for the current font, and add the extra space if $f \geq 2000$. Then the stretch component is multiplied by $f/1000$, while the shrink component is multiplied by $1000/f$. (Look up the Appendix for the values of normal space, normal stretch, normal shrink, and extra space for some of CM fonts.)

INITEX sets space factor codes: `\sfcode x = 1000` for all x , except that `\sfcode‘X = 999` for uppercase letters.

The characters ‘)’, ‘’’, and ‘]’ does not change space factor.

⟨ Assign space factor codes ⟩≡

```
134      \sfcode‘\)=0 \sfcode‘\’=0 \sfcode‘\]=0
```

Math fonts

⟨ Set up math fonts ⟩≡

```

  ⟨ Define math fonts ⟩
  ⟨ Encode math accents ⟩
  ⟨ Establish spacing around mathematical objects ⟩
  ⟨ Assign math codes ⟩
  ⟨ Assign delimiter codes ⟩
  ⟨ Define font families ⟩
```

As was said earlier, the font metric information about preloaded font will be build into the format. But, if another set of macros says `\font\fiftyfiverm = cmr9` at 55pt, \TeX will *have to reload again* the font metric information for `cmr9`.

⟨ Define math fonts ⟩≡

```
135      \font\teni=cmmi10 % math italic
136      \font\preloaded=cmmi9
137      \font\preloaded=cmmi8
138      \font\seveni=cmmi7
139      \font\preloaded=cmmi6
140      \font\fivei=cmmi5
141
142      \font\tensy=cmsy10 % math symbols
143      \font\preloaded=cmsy9
144      \font\preloaded=cmsy8
145      \font\sevensy=cmsy7
146      \font\preloaded=cmsy6
147      \font\fivesy=cmsy5
148
149      \font\tenex=cmex10 % math extension
150
```

```

151     \font\preloaded=cmmib10 % bold math italic
152     \font\preloaded=cmsy10 % bold math symbols

```

Mathematical spacing

Spacing around mathematical object is measured in μ —‘math units.’ 1μ is equal to $1/18$ th part of `\fontdimen 6` of the font in family 2.

`\quad` spacing does not change with the style of formula, nor does it depend on the math font families that are being used. But thin spaces, medium spaces, and thick spaces do get bigger and smaller as the size of type gets bigger and smaller; this is because they are defined in terms of `\muglue`.

According to these specifications, thin spaces in plain \TeX do not stretch or shrink; medium spaces can stretch a little, and they can shrink to zero; thick spaces can stretch a lot, but they never shrink.

The following table gives the complete definition of `\muglue` between mathematical objects. A formula is converted to a math list, and the math list consists chiefly of “atoms” of eight basic types: Ord (ordinary), Op (large operator), Bin (binary operation), Rel (relation), Open (opening), Close (closing), Punct (punctuation), and Inner (a delimited subformula). Other kinds of atoms, which arise from commands like `\overline` or `\mathaccent` or `\vcenter`, etc., are all treated as type Ord; fractions are treated as type Inner. The following (non-symmetric) table is used to determine the spacing between pairs of adjacent atoms:

		<i>Right atom</i>							
		Ord	Op	Bin	Rel	Open	Close	Punct	Inner
<i>Left atom</i>	Ord	0	1	(2)	(3)	0	0	0	(1)
	Op	1	1	*	(3)	0	0	0	(1)
	Bin	(2)	(2)	*	*	(2)	*	*	(2)
	Rel	(3)	(3)	*	0	(3)	0	0	(3)
	Open	0	0	*	0	0	0	0	0
	Close	0	1	(2)	(3)	0	0	0	(1)
	Punct	(1)	(1)	*	(1)	(1)	(1)	(1)	(1)
	Inner	(1)	1	(2)	(3)	(1)	0	(1)	(1)

Here 0, 1, 2, and 3 stand for no space, thin space, medium space, and thick space, respectively. Thin space, medium space, and thin space are equal to values of `\thinmuskip`, `\medmuskip`, `\thickmuskip` parameters, respectively. The table entry is parenthesized if the space is to be inserted only in display and text styles, not in script and scriptscript styles. For example, many of the entries in the Rel row and the Rel column are ‘(3)’; this means that thick spaces are normally inserted before and after relational symbols like ‘=’, but not in subscripts. Some of the entries in the table are ‘*’; such cases never arise, because Bin atoms must be preceded and followed by atoms compatible with the nature of binary operations. The conversion of math lists to horizontal lists is done whenever \TeX is about to leave math mode, and the inter-atomic spacing is inserted at that time.

\langle *Establish spacing around mathematical objects* \equiv

```

153     \thinmuskip=3mu
154     \medmuskip=4mu plus 2mu minus 4mu
155     \thickmuskip=5mu plus 5mu

```

For the positioning of accents over single character the width of `\skewchar` is used. For most of fonts the default value of `\skewchar` is `-1`; but the math italic (family 1) and math symbol fonts (family 2) have special `\skewchar` values equal to `'177` and `'60`, respectively. These are characters `'^` and `'r`.

< Encode math accents >≡

```
156 \skewchar\teni='177 \skewchar\seveni='177 \skewchar\fivei='177
157 \skewchar\tensy='60 \skewchar\sevensy='60 \skewchar\fivesy='60
```

A math code is relevant only when the corresponding category code is 11 or 12. When processing in math mode characters of categories 11 and 12, `\char` and `\chardef` characters are replaced by their math code.

If we denote 15-bit number by "uvwz, then math codes are assigned by

`\mathcode <8-bit number> = "uvwz`, where

- u — the class code (see below for the list)
- v — the font family number (see the font tables at the end of this document)
- wz — the position of the character in the font

<i>Class</i>	<i>Meaning</i>	<i>Example</i>	<i>Class</i>	<i>Meaning</i>	<i>Example</i>
0	Ordinary	/	4	Opening	(
1	Large operator	\sum	5	Closing)
2	Binary operation	+	6	Punctuation	,
3	Relation	=	7	Variable family	x

A `\mathcode` can also have the special value "8000, which causes the character to behave as if it has catcode 13 (active). This feature makes `'` apostrophe expand to `\prime`. The mathcode of `'` does not interfere with the use of `'` in octal constants. The mathcode of "8000 is also assigned to space and underscore.

INITEX sets up `\mathcode x = x` for $x = 0..255$, except that `\mathcode x = x + "7000` for each of the ten digits $x = '0$ to `'9`; `\mathcode x = x + "7100` for each of the 52 letters. T_EX looks at the mathcode only when it is typesetting a character whose catcode is 11 (letter) or 12 (other), or when it encounters a character that is given explicitly as `\char<number>`.

Class 7 is a special case that allows math symbols to change families. It behaves exactly like class 0, except that the specified family is replaced by the current value of an integer parameter called `\fam`, provided that `\fam` is a legal family number (i.e., if it lies between 0 and 15). T_EX automatically sets `\fam=-1` whenever math mode is entered; therefore class 7 and class 0 are equivalent unless `\fam` has been given a new value. Plain T_EX changes `\fam` to 0 when the user types `\rm`; this makes it convenient to get roman letters in formulas, since letters belong to class 7. The control sequence `\rm` is an abbreviation for `\fam=0 \tenrm`; thus, `\rm` causes `\fam` to become zero, and it makes `\tenrm` the "current font." In horizontal mode, the `\fam` value is irrelevant and the current font governs the typesetting of letters; but in math mode, the current font is irrelevant and the `\fam` value governs the letters. The current font affects math mode only if control space (`\`) is used or if dimensions are given in `ex` or `em` units; it also has an effect if an `\hbox` appears inside a formula, since the contents of an `hbox` are typeset in horizontal mode.

< Assign math codes >≡

```
158 \mathcode'\^^@="2201 % \cdot
159 \mathcode'\^^A="3223 % \downarrow
160 \mathcode'\^^B="010B % \alpha
161 \mathcode'\^^C="010C % \beta
162 \mathcode'\^^D="225E % \land
163 \mathcode'\^^E="023A % \lnot
```

```

164     \mathcode'\^^F="3232 % \in
165     \mathcode'\^^G="0119 % \pi
166     \mathcode'\^^H="0115 % \lambda
167     \mathcode'\^^I="010D % \gamma
168     \mathcode'\^^J="010E % \delta
169     \mathcode'\^^K="3222 % \uparrow
170     \mathcode'\^^L="2206 % \pm
171     \mathcode'\^^M="2208 % \oplus
172     \mathcode'\^^N="0231 % \infty
173     \mathcode'\^^O="0140 % \partial
174     \mathcode'\^^P="321A % \subset
175     \mathcode'\^^Q="321B % \supset
176     \mathcode'\^^R="225C % \cap
177     \mathcode'\^^S="225B % \cup
178     \mathcode'\^^T="0238 % \forall
179     \mathcode'\^^U="0239 % \exists
180     \mathcode'\^^V="220A % \otimes
181     \mathcode'\^^W="3224 % \leftrightarrows
182     \mathcode'\^^X="3220 % \leftarrow
183     \mathcode'\^^Y="3221 % \rightarrow
184     \mathcode'\^^Z="8000 % \neq
185     \mathcode'\^^[="2205 % \diamond
186     \mathcode'\^^\="3214 % \leq
187     \mathcode'\^^]="3215 % \geq
188     \mathcode'\^^^="3211 % \equiv
189     \mathcode'\^^_="225F % \lor
190     \mathcode'\ =="8000 % \space
191     \mathcode'\!="5021
192     \mathcode'\'="8000 % \prime
193     \mathcode'\(="4028
194     \mathcode'\)="5029
195     \mathcode'\*="2203 % \ast
196     \mathcode'\+="202B
197     \mathcode'\,="613B
198     \mathcode'\-="2200
199     \mathcode'\.="013A
200     \mathcode'\/"="013D
201     \mathcode'\:="303A
202     \mathcode'\;="603B
203     \mathcode'\<="313C
204     \mathcode'\=="303D
205     \mathcode'\>="313E
206     \mathcode'\?="503F
207     \mathcode'\["="405B
208     \mathcode'\\"="026E % \backslash
209     \mathcode'\]="505D
210     \mathcode'\_="8000 % \_
211     \mathcode'\{"="4266
212     \mathcode'\|"="026A
213     \mathcode'\}"="5267
214     \mathcode'\^^?="1273 % \smallint
215

```

Delimiter codes are used after `\left` and `\right` commands, when \TeX is looking for a delimiter. If we denote 24-bit number by "qrstuv", then delimiter codes are assigned by

`\delcode` <8-bit number> = "qrstuv, where

- q — the font family number of
- rs — the position of the the small variant of the delimiter
- t — the font family number of
- uv — the position of the the large variant of the delimiter

INITEX sets all `\delcode` values to -1, which means that no characters are recognized as delimiters in math formulas, except `\delcode` ' . = 0, so that ' . ' stands for "null delimiter". { and } should *not get* delcodes; otherwise parameter grouping fails!

< *Assign delimiter codes* >≡

```
216     \delcode`\="028300
217     \delcode`\)"029301
218     \delcode`\["05B302
219     \delcode`\]"05D303
220     \delcode`\<"26830A
221     \delcode`\>"26930B
222     \delcode`\/"02F30E
223     \delcode`\|"26A30C
224     \delcode`\\"26E30F
```

All characters that are typeset in math mode belong to one of sixteen *families* of fonts, numbered internally from 0 to 15. Each of these families consists of three fonts: one for text size, one for script size, and one for scriptscriptsize. The commands `\textfont`, `\scriptfont`, and `\scriptscriptfont` are used to specify the members of each family. Since there are up to 256 characters per font, and 3 fonts per family, and 16 families, \TeX can access up to 12,288 characters in any one formula (4096 in each of the three sizes).

The `plain.tex` format uses family 1 for math italic letters, family 2 for ordinary math symbols, and family 3 for large symbols. Text italic is put in family 4, slanted roman in family 5, bold roman in family 6, and typewriter type in family 7. A macro `\newfam` will assign symbolic names to families that aren't already used.

INITEX initializes the mathcodes of all letters **A** to **Z** and **a** to **z** so that they are symbols of class 7 and family 1; that's why it is natural to use family 1 for math italics. Similarly, the digits **0** to **9** are class 7 and family 0. None of the other families is treated in any special way by \TeX .

\TeX doesn't check to see if the families are sensibly organized. The only constraint is that the fonts in families 2 and 3 have special `\fontdimen` parameters, which govern mathematical spacing (see Appendix). In Computer Modern only `cmsy` and `cmex` have these parameters, so their assignment to families 2 and 3 is almost mandatory.

During the time that a math formula is being read, \TeX remembers each symbol as being "character position so-and-so in family number such-and-such," but it does not take note of what fonts are actually in the families until reaching the end of the formula.

< *Define font families* >≡

```
225     \textfont0=\tenrm \scriptfont0=\sevenrm \scriptscriptfont0=\fiverm
226     \def\rm{\fam\z@\tenrm}
227     \textfont1=\teni \scriptfont1=\seveni \scriptscriptfont1=\fivei
228     \def\mit{\fam\@ne} \def\oldstyle{\fam\@ne\teni}
229     \textfont2=\tensy \scriptfont2=\sevensy \scriptscriptfont2=\fivesy
230     \def\cal{\fam\tw@}
231     \textfont3=\tenex \scriptfont3=\tenex \scriptscriptfont3=\tenex
```

```

232     \newfam\itfam \def\it{\fam\itfam\tenit} % \it is family 4
233     \textfont\itfam=\tenit
234     \newfam\slfam \def\sl{\fam\slfam\tensl} % \sl is family 5
235     \textfont\slfam=\tensl
236     \newfam\bffam \def\bf{\fam\bffam\tenbf} % \bf is family 6
237     \textfont\bffam=\tenbf \scriptfont\bffam=\sevenbf
238     \scriptscriptfont\bffam=\fivebf
239     \newfam\ttfam \def\tt{\fam\ttfam\tentt} % \tt is family 7
240     \textfont\ttfam=\tentt

```

Registers allocation

Here are macros for the automatic allocation of `\count`, `\box`, `\dimen`, `\skip`, `\muskip`, and `\toks` registers, as well as `\read` and `\write` stream numbers, `\fam` codes, `\language` codes, and `\insert` numbers.

The main use of these macros is for registers that are defined by one macro and used by others, possibly at different nesting levels.

The following counters are reserved:

0–9	page numbering
10	count allocation
11	dimen allocation
12	skip allocation
13	muskip allocation
14	box allocation
15	toks allocation
16	read file allocation
17	write file allocation
18	math family allocation
19	language allocation
20	insert allocation
21	the most recently allocated number
22	constant -1

New counters are allocated starting with 23, 24, etc. Other registers are allocated starting with 10. This leaves 0 through 9 for the user to play with safely, except that counts 0 to 9 are considered to be the page and subpage numbers (since they are displayed during output). In this scheme, `\count 10` always contains the number of the highest-numbered counter that has been allocated, `\count 14` the highest-numbered box, etc. Inserts are given numbers 254, 253, etc., since they require a `\count`, `\dimen`, `\skip`, and `\box` all with the same number; `\count 20` contains the lowest-numbered insert that has been allocated. `\box255` is reserved for `\output`; `\count255`, `\dimen255`, and `\skip255` can be used freely.

It is recommended that macro designers always use global assignments with respect to registers numbered 1, 3, 5, 7, 9, and always non-global assignments with respect to registers 0, 2, 4, 6, 8, 255. This will prevent “save stack buildup” that might otherwise occur.

< Allocate registers >≡

```

241     \count10=22 % allocates \count registers 23, 24, ...
242     \count11=9 % allocates \dimen registers 10, 11, ...
243     \count12=9 % allocates \skip registers 10, 11, ...
244     \count13=9 % allocates \muskip registers 10, 11, ...
245     \count14=9 % allocates \box registers 10, 11, ...

```

```

246     \count15=9 % allocates \toks registers 10, 11, ...
247     \count16=-1 % allocates input streams 0, 1, ...
248     \count17=-1 % allocates output streams 0, 1, ...
249     \count18=3 % allocates math families 4, 5, ...
250     \count19=0 % allocates \language codes 1, 2, ...
251     \count20=255 % allocates insertions 254, 253, ...
252     \countdef\insc@unt=20 % the insertion counter
253     \countdef\allocationnumber=21 % the most recent allocation
254     \countdef@m@ne=22 \m@ne=-1 % a handy constant
255     \def\wlog{\immediate\write\m@ne} % write on log file (only)
     \langle Allocate scratch registers \rangle
     \langle Provide user-level register allocation macros \rangle
     \langle Define implementation-level register allocation macros \rangle
     \langle Initialize register constants \rangle

```

Here are abbreviations for the names of scratch registers that don't need to be allocated.

\langle *Allocate scratch registers* \rangle≡

```

256     \countdef\count@=255
257     \dimendef\dimen@=0
258     \dimendef\dimen@i=1 % global only
259     \dimendef\dimen@ii=2
260     \skipdef\skip@=0
261     \toksdef\toks@=0

```

Now, we define `\newcount`, `\newbox`, etc. so that you can say `\newcount\foo` and `\foo` will be defined (with `\countdef`) to be the next counter. To find out which counter `\foo` is, you can look at `\allocationnumber`. Since there's no `\boxdef` command, `\chardef` is used to define a `\newbox`, `\newinsert`, `\newfam`, and so on.

\langle *Provide user-level register allocation macros* \rangle≡

```

262     \outer\def\newcount{\alloc@0\count\countdef\insc@unt}
263     \outer\def\newdimen{\alloc@1\dimen\dimendef\insc@unt}
264     \outer\def\newskip{\alloc@2\skip\skipdef\insc@unt}
265     \outer\def\newmuskip{\alloc@3\muskip\muskipdef\@cclvi}
266     \outer\def\newbox{\alloc@4\box\chardef\insc@unt}
267     \let\newtoks=\relax % we do this to allow plain.tex to be read in twice
268     \outer\def\newhelp#1#2{\newtoks#1#1\expandafter{\csname#2\endcsname}}
269     \outer\def\newtoks{\alloc@5\toks\toksdef\@cclvi}
270     \outer\def\newread{\alloc@6\read\chardef\sixt@n}
271     \outer\def\newwrite{\alloc@7\write\chardef\sixt@n}
272     \outer\def\newfam{\alloc@8\fam\chardef\sixt@n}
273     \outer\def\newlanguage{\alloc@9\language\chardef\@cclvi}

```

\langle *Define implementation-level register allocation macros* \rangle≡

```

274     \def\alloc@#1#2#3#4#5{\global\advance\count1#1by\@ne
275     \ch@ck#1#4#2% make sure there's still room
276     \allocationnumber=\count1#1%
277     \global#3#5=\allocationnumber
278     \wlog{\string#5=\string#2the\allocationnumber}}
279     \outer\def\newinsert#1{\global\advance\insc@unt by\m@ne
280     \ch@ck0\insc@unt\count

```

```

281     \ch@ck1\insc@unt\dimen
282     \ch@ck2\insc@unt\skip
283     \ch@ck4\insc@unt\box
284     \allocationnumber=\insc@unt
285     \global\chardef#1=\allocationnumber
286     \wlog{\string#1=\string\insert\the\allocationnumber}}
287 \def\ch@ck#1#2#3{\ifnum\count1#1<#2%
288     \else\errmessage{No room for a new #3}\fi}

```

We finish with the initialization of some constants.

⟨ Initialize register constants ⟩≡

```

289     \newdimen\maxdimen \maxdimen=16383.99999pt % the largest legal <dimen>
290     \newskip\hideskip \hideskip=-1000pt plus 1fill % negative but can grow
291     \newskip\centering \centering=0pt plus 1000pt minus 1000pt
292     \newdimen\p@ \p@=1pt % this saves macro space and time
293     \newdimen\z@ \z@=0pt % can be used both for 0pt and 0
294     \newskip\z@skip \z@skip=0pt plus0pt minus0pt
295     \newbox\voidb@x % permanently void box register

```

Parameters

Let's turn now to \TeX 's parameters, which the previous chapters have introduced one at a time; it will be convenient to assemble them all together.

An *⟨integer parameter⟩* is one of the following tokens:

```

\pretolerance    badness tolerance before hyphenation
\tolerance       badness tolerance after hyphenation
\hbadness        badness above which bad hboxes will be shown
\vbadness        badness above which bad vboxes will be shown
\linepenalty     amount added to badness of every line in a paragraph
\hyphenpenalty   penalty for line break after discretionary hyphen
\exhyphenpenalty penalty for line break after explicit hyphen
\binoppenalty    penalty for line break after binary operation
\relpenalty      penalty for line break after math relation
\clubpenalty     penalty for creating a club line at bottom of page
\widowpenalty    penalty for creating a widow line at top of page
\displaywidowpenalty ditto, before a display
\brokenpenalty   penalty for page break after a hyphenated line
\predisplaypenalty penalty for page break just before a display
\postdisplaypenalty penalty for page break just after a display
\interlinepenalty additional penalty for page break between lines
\floatingpenalty penalty for insertions that are split
\outputpenalty   penalty at the current page break
\doublehyphendemerits demerits for consecutive broken lines
\finalhyphendemerits demerits for a penultimate broken line
\adjdemerits     demerits for adjacent incompatible lines
\looseness       change to the number of lines in a paragraph
\pausing         positive if pausing after each line is read from a file
\holdinginserts  positive if insertions remain dormant in output box
\tracingonline   positive if showing diagnostic info on the terminal
\tracingmacros   positive if showing macros as they are expanded

```

`\tracingstats` positive if showing statistics about memory usage
`\tracingparagraphs` positive if showing line-break calculations
`\tracingpages` positive if showing page-break calculations
`\tracingoutput` positive if showing boxes that are shipped out
`\tracinglostchars` positive if showing characters not in the font
`\tracingcommands` positive if showing commands before they are executed
`\tracingrestores` positive if showing deassignments when groups end
`\language` the current set of hyphenation rules
`\uchyph` positive if hyphenating words beginning with capital letters
`\lefthyphenmin` smallest fragment at beginning of hyphenated word
`\righthyphenmin` smallest fragment at end of hyphenated word
`\globaldefs` nonzero if overriding `\global` specifications
`\defaultshyphenchar` `\hyphenchar` value when a font is loaded
`\defaultsskewchar` `\skewchar` value when a font is loaded
`\escapechar` escape character in the output of control sequence tokens
`\endlinechar` character placed at the right end of an input line
`\newlinechar` character that starts a new output line
`\maxdeadcycles` upper bound on `\deadcycles`
`\hangafter` hanging indentation changes after this many lines
`\fam` the current family number
`\mag` magnification ratio, times 1000
`\delimiterfactor` ratio for variable delimiters, times 1000
`\time` current time of day in minutes since midnight
`\day` current day of the month
`\month` current month of the year
`\year` current year of our Lord
`\showboxbreadth` maximum items per level when boxes are shown
`\showboxdepth` maximum level when boxes are shown
`\errorcontextlines` maximum extra context shown when errors occur

The first few of these parameters have values in units of “badness” and “penalties” that affect line breaking and page breaking. Then come demerit-oriented parameters; demerits are essentially given in units of “badness squared,” so those parameters tend to have larger values. By contrast, the next few parameters (`\looseness`, `\pausing`, etc.) generally have quite small values (either -1 or 0 or 1 or 2). Miscellaneous parameters complete the set.

A \langle dimen parameter \rangle is one of the following:

`\hfuzz` maximum overrun before overfull hbox messages occur
`\vfuzz` maximum overrun before overfull vbox messages occur
`\overfullrule` width of rules appended to overfull boxes
`\emergencystretch` reduces badnesses on final pass of line-breaking
`\hsize` line width in horizontal mode
`\vsize` page height in vertical mode
`\maxdepth` maximum depth of boxes on main pages
`\splitmaxdepth` maximum depth of boxes on split pages
`\boxmaxdepth` maximum depth of boxes on explicit pages
`\lineskiplimit` threshold where `\baselineskip` changes to `\lineskip`
`\delimitershortfall` maximum space not covered by a delimiter
`\nulldelimiterspace` width of a null delimiter
`\scriptspace` extra space after subscript or superscript
`\mathsurround` kerning before and after math in text
`\prelaysize` length of text preceding a display
`\displaywidth` length of line for displayed equation

`\displayindent` indentation of line for displayed equation
`\parindent` width of `\indent`
`\hangindent` amount of hanging indentation
`\hoffset` horizontal offset in `\shipout`
`\voffset` vertical offset in `\shipout`

Before typesetting a delimiter T_EX determines the size f of the formula to be covered as twice the maximum of the height and the depth of the formula. The size d of the delimiter should be

$$\min \left\{ f - \text{\delimitershortfall}, \text{\delimiterfactor} \cdot \frac{f}{1000} \right\} \leq d$$

And the possibilities for \langle glue parameter \rangle are:

`\baselineskip` desired glue between baselines
`\lineskip` interline glue if `\baselineskip` isn't feasible
`\parskip` extra glue just above paragraphs
`\abovedisplayskip` extra glue just above displays
`\abovedisplayshortskip` ditto, following short lines
`\belowdisplayskip` extra glue just below displays
`\belowdisplayshortskip` ditto, following short lines
`\leftskip` glue at left of justified lines
`\rightskip` glue at right of justified lines
`\topskip` glue at top of main pages
`\splittopskip` glue at top of split pages
`\tabskip` glue between aligned entries
`\spaceskip` glue between words, if nonzero
`\xspaceskip` glue between sentences, if nonzero
`\parfillskip` additional `\rightskip` at end of paragraphs

To above parameters (except `\parfillskip`) are assigned values appropriate for CM family typeset at 12pt baseline.

Finally, there are three permissible \langle mu glue parameter \rangle tokens:

`\thinmuskip` thin space in math formulas
`\medmuskip` medium space in math formulas
`\thickmuskip` thick space in math formulas

T_EX also has parameters that are token lists. Such parameters do not enter into the definitions of \langle number \rangle and such things. A \langle token parameter \rangle is any of:

`\output` the user's output routine
`\everypar` tokens to insert when a paragraph begins
`\everymath` tokens to insert when math in text begins
`\everydisplay` tokens to insert when display math begins
`\everyhbox` tokens to insert when an hbox begins
`\everyvbox` tokens to insert when a vbox begins
`\everyjob` tokens to insert when the job begins
`\everycr` tokens to insert after every `\cr` or nonredundant `\crcr`
`\errhelp` tokens that supplement an `\errmessage`

All of numeric parameters are listed below, but the code is commented out if no special value needs to be set. INITEX makes all parameters zero except where noted.

⟨ Assign initial values to parameters ⟩≡

⟨ Assign values to integer parameters ⟩
⟨ Assign values to dimen parameters ⟩
⟨ Assign values to glue parameters ⟩
⟨ Assign values to special registers ⟩

⟨ Assign values to integer parameters ⟩≡

```

296     \pretolerance=100
297     \tolerance=200 % INITEX sets this to 10000
298     \hbadness=1000
299     \vbadness=1000
300     \linepenalty=10
301     \hyphenpenalty=50
302     \exhyphenpenalty=50
303     \binoppenalty=700
304     \relpenalty=500
305     \clubpenalty=150
306     \widowpenalty=150
307     \displaywidowpenalty=50
308     \brokenpenalty=100
309     \predisplaypenalty=10000
310     % \postdisplaypenalty=0
311     % \interlinepenalty=0
312     % \floatingpenalty=0, set during \insert
313     % \outputpenalty=0, set before TeX enters \output
314     \doublehyphendemerits=10000
315     \finalhyphendemerits=5000
316     \adjdemerits=10000
317     % \looseness=0, cleared by TeX after each paragraph
318     % \pausing=0
319     % \holdinginserts=0
320     % \tracingonline=0
321     % \tracingmacros=0
322     % \tracingstats=0
323     % \tracingparagraphs=0
324     % \tracingpages=0
325     % \tracingoutput=0
326     \tracinglostchars=1
327     % \tracingcommands=0
328     % \tracingrestores=0
329     % \language=0
330     \uchyph=1
331     % \lefthyphenmin=2 \righthyphenmin=3 set below
332     % \globaldefs=0
333     % \maxdeadcycles=25 % INITEX does this
334     % \hangafter=1 % INITEX does this, also TeX after each paragraph
335     % \fam=0
336     % \mag=1000 % INITEX does this
337     % \escapechar='\ % INITEX does this
338     \defaultthyphenchar='\-
339     \defaultskewchar=-1
340     % \endlinechar='\^M % INITEX does this

```

```

341     \newlinechar=-1
342     \delimiterfactor=901
343     % \time=now % TeX does this at beginning of job
344     % \day=now % TeX does this at beginning of job
345     % \month=now % TeX does this at beginning of job
346     % \year=now % TeX does this at beginning of job
347     \showboxbreadth=5
348     \showboxdepth=3
349     \errorcontextlines=5

```

⟨ *Assign values to dimen parameters* ⟩≡

```

350     \hfuzz=0.1pt
351     \vfuzz=0.1pt
352     \overfullrule=5pt
353     \hsize=6.5in
354     \vsize=8.9in
355     \maxdepth=4pt
356     \splitmaxdepth=\maxdimen
357     \boxmaxdepth=\maxdimen
358     % \lineskiplimit=0pt, changed by \normalbaselines
359     \delimitershortfall=5pt
360     \nulldelimiterspace=1.2pt
361     \scriptspace=0.5pt
362     % \mathsurround=0pt
363     % \predisplaysize=0pt, set before TeX enters $$
364     % \displaywidth=0pt, set before TeX enters $$
365     % \displayindent=0pt, set before TeX enters $$
366     \parindent=20pt
367     % \hangindent=0pt, zeroed by TeX after each paragraph
368     % \hoffset=0pt
369     % \voffset=0pt

```

⟨ *Assign values to glue parameters* ⟩≡

```

370     % \baselineskip=0pt, changed by \normalbaselines
371     % \lineskip=0pt, changed by \normalbaselines
372     \parskip=0pt plus 1pt
373     \abovedisplayskip=12pt plus 3pt minus 9pt
374     \abovedisplayshortskip=0pt plus 3pt
375     \belowdisplayskip=12pt plus 3pt minus 9pt
376     \belowdisplayshortskip=7pt plus 3pt minus 4pt
377     % \leftskip=0pt
378     % \rightskip=0pt
379     \topskip=10pt
380     \splittopskip=10pt
381     % \tabskip=0pt
382     % \spaceskip=0pt
383     % \xspaceskip=0pt
384     \parfillskip=0pt plus 1fil

```

We also define special registers that function like parameters:

```

385 \newskip\smallskipamount \smallskipamount=3pt plus 1pt minus 1pt
386 \newskip\medskipamount \medskipamount=6pt plus 2pt minus 2pt
387 \newskip\bigskipamount \bigskipamount=12pt plus 4pt minus 4pt
388 \newskip\normalbaselineskip \normalbaselineskip=12pt
389 \newskip\normallineskip \normallineskip=1pt
390 \newdimen\normallineskiplimit \normallineskiplimit=0pt
391 \newdimen\jot \jot=3pt
392 \newcount\interdisplaylinepenalty \interdisplaylinepenalty=100
393 \newcount\interfootnotelinepenalty \interfootnotelinepenalty=100
394 \def\normalbaselines{\lineskip\normallineskip
395 \baselineskip\normalbaselineskip \lineskiplimit\normallineskiplimit}

```

Macros for text

Here we introduce macros that are used for basic formatting unrelated to mathematics.

```

< Provide macros for text formatting >≡
  < Supply basic macros for text formatting >
  < Define space macros >
  < Supply various paragraph shapes >
  < Define sectioning macros >
  < Supply ragged setting >
  < Supply 'boxing' macros >
  < Supply strut >
  < Provide alignment macros >
  < Establish spacing after punctuation characters >
  < Supply various ways to fill space >
  < Define \showhyphens macro >
  < Completing the job >

```

`\<tab>` and `\<return>` are defined so that they expand to `\<space>`; this helps to prevent confusion, since all three cases look identical when displayed on most computer terminals.

```

< Supply basic macros for text formatting >≡
396 \def\^^M{\ } % control <return> = control <space>
397 \def\^^I{\ } % same for <tab>

```

The control sequences `\endgraf` and `\endline` are made equivalent to TeX's primitive `\par` and `\cr` operations, since it is often useful to redefine the meanings of `\par` and `\cr` themselves. Then come the definitions of `\space` (a blank space), `\empty` (a list of no tokens), and `\null` (an empty hbox).

`\bgroup` and `\egroup` are made to provide “implicit” grouping characters that turn out to be especially useful in macro definitions.

```

< Supply basic macros for text formatting >+≡
398 \let\endgraf=\par \let\endline=\cr
399
400 \def\space{ }

```

```

401     \def\empty{}
402     \def\null{\hbox{}}
403
404     \let\bgroup={ \let\egroup=}

```

The `\obeylines` macro says ‘`\let^^M=\par`’ instead of ‘`\def^^M{\par}`’ because the `\let` technique allows constructions such as ‘`\let\par=\cr \obeylines \halign{...}`’ in which `\cr`’s need not be given within the alignment.

⟨ *Supply basic macros for text formatting* ⟩+≡

```

405     {\catcode'\^^M=active % these lines must end with %
406     \gdef\obeylines{\catcode'\^^M=active \let^^M\par}%
407     \global\let^^M\par} % this is in case ^^M appears in a \write
408     \def\obeyspaces{\catcode'\ \active}
409     {\obeyspaces\global\let =\space}

```

The macros `\lq`, `\rq`, `\lbrack`, and `\rbrack` are defined, for people who have difficulty typing quotation marks and/or brackets.

⟨ *Supply basic macros for text formatting* ⟩+≡

```

410     \def\lq{'} \def\rq{'}
411     \def\lbrack{[} \def\rbrack{]}

```

The macros `\enskip`, `\quad`, and `\qqquad` provide spaces that are legitimate breakpoints within a paragraph; `\enspace`, `\thinspace`, and `\negthinspace` produce space that cannot cause a break (although the space will disappear if it occurs just next to certain kinds of breaks). All six of these spaces are relative to the current font.

You can get horizontal space that never disappears by saying ‘`\hglue<glue>`’; this space is able to stretch or shrink. Similarly, there’s a vertical analog, ‘`\vglue<glue>`’.

The `\nointerlineskip` macro suppresses interline glue that would ordinarily be inserted before the next box in vertical mode; this is a “one shot” macro, but `\offinterlineskip` is more drastic—it sets things up so that future interline glue will be present, but zero. There also are macros for potentially breakable vertical spaces: `\smallskip`, `\medskip`, and `\bigskip`.

⟨ *Define space macros* ⟩≡

```

412     \def\thinspace{\kern .16667em }
413     \def\negthinspace{\kern-.16667em }
414     \def\enspace{\kern.5em }
415
416     \def\enskip{\hskip.5em\relax}
417     \def\quad{\hskip1em\relax}
418     \def\qqquad{\hskip2em\relax}
419
420     \def\smallskip{\vskip\smallskipamount}
421     \def\medskip{\vskip\medskipamount}
422     \def\bigskip{\vskip\bigskipamount}
423
424     \def\nointerlineskip{\prevdepth-1000\p@}
425     \def\offinterlineskip{\baselineskip-1000\p@
426     \lineskip\z@ \lineskiplimit\maxdimen}
427
428     \def\topglue{\nointerlineskip\vglue-\topskip\vglue} % for top of page
429     \def\vglue{\afterassignment\vgl@skip@=}

```

```

430     \def\vgl@{\par \dimen@\prevdepth \hrule height\z@
431         \nobreak\vskip\skip@ \prevdepth\dimen@}
432     \def\hgllue{\afterassignment\hgll@\skip@=}
433     \def\hgll@{\leavevmode \count@\spacefactor \vrule width\z@
434         \nobreak\hskip\skip@ \spacefactor\count@}

```

The following macros introduce penalty markers that make breaking less, or more, desirable. The `\break`, `\nobreak`, and `\allowbreak` macros are intended for use in any mode; the `~` (tie) and `\slash` (hyphen-like ‘/’) macros are intended for horizontal mode. The others are intended only for vertical mode, i.e., between paragraphs, so they begin with `\par`.

⟨ Define space macros ⟩≡

```

435     \def~{\penalty\@M \ } % tie
436     \def\slash/{\penalty\exhyphenpenalty} % a ‘/’ that acts like a ‘-’
437
438     \def\break{\penalty-\@M}
439     \def\nobreak{\penalty \@M}
440     \def\allowbreak{\penalty \z@}
441
442     \def\filbreak{\par\vfil\penalty-200\vfilneg}
443     \def\goodbreak{\par\penalty-500 }
444     \def\eject{\par\break}
445     \def\supereject{\par\penalty-\@MM}
446
447     \def\removelastskip{\ifdim\lastskip=\z@\else\vskip-\lastskip\fi}
448     \def\smallbreak{\par\ifdim\lastskip<\smallskipamount
449         \removelastskip\penalty-50\smallskip\fi}
450     \def\medbreak{\par\ifdim\lastskip<\medskipamount
451         \removelastskip\penalty-100\medskip\fi}
452     \def\bigbreak{\par\ifdim\lastskip<\bigskipamount
453         \removelastskip\penalty-200\bigskip\fi}

```

`\line`, `\leftline`, `\rightline`, and `\centerline` produce boxes of the full line width, while `\llap` and `\rlap` make boxes whose effective width is zero. The `\underbar` macro puts its argument into an hbox with a straight line at a fixed distance under it.

`\underbar` uses math mode to do its job, although the operation is essentially non-mathematical in nature. A few of the other macros below use math mode in similar ways; thus, \TeX ’s mathematical abilities prove to be useful even when no mathematical typesetting is actually being done. A special control sequence `\m@th` is used to “turn off” `\mathsurround` when such constructions are being performed.)

⟨ Supply ‘boxing’ macros ⟩≡

```

454     \def\line{\hbox to\hsize}
455     \def\leftline#1{\line{#1\hss}}
456     \def\rightline#1{\line{\hss#1}}
457     \def\centerline#1{\line{\hss#1\hss}}
458
459     \def\rlap#1{\hbox to\z@{#1\hss}}
460     \def\llap#1{\hbox to\z@{\hss#1}}
461
462     \def\m@th{\mathsurround\z@}
463     \def\underbar#1{\setbox\z@\hbox{#1}\dp\z@\z@
464         \m@th \underline{\box\z@}$}

```

A `\strut` is implemented here as a rule of width zero. The ‘`\relax`’ in this macro and in others below is necessary in case `\strut` appears first in an alignment entry, because T_EX is in a somewhat unpredictable mode at such times.

⟨ *Supply strut* ⟩≡

```
465 \newbox\strutbox
466 \setbox\strutbox=\hbox{\vrule height8.5pt depth3.5pt width\z@}
467 \def\strut{\relax\ifmmode\copy\strutbox\else\unhcopy\strutbox\fi}
```

The `\ialign` macro provides for alignments when it is necessary to be sure that `\tabskip` is initially zero. The `\hidewidth` macro can be used essentially as `\hfill` in alignment entries that are permitted to “stick out” of their column. There’s also `\multispan`, which permits alignment entries to span one or more columns.

⟨ *Provide alignment macros* ⟩≡

```
468 \def\hidewidth{\hskip\hideskip} % for alignment entries that can stick out
469 \def\ialign{\everycr{\tabskip\z@skip\halign} % initialized \halign
470 \newcount\mscount
471 \def\multispan#1{\omit \mscount#1\relax
472 \loop\ifnum\mscount>\@ne \sp@n\repeat}
473 \def\sp@n{\span\omit\advance\mscount\m@ne}
```

Now we get to the “tabbing” macros. They keep track of the tab positions by maintaining boxes full of empty boxes having the specified widths.

The macro `\+` has been declared ‘`\outer`’ here, so that T_EX will be better able to detect runaway arguments and definitions. A non-`\outer` version, called `\tabalign`, has also been provided in case it is necessary to use `\+` in some “inner” place. You can use `\tabalign` just like `\+`, except after `\settabs`.

⟨ *Provide alignment macros* ⟩+≡

```
474 \newif\ifus@ \newif\if@cr
475 \newbox\tabs \newbox\tabsyet \newbox\tabsdone
476
477 \def\cleartabs{\global\setbox\tabsyet\null \setbox\tabs\null}
478 \def\settabs{\setbox\tabs\null \futurelet\next\sett@b}
479 \let\+=\relax % in case this file is being read in twice
480 \def\sett@b{\ifx\next+\def\nxt{\afterassignment\s@tt@b\let\nxt}%
481 \else\let\nxt\s@tcols\fi \let\next\relax \nxt}
482 \def\s@tt@b{\let\nxt\relax \us@false\m@ketabbox}
483 \def\tabalign{\us@true\m@ketabbox} % non-\outer version of \+
484 \outer\def\+{\tabalign}
485 \def\s@tcols#1\columns{\count@#1\dimen@\hsize
486 \loop\ifnum\count@>\z@ \@nother \repeat}
487 \def\@nother{\dimen@ii\dimen@ \divide\dimen@ii\count@
488 \setbox\tabs\hbox{\hbox to\dimen@ii{\unhbox\tabs}}%
489 \advance\dimen@-\dimen@ii \advance\count@\m@ne}
490
491 \def\m@ketabbox{\begingroup
492 \global\setbox\tabsyet\copy\tabs
493 \global\setbox\tabsdone\null
494 \def\cr{\@crtrue\cr\egroup\egroup
495 \ifus@\unvbox\z@\lastbox\fi\endgroup
496 \setbox\tabs\hbox{\unhbox\tabsyet\unhbox\tabsdone}}%
```

```

497     \setbox\z@\vbox\bgroup\@crfalse
498     \ialign\bgroup&\t@bbox#\t@bb@x\cr}
499
500     \def\t@bbox{\setbox\z@\hbox\bgroup}
501     \def\t@bb@x{\if@cr\egroup % now \box\z@ holds the column
502     \else\hss\egroup \global\setbox\tabsyet\hbox{\unhbox\tabsyet
503     \global\setbox\@ne\lastbox}% now \box\@ne holds its size
504     \ifvoid\@ne\global\setbox\@ne\hbox to\wd\z@}%
505     \else\setbox\z@\hbox to\wd\@ne{\unhbox\z@}\fi
506     \global\setbox\tabsdone\hbox{\box\@ne\unhbox\tabsdone}\fi
507     \box\z@}

```

Paragraph shapes of a limited but important kind are provided by `\item`, `\itemitem`, and `\narrower`. A macro `\hang` causes hanging indentation by the normal amount of `\parindent`, after the first line; thus, the entire paragraph will be indented by the same amount (unless it began with `\noindent`). `\textindent{stuff}` is like `\indent`, but it puts the ‘stuff’ into the indentation, flush right except for an en space; it also removes spaces that might follow the right brace in ‘{stuff}’.

< Supply various paragraph shapes >≡

```

508     \def\hang{\hangindent\parindent}
509     \def\textindent#1{\indent\llap{#1\enspace}\ignorespaces}
510     \def\item{\par\hang\textindent}
511     \def\itemitem{\par\indent \hangindent2\parindent \textindent}
512     \def\narrower{\advance\leftskip\parindent
513     \advance\rightskip\parindent}

```

The `\beginsection` macro is intended to mark the beginning of a new major subdivision in a document; to use it, you say ‘`\beginsection`(section title)’ followed by a blank line (or `\par`). The macro first emits glue and penalties, designed to start a new page if the present page is nearly full; then it makes a `\bigskip` and puts the section title flush left on a line by itself, in boldface type. The section title is also displayed on the terminal. After a `\smallskip`, with page break prohibited, a `\noindent` command is given; this suppresses indentation in the next paragraph, i.e., in the first paragraph of the new section. (However, the next “paragraph” will be empty if vertical mode material immediately follows the `\beginsection` command.)

Special statements in a mathematical paper are often called theorems, lemmas, definitions, axioms, postulates, remarks, corollaries, algorithms, facts, conjectures, or some such things, and they generally are given special typographic treatment. The `\proclaim` macro puts the title of the proclamation in boldface, then sets the rest of the paragraph in slanted type. The paragraph is followed by something similar to `\medbreak`, except that the amount of penalty is different so that page breaks are discouraged:

< Define sectioning macros >≡

```

514     \outer\def\beginsection#1\par{\vskip\z@ plus.3\size\penalty-250
515     \vskip\z@ plus-.3\size\bigskip\vskip\parskip
516     \message{#1}\leftline{\bf#1}\nobreak\smallskip\noindent}
517     \outer\def\proclaim #1. #2\par{\medbreak
518     \noindent{\bf#1.\enspace}{\sl#2\par}%
519     \ifdim\lastskip<\medskipamount \removelastskip\penalty55\medskip\fi}

```

Ragged-right setting is initiated by restricting the spaces between words to have a fixed width, and by putting variable space at the right of each line. You should not call `\raggedright` until your text font has already been specified.

It is assumed that the ragged-right material will not be in a variety of different sizes. If this assumption is not valid, a different approach should be used: `\fontdimen` parameters 3 and 4 of the fonts you will be using should be set to zero, by saying, e.g., `'\fontdimen3\tenrm=0pt'`. These parameters specify the stretchability and shrinkability of interword spaces. A special macro `\ttraggedright` should be used for ragged-right setting in `typewriter type`, since the spaces between words are generally bigger in that style. (Spaces are already unstretchable and unshrinkable in font `cmtt`.)

< Supply ragged setting >≡

```
520 \def\raggedright
521   {\rightskip\z@ plus2em \spaceskip.3333em \xspaceskip.5em\relax}
522 \def\ttraggedright
523   {\tt\rightskip\z@ plus2em\relax} % for use with \tt only
```

The `\nonfrenchspacing` macro is be used to change the default `sfcodes` (set by `INITEX`) of punctuation marks. This macro, when called, affects spacing after punctuation marks.

< Establish spacing after punctuation characters >≡

```
524 \def\frenchspacing{\sfcode'\.\@m \sfcode'\?\@m \sfcode'\!\@m
525   \sfcode'\:\@m \sfcode'\;\@m \sfcode'\,\@m}
526 \def\nonfrenchspacing{\sfcode'\.3000\sfcode'\?3000\sfcode'\!3000%
527   \sfcode'\:2000\sfcode'\;1500\sfcode'\,1250 }
```

`\rightarrowfill` and `\leftarrowfill` macros use `\cleaders` with a repeatable box consisting of the middle 10 units of a minus sign, where one unit is $\frac{1}{18}$ em. The leaders are preceded and followed by `–` and `→`; there's enough backspacing to compensate for up to 5 units of extra space, fore and aft, that `\cleaders` might leave blank. In this way a macro is obtained such that `\hbox to 100pt{\rightarrowfill}` yields `'—————→'`.

The `\overbrace` and `\underbrace` macros of plain `TeX` are constructed by combining characters with rules. Font `cmex10` contains four symbols `⏟`, `⏞`, `⏟`, each of which has depth zero and height equal to the thickness of a rule that joins them properly. Therefore it's easy to define `\upbracefill` and `\downbracefill` macros so that you can obtain, e.g.,



by saying `'\hbox to 100pt{\downbracefill}\hbox to 50pt{\upbracefill}'` in vertical mode.

< Supply various ways to fill space >≡

```
528 \def\hrulefill{\leaders\hrule\hfill}
529 \def\dotfill{\cleaders\hbox{\$m@th \mkern1.5mu.\mkern1.5mu$}\hfill}
530 \def\rightarrowfill{\$m@th\smash-\mkern-6mu%
531   \cleaders\hbox{\$mkern-2mu\smash-\mkern-2mu$}\hfill
532   \mkern-6mu\mathord\rightarrow$}
533 \def\leftarrowfill{\$m@th\mathord\leftarrow\mkern-6mu%
534   \cleaders\hbox{\$mkern-2mu\smash-\mkern-2mu$}\hfill
535   \mkern-6mu\smash-$}
536 \mathchardef\bracedl="37A \mathchardef\bracerd="37B
537 \mathchardef\bracelu="37C \mathchardef\braceru="37D
538 \def\downbracefill{\$m@th \setbox\z@\hbox{\$bracedl$}%
539   \bracedl\leaders\vrule height\ht\z@ depth\z@\hfill\braceru
```

```

540     \bracelu\leaders\vrule height\ht\z@ depth\z@\hfill\bracerd$}
541     \def\upbracefill{${\m@th \setbox\z@\hbox{${\braceld$}}%
542     \bracelu\leaders\vrule height\ht\z@ depth\z@\hfill\bracerd
543     \braceld\leaders\vrule height\ht\z@ depth\z@\hfill\braceru$}

```

If you want to see all of the hyphens that plain T_EX will find in some random text, you can say ‘\showhyphens{<random text>}’ and the results will appear on your terminal (and in the log file). The \showhyphens macro creates an hbox that is intentionally underfull.

< Define \showhyphens macro >≡

```

544     \def\showhyphens#1{\setbox0\vbox{\parfillskip\z@skip\hsize\maxdimen\tenrm
545     \pretolerance\m@ne\tolerance\m@ne\hbadness0\showboxdepth0\ #1}}

```

At the end of a T_EX manuscript it’s usually best to finish everything off by typing ‘\bye’. The ‘\vfill\supereject’ gets T_EX to flush out all remaining insertions, with blank space filling the bottom of incomplete pages, and ‘\end’ sends the computer into its endgame routine.

< Completing the job >≡

```

546     \outer\def\bye{\par\vfill\supereject\end}

```

Macros for math

Most of this section consists of long listings of special symbols together with their font locations.

< Provide macros for math formatting >≡

```

< Define math space macros >
< Define Greek letters >
< Define math symbols >
< Encode large operators >
< Encode binary operations >
< Encode relations >
< Supply vertical and diagonal dots >
< Supply variable-width math accents >
< Supply extensible delimiters >
< Provide access to delimiters of various sizes >
< Supply common math functions >
< Provide \cases and \matrix macros >
< Provide support to typeset displayed equations >

```

T_EX does automatic spacing of math formulas so that they look right, and this is almost true. But occasionally you must give T_EX some help. The number of possible math formulas is vast, and T_EX’s spacing rules are rather simple, so it is natural that exceptions should arise. The basic elements of space that T_EX puts into formulas are called: thin spaces, medium spaces, thick spaces. The normal space between words of a paragraph is approximately equal to two thin spaces.

You can add your own spacing whenever you want to, by using the control sequences

```

\, thin space (normally 1/6 of a quad);
\> medium space (normally 2/9 of a quad);
\; thick space (normally 5/18 of a quad);
\! negative thin space (normally -1/6 of a quad).

```

⟨ Define math space macros ⟩≡

```
547     \def\,{\mskip\thinmuskip}
548     \def\>{\mskip\medmuskip}
549     \def\;{\mskip\thickmuskip}
550     \def\!{\mskip-\thinmuskip}
```

The next job is to define Greek letters and other symbols of type Ord. Uppercase Greek letters are assigned hexadecimal codes of the form "7xxx, so that they will change families when `\fam` changes.

`\mathchardef` defines a control sequence to be a synonym for a math character (check `\mathchar` for the meaning of hex number).

⟨ Define Greek letters ⟩≡

```
551     \mathchardef\alpha="010B
552     \mathchardef\beta="010C
553     \mathchardef\gamma="010D
554     \mathchardef\delta="010E
555     \mathchardef\epsilon="010F
556     \mathchardef\zeta="0110
557     \mathchardef\eta="0111
558     \mathchardef\theta="0112
559     \mathchardef\iota="0113
560     \mathchardef\kappa="0114
561     \mathchardef\lambda="0115
562     \mathchardef\mu="0116
563     \mathchardef\nu="0117
564     \mathchardef\xi="0118
565     \mathchardef\pi="0119
566     \mathchardef\rho="011A
567     \mathchardef\sigma="011B
568     \mathchardef\tau="011C
569     \mathchardef\upsilon="011D
570     \mathchardef\phi="011E
571     \mathchardef\chi="011F
572     \mathchardef\psi="0120
573     \mathchardef\omega="0121
574     \mathchardef\varepsilon="0122
575     \mathchardef\vartheta="0123
576     \mathchardef\varpi="0124
577     \mathchardef\varrho="0125
578     \mathchardef\varsigma="0126
579     \mathchardef\varphi="0127
580     \mathchardef\Gamma="7000
581     \mathchardef\Delta="7001
582     \mathchardef\Theta="7002
583     \mathchardef\Lambda="7003
584     \mathchardef\Xi="7004
585     \mathchardef\Pi="7005
586     \mathchardef\Sigma="7006
587     \mathchardef\Upsilon="7007
588     \mathchardef\Phi="7008
589     \mathchardef\Psi="7009
590     \mathchardef\Omega="700A
```

⟨ Define math symbols ⟩≡

```

591     \mathchardef\aleph="0240
592     \def\hbar{{\mathchar'26\mkern-9muh}}
593     \mathchardef\imath="017B
594     \mathchardef\jmath="017C
595     \mathchardef\ell="0160
596     \mathchardef\wp="017D
597     \mathchardef\Re="023C
598     \mathchardef\Im="023D
599     \mathchardef\partial="0140
600     \mathchardef\infty="0231
601     \mathchardef\prime="0230
602     \mathchardef\emptyset="023B
603     \mathchardef\nabla="0272
604     \def\surd{{\mathchar"1270}}
605     \mathchardef\top="023E
606     \mathchardef\bot="023F
607     \def\angle{{\vbox{\ialign{${\m@th\scriptstyle##$\crrc
608         \not\mathrel{\mkern14mu}\crrc
609         \noalign{\nointerlineskip}
610         \mkern2.5mu\leaders\hrule height.34pt\hfill\mkern2.5mu\crrc}}}}
611     \mathchardef\triangle="0234
612     \mathchardef\forall="0238
613     \mathchardef\exists="0239
614     \mathchardef\neg="023A \let\lnot=\neg
615     \mathchardef\flat="015B
616     \mathchardef\natural="015C
617     \mathchardef\sharp="015D
618     \mathchardef\clubsuit="027C
619     \mathchardef\diamondsuit="027D
620     \mathchardef\heartsuit="027E
621     \mathchardef\spadesuit="027F

```

Integral signs get special treatment so that their limits won't be set above and below.

⟨ Encode large operators ⟩≡

```

622     \mathchardef\coprod="1360
623     \mathchardef\bigvee="1357
624     \mathchardef\bigwedge="1356
625     \mathchardef\biguplus="1355
626     \mathchardef\bigcap="1354
627     \mathchardef\bigcup="1353
628     \mathchardef\intop="1352 \def\int{\intop\nolimits}
629     \mathchardef\prod="1351
630     \mathchardef\sum="1350
631     \mathchardef\bigotimes="134E
632     \mathchardef\bigoplus="134C
633     \mathchardef\bigodot="134A
634     \mathchardef\ointop="1348 \def\oint{\ointop\nolimits}
635     \mathchardef\bigsqcup="1346
636     \mathchardef\smallint="1273

```

⟨ Encode binary operations ⟩≡

```

637     \mathchardef\triangleleft="212F
638     \mathchardef\triangleright="212E
639     \mathchardef\bigtriangleup="2234
640     \mathchardef\bigtriangledown="2235
641     \mathchardef\wedge="225E \let\land=\wedge
642     \mathchardef\vee="225F \let\lor=\vee
643     \mathchardef\cap="225C
644     \mathchardef\cup="225B
645     \mathchardef\ddagger="227A
646     \mathchardef\dagger="2279
647     \mathchardef\sqcap="2275
648     \mathchardef\sqcup="2274
649     \mathchardef\uplus="225D
650     \mathchardef\amalg="2271
651     \mathchardef\diamond="2205
652     \mathchardef\bullet="220F
653     \mathchardef\wr="226F
654     \mathchardef\div="2204
655     \mathchardef\odot="220C
656     \mathchardef\oslash="220B
657     \mathchardef\otimes="220A
658     \mathchardef\ominus="2209
659     \mathchardef\oplus="2208
660     \mathchardef\mp="2207
661     \mathchardef\pm="2206
662     \mathchardef\circ="220E
663     \mathchardef\bigcirc="220D
664     \mathchardef\setminus="226E % for set difference \setminus B
665     \mathchardef\cdot="2201
666     \mathchardef\ast="2203
667     \mathchardef\times="2202
668     \mathchardef\star="213F

```

Relations are also fairly straightforward, except for the ones that are constructed from other characters. The `\mapstochar` is a character ‘`’` of width zero that is quite useless by itself, but it combines with right arrows to make `\mapsto` ‘`↦`’ and `\longmapsto` ‘`↪`’. Similarly, `\not` is a relation character of width zero that puts a slash over the character that follows. When two relations are adjacent in a math formula, `TEX` puts no space between them.

⟨ Encode relations ⟩≡

```

669     \mathchardef\propto="322F
670     \mathchardef\sqssubseteq="3276
671     \mathchardef\sqsupseteq="3277
672     \mathchardef\parallel="326B
673     \mathchardef\mid="326A
674     \mathchardef\dashv="3261
675     \mathchardef\vdash="3260
676     \mathchardef\nearrow="3225
677     \mathchardef\searrow="3226
678     \mathchardef\nwarrow="322D
679     \mathchardef\swarrow="322E
680     \mathchardef\Leftarrow="322C

```

```

681     \mathchardef\Leftarrow="3228
682     \mathchardef\Rightarrow="3229
683     \def\neq{\not=} \let\ne=\neq
684     \mathchardef\leq="3214 \let\le=\leq
685     \mathchardef\geq="3215 \let\ge=\geq
686     \mathchardef\succ="321F
687     \mathchardef\prec="321E
688     \mathchardef\approx="3219
689     \mathchardef\succeq="3217
690     \mathchardef\preceq="3216
691     \mathchardef\supset="321B
692     \mathchardef\subset="321A
693     \mathchardef\supseteq="3213
694     \mathchardef\subseteq="3212
695     \mathchardef\in="3232
696     \mathchardef\ni="3233 \let\owns=\ni
697     \mathchardef\gg="321D
698     \mathchardef\ll="321C
699     \mathchardef\not="3236
700     \mathchardef\leftrightharpoonup="3224
701     \mathchardef\leftarrow="3220 \let\gets=\leftarrow
702     \mathchardef\rightarrow="3221 \let\to=\rightarrow
703     \mathchardef\mapstochar="3237 \def\mapsto{\mapstochar\rightarrow}
704     \mathchardef\sim="3218
705     \mathchardef\simeq="3227
706     \mathchardef\perp="323F
707     \mathchardef\equiv="3211
708     \mathchardef\asymp="3210
709     \mathchardef\smile="315E
710     \mathchardef\frown="315F
711     \mathchardef\leftharpoonup="3128
712     \mathchardef\leftharpoondown="3129
713     \mathchardef\rightharpoonup="312A
714     \mathchardef\rightharpoondown="312B
715
716     \def\joinrel{\mathrel{\mkern-3mu}}
717     \def\relbar{\mathrel{\smash-}} % \smash, because - has the same height as +
718     \def\Relbar{\mathrel=}
719     \mathchardef\lhook="312C \def\hookrightarrow{\lhook\joinrel\rightarrow}
720     \mathchardef\rhook="312D \def\hookleftarrow{\leftarrow\joinrel\rhook}
721     \def\bowtie{\mathrel\triangleright\joinrel\mathrel\triangleleft}
722     \def\models{\mathrel|\joinrel=}
723     \def\Longrightarrow{\Relbar\joinrel\rightarrow}
724     \def\longrightarrow{\relbar\joinrel\rightarrow}
725     \def\longleftarrow{\leftarrow\joinrel\relbar}
726     \def\Longleftarrow{\Leftarrow\joinrel\Relbar}
727     \def\longmapsto{\mapstochar\longrightarrow}
728     \def\longleftrightharpoonup{\leftarrow\joinrel\rightarrow}
729     \def\Longleftrightharpoonup{\Leftarrow\joinrel\rightarrow}
730     \def\iff{\;\Longleftrightharpoonup\;}

```

After defining characters `\ldotp` and `\cdotp` that act as math punctuation, `\ldots` and `\cdots` macros are defined that give the proper spacing in most circumstances. Vertical and diagonal dots (`\vdots` and `\ddots`) are also provided here:

⟨ Supply vertical and diagonal dots ⟩≡

```

731     \mathchardef\ldotp="613A % ldot as a punctuation mark
732     \mathchardef\cdotp="6201 % cdot as a punctuation mark
733     \mathchardef\colon="603A % colon as a punctuation mark
734     \def\ldots{\mathinner{\ldotp\ldotp\ldotp}}
735     \def\cdots{\mathinner{\cdotp\cdotp\cdotp}}
736     \def\vdots{\vbox{\baselineskip4\p@ \lineskiplimit\z@
737         \kern6\p@\hbox{.}\hbox{.}\hbox{.}}}
738     \def\ddots{\mathinner{\mkern1mu\raise7\p@\vbox{\kern7\p@\hbox{.}}\mkern2mu
739         \raise4\p@\hbox{.}\mkern2mu\raise\p@\hbox{.}\mkern1mu}}

```

Most of the math accents are handled entirely by the `\mathaccent` primitive, but a few of the variable-width ones are constructed the hard way.

⟨ Supply variable-width math accents ⟩≡

```

740     \def\acute{\mathaccent"7013 }
741     \def\grave{\mathaccent"7012 }
742     \def\ddot{\mathaccent"707F }
743     \def\tilde{\mathaccent"707E }
744     \def\bar{\mathaccent"7016 }
745     \def\breve{\mathaccent"7015 }
746     \def\check{\mathaccent"7014 }
747     \def\hat{\mathaccent"705E }
748     \def\vec{\mathaccent"017E }
749     \def\dot{\mathaccent"705F }
750     \def\widetilde{\mathaccent"0365 }
751     \def\widehat{\mathaccent"0362 }
752     \def\overrightarrow#1{\vbox{\m@th\ialign{##\crrc
753         \rightarrowfill\crrc\noalign{\kern-\p@\nointerlineskip}
754         $\hfil\displaystyle{#1}\hfil$\crrc}}}}
755     \def\overleftarrow#1{\vbox{\m@th\ialign{##\crrc
756         \leftarrowfill\crrc\noalign{\kern-\p@\nointerlineskip}
757         $\hfil\displaystyle{#1}\hfil$\crrc}}}}
758     \def\overbrace#1{\mathop{\vbox{\m@th\ialign{##\crrc\noalign{\kern3\p@}
759         \downbracefill\crrc\noalign{\kern3\p@\nointerlineskip}
760         $\hfil\displaystyle{#1}\hfil$\crrc}}}\limits}
761     \def\underbrace#1{\mathop{\vtop{\m@th\ialign{##\crrc
762         $\hfil\displaystyle{#1}\hfil$\crrc\noalign{\kern3\p@\nointerlineskip}
763         \upbracefill\crrc\noalign{\kern3\p@}}}\limits}
764     \def\skew#1#2#3{\muskip\z@#1mu\divide\muskip\z@\tw@ \mkern\muskip\z@
765         #2{\mkern-\muskip\z@#3}\mkern\muskip\z@}\mkern-\muskip\z@}{}}

```

Now we come to 24 delimiters that can change their size. These are denoted explicitly by a ⟨27-bit number⟩.

If we denote 27-bit number by "cqrstuvwxyz, then delimiter codes are assigned by

`\delimiter "cqrstuvwxyz`, where

c — the class

q — the font family number of

- rs — the position of the the small variant of the delimiter
- t — the font family number of
- uv — the position of the the large variant of the delimiter

After `\left` and `\right` commands the class digit is ignored. When \TeX is not looking for a delimiter the righthmost three digits `tuv` are ignored, and the remaining four `cqrs` are treated as a `\mathchar`.

< Supply extensible delimiters >≡

```

766     \def\lmoustache{\delimiter"437A340 } % top from (, bottom from )
767     \def\rmoustache{\delimiter"537B341 } % top from ), bottom from (
768     \def\lgroup{\delimiter"462833A } % extensible ( with sharper tips
769     \def\rgroup{\delimiter"562933B } % extensible ) with sharper tips
770     \def\arrowvert{\delimiter"26A33C } % arrow without arrowheads
771     \def\Arrowvert{\delimiter"26B33D } % double arrow without arrowheads
772     \def\bracevert{\delimiter"77C33E } % the vertical bar that extends braces
773     \def\Vert{\delimiter"26B30D } \let\|= \Vert
774     \def\vert{\delimiter"26A30C }
775     \def\uparrow{\delimiter"3222378 }
776     \def\downarrow{\delimiter"3223379 }
777     \def\updownarrow{\delimiter"326C33F }
778     \def\Uparrow{\delimiter"322A37E }
779     \def\Downarrow{\delimiter"322B37F }
780     \def\Updownarrow{\delimiter"326D377 }
781     \def\backslash{\delimiter"26E30F } % for double coset G\backslash H
782     \def\rangle{\delimiter"526930B }
783     \def\langle{\delimiter"426830A }
784     \def\rbrace{\delimiter"5267309 } \let\}= \rbrace
785     \def\lbrace{\delimiter"4266308 } \let\{= \lbrace
786     \def\rceil{\delimiter"5265307 }
787     \def\lceil{\delimiter"4264306 }
788     \def\rfloor{\delimiter"5263305 }
789     \def\lfloor{\delimiter"4262304 }

```

In the `plain.tex` format and in the Computer Modern math fonts there is only one radical—the square root. The meaning of `\radical` is analogous to the `\delimiter` commands. Only the class number is dropped. Joining the radical character and the horizontal rule is done by letting the radical character have a large depth, and the height which is equal to the rule thickness. The rule is placed on the baseline and the radical character is placed below. Then the whole is centered around math axis.

< Supply extensible delimiters >+≡

```

790     \def\choose{\atopwithdelims()}
791     \def\brack{\atopwithdelims[]}
792     \def\brace{\atopwithdelims\{\}}
793
794     \def\sqrt{\radical"270370 }

```

These macros depend on actual sizes of delimiters.

⟨ Provide access to delimiters of various sizes ⟩≡

```

795     \def\bigl{\mathopen\big}
796     \def\bigm{\mathrel\big}
797     \def\bigr{\mathclose\big}
798     \def\Bigl{\mathopen\Big}
799     \def\Bigm{\mathrel\Big}
800     \def\Bigr{\mathclose\Big}
801     \def\biggl{\mathopen\bigg}
802     \def\biggm{\mathrel\bigg}
803     \def\biggr{\mathclose\bigg}
804     \def\Biggl{\mathopen\Bigg}
805     \def\Biggm{\mathrel\Bigg}
806     \def\Biggr{\mathclose\Bigg}
807     \def\big#1{\hbox{$\left#1\lrcorner to8.5\p@{}\right.\n@space$}}
808     \def\Big#1{\hbox{$\left#1\lrcorner to11.5\p@{}\right.\n@space$}}
809     \def\bigg#1{\hbox{$\left#1\lrcorner to14.5\p@{}\right.\n@space$}}
810     \def\Bigg#1{\hbox{$\left#1\lrcorner to17.5\p@{}\right.\n@space$}}
811     \def\n@space{\nulldelimiterspace\z@ \m@th}

```

The `\mathpalette` operation constructs a formula in all four styles; it is applied here in the implementation of `\phantom`, `\smash`, `\root`, and other operations. (Actually `\phantom` and `\smash` are not perfect: They assume that the current style is uncramped.)

These definitions illustrate how other built-up symbol combinations could be defined to work in all four styles.

The control sequences `\sp` and `\sb` are provided for people who can't easily type $\hat{\ }_-$; a “discretionary times sign” `*` is defined.

⟨ Provide macros for math formatting ⟩+≡

```

812     \def\mathpalette#1#2{\mathchoice{#1\displaystyle{#2}}%
813       {#1\textstyle{#2}}{#1\scriptstyle{#2}}{#1\scriptscriptstyle{#2}}}
814     \newbox\rootbox
815     \def\root#1\of{\setbox\rootbox\hbox{$\m@th\scriptscriptstyle{#1}$}
816       \mathpalette\root@}
817     \def\root@#1#2{\setbox\z@\hbox{$\m@th#1\sqrt{#2}$}
818       \dimen@ht\z@ \advance\dimen@-\dp\z@
819       \mkern5mu\raise.6\dimen@copy\rootbox \mkern-10mu \box\z@}
820     \newif\ifv@ \newif\ifh@
821     \def\vphantom{\v@true\h@false\ph@nt}
822     \def\hphantom{\v@false\h@true\ph@nt}
823     \def\phantom{\v@true\h@true\ph@nt}
824     \def\ph@nt{\ifmmode\def\next{\mathpalette\mathph@nt}%
825       \else\let\next\makeph@nt\fi\next}
826     \def\makeph@nt#1{\setbox\z@\hbox{#1}\finph@nt}
827     \def\mathph@nt#1#2{\setbox\z@\hbox{$\m@th#1{#2}$}\finph@nt}
828     \def\finph@nt{\setbox\tw@null
829       \ifv@ \ht\tw@\ht\z@ \dp\tw@\dp\z@\fi
830       \ifh@ \wd\tw@\wd\z@\fi \box\tw@}
831     \def\mathstrut{\vphantom{}}
832     \def\smash{\relax % \relax, in case this comes first in \halign
833       \ifmmode\def\next{\mathpalette\mathsm@sh}\else\let\next\makesm@sh
834       \fi\next}
835     \def\makesm@sh#1{\setbox\z@\hbox{#1}\finsm@sh}
836     \def\mathsm@sh#1#2{\setbox\z@\hbox{$\m@th#1{#2}$}\finsm@sh}

```

```

837     \def\finism@sh{\ht\z@\z@ \dp\z@\z@ \box\z@}
838
839     \def\cong{\mathrel{\mathpalette@vereq\sim}} % congruence sign
840     \def@vereq#1#2{\lower.5p@\vbox{\lineskiplimit\maxdimen\lineskip-.5p@
841       \ialign{\$m@th#1\hfil#\hfil$\crcr#2\crcr=\crcr}}
842     \def\notin{\mathrel{\mathpalette@cncel\in}}
843     \def@cncel#1#2{\m@th\oalign{\$hfil#1\mkern1mu/\hfil$\crcr$#1#2$}}
844     \def\rightharpoons{\mathrel{\mathpalette\rlh@{}}}
845     \def\rlh@#1{\vcenter{\m@th\hbox{\oalign{\raise2pt
846       \hbox{$#1\rightharpoonup$}\crcr
847       $#1\leftharpoondown$}}}}
848     \def\buildrel#1\over#2{\mathrel{\mathop{\kern\z@#2}\limits^{#1}}}
849     \def\doteq{\buildrel\textstyle.\over=}
850
851     \let\sp=^ \let\sb=_
852     \def\*{\discretionary{\thinspace\the\textfont2\char2}{}}{}}

```

The 32 common functions whose names generally appear in roman letters.

⟨ *Supply common math functions* ⟩≡

```

853     \def\log{\mathop{\rm log}\nolimits}
854     \def\lg{\mathop{\rm lg}\nolimits}
855     \def\ln{\mathop{\rm ln}\nolimits}
856     \def\lim{\mathop{\rm lim}}
857     \def\limsup{\mathop{\rm lim}\,sup}
858     \def\liminf{\mathop{\rm lim}\,inf}
859     \def\sin{\mathop{\rm sin}\nolimits}
860     \def\arcsin{\mathop{\rm arcsin}\nolimits}
861     \def\sinh{\mathop{\rm sinh}\nolimits}
862     \def\cos{\mathop{\rm cos}\nolimits}
863     \def\arccos{\mathop{\rm arccos}\nolimits}
864     \def\cosh{\mathop{\rm cosh}\nolimits}
865     \def\tan{\mathop{\rm tan}\nolimits}
866     \def\arctan{\mathop{\rm arctan}\nolimits}
867     \def\tanh{\mathop{\rm tanh}\nolimits}
868     \def\cot{\mathop{\rm cot}\nolimits}
869     \def\coth{\mathop{\rm coth}\nolimits}
870     \def\sec{\mathop{\rm sec}\nolimits}
871     \def\csc{\mathop{\rm csc}\nolimits}
872     \def\max{\mathop{\rm max}}
873     \def\min{\mathop{\rm min}}
874     \def\sup{\mathop{\rm sup}}
875     \def\inf{\mathop{\rm inf}}
876     \def\arg{\mathop{\rm arg}\nolimits}
877     \def\ker{\mathop{\rm ker}\nolimits}
878     \def\dim{\mathop{\rm dim}\nolimits}
879     \def\hom{\mathop{\rm hom}\nolimits}
880     \def\det{\mathop{\rm det}}
881     \def\exp{\mathop{\rm exp}\nolimits}
882     \def\Pr{\mathop{\rm Pr}}
883     \def\gcd{\mathop{\rm gcd}}
884     \def\deg{\mathop{\rm deg}\nolimits}
885

```

```

886     \def\bmod{\nonscript\mskip-\medmuskip\mkern5mu
887     \mathbin{\rm mod}\penalty900\mkern5mu\nonscript\mskip-\medmuskip}
888     \def\pmod#1{\allowbreak\mkern18mu({\rm mod}\,\,\,#1)}

```

The definition of `\matrix` goes to some pains to ensure that two n -rowed matrices will have the same height and the same depth, unless at least one of their rows is unusually big. The definition of `\bordermatrix` is even more complicated, but it seems to work reasonably well; it uses a constant `\p@renwd` that represents the width of a big extensible left parenthesis.

< Provide \cases and \matrix macros >≡

```

889     \def\cases#1{\left\{\,\,\vcenter{\normalbaselines\m@th
890     \ialign{###\hfil$&\quad##\hfil\crr#1\crr}}\right.}
891     \def\matrix#1{\null\,\vcenter{\normalbaselines\m@th
892     \ialign{\hfil$###\hfil&\quad\hfil$##\hfil\crr
893     \mathstrut\crr\noalign{\kern-\baselineskip}
894     #1\crr\mathstrut\crr\noalign{\kern-\baselineskip}}\,\,}
895     \def\pmatrix#1{\left(\matrix{#1}\right)}
896     \newdimen\p@renwd
897     \setbox0=\hbox{\tenex B} \p@renwd=\wd0 % width of the big left (
898     \def\bordermatrix#1{\begingroup \m@th
899     \setbox\z@\vbox{\def\crr{\crr\noalign{\kern2\p@global\let\cr\endline}}%
900     \ialign{###\hfil\kern2\p@\kern\p@renwd&\thinspace\hfil$##\hfil
901     &\quad\hfil$##\hfil\crr
902     \omit\strut\hfil\crr\noalign{\kern-\baselineskip}%
903     #1\crr\omit\strut\crr}%
904     \setbox\tw@\vbox{\unvcopy\z@\global\setbox\@ne\lastbox}%
905     \setbox\tw@\hbox{\unhbox\@ne\unskip\global\setbox\@ne\lastbox}%
906     \setbox\tw@\hbox{\$ \kern\wd\@ne\kern-\p@renwd\left(\kern-\wd\@ne
907     \global\setbox\@ne\vbox{\box\@ne\kern2\p@}%
908     \vcenter{\kern-\ht\@ne\unvbox\z@\kern-\baselineskip}\,\,\right)$}%
909     \null\;\vbox{\kern\ht\@ne\box\tw@}\endgroup}

```

The value of `\lineskiplimit` is assumed to be `\normallineskiplimit` plus the accumulated amount of “opening up.” Thus, the `\vskip` instructions in `\display` will compensate for the fact that the first baseline of an alignment is separated by an opened-up `baselineskip` from the last line preceding the display.

< Provide support to typeset displayed equations >≡

```

910     \def\openup{\afterassignment\@openup\dimen@=}
911     \def\@openup{\advance\lineskip\dimen@
912     \advance\baselineskip\dimen@
913     \advance\lineskiplimit\dimen@}
914     \def\eqalign#1{\null\,\vcenter{\openup\jot\m@th
915     \ialign{\strut\hfil$\displaystyle{##}$&\displaystyle{\{##}$\hfil
916     \crr#1\crr}}\,\,}
917     \newif\ifdt@p
918     \def\displayy{\global\dt@ptrue\openup\jot\m@th
919     \everycr{\noalign{\ifdt@p \global\dt@pfalse \ifdim\prevdepth>-1000\p@
920     \vskip-\lineskiplimit \vskip\normallineskiplimit \fi
921     \else \penalty\interdisplaylinepenalty \fi}}
922     \def\@lign{\tabskip\z@skip\everycr{}} % restore inside \display
923     \def\displaylines#1{\display \tabskip\z@skip
924     \halign{\hbox to\displaywidth{\$@lign\hfil\displaystyle##\hfil}$\crr

```

```

925     #1\crrc}}
926 \def\eqalignno#1{\displ@y \tabskip\centering
927   \halign to\displaywidth{\hfil$\@lign\displaystyle{##}$\tabskip\z@skip
928     & $\@lign\displaystyle{\{}##\}$\hfil\tabskip\centering
929     & \llap{ $\@lign##$ }\tabskip\z@skip\crrc
930     #1\crrc}}
931 \def\leqalignno#1{\displ@y \tabskip\centering
932   \halign to\displaywidth{\hfil$\@lign\displaystyle{##}$\tabskip\z@skip
933     & $\@lign\displaystyle{\{}##\}$\hfil\tabskip\centering
934     & \kern-\displaywidth\rlap{ $\@lign##$ }\tabskip\displaywidth\crrc
935     #1\crrc}}

```

Below we have an interesting set of macros that convert f''' into $f^{\prime\prime\prime}$.

< Supply common math functions >+≡

```

936   {\catcode'\=' \active \gdef'\bgroup\prim@s}}
937   \def\prim@s{\prime\futurelet\next\pr@m@s}
938   \def\pr@m@s{\ifx'\next\let\nxt\pr@@@s \else\ifx'\next\let\nxt\pr@@@t
939     \else\let\nxt\egroup\fi\fi \nxt}
940   \def\pr@@@s#1{\prim@s} \def\pr@@@t#1#2{#2\egroup}
941   {\catcode'\^^Z=\active \gdef^^Z{\not=}} % ^^Z is like \ne in math
942   {\catcode'\_=\active \global\let\_=_} % _ in math is either subscript or \_

```

Macros for output

< Prepare page for output >≡

```

  < Supply headers and footers >
  < Supply footnotes >
  < Supply floating insertions >
  < Supply raggedbottom setting >
  < Set up the output routine >

```

The `\makeheadline` macro constructs a vbox of height and depth zero. The magic constant -22.5pt is equal to

$$\backslashtopskip - (\text{height of strut}) - 2\backslashbaselineskip$$

i.e., $10\text{pt} - 8.5\text{pt} - 24\text{pt}$ (assuming default values of `\topskip` and the height of the strut); this places the reference point of the headline exactly 24pt above the reference point of the top line on the page, unless the headline or the top line are excessively large.

The `\advancepageno` macro normally advances `\pageno` by $+1$; but if `\pageno` is negative (for roman numerals), the advance is by -1 . The new value of `\pageno` will be appropriate for the next time the output routine is called into action.

< Supply headers and footers >≡

```

943   \countdef\pageno=0 \pageno=1 % first page is number 1
944   \newtoks\headline \headline={\hfil} % headline is normally blank
945   \newtoks\footline \footline={\hss\tenrm\folio\hss}
946   % footline is normally a centered page number in font \tenrm
947   %
948   \def\makeheadline{\vbox to\z@{\vskip-22.5\p@
949     \line{\vbox to8.5\p@{\the\headline}\vss}\nointerlineskip}

```

```

950     \def\makefootline{\baselineskip24\p@\line{\the\footline}}
951     %
952     \def\folio{\ifnum\pageno<\z@ \romannumeral-\pageno \else\number\pageno \fi}
953     \def\nopagenumbers{\footline{\hfil}} % blank out the footline
954     \def\advancepageno{\ifnum\pageno<\z@ \global\advance\pageno\m@ne
955     \else\global\advance\pageno\@ne \fi} % increase |pageno|

```

Ragged-bottom setting is achieved by inserting infinite glue, which overpowers the stretchability of `\topskip`. This macros assume that `\topskip = 10pt`

⟨ Supply raggedbottom setting ⟩≡

```

956     \newif\ifr@ggedbottom
957     \def\raggedbottom{\topskip 10\p@ plus60\p@ \r@ggedbottomtrue}
958     \def\normalbottom{\topskip 10\p@ \r@ggedbottomfalse} % undoes \raggedbottom

```

There are 255 classes of insertions, `\insert0` to `\insert254`, and they are tied to other registers of the same number. For example, `\insert100` is connected with `\count100`, `\dimen100`, `\skip100`, and `\box100`.

For our purposes let's consider a particular class of insertions called class n ; we will then be dealing with $\text{T}_{\text{E}}\text{X}$'s primitive command

```
\insert n{⟨vertical mode material⟩}
```

which puts an insertion item into a horizontal or vertical list. For this class of insertions

```

\box n    is where the material appears when a page is output;
\count n  is the magnification factor for page breaking;
\dimen n  is the maximum insertion size per page;
\skip n   is the extra space to allocate on a page.

```

For example, material inserted with `\insert100` will eventually appear in `\box100`.

`\footnote` macro depends on the value of `\bigskipamount`, and `\parindent`. Because the value of `\vsize` (8.9 in) is greater than `\dimen\footins` (8 in), footnotes never fill up the whole page.

⟨ Supply footnotes ⟩≡

```

959     \newinsert\footins
960     \def\footnote#1{\let\@sf\empty % parameter #2 (the text) is read later
961     \ifhmode\edef\@sf{\spacefactor\the\spacefactor}\fi
962     #1\@sf\vfootnote{#1}}
963     \def\vfootnote#1{\insert\footins\bgroup
964     \interlinepenalty\interfootnotelinepenalty
965     \splittopskip\ht\strutbox % top baseline for broken footnotes
966     \splitmaxdepth\dp\strutbox \floatingpenalty\@MM
967     \leftskip\z@skip \rightskip\z@skip \spaceskip\z@skip \xspaceskip\z@skip
968     \textindent{#1}\footstrut\futurelet\next\fo@t}
969     \def\fo@t{\ifcat\bgroup\noexpand\next \let\next\fo@t
970     \else\let\next\fo@t\fi \next}
971     \def\fo@t{\bgroup\aftergroup\@foot\let\next}
972     \def\fo@t#1{#1\@foot}
973     \def\@foot{\strut\egroup}
974     \def\footstrut{\vbox to\splittopskip{}}
975
976     \skip\footins=\bigskipamount % space added when footnote is present

```

```

977     \count\footins=1000 % footnote magnification factor (1 to 1)
978     \dimen\footins=8in % maximum footnotes per page
979
980     \def\footnoterule{\kern-3\p@
981       \hrule width 2truein \kern 2.6\p@} % the \hrule is .4pt high

```

Here the constant $12\text{\p@} = 12\text{pt}$ is hard coded into the format.

〈 Supply floating insertions 〉≡

```

982     \newinsert\topins
983     \newif\ifp@ge \newif\if@mid
984     \def\topinsert{\@midfalse\p@gefalse\@ins}
985     \def\midinsert{\@midtrue\@ins}
986     \def\pageinsert{\@midfalse\p@getrue\@ins}
987     \skip\topins=\z@skip % no space added when a topinsert is present
988     \count\topins=1000 % magnification factor (1 to 1)
989     \dimen\topins=\maxdimen % no limit per page
990     \def\@ins{\par\begingroup\setbox\z@\vbox\bgroup} % start a \vbox
991     \def@endinsert{\egroup % finish the \vbox
992       \if@mid \dimen@ht\z@ \advance\dimen@dp\z@ \advance\dimen@12\p@
993       \advance\dimen@pagetotal \advance\dimen@-\pageshrink
994       \ifdim\dimen@>\pagegoal\@midfalse\p@gefalse\fi\fi
995       \if@mid \bigskip\box\z@\bigbreak
996       \else\insert\topins{\penalty100 % floating insertion
997         \splittopskip\z@skip
998         \splitmaxdepth\maxdimen \floatingpenalty\z@
999         \ifp@ge \dimen@dp\z@
1000         \vbox to\vsizel{\unvbox\z@\kern-\dimen@}% depth is zero
1001         \else \box\z@\nobreak\bigskip\fi}\fi\endgroup}

```

The value of \boxmaxdepth is set to \maxdepth so that the \vbox will be constructed under the assumptions that \TeX 's page builder has used to set up \box255 .

The \pagecontents macro produces a vertical list for everything that belongs on the main body of the page, namely the contents of \box255 together with illustrations (inserted at the top) and footnotes (inserted at the bottom). \topins and \footins are the insertion class numbers for the two kinds of insertions used in plain \TeX ; if more classes of insertions are added, \pagecontents should be changed accordingly. Notice that the boxes are unboxed so that the glue coming from insertions can help out the glue on the main page. The \footnoterule macro places a dividing line between the page and its footnotes; it makes a net contribution of 0 pt to the height of the vertical list.

The \dosupereject macro is designed to clear out any insertions that have been held over, whether they are illustrations or footnotes or both. The negative \kern here cancels out the natural space of the \topskip glue that goes above the empty \line ; that empty line box prevents the \vfill from disappearing into a page break. The vertical list that results from \dosupereject is placed on \TeX 's list of things to put out next, just after the straggling insertions have been reconsidered. Hence another super-eject will occur, and the process will continue until no insertions remain.

〈 Set up the output routine 〉≡

```

1002     \output{\plainoutput}
1003     \def\plainoutput{\shipout\vbox{\makeheadline\pagebody\makefootline}%
1004       \advancepageno
1005       \ifnum\outputpenalty>-\@MM \else\dosupereject\fi}
1006

```

```

1007     \def\pagebody{\vbox to\vsizel{\boxmaxdepth\maxdepth \pagecontents}}
1008
1009     \def\dosupereject{\ifnum\insertpenalties>\z@ % something is being held over
1010       \line{\kern-\topskip\nobreak\vfill\supereject\fi}
1011
1012     \def\pagecontents{\ifvoid\topins\else\unvbox\topins\fi
1013       \dimen@=\dp\@cclv \unvbox\@cclv % open up \box255
1014       \ifvoid\footins\else % footnote info is present
1015         \vskip\skip\footins
1016         \footnoterule
1017         \unvbox\footins\fi
1018       \ifr@ggedbottom \kern-\dimen@ \vfil \fi}

```

Hyphenation

A discretionary break consists of three sequences of characters called the *pre-break*, *post-break*, and *no-break* texts. The idea is that if a line break occurs here, the pre-break text will appear at the end of the current line and the post-break text will occur at the beginning of the next line; but if no break occurs, the no-break text will appear in the current line. The discretionary are specified by writing

```
\discretionary{⟨pre-break text⟩}{⟨post-break text⟩}{⟨no-break text⟩}
```

where the three texts consist entirely of characters, boxes, and kerns. If a word contains discretionary breaks \TeX will not hyphenate it.

Hyphenation exceptions are specified with the statements like

```
\hyphenation{gal-axy iso-peri-met-ric}
```

which gives to \TeX locations where these words may be hyphenated.

The default values of `\hyphenchar` is ‘-’.

Sometimes the typewriter fonts are given `\hyphenchar(font name)=-1` which value inhibits hyphenation.

⟨ *Read hyphenation patterns* ⟩≡

```

1019     \lefthyphenmin=2 \righthyphenmin=3 % disallow x- or -xx breaks
1020     \input hyphen

```

Initialization

⟨ *Initialize the layout* ⟩≡

```

1021     \normalbaselines\rm % select roman font
1022     \nonfrenchspacing % punctuation affects the spacing

```

Programming support

⟨ Provide programming constructs ⟩≡

⟨ Supply loops and conditionals ⟩
⟨ Define \tracingall macro ⟩

The `\loop...repeat` macro provides for iterative operations. In this macro and several others, the control sequence ‘`\next`’ is given a temporary value that is not going to be needed later; thus, `\next` acts like a “scratch control sequence.”

The macro `\newif` to be used for definitions of new conditionals. For example, `\newif\iffoo` creates `\footrue`, `\foofalse` to go with `\iffoo`.

⟨ Supply loops and conditionals ⟩≡

```

1023     \def\loop#1\repeat{\def\body{#1}\iterate}
1024     \def\iterate{\body \let\next\iterate \else\let\next\relax\fi \next}
1025     \let\repeat=\fi % this makes \loop...\if...\repeat skippable
1026
1027     \outer\def\newif#1{\count@\escapechar \escapechar\m@ne
1028     \expandafter\expandafter\expandafter
1029     \edef@if#1{true}{\let\noexpand#1=noexpand\iftrue}%
1030     \expandafter\expandafter\expandafter
1031     \edef@if#1{false}{\let\noexpand#1=noexpand\iffalse}%
1032     \@if#1{false}\escapechar\count@} % the condition starts out false
1033     \def@if#1#2{\csname\expandafter\if@string#1#2\endcsname}
1034     {\\uccode'1='i \uccode'2='f \uppercase{\gdef\if@12{}} % 'if' is required

```

⟨ Define \tracingall macro ⟩≡

```

1035     \def\tracingall{\tracingonline\@ne\tracingcommands\tw@\tracingstats\tw@
1036     \tracingpages\@ne\tracingoutput\@ne\tracinglostchars\@ne
1037     \tracingmacros\tw@\tracingparagraphs\@ne\tracingrestores\@ne
1038     \showboxbreadth\maxdimen\showboxdepth\maxdimen\errorstopmode}

```

The format name and version number are recorded in control sequences, in order to help the people who might have to explain why something doesn't work.

⟨ Identify the format ⟩≡

```

1039     \def\fmtname{plain+W}\def\fmtversion{3.14159+W}

```

Appendices

Efficiency and memory-space considerations

One difficulty with large sets of macros is that they take up space. It would be nice to preload every macro that every T_EX user has ever dreamed up; but there might not be enough room, because T_EX's memory capacity is finite. You might find it necessary to hold back and to load only the macros that are really needed.

How much memory space does a macro require?

There are four kinds of memory involved: token memory, name memory, string memory, and character memory. (If any of these becomes too full, it will be necessary to increase what T_EX calls the macro memory size, the hash size, the number of strings, and/or the pool size, respectively. The token memory is most important; a macro takes one cell of token memory for each token in its definition, including the '{' and the '}'. For example, the comparatively short definition

```
\def\example#1\two{\four}
```

takes five tokens: #1, \two, {1, \four, and }2. Each control sequence also takes up one cell of name memory, one cell of string memory, and as many cells of character memory as there are characters in the name (seven in the case of \example). Character memory is comparatively cheap; four characters, or in some cases five, will fit in the same number of bits as a single cell of token memory, inside the machine. Therefore you don't save much by choosing short macro names.

T_EX will tell you how close you come to exceeding its current memory capacity if you say \tracingstats=1. T_EX governs fourteen kinds of memory:

```
number of strings    (names of control sequences and files)
pool size           (the characters in such names)
main memory size    (boxes, glue, breakpoints, token lists, characters, etc.)
hash size          (control sequence names)
font memory        (font metric data)
exception dictionary (hyphenation exceptions)
input stack size    (simultaneous input sources)
semantic nest size  (unfinished lists being constructed)
parameter stack size (macro parameters)
buffer size        (characters in lines being read from files)
save size          (values to restore at group ends)
text input levels   (\input files and error insertions)
grouping levels     (unfinished groups)
pattern memory     (hyphenation pattern data)
```

The current amount of memory available will also be shown.

One obvious way to keep from loading too many macros is to keep the macro files short and to \input only the ones that you need.

Extensible delimiters

\TeX builds large delimiters by using “extensible” characters, which are specified by giving top, middle, bottom, and repeatable characters in an **extensible** command. For example, the extensible left parentheses in `cmex10` are defined by (see Figure 4)

```
extensible oct"060": oct"060", 0, oct"100", oct"102";
```

this says that character code `oct"060"` specifies an extensible delimiter constructed from itself as the top piece, from character number `oct"100"` as the bottom piece, and from character number `oct"102"` as the piece which should be repeated as often as necessary to reach a desired size. In this particular example there is no middle piece, but characters like curly braces have a middle piece as well. A zero value in the top, middle, or bottom position means that no character should be used in that part of the construction; but a zero value in the final position means that character number zero is the repeater. The width of an extensible character is taken to be the width of the repeater.

Also several characters of various sizes can be linked together in a series by means of a **charlist** command. For example (see Figure 4),

```
charlist oct"000": oct"020": oct"022": oct"040";
```

is used in the font `cmex10` to specify the left parentheses that \TeX uses in displayed math formulas. \TeX follows `charlist` to make variable-size delimiters and variable-size accents, as well to link `\textstyle` and the `\displaystyle` operators.

Font dimensions

The main information about font consists of the dimensions of the characters. These numbers \TeX finds in the font metric files. Except character dimensions, font metric files contain: values for `\fontdimen` parameters, italic correction of characters, ligature and kerning programs for characters. We change the `fontdimen` parameters with the (global) assignment:

```
\fontdimen<number>\font\equals<dimen>,
```

for example, the assignment `\fontdimen8\tenex = 0.6pt` changes width of the fraction bar from default 0.4 pt to 0.6 pt.

The first seven `fontdimen` parameters have the following meaning:

1. the slant per point
2. the interword space; that is used unless `\spaceskip` is specified
3. interword stretch
4. interword shrink
5. the x-height
6. the quad width (for the font in family 2, 1/18 th quad width is equal to 1 mu)
7. the extra space; that value is added to the interword space used whenever `\spacefactor` \geq 2000, unless `\xspaceskip` is specified.

For the font in family 2 attributes 8–19 specify positioning of fractions, subscripts, superscripts.

fraction numerator attributes: minimum shift up, from the main baseline, of the baseline of the numerator of a generalized fraction

8. num1: for display style
9. num2: for text style or smaller if a fraction bar is present
10. num3: for text style or smaller if no a fraction bar is present

fraction denominator attributes: minimum shift down, from the main baseline, of the baseline of the denominator of a generalized fraction

11. denom1: for display style
12. denom2: for text style or smaller

superscript attributes: minimum shift up, from the main baseline, of the baseline of the superscript

13. sup1: for display style
14. sup2: for text style or smaller, non-cramped
15. sup3: for text style or smaller, cramped

subscript attributes: minimum shift down, from the main baseline, of the baseline of a subscript

16. sub1: when no superscript is present
17. sub2: when superscript is present

script adjustment attributes: for use only with non-glyph, that is, composite objects

18. sup_drop: maximum distance of superscript baseline below top of nucleus
19. sub_drop: minimum distance of subscript baseline below bottom of nucleus

Delimiter span attributes: height plus depth of delimiter enclosing a generalized fraction.

20. delim1: in display style
21. delim2: in text style or smaller

The last parameter, the height of the math axis, specifies the height above the baseline of the fraction bar, and the centre of large delimiters and most operators and relations. This position is used in vertical centering.

22. axis_height.

For the font in family 3 attributes 9–12 determine extra space added when limits are attached to operators. The attribute 8 specifies thickness of the rule used for overlines, underlines, radical extenders, and fraction bars. From that dimension are derived ‘clearances’ around fraction bar. The attribute 13 specifies extra space added above and below attached limits.

8. default_rule_thickness
9. big_op_spacing1
10. big_op_spacing2
11. big_op_spacing3
12. big_op_spacing4
13. big_op_spacing5

We have: $\text{big_op_spacing1}(2) \leq \text{space between upper (lower) limit and top (bottom) of large operator} \leq \text{big_op_spacing3}(4)$

<i>fontdimen</i>	<i>cmsy10</i>	<i>cmex10</i>	<i>cmtt10</i>	<i>cmr10</i>	<i>cmti10</i>	<i>cmbx10</i>
1.	0.25pt	0.0pt	0.0pt	0.0pt	0.25pt	0.0pt
2.	0.0pt	0.0pt	5.24995pt	3.33333pt	3.57774pt	3.83331pt
3.	0.0pt	0.0pt	0.0pt	1.66666pt	1.53333pt	1.91666pt
4.	0.0pt	0.0pt	0.0pt	1.11111pt	1.0222pt	1.27777pt
5.	4.30554pt	4.30554pt	4.30554pt	4.30554pt	4.30554pt	4.44444pt
6.	10.00002pt	10.00002pt	10.4999pt	10.00002pt	10.22217pt	11.49994pt
7.	0.0pt	0.0pt	5.24995pt	1.11111pt	1.0222pt	1.27777pt
8.	6.76508pt	0.39998pt				
9.	3.93732pt	1.11111pt				
10.	4.4373pt	1.66666pt				
11.	6.85951pt	1.99998pt				
12.	3.44841pt	6.0pt				

13.	4.12892pt	1.0pt
14.	3.62892pt	
15.	2.88889pt	
16.	1.49998pt	
17.	2.47217pt	
18.	3.86108pt	
19.	0.5pt	
20.	23.9pt	
21.	10.09999pt	
22.	2.5pt	

Font tables

Figure 1. cmr10—family 0

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	Γ	Δ	Θ	Λ	Ξ	Π	Σ	Υ	"0x
'01x	Φ	Ψ	Ω	ff	fi	fl	ffi	ffl	
'02x	ı	ı	`	'	˘	˙	-	°	"1x
'03x	˚	ß	æ	œ	ø	Æ	Œ	Ø	
'04x	-	!	"	#	\$	%	&	'	"2x
'05x	()	*	+	,	-	.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;	i	=	ı	?	
'10x	@	A	B	C	D	E	F	G	"4x
'11x	H	I	J	K	L	M	N	O	
'12x	P	Q	R	S	T	U	V	W	"5x
'13x	X	Y	Z	["]	^	·	
'14x	`	a	b	c	d	e	f	g	"6x
'15x	h	i	j	k	l	m	n	o	
'16x	p	q	r	s	t	u	v	w	"7x
'17x	x	y	z	-	—	"	~	..	
	"8	"9	"A	"B	"C	"D	"E	"F	

Figure 2. cmmi10—family 1

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	Γ	Δ	Θ	Λ	Ξ	Π	Σ	Υ	"0x
'01x	Φ	Ψ	Ω	α	β	γ	δ	ϵ	
'02x	ζ	η	θ	ι	κ	λ	μ	ν	"1x
'03x	ξ	π	ρ	σ	τ	υ	ϕ	χ	
'04x	ψ	ω	ε	ϑ	ϖ	ϱ	ς	φ	"2x
'05x	\leftarrow	\nrightarrow	\rightarrow	\dashrightarrow	\leftarrow	\rightarrow	\triangleright	\triangleleft	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	.	,	<	/	>	*	
'10x	∂	A	B	C	D	E	F	G	"4x
'11x	H	I	J	K	L	M	N	O	
'12x	P	Q	R	S	T	U	V	W	"5x
'13x	X	Y	Z	\flat	\natural	\sharp	\smile	\frown	
'14x	ℓ	a	b	c	d	e	f	g	"6x
'15x	h	i	j	k	l	m	n	o	
'16x	p	q	r	s	t	u	v	w	"7x
'17x	x	y	z	ι	j	\wp	\rightarrow	\leftarrow	
	"8	"9	"A	"B	"C	"D	"E	"F	

Figure 3. cmsy10—family 2

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	—	·	×	*	÷	◇	±	∓	"0x
'01x	⊕	⊖	⊗	⊙	⊚	○	◦	●	
'02x	∝	≡	⊆	⊇	≤	≥	≲	≳	"1x
'03x	~	≈	⊂	⊃	≪	≫	↖	↗	
'04x	←	→	↑	↓	↔	↗	↘	≈	"2x
'05x	⇐	⇒	↑	↓	⇔	↖	↘	∞	
'06x	∫	∞	∈	∃	△	▽	/	∓	"3x
'07x	∀	∃	¬	∅	ℜ	ℑ	⊤	⊥	
'10x	ℵ	ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	"4x
'11x	ℋ	ℐ	ℐ	ℐ	ℐ	ℐ	ℐ	ℐ	
'12x	ℙ	ℚ	ℚ	ℚ	ℚ	ℚ	ℚ	ℚ	"5x
'13x	ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	ℳ	
'14x	⊢	⊣	⊤	⊥	⊦	⊧	{	}	"6x
'15x	⟨	⟩			↕	↕	\	∩	
'16x	√	∏	∇	∫	⊔	⊓	⊆	⊇	"7x
'17x	§	†	‡	♣	♣	◇	♥	♠	
	"8	"9	"A	"B	"C	"D	"E	"F	

Figure 4. cmex10—family 3

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	()	[]	[]	[]	"0x
'01x	{	}	<	>			/	\	
'02x	()	()	[]	[]	"1x
'03x	[]	{	}	<	>	/	\	
'04x	()	[]	[]	[]	"2x
'05x	{	}	<	>	/	\	/	\	
'06x	()	[]	[]			"3x
'07x	()	()	{	}	·		
'10x	\)			<	>	⊥	⊥	"4x
'11x	ℱ	ℱ	⊙	⊙	⊕	⊕	⊗	⊗	
'12x	∑	∏	∫	∪	∩	⊕	∧	∨	"5x
'13x	∑	∏	∫	∪	∩	⊕	∧	∨	
'14x	∏	∏	∧	∧	∧	∧	∧	∧	"6x
'15x	[]	[]	[]	{	}	
'16x	√	√	√	√	√		┌		"7x
'17x	↑	↓	↖	↗	↘	↙	↕	↕	
	"8	"9	"A	"B	"C	"D	"E	"F	

User-level definitions

<code>\@M</code> 4 ₂₄	<code>\Biggr</code> 33 ₈₀₆	<code>\copyright</code> 7 ₁₁₇
<code>\@MM</code> 4 ₂₅	<code>\biggr</code> 33 ₈₀₃	<code>\cos</code> 34 ₈₆₂
<code>\@cclv</code> 4 ₂₁	<code>\Bigl</code> 33 ₇₉₈	<code>\cosh</code> 34 ₈₆₄
<code>\@cclvi</code> 4 ₂₂ , 14 ₂₆₅ , 14 ₂₆₉ , 14 ₂₇₃	<code>\bigl</code> 33 ₇₉₅	<code>\cot</code> 34 ₈₆₈
<code>\@foot</code> 37 ₉₇₃	<code>\Bigm</code> 33 ₇₉₉	<code>\coth</code> 34 ₈₆₉
<code>\@if</code> 40 ₁₀₂₉ , 40 ₁₀₃₁ , 40 ₁₀₃₃	<code>\bigm</code> 33 ₇₉₆	<code>\count@</code> 14 ₂₅₆
<code>\@ins</code> 38 ₉₉₀	<code>\bigodot</code> 28 ₆₃₃	<code>\cr</code> 23 ₄₉₄ , 35 ₈₉₉
<code>\@lign</code> 35 ₉₂₂	<code>\bigoplus</code> 28 ₆₃₂	<code>\csc</code> 34 ₈₇₁
<code>\@m</code> 4 ₂₃	<code>\bigotimes</code> 28 ₆₃₁	<code>\cup</code> 29 ₆₄₄
<code>\@ne</code> 4 ₁₇	<code>\Bigr</code> 33 ₈₀₀	<code>\d</code> 7 ₁₁₂
<code>\@another</code> 23 ₄₈₇	<code>\bigr</code> 33 ₇₉₇	<code>\dag</code> 7 ₁₀₃
<code>\@penup</code> 35 ₉₁₁	<code>\bigskip</code> 21 ₄₂₂	<code>\dagger</code> 29 ₆₄₆
<code>\@sf</code> 37 ₉₆₁	<code>\bigskipamount</code> 20 ₃₈₇	<code>\dashv</code> 29 ₆₇₄
<code>\@vereq</code> 34 ₈₄₀	<code>\bigsqcup</code> 28 ₆₃₅	<code>\ddag</code> 7 ₁₀₄
<code>\AA</code> 7 ₉₈	<code>\bigtriangledown</code> 29 ₆₄₀	<code>\ddagger</code> 29 ₆₄₅
<code>\aa</code> 7 ₉₂	<code>\bigtriangleup</code> 29 ₆₃₉	<code>\ddot</code> 31 ₇₄₂
<code>\active</code> 4 ₁₂	<code>\biguplus</code> 28 ₆₂₅	<code>\ddots</code> 31 ₇₃₈
<code>\acute</code> 31 ₇₄₀	<code>\bigvee</code> 28 ₆₂₃	<code>\deg</code> 34 ₈₈₄
<code>\advancepageno</code> 37 ₉₅₄	<code>\bigwedge</code> 28 ₆₂₄	<code>\Delta</code> 27 ₅₈₁
<code>\AE</code> 7 ₈₈	<code>\bmod</code> 35 ₈₈₆	<code>\delta</code> 27 ₅₅₄
<code>\ae</code> 6 ₈₅	<code>\body</code> 40 ₁₀₂₃	<code>\det</code> 34 ₈₈₀
<code>\aleph</code> 28 ₅₉₁	<code>\bordermatrix</code> 35 ₈₉₈	<code>\diamond</code> 29 ₆₅₁
<code>\alloc@</code> 14 ₂₇₄	<code>\bot</code> 28 ₆₀₆	<code>\diamondsuit</code> 28 ₆₁₉
<code>\allocationnumber</code> 14 ₂₅₃ , 15 ₂₈₅	<code>\bowtie</code> 30 ₇₂₁	<code>\dim</code> 34 ₈₇₈
<code>\allowbreak</code> 22 ₄₄₀	<code>\brace</code> 32 ₇₉₂	<code>\dimen@</code> 14 ₂₅₇
<code>\alpha</code> 27 ₅₅₁	<code>\braceld</code> 25 ₅₃₆	<code>\dimen@i</code> 14 ₂₅₈
<code>\amalg</code> 29 ₆₅₀	<code>\bracelu</code> 25 ₅₃₇	<code>\dimen@ii</code> 14 ₂₅₉
<code>\angle</code> 28 ₆₀₇	<code>\bracerd</code> 25 ₅₃₆	<code>\displ@y</code> 35 ₉₁₈
<code>\approx</code> 30 ₆₈₈	<code>\braceru</code> 25 ₅₃₇	<code>\displaylines</code> 35 ₉₂₃
<code>\arccos</code> 34 ₈₆₃	<code>\bracevert</code> 32 ₇₇₂	<code>\div</code> 29 ₆₅₄
<code>\arcsin</code> 34 ₈₆₀	<code>\brack</code> 32 ₇₉₁	<code>\dospecials</code> 4 ₁₅
<code>\arctan</code> 34 ₈₆₆	<code>\break</code> 22 ₄₃₈	<code>\dosupereject</code> 39 ₁₀₀₉
<code>\arg</code> 34 ₈₇₆	<code>\breve</code> 31 ₇₄₅	<code>\dot</code> 31 ₇₄₉
<code>\Arrowvert</code> 32 ₇₇₁	<code>\buildrel</code> 34 ₈₄₈	<code>\doteq</code> 34 ₈₄₉
<code>\arrowvert</code> 32 ₇₇₀	<code>\bullet</code> 29 ₆₅₂	<code>\dotfill</code> 25 ₅₂₉
<code>\ast</code> 29 ₆₆₆	<code>\bye</code> 26 ₅₄₆	<code>\dots</code> 7 ₁₁₉
<code>\asympt</code> 30 ₇₀₈	<code>\c</code> 7 ₁₁₅	<code>\Downarrow</code> 32 ₇₇₉
<code>\b</code> 7 ₁₁₃	<code>\c@ncel</code> 34 ₈₄₃	<code>\downarrow</code> 32 ₇₇₆
<code>\backslash</code> 32 ₇₈₁	<code>\cal</code> 12 ₂₃₀	<code>\downbracefill</code> 25 ₅₃₈
<code>\bar</code> 31 ₇₄₄	<code>\cap</code> 29 ₆₄₃	<code>\eject</code> 22 ₄₄₄
<code>\beginsection</code> 24 ₅₁₄	<code>\cases</code> 35 ₈₈₉	<code>\ell</code> 28 ₅₉₅
<code>\beta</code> 27 ₅₅₂	<code>\cdot</code> 29 ₆₆₅	<code>\empty</code> 21 ₄₀₁
<code>\bf</code> 13 ₂₃₆	<code>\cdotp</code> 31 ₇₃₂	<code>\emptyset</code> 28 ₆₀₂
<code>\Big</code> 33 ₈₀₈	<code>\cdots</code> 31 ₇₃₅	<code>\endinsert</code> 38 ₉₉₁
<code>\big</code> 33 ₈₀₇	<code>\centering</code> 15 ₂₉₁	<code>\enskip</code> 21 ₄₁₆
<code>\bigbreak</code> 22 ₄₅₂	<code>\centerline</code> 22 ₄₅₇	<code>\enspace</code> 21 ₄₁₄
<code>\bigcap</code> 28 ₆₂₆	<code>\ch@ck</code> 15 ₂₈₇	<code>\epsilon</code> 27 ₅₅₅
<code>\bigcirc</code> 29 ₆₆₃	<code>\check</code> 31 ₇₄₆	<code>\equalign</code> 35 ₉₁₄
<code>\bigcup</code> 28 ₆₂₇	<code>\chi</code> 27 ₅₇₁	<code>\equalignno</code> 36 ₉₂₆
<code>\Bigg</code> 33 ₈₁₀	<code>\choose</code> 32 ₇₉₀	<code>\equiv</code> 30 ₇₀₇
<code>\bigg</code> 33 ₈₀₉	<code>\circ</code> 29 ₆₆₂	<code>\eta</code> 27 ₅₅₇
<code>\Biggl</code> 33 ₈₀₄	<code>\cleartabs</code> 23 ₄₇₇	<code>\exists</code> 28 ₆₁₃
<code>\biggl</code> 33 ₈₀₁	<code>\clubsuit</code> 28 ₆₁₈	<code>\exp</code> 34 ₈₈₁
<code>\Biggm</code> 33 ₈₀₅	<code>\colon</code> 31 ₇₃₃	<code>\expandafter</code> 14 ₂₆₈
<code>\biggm</code> 33 ₈₀₂	<code>\cong</code> 34 ₈₃₉	<code>\f@t</code> 37 ₉₇₁
	<code>\coprod</code> 28 ₆₂₂	<code>\f@t</code> 37 ₉₇₂

<code>\filbreak</code> 22 ₄₄₂	<code>\iota</code> 27 ₅₅₉	<code>\magstep</code> 6 ₇₆
<code>\finph@nt</code> 33 ₈₂₈	<code>\it</code> 13 ₂₃₂	<code>\magstephalf</code> 6 ₇₅
<code>\finsm@sh</code> 34 ₈₃₇	<code>\item</code> 24 ₅₁₀	<code>\makefootline</code> 37 ₉₅₀
<code>\flat</code> 28 ₆₁₅	<code>\itemitem</code> 24 ₅₁₁	<code>\makeheadline</code> 36 ₉₄₈
<code>\fmtname</code> 40 ₁₀₃₉	<code>\iterate</code> 40 ₁₀₂₄	<code>\makeph@nt</code> 33 ₈₂₆
<code>\fmtversion</code> 40 ₁₀₃₉	<code>\j</code> 7 ₉₁	<code>\makesm@sh</code> 33 ₈₃₅
<code>\fo@t</code> 37 ₉₆₉	<code>\jmath</code> 28 ₅₉₄	<code>\mapsto</code> 30 ₇₀₃
<code>\folio</code> 37 ₉₅₂	<code>\joinrel</code> 30 ₇₁₆	<code>\mapstochar</code> 30 ₇₀₃
<code>\footline</code> 36 ₉₄₅	<code>\jot</code> 20 ₃₉₁	<code>\mathhexbox</code> 7 ₁₀₁
<code>\footnote</code> 37 ₉₆₀	<code>\kappa</code> 27 ₅₆₀	<code>\mathpalette</code> 33 ₈₁₂
<code>\footnoterule</code> 38 ₉₈₀	<code>\ker</code> 34 ₈₇₇	<code>\mathph@nt</code> 33 ₈₂₇
<code>\footstrut</code> 37 ₉₇₄	<code>\L</code> 7 ₉₄	<code>\mathsm@sh</code> 33 ₈₃₆
<code>\forall</code> 28 ₆₁₂	<code>\l</code> 7 ₉₃	<code>\mathstrut</code> 33 ₈₃₁
<code>\frenchspacing</code> 25 ₅₂₄	<code>\Lambda</code> 27 ₅₈₃	<code>\matrix</code> 35 ₈₉₁
<code>\frown</code> 30 ₇₁₀	<code>\lambda</code> 27 ₅₆₁	<code>\max</code> 34 ₈₇₂
<code>\Gamma</code> 27 ₅₈₀	<code>\langle</code> 32 ₇₈₃	<code>\maxdimen</code> 15 ₂₈₉
<code>\gamma</code> 27 ₅₅₃	<code>\lbrace</code> 32 ₇₈₅	<code>\medbreak</code> 22 ₄₅₀
<code>\gcd</code> 34 ₈₈₃	<code>\lbrack</code> 21 ₄₁₁	<code>\medskip</code> 21 ₄₂₁
<code>\geq</code> 30 ₆₈₅	<code>\lceil</code> 32 ₇₈₇	<code>\medskipamount</code> 20 ₃₈₆
<code>\gg</code> 30 ₆₉₇	<code>\ldotp</code> 31 ₇₃₁	<code>\mid</code> 29 ₆₇₃
<code>\goodbreak</code> 22 ₄₄₃	<code>\ldots</code> 31 ₇₃₄	<code>\midinsert</code> 38 ₉₈₅
<code>\grave</code> 31 ₇₄₁	<code>\leavevmode</code> 7 ₉₆	<code>\min</code> 34 ₈₇₃
<code>\H</code> 7 ₁₂₉	<code>\Leftarrow</code> 30 ₆₈₁	<code>\mit</code> 12 ₂₂₈
<code>\hang</code> 24 ₅₀₈	<code>\leftarrow</code> 30 ₇₀₁	<code>\models</code> 30 ₇₂₂
<code>\hat</code> 31 ₇₄₇	<code>\leftarrowfill</code> 25 ₅₃₃	<code>\mp</code> 29 ₆₆₀
<code>\hbar</code> 28 ₅₉₂	<code>\leftharpoondown</code> 30 ₇₁₂	<code>\mscount</code> 23 ₄₇₀
<code>\headline</code> 36 ₉₄₄	<code>\leftharpoonup</code> 30 ₇₁₁	<code>\mu</code> 27 ₅₆₂
<code>\heartsuit</code> 28 ₆₂₀	<code>\leftline</code> 22 ₄₅₅	<code>\multispan</code> 23 ₄₇₁
<code>\hgl@</code> 22 ₄₃₃	<code>\Leftrightarrow</code> 29 ₆₈₀	<code>\n@space</code> 33 ₈₁₁
<code>\hglue</code> 22 ₄₃₂	<code>\leftrightarrow</code> 30 ₇₀₀	<code>\nabla</code> 28 ₆₀₃
<code>\hideskip</code> 15 ₂₉₀	<code>\leq</code> 30 ₆₈₄	<code>\narrower</code> 24 ₅₁₂
<code>\hidewidth</code> 23 ₄₆₈	<code>\leqalignno</code> 36 ₉₃₁	<code>\natural</code> 28 ₆₁₆
<code>\hom</code> 34 ₈₇₉	<code>\lfloor</code> 32 ₇₈₉	<code>\nearrow</code> 29 ₆₇₆
<code>\hookleftarrow</code> 30 ₇₂₀	<code>\lg</code> 34 ₈₅₄	<code>\neg</code> 28 ₆₁₄
<code>\hookrightarrow</code> 30 ₇₁₉	<code>\lgroup</code> 32 ₇₆₈	<code>\negthinspace</code> 21 ₄₁₃
<code>\hphantom</code> 33 ₈₂₂	<code>\lhook</code> 30 ₇₁₉	<code>\neq</code> 30 ₆₈₃
<code>\hrulefill</code> 25 ₅₂₈	<code>\lim</code> 34 ₈₅₆	<code>\newbox</code> 14 ₂₆₆
<code>\i</code> 7 ₉₁	<code>\liminf</code> 34 ₈₅₈	<code>\newcount</code> 14 ₂₆₂
<code>\ialign</code> 23 ₄₆₉	<code>\limsup</code> 34 ₈₅₇	<code>\newdimen</code> 14 ₂₆₃
<code>\if@cr</code> 23 ₄₇₄	<code>\line</code> 22 ₄₅₄	<code>\newfam</code> 14 ₂₇₂
<code>\if@mid</code> 38 ₉₈₃	<code>\ll</code> 30 ₆₉₈	<code>\newhelp</code> 14 ₂₆₈
<code>\ifdt@p</code> 35 ₉₁₇	<code>\llap</code> 22 ₄₆₀	<code>\newif</code> 40 ₁₀₂₇
<code>\iff</code> 30 ₇₃₀	<code>\lmoustache</code> 32 ₇₆₆	<code>\newinsert</code> 14 ₂₇₉
<code>\ifh@</code> 33 ₈₂₀	<code>\ln</code> 34 ₈₅₅	<code>\newlanguage</code> 14 ₂₇₃
<code>\ifp@ge</code> 38 ₉₈₃	<code>\log</code> 34 ₈₅₃	<code>\newmuskup</code> 14 ₂₆₅
<code>\ifr@ggedbottom</code> 37 ₉₅₆	<code>\Longleftarrow</code> 30 ₇₂₆	<code>\newread</code> 14 ₂₇₀
<code>\ifus@</code> 23 ₄₇₄	<code>\longleftarrow</code> 30 ₇₂₅	<code>\newskip</code> 14 ₂₆₄
<code>\ifv@</code> 33 ₈₂₀	<code>\Longleftrightarrow</code> 30 ₇₂₉	<code>\newtoks</code> 14 ₂₆₉
<code>\Im</code> 28 ₅₉₈	<code>\longleftrightarrow</code> 30 ₇₂₈	<code>\newwrite</code> 14 ₂₇₁
<code>\imath</code> 28 ₅₉₃	<code>\longmapsto</code> 30 ₇₂₇	<code>\next</code> 7 ₁₃₂ , 33 ₈₂₄ , 33 ₈₃₃
<code>\in</code> 30 ₆₉₅	<code>\Longrightarrow</code> 30 ₇₂₃	<code>\ni</code> 30 ₆₉₆
<code>\inf</code> 34 ₈₇₅	<code>\longrightarrow</code> 30 ₇₂₄	<code>\nobreak</code> 22 ₄₃₉
<code>\infty</code> 28 ₆₀₀	<code>\loop</code> 40 ₁₀₂₃	<code>\nointerlineskip</code> 21 ₄₂₄
<code>\insc@unt</code> 14 ₂₅₂ , 14 ₂₆₂ , 14 ₂₆₃ , 14 ₂₆₄ , 14 ₂₆₆	<code>\lq</code> 21 ₄₁₀	<code>\nonfrenchspacing</code> 25 ₅₂₆
<code>\int</code> 28 ₆₂₈	<code>\m@g</code> 6 ₇₈	<code>\nopagenumbers</code> 37 ₉₅₃
<code>\interdisplaylinepenalty</code> 20 ₃₉₂	<code>\m@ketabbox</code> 23 ₄₉₁	<code>\normalbaselines</code> 20 ₃₉₄
<code>\interfootnotelinepenalty</code> 20 ₃₉₃	<code>\m@ne</code> 14 ₂₅₄	<code>\normalbaselineskip</code> 20 ₃₈₈
<code>\intop</code> 28 ₆₂₈	<code>\m@th</code> 22 ₄₆₂	<code>\normalbottom</code> 37 ₉₅₈
	<code>\magnification</code> 6 ₇₇	<code>\normallineskip</code> 20 ₃₈₉

<code>\normallineskiplimit</code> 20 ₃₉₀	<code>\prime</code> 28 ₆₀₁	<code>\sl</code> 13 ₂₃₄
<code>\not</code> 30 ₆₉₉	<code>\proclaim</code> 24 ₅₁₇	<code>\slash</code> 22 ₄₃₆
<code>\notin</code> 34 ₈₄₂	<code>\prod</code> 28 ₆₂₉	<code>\smallbreak</code> 22 ₄₄₈
<code>\nu</code> 27 ₅₆₃	<code>\propto</code> 29 ₆₆₉	<code>\smallint</code> 28 ₆₃₆
<code>\null</code> 21 ₄₀₂	<code>\Psi</code> 27 ₅₈₉	<code>\smallskip</code> 21 ₄₂₀
<code>\narrow</code> 29 ₆₇₈	<code>\psi</code> 27 ₅₇₂	<code>\smallskipamount</code> 20 ₃₈₅
<code>\nxt</code> 23 ₄₈₀	<code>\qqquad</code> 21 ₄₁₈	<code>\smash</code> 33 ₈₃₂
<code>\O</code> 7 ₉₀	<code>\quad</code> 21 ₄₁₇	<code>\smile</code> 30 ₇₀₉
<code>\o</code> 6 ₈₇	<code>\r@t</code> 33 ₈₁₇	<code>\sp@n</code> 23 ₄₇₃
<code>\o@lign</code> 7 ₁₀₈	<code>\raggedbottom</code> 37 ₉₅₇	<code>\space</code> 20 ₄₀₀
<code>\oalign</code> 7 ₁₀₇	<code>\raggedright</code> 25 ₅₂₀	<code>\spadesuit</code> 28 ₆₂₁
<code>\obeyspaces</code> 21 ₄₀₈	<code>\rangle</code> 32 ₇₈₂	<code>\sqcap</code> 29 ₆₄₇
<code>\odot</code> 29 ₆₅₅	<code>\rbrace</code> 32 ₇₈₄	<code>\sqcup</code> 29 ₆₄₈
<code>\OE</code> 7 ₈₉	<code>\rbrack</code> 21 ₄₁₁	<code>\sqrt</code> 32 ₇₉₄
<code>\oe</code> 6 ₈₆	<code>\rceil</code> 32 ₇₈₆	<code>\sqsubsetq</code> 29 ₆₇₀
<code>\offinterlineskip</code> 21 ₄₂₅	<code>\Re</code> 28 ₅₉₇	<code>\sqsupsetq</code> 29 ₆₇₁
<code>\oint</code> 28 ₆₃₄	<code>\relax</code> 14 ₂₆₇	<code>\ss</code> 6 ₈₄
<code>\ointop</code> 28 ₆₃₄	<code>\Relbar</code> 30 ₇₁₈	<code>\star</code> 29 ₆₆₈
<code>\oldstyle</code> 12 ₂₂₈	<code>\relbar</code> 30 ₇₁₇	<code>\strut</code> 23 ₄₆₇
<code>\Omega</code> 27 ₅₉₀	<code>\removelastskip</code> 22 ₄₄₇	<code>\strutbox</code> 23 ₄₆₅
<code>\omega</code> 27 ₅₇₃	<code>\rfloor</code> 32 ₇₈₈	<code>\subset</code> 30 ₆₉₂
<code>\ominus</code> 29 ₆₅₈	<code>\rgroup</code> 32 ₇₆₉	<code>\subsetq</code> 30 ₆₉₄
<code>\oalign</code> 7 ₁₀₉	<code>\rho</code> 27 ₅₆₆	<code>\succ</code> 30 ₆₈₆
<code>\openup</code> 35 ₉₁₀	<code>\rhook</code> 30 ₇₂₀	<code>\succeq</code> 30 ₆₈₉
<code>\oplus</code> 29 ₆₅₉	<code>\Rightarrow</code> 30 ₆₈₂	<code>\sum</code> 28 ₆₃₀
<code>\oslash</code> 29 ₆₅₆	<code>\rightarrow</code> 30 ₇₀₂	<code>\sup</code> 34 ₈₇₄
<code>\otimes</code> 29 ₆₅₇	<code>\rightarrowfill</code> 25 ₅₃₀	<code>\supereject</code> 22 ₄₄₅
<code>\overbrace</code> 31 ₇₅₈	<code>\rightharpoondown</code> 30 ₇₁₄	<code>\supset</code> 30 ₆₉₁
<code>\overleftarrow</code> 31 ₇₅₅	<code>\rightharpoonup</code> 30 ₇₁₃	<code>\supsetq</code> 30 ₆₉₃
<code>\overrightarrow</code> 31 ₇₅₂	<code>\rightleftharpoons</code> 34 ₈₄₄	<code>\surd</code> 28 ₆₀₄
<code>\P</code> 7 ₁₀₆	<code>\rightline</code> 22 ₄₅₆	<code>\swarrow</code> 29 ₆₇₉
<code>\p@</code> 15 ₂₉₂	<code>\rlap</code> 22 ₄₅₉	<code>\t</code> 7 ₁₃₂
<code>\p@renwd</code> 35 ₈₉₆	<code>\rlh@</code> 34 ₈₄₅	<code>\t@bb@x</code> 24 ₅₀₁
<code>\pagebody</code> 39 ₁₀₀₇	<code>\rm</code> 12 ₂₂₆	<code>\t@b@box</code> 24 ₅₀₀
<code>\pagecontents</code> 39 ₁₀₁₂	<code>\rmoustache</code> 32 ₇₆₇	<code>\tabalign</code> 23 ₄₈₃
<code>\pageinsert</code> 38 ₉₈₆	<code>\root</code> 33 ₈₁₅	<code>\tabs</code> 23 ₄₇₅
<code>\pageno</code> 36 ₉₄₃	<code>\rootbox</code> 33 ₈₁₄	<code>\tabsdone</code> 23 ₄₇₅
<code>\par</code> 4 ₁₃	<code>\rq</code> 21 ₄₁₀	<code>\tabset</code> 23 ₄₇₅
<code>\parallel</code> 29 ₆₇₂	<code>\S</code> 7 ₁₀₅	<code>\tan</code> 34 ₈₆₅
<code>\partial</code> 28 ₅₉₉	<code>\s@tcols</code> 23 ₄₈₅	<code>\tanh</code> 34 ₈₆₇
<code>\penalty</code> 22 ₄₃₅	<code>\s@t@b</code> 23 ₄₈₂	<code>\tau</code> 27 ₅₆₈
<code>\perp</code> 30 ₇₀₆	<code>\searrow</code> 29 ₆₇₇	<code>\TeX</code> 7 ₁₂₀
<code>\ph@nt</code> 33 ₈₂₄	<code>\sec</code> 34 ₈₇₀	<code>\textindent</code> 24 ₅₀₉
<code>\phantom</code> 33 ₈₂₃	<code>\setminus</code> 29 ₆₆₄	<code>\Theta</code> 27 ₅₈₂
<code>\Phi</code> 27 ₅₈₈	<code>\sett@b</code> 23 ₄₈₀	<code>\theta</code> 27 ₅₅₈
<code>\phi</code> 27 ₅₇₀	<code>\settabs</code> 23 ₄₇₈	<code>\thinspace</code> 21 ₄₁₂
<code>\Pi</code> 27 ₅₈₅	<code>\sh@ft</code> 7 ₁₁₀	<code>\thr@@</code> 4 ₁₉
<code>\pi</code> 27 ₅₆₅	<code>\sharp</code> 28 ₆₁₇	<code>\tilde</code> 31 ₇₄₃
<code>\plainoutput</code> 38 ₁₀₀₃	<code>\showhyphens</code> 26 ₅₄₄	<code>\times</code> 29 ₆₆₇
<code>\pm</code> 29 ₆₆₁	<code>\Sigma</code> 27 ₅₈₆	<code>\toks@</code> 14 ₂₆₁
<code>\pmatrix</code> 35 ₈₉₅	<code>\sigma</code> 27 ₅₆₇	<code>\top</code> 28 ₆₀₅
<code>\pmod</code> 35 ₈₈₈	<code>\sim</code> 30 ₇₀₄	<code>\topglue</code> 21 ₄₂₈
<code>\Pr</code> 34 ₈₈₂	<code>\simeq</code> 30 ₇₀₅	<code>\topinsert</code> 38 ₉₈₄
<code>\pr@@@s</code> 36 ₉₄₀	<code>\sin</code> 34 ₈₅₉	<code>\tracingall</code> 40 ₁₀₃₅
<code>\pr@@@t</code> 36 ₉₄₀	<code>\sinh</code> 34 ₈₆₁	<code>\triangle</code> 28 ₆₁₁
<code>\pr@m@s</code> 36 ₉₃₈	<code>\sixt@@n</code> 4 ₂₀ , 14 ₂₇₀ , 14 ₂₇₁ ,	<code>\triangleleft</code> 29 ₆₃₇
<code>\prec</code> 30 ₆₈₇	14 ₂₇₂	<code>\triangleright</code> 29 ₆₃₈
<code>\preceq</code> 30 ₆₉₀	<code>\skew</code> 31 ₇₆₄	<code>\tt</code> 13 ₂₃₉
<code>\prim@s</code> 36 ₉₃₇	<code>\skip@</code> 14 ₂₆₀	<code>\ttragedright</code> 25 ₅₂₂

<code>\tw@</code>	4 ₁₈	<code>\varphi</code>	27 ₅₇₉	<code>\voidb@x</code>	15 ₂₉₅
<code>\u</code>	7 ₁₂₅	<code>\varpi</code>	27 ₅₇₆	<code>\vphantom</code>	33 ₈₂₁
<code>\underbar</code>	22 ₄₆₃	<code>\varrho</code>	27 ₅₇₇	<code>\wedge</code>	29 ₆₄₁
<code>\underbrace</code>	31 ₇₆₁	<code>\varsigma</code>	27 ₅₇₈	<code>\widehat</code>	31 ₇₅₁
<code>\Uparrow</code>	32 ₇₇₈	<code>\vartheta</code>	27 ₅₇₅	<code>\widetilde</code>	31 ₇₅₀
<code>\uparrow</code>	32 ₇₇₅	<code>\vdash</code>	29 ₆₇₅	<code>\wlog</code>	14 ₂₅₅
<code>\upbracefill</code>	26 ₅₄₁	<code>\vdots</code>	31 ₇₃₆	<code>\wp</code>	28 ₅₉₆
<code>\Udownarrow</code>	32 ₇₈₀	<code>\vec</code>	31 ₇₄₈	<code>\wr</code>	29 ₆₅₃
<code>\updownarrow</code>	32 ₇₇₇	<code>\vee</code>	29 ₆₄₂	<code>\Xi</code>	27 ₅₈₄
<code>\uplus</code>	29 ₆₄₉	<code>\Vert</code>	32 ₇₇₃	<code>\xi</code>	27 ₅₆₄
<code>\Upsilon</code>	27 ₅₈₇	<code>\vert</code>	32 ₇₇₄	<code>\z@</code>	15 ₂₉₃
<code>\upsilon</code>	27 ₅₆₉	<code>\tfootnote</code>	37 ₉₆₃	<code>\z@skip</code>	15 ₂₉₄
<code>\v</code>	7 ₁₂₄	<code>\vgl@</code>	22 ₄₃₀	<code>\zeta</code>	27 ₅₅₆
<code>\varepsilon</code>	27 ₅₇₄	<code>\vglue</code>	21 ₄₂₉		

Definitions

`\@M` 22₄₃₅, 22₄₃₈, 22₄₃₉
`\@MM` 22₄₄₅, 37₉₆₆, 38₁₀₀₅
`\@cclv` 39₁₀₁₃
`\@crfalse` 24₄₉₇
`\@crtrue` 23₄₉₄
`\@foot` 37₉₇₁, 37₉₇₂
`\@if` 40₁₀₃₂
`\@ins` 38₉₈₄, 38₉₈₅, 38₉₈₆
`\@lign` 35₉₂₄, 36₉₂₇, 36₉₂₈,
36₉₂₉, 36₉₃₂, 36₉₃₃, 36₉₃₄
`\@m` 6₇₆, 25₅₂₄, 25₅₂₅
`\@midfalse` 38₉₈₄, 38₉₈₆, 38₉₉₄
`\@midtrue` 38₉₈₅
`\@ne` 12₂₂₈, 14₂₇₄, 23₄₇₂, 24₅₀₃,
24₅₀₄, 24₅₀₅, 24₅₀₆, 35₉₀₄,
35₉₀₅, 35₉₀₆, 35₉₀₇, 35₉₀₈,
35₉₀₉, 37₉₅₅, 40₁₀₃₅, 40₁₀₃₆,
40₁₀₃₇
`\@nother` 23₄₈₆
`\@penup` 35₉₁₀
`\@sf` 37₉₆₀, 37₉₆₂
`\@vereq` 34₈₃₉
`\abovedisplaysshortskip` 19₃₇₄
`\abovedisplayskip` 19₃₇₃
`\accent` 7₁₁₅, 7₁₂₂, 7₁₂₃, 7₁₂₄,
7₁₂₅, 7₁₂₆, 7₁₂₇, 7₁₂₈,
7₁₂₉, 7₁₃₀, 7₁₃₁, 7₁₃₂,
7₉₂
`\active` 4₁₂, 4₁₃, 21₄₀₅, 21₄₀₆,
21₄₀₈, 36₉₃₆, 36₉₄₁, 36₉₄₂
`\adjdemerits` 18₃₁₆
`\advance` 7₉₈, 14₂₇₄, 14₂₇₉,
23₄₇₃, 23₄₈₉, 24₅₁₂, 24₅₁₃,
33₈₁₈, 35₉₁₁, 35₉₁₂, 35₉₁₃,
37₉₅₄, 37₉₅₅, 38₉₉₂, 38₉₉₃
`\advancepageno` 38₁₀₀₄
`\afterassignment` 6₇₇, 21₄₂₉,
22₄₃₂, 23₄₈₀, 35₉₁₀
`\aftergroup` 37₉₇₁
`\alloc@` 14₂₆₂, 14₂₆₃, 14₂₆₄,
14₂₆₅, 14₂₆₆, 14₂₆₉, 14₂₇₀,
14₂₇₁, 14₂₇₂, 14₂₇₃
`\allocationnumber` 14₂₇₆, 14₂₇₇,
14₂₇₈, 15₂₈₄, 15₂₈₆
`\allowbreak` 35₈₈₈
`\atopwithdelims` 32₇₉₀, 32₇₉₁,
32₇₉₂
`\baselineskip` 7₁₀₇, 20₃₉₅,
21₄₂₅, 31₇₃₆, 35₈₉₃, 35₈₉₄,
35₉₀₂, 35₉₀₈, 35₉₁₂, 37₉₅₀
`\begingroup` 23₄₉₁, 35₈₉₈, 38₉₉₀
`\belowdisplaysshortskip` 19₃₇₆
`\belowdisplayskip` 19₃₇₅
`\bf` 24₅₁₆, 24₅₁₈
`\bffam` 13₂₃₆, 13₂₃₇, 13₂₃₈
`\bgroup` 21₄₀₄, 24₄₉₇, 24₄₉₈,
24₅₀₀, 36₉₃₆, 37₉₆₃, 37₉₆₉,
37₉₇₁, 38₉₉₀
`\Big` 33₇₉₈, 33₇₉₉, 33₈₀₀
`\big` 33₇₉₅, 33₇₉₆, 33₇₉₇
`\bigbreak` 38₉₉₅
`\Bigg` 33₈₀₄, 33₈₀₅, 33₈₀₆
`\bigg` 33₈₀₁, 33₈₀₂, 33₈₀₃
`\bigskip` 22₄₅₃, 24₅₁₅, 38₁₀₀₁,
38₉₉₅
`\bigskipamount` 20₃₈₇, 21₄₂₂,
22₄₅₂, 37₉₇₆
`\binoppenalty` 18₃₀₃
`\body` 40₁₀₂₄
`\box` 14₂₆₆, 15₂₈₃, 22₄₆₄, 24₅₀₆,
24₅₀₇, 33₈₁₉, 33₈₃₀, 34₈₃₇,
35₉₀₇, 35₉₀₉, 38₁₀₀₁, 38₉₉₅
`\boxmaxdepth` 19₃₅₇, 39₁₀₀₇
`\braced` 25₅₃₈, 25₅₃₉, 26₅₄₁,
26₅₄₃
`\bracelu` 26₅₄₀, 26₅₄₂
`\bracerd` 26₅₄₀, 26₅₄₂
`\braceru` 25₅₃₉, 26₅₄₃
`\break` 22₄₄₄
`\brokenpenalty` 18₃₀₈
`\buildrel` 34₈₄₉
`\c@ncel` 34₈₄₂
`\catcode` 3₁, 3₂, 4₁₀, 4₁₂,
4₁₃, 4₃, 4₄, 4₅, 4₆, 4₇,
4₈, 4₉, 21₄₀₅, 21₄₀₆, 21₄₀₈,
36₉₃₆, 36₉₄₁, 36₉₄₂
`\cdotp` 31₇₃₅
`\centering` 15₂₉₁, 36₉₂₆, 36₉₂₈,
36₉₃₁, 36₉₃₃
`\ch@ck` 14₂₇₅, 14₂₈₀, 15₂₈₁,
15₂₈₂, 15₂₈₃
`\char` 7₁₁₄, 7₁₁₆, 7₉₃, 7₉₄,
7₉₉, 34₈₅₂
`\cleaders` 25₅₂₉, 25₅₃₁, 25₅₃₄
`\clubpenalty` 18₃₀₅
`\columns` 23₄₈₅
`\copy` 23₄₆₇, 23₄₉₂, 33₈₁₉
`\count` 13₂₄₁, 13₂₄₂, 13₂₄₃,
13₂₄₄, 13₂₄₅, 14₂₄₆, 14₂₄₇,
14₂₄₈, 14₂₄₉, 14₂₅₀, 14₂₅₁,
14₂₆₂, 14₂₇₄, 14₂₇₆, 14₂₈₀,
15₂₈₇, 38₉₇₇, 38₉₈₈
`\count@` 6₇₇, 6₇₈, 22₄₃₃, 22₄₃₄,
23₄₈₅, 23₄₈₆, 23₄₈₇, 23₄₈₉,
40₁₀₂₇, 40₁₀₃₂
`\cr` 20₃₉₈, 35₈₉₉, 35₉₀₃
`\crrc` 7₁₀₈, 7₁₁₂, 7₁₁₃, 7₁₁₆,
7₁₁₇, 23₄₉₄, 24₄₉₈, 28₆₀₇,
28₆₀₈, 28₆₁₀, 31₇₅₂, 31₇₅₃,
31₇₅₄, 31₇₅₅, 31₇₅₆, 31₇₅₇,
31₇₅₈, 31₇₅₉, 31₇₆₀, 31₇₆₁,
31₇₆₂, 31₇₆₃, 34₈₄₁, 34₈₄₃,
34₈₄₆, 35₈₉₀, 35₈₉₂, 35₈₉₃,
35₈₉₄, 35₈₉₉, 35₉₀₁, 35₉₀₂,
35₉₀₃, 35₉₁₆, 35₉₂₄, 36₉₂₅,
36₉₂₉, 36₉₃₀, 36₉₃₄, 36₉₃₅
`\csname` 14₂₆₈, 40₁₀₃₃
`\defaultthyphenchar` 18₃₃₈
`\defaultskewchar` 18₃₃₉
`\delcode` 12₂₁₆, 12₂₁₇, 12₂₁₈,
12₂₁₉, 12₂₂₀, 12₂₂₁, 12₂₂₂,
12₂₂₃, 12₂₂₄
`\delimiter` 32₇₆₆, 32₇₆₇, 32₇₆₈,
32₇₆₉, 32₇₇₀, 32₇₇₁, 32₇₇₂,
32₇₇₃, 32₇₇₄, 32₇₇₅, 32₇₇₆,
32₇₇₇, 32₇₇₈, 32₇₇₉, 32₇₈₀,
32₇₈₁, 32₇₈₂, 32₇₈₃, 32₇₈₄,
32₇₈₅, 32₇₈₆, 32₇₈₇, 32₇₈₈,
32₇₈₉
`\delimiterfactor` 19₃₄₂
`\delimitershortfall` 19₃₅₉
`\dimen` 6₇₉, 7₁₁₀, 7₁₁₁, 14₂₆₃,
15₂₈₁, 38₉₇₈, 38₉₈₉
`\dimen@` 7₉₈, 7₉₉, 22₄₃₀, 22₄₃₁,
23₄₈₅, 23₄₈₇, 23₄₈₉, 33₈₁₈,
33₈₁₉, 35₉₁₀, 35₉₁₁, 35₉₁₂,
35₉₁₃, 38₁₀₀₀, 38₉₉₂, 38₉₉₃,
38₉₉₄, 38₉₉₉, 39₁₀₁₃, 39₁₀₁₈
`\dimen@ii` 23₄₈₇, 23₄₈₈, 23₄₈₉
`\discretionary` 34₈₅₂
`\displ@y` 35₉₂₃, 36₉₂₆, 36₉₃₁
`\displaystyle` 31₇₅₄, 31₇₅₇,
31₇₆₀, 31₇₆₂, 33₈₁₂, 35₉₁₅,
35₉₂₄, 36₉₂₇, 36₉₂₈, 36₉₃₂,
36₉₃₃
`\displaywidowpenalty` 18₃₀₇
`\displaywidth` 35₉₂₄, 36₉₂₇,
36₉₃₂, 36₉₃₄
`\divide` 23₄₈₇, 31₇₆₄
`\do` 4₁₅, 4₁₆
`\dosupereject` 38₁₀₀₅
`\doublehyphdemerits` 18₃₁₄
`\downbracefill` 31₇₅₉
`\dp` 22₄₆₃, 33₈₁₈, 33₈₂₉, 34₈₃₇,
37₉₆₆, 38₉₉₂, 38₉₉₉, 39₁₀₁₃
`\dt@pfalse` 35₉₁₉
`\dt@ptrue` 35₉₁₈
`\egroup` 21₄₀₄, 23₄₉₄, 24₅₀₁,
24₅₀₂, 36₉₃₉, 36₉₄₀, 37₉₇₃,
38₉₉₁
`\else` 7₁₁₆, 7₁₁₉, 15₂₈₈, 22₄₄₇,
23₄₆₇, 23₄₈₁, 24₅₀₂, 24₅₀₅,
33₈₂₅, 33₈₃₃, 35₉₂₁, 36₉₃₈,
36₉₃₉, 37₉₅₂, 37₉₅₅, 37₉₇₀,
38₁₀₀₁, 38₁₀₀₅, 38₉₉₆, 39₁₀₁₂,
39₁₀₁₄, 40₁₀₂₄
`\empty` 37₉₆₀
`\end` 26₅₄₆

`\endcsname` 14₂₆₈, 40₁₀₃₃
`\endgraf` 20₃₉₈
`\endgroup` 23₄₉₅, 35₉₀₉, 38₁₀₀₁
`\endline` 20₃₉₈, 35₈₉₉
`\enspace` 24₅₀₉, 24₅₁₈
`\errmessage` 15₂₈₈
`\errorcontextlines` 19₃₄₉
`\errorstopmode` 40₁₀₃₈
`\escapechar` 40₁₀₂₇, 40₁₀₃₂
`\everycr` 23₄₆₉, 35₉₁₉, 35₉₂₂
`\exhyphenpenalty` 18₃₀₂, 22₄₃₆
`\expandafter` 40₁₀₂₈, 40₁₀₃₀,
40₁₀₃₃
`\f@t` 37₉₆₉
`\fot` 37₉₇₀
`\fam` 12₂₂₆, 12₂₂₈, 12₂₃₀, 13₂₃₂,
13₂₃₄, 13₂₃₆, 13₂₃₉, 14₂₇₂
`\fi` 6₇₆, 7₁₁₆, 7₁₁₉, 15₂₈₈,
22₄₄₇, 22₄₄₉, 22₄₅₁, 22₄₅₃,
23₄₆₇, 23₄₈₁, 23₄₉₅, 24₅₀₅,
24₅₀₆, 24₅₁₉, 33₈₂₅, 33₈₂₉,
33₈₃₀, 33₈₃₄, 35₉₂₀, 35₉₂₁,
36₉₃₉, 37₉₅₂, 37₉₅₅, 37₉₆₁,
37₉₇₀, 38₁₀₀₁, 38₁₀₀₅, 38₉₉₄,
39₁₀₁₀, 39₁₀₁₂, 39₁₀₁₇,
39₁₀₁₈, 40₁₀₂₄, 40₁₀₂₅
`\finalhyphendemerits` 18₃₁₅
`\finph@nt` 33₈₂₆, 33₈₂₇
`\finsm@sh` 33₈₃₅, 33₈₃₆
`\fivebf` 5₄₄, 13₂₃₈
`\fivei` 8₁₄₀, 10₁₅₆, 12₂₂₇
`\fiverm` 5₃₁, 12₂₂₅
`\fivesy` 8₁₄₇, 10₁₅₇, 12₂₂₉
`\floatingpenalty` 37₉₆₆, 38₉₉₈
`\fo@t` 37₉₆₈
`\folio` 36₉₄₅
`\font` 5₂₆, 5₂₇, 5₂₈, 5₂₉, 5₃₀,
5₃₁, 5₃₃, 5₃₄, 5₃₆, 5₃₇,
5₃₉, 5₄₀, 5₄₁, 5₄₂, 5₄₃,
5₄₄, 5₄₆, 5₄₇, 5₄₈, 5₅₀,
5₅₂, 5₅₃, 5₅₄, 5₅₆, 5₅₇,
5₅₈, 5₅₉, 5₆₁, 6₆₃, 6₆₅,
6₆₇, 6₆₉, 6₇₀, 6₇₁, 6₇₃,
7₁₁₀, 7₁₃₂, 8₁₃₅, 8₁₃₆,
8₁₃₇, 8₁₃₈, 8₁₃₉, 8₁₄₀,
8₁₄₂, 8₁₄₃, 8₁₄₄, 8₁₄₅,
8₁₄₆, 8₁₄₇, 8₁₄₉, 9₁₅₁,
9₁₅₂
`\fontdimen` 7₁₁₀
`\footins` 6₇₉, 37₉₅₉, 37₉₆₃,
37₉₇₆, 38₉₇₇, 38₉₇₈, 39₁₀₁₄,
39₁₀₁₅, 39₁₀₁₇
`\footline` 36₉₄₅, 37₉₅₀, 37₉₅₃
`\footnoterule` 39₁₀₁₆
`\footstrut` 37₉₆₈
`\futurelet` 23₄₇₈, 36₉₃₇, 37₉₆₈
`\gdef` 21₄₀₆, 36₉₃₆, 36₉₄₁,
40₁₀₃₄
`\ge` 30₆₈₅

`\geq` 30₆₈₅
`\gets` 30₇₀₁
`\global` 14₂₇₄, 14₂₇₇, 14₂₇₉,
15₂₈₅, 21₄₀₇, 21₄₀₉, 23₄₇₇,
23₄₉₂, 23₄₉₃, 24₅₀₂, 24₅₀₃,
24₅₀₄, 24₅₀₆, 35₈₉₉, 35₉₀₄,
35₉₀₅, 35₉₀₇, 35₉₁₈, 35₉₁₉,
36₉₄₂, 37₉₅₄, 37₉₅₅
`\h@false` 33₈₂₁
`\h@true` 33₈₂₂, 33₈₂₃
`\halign` 23₄₆₉, 35₉₂₄, 36₉₂₇,
36₉₃₂
`\hang` 24₅₁₀
`\hangindent` 24₅₀₈, 24₅₁₁
`\hbadness` 18₂₉₈, 26₅₄₅
`\hbox` 7₁₀₂, 7₁₁₄, 7₁₁₅, 7₁₁₇,
7₁₂₀, 7₉₄, 7₉₈, 7₉₉, 21₄₀₂,
22₄₅₄, 22₄₅₉, 22₄₆₀, 22₄₆₃,
23₄₆₆, 23₄₈₈, 23₄₉₆, 24₅₀₀,
24₅₀₂, 24₅₀₄, 24₅₀₅, 24₅₀₆,
25₅₂₉, 25₅₃₁, 25₅₃₄, 25₅₃₈,
26₅₄₁, 31₇₃₇, 31₇₃₈, 31₇₃₉,
33₈₀₇, 33₈₀₈, 33₈₀₉, 33₈₁₀,
33₈₁₅, 33₈₁₇, 33₈₂₆, 33₈₂₇,
33₈₃₅, 33₈₃₆, 34₈₄₅, 34₈₄₆,
35₈₉₇, 35₉₀₅, 35₉₀₆, 35₉₂₄
`\headline` 36₉₄₄, 36₉₄₉
`\hfil` 7₁₁₇, 31₇₅₄, 31₇₅₇, 31₇₆₀,
31₇₆₂, 34₈₄₁, 34₈₄₃, 35₈₉₀,
35₈₉₂, 35₉₀₀, 35₉₀₁, 35₉₀₂,
35₉₁₅, 35₉₂₄, 36₉₂₇, 36₉₂₈,
36₉₃₂, 36₉₃₃, 36₉₄₄, 37₉₅₃
`\hfill` 25₅₂₈, 25₅₂₉, 25₅₃₁,
25₅₃₄, 25₅₃₉, 26₅₄₀, 26₅₄₂,
26₅₄₃, 28₆₁₀
`\hfuzz` 19₃₅₀
`\hgl@` 22₄₃₂
`\hideskip` 15₂₉₀, 23₄₆₈
`\hidewidth` 7₁₁₂, 7₁₁₃, 7₁₁₄,
7₁₁₆
`\hrule` 7₉₇, 22₄₃₀, 25₅₂₈, 28₆₁₀,
38₉₈₁
`\hsize` 6₇₉, 19₃₅₃, 22₄₅₄, 23₄₈₅,
26₅₄₄
`\hskip` 21₄₁₆, 21₄₁₇, 21₄₁₈,
22₄₃₄, 23₄₆₈
`\hss` 7₉₄, 22₄₅₅, 22₄₅₆, 22₄₅₇,
22₄₅₉, 22₄₆₀, 24₅₀₂, 36₉₄₅
`\ht` 7₁₁₅, 7₉₈, 25₅₃₉, 26₅₄₀,
26₅₄₂, 26₅₄₃, 33₈₁₈, 33₈₂₉,
34₈₃₇, 35₉₀₈, 35₉₀₉, 37₉₆₅,
38₉₉₂
`\hyphenpenalty` 18₃₀₁
`\ialign` 7₁₀₈, 24₄₉₈, 28₆₀₇,
31₇₅₂, 31₇₅₅, 31₇₅₈, 31₇₆₁,
34₈₄₁, 35₈₉₀, 35₈₉₂, 35₉₀₀,
35₉₁₅
`\if@` 40₁₀₃₃, 40₁₀₃₄
`\if@cr` 24₅₀₁

`\if@mid` 38₉₉₂, 38₉₉₅
`\ifcase` 6₇₆
`\ifcat` 37₉₆₉
`\ifdim` 7₁₁₅, 22₄₄₇, 22₄₄₈,
22₄₅₀, 22₄₅₂, 24₅₁₉, 35₉₁₉,
38₉₉₄
`\ifdt@p` 35₉₁₉
`\iffalse` 40₁₀₃₁
`\ifh` 33₈₃₀
`\ifhmode` 37₉₆₁
`\ifmmode` 7₁₁₉, 23₄₆₇, 33₈₂₄,
33₈₃₃
`\ifnum` 15₂₈₇, 23₄₇₂, 23₄₈₆,
37₉₅₂, 37₉₅₄, 38₁₀₀₅, 39₁₀₀₉
`\ifp@ge` 38₉₉₉
`\ifrggedbottom` 39₁₀₁₈
`\iftrue` 40₁₀₂₉
`\ifus@` 23₄₉₅
`\ifv@` 33₈₂₉
`\ifvoid` 24₅₀₄, 39₁₀₁₂, 39₁₀₁₄
`\ifx` 23₄₈₀, 36₉₃₈
`\ignorespaces` 24₅₀₉
`\immediate` 14₂₅₅
`\in` 34₈₄₂
`\indent` 24₅₀₉, 24₅₁₁
`\input` 39₁₀₂₀
`\insecunt` 14₂₇₉, 14₂₈₀, 15₂₈₁,
15₂₈₂, 15₂₈₃, 15₂₈₄
`\insert` 15₂₈₆, 37₉₆₃, 38₉₉₆
`\insertpenalties` 39₁₀₀₉
`\interdisplaylinepenalty` 20₃₉₂,
35₉₂₁
`\interfootnotelinepenalty` 20₃₉₃,
37₉₆₄
`\interlinepenalty` 37₉₆₄
`\intop` 28₆₂₈
`\iterate` 40₁₀₂₃, 40₁₀₂₄
`\itfam` 13₂₃₂, 13₂₃₃
`\joinrel` 30₇₁₉, 30₇₂₀, 30₇₂₁,
30₇₂₂, 30₇₂₃, 30₇₂₄, 30₇₂₅,
30₇₂₆, 30₇₂₈, 30₇₂₉
`\jot` 20₃₉₁, 35₉₁₄, 35₉₁₈
`\kern` 7₁₁₁, 7₁₂₀, 7₉₇, 21₄₁₂,
21₄₁₃, 21₄₁₄, 31₇₃₇, 31₇₃₈,
31₇₅₃, 31₇₅₆, 31₇₅₈, 31₇₅₉,
31₇₆₂, 31₇₆₃, 34₈₄₈, 35₈₉₃,
35₈₉₄, 35₈₉₉, 35₉₀₀, 35₉₀₂,
35₉₀₆, 35₉₀₇, 35₉₀₈, 35₉₀₉,
36₉₃₄, 38₁₀₀₀, 38₉₈₀, 38₉₈₁,
39₁₀₁₀, 39₁₀₁₈
`\land` 29₆₄₁
`\language` 14₂₇₃
`\lastbox` 23₄₉₅, 24₅₀₃, 35₉₀₄,
35₉₀₅
`\lastskip` 22₄₄₇, 22₄₄₈, 22₄₅₀,
22₄₅₂, 24₅₁₉
`\lbrace` 32₇₈₅
`\ldotp` 31₇₃₄
`\ldots` 7₁₁₉

`\le` 30₆₈₄
`\leaders` 25₅₂₈, 25₅₃₉, 26₅₄₀,
26₅₄₂, 26₅₄₃, 28₆₁₀
`\leavevmode` 7₁₀₁, 7₁₀₇, 7₉₄,
7₉₇, 7₉₈, 22₄₃₃
`\left` 33₈₀₇, 33₈₀₈, 33₈₀₉,
33₈₁₀, 35₈₈₉, 35₈₉₅, 35₉₀₆
`\Leftarrow` 30₇₂₆, 30₇₂₉
`\leftarrow` 25₅₃₃, 30₇₀₁, 30₇₂₀,
30₇₂₅, 30₇₂₈
`\leftarrowfill` 71₇₅₆
`\leftarrowpoondown` 34₈₄₇
`\leftthyphenmin` 39₁₀₁₉
`\leftline` 24₅₁₆
`\leftskip` 24₅₁₂, 37₉₆₇
`\leq` 30₆₈₄
`\let` 6₇₄, 7₁₂₄, 7₁₂₅, 7₁₂₇,
14₂₆₇, 20₃₉₈, 21₄₀₄, 21₄₀₆,
21₄₀₇, 21₄₀₉, 23₄₇₉, 23₄₈₀,
23₄₈₁, 23₄₈₂, 28₆₁₄, 29₆₄₁,
29₆₄₂, 30₆₈₃, 30₆₈₄, 30₆₈₅,
30₆₉₆, 30₇₀₁, 30₇₀₂, 32₇₇₃,
32₇₈₄, 32₇₈₅, 33₈₂₅, 33₈₃₃,
34₈₅₁, 35₈₉₉, 36₉₃₈, 36₉₃₉,
36₉₄₂, 37₉₆₀, 37₉₆₉, 37₉₇₀,
37₉₇₁, 40₁₀₂₄, 40₁₀₂₅, 40₁₀₂₉,
40₁₀₃₁
`\lhook` 30₇₁₉
`\limits` 31₇₆₀, 31₇₆₃, 34₈₄₈
`\line` 22₄₅₅, 22₄₅₆, 22₄₅₇,
36₉₄₉, 37₉₅₀, 39₁₀₁₀
`\linepenalty` 18₃₀₀
`\lineskip` 7₁₀₇, 20₃₉₄, 21₄₂₆,
34₈₄₀, 35₉₁₁
`\lineskiplimit` 7₁₀₈, 7₁₀₉,
20₃₉₅, 21₄₂₆, 31₇₃₆, 34₈₄₀,
35₉₁₃, 35₉₂₀
`\llap` 24₅₀₉, 36₉₂₉
`\lnot` 28₆₁₄
`\Longleftarrow` 30₇₃₀
`\longrightarrow` 30₇₂₇
`\loop` 23₄₇₂, 23₄₈₆
`\lor` 29₆₄₂
`\lower` 7₁₂₀, 34₈₄₀
`\m@g` 6₇₇
`\m@ketabbox` 23₄₈₂, 23₄₈₃
`\m@ne` 14₂₅₄, 14₂₅₅, 14₂₇₉,
23₄₇₃, 23₄₈₉, 26₅₄₅, 37₉₅₄,
40₁₀₂₇
`\m@th` 7₁₀₂, 7₁₁₉, 22₄₆₄, 25₅₂₉,
25₅₃₀, 25₅₃₃, 25₅₃₈, 26₅₄₁,
28₆₀₇, 31₇₅₂, 31₇₅₅, 31₇₅₈,
31₇₆₁, 33₈₁₁, 33₈₁₅, 33₈₁₇,
33₈₂₇, 33₈₃₆, 34₈₄₁, 34₈₄₃,
34₈₄₅, 35₈₈₉, 35₈₉₁, 35₈₉₈,
35₉₁₄, 35₉₁₈
`\mag` 6₇₈
`\magstep` 6₆₉, 6₇₀, 6₇₁
`\makefootline` 38₁₀₀₃
`\makeheadline` 38₁₀₀₃
`\makeph@nt` 33₈₂₅
`\makesm@sh` 33₈₃₃
`\mapstochar` 30₇₀₃, 30₇₂₇
`\mathaccent` 31₇₄₀, 31₇₄₁, 31₇₄₂,
31₇₄₃, 31₇₄₄, 31₇₄₅, 31₇₄₆,
31₇₄₇, 31₇₄₈, 31₇₄₉, 31₇₅₀,
31₇₅₁
`\mathbin` 35₈₈₇
`\mathchar` 7₁₀₂, 28₅₉₂, 28₆₀₄
`\mathchoice` 33₈₁₂
`\mathclose` 33₇₉₇, 33₈₀₀, 33₈₀₃,
33₈₀₆
`\mathcode` 10₁₅₈, 10₁₅₉, 10₁₆₀,
10₁₆₁, 10₁₆₂, 10₁₆₃, 11₁₆₄,
11₁₆₅, 11₁₆₆, 11₁₆₇, 11₁₆₈,
11₁₆₉, 11₁₇₀, 11₁₇₁, 11₁₇₂,
11₁₇₃, 11₁₇₄, 11₁₇₅, 11₁₇₆,
11₁₇₇, 11₁₇₈, 11₁₇₉, 11₁₈₀,
11₁₈₁, 11₁₈₂, 11₁₈₃, 11₁₈₄,
11₁₈₅, 11₁₈₆, 11₁₈₇, 11₁₈₈,
11₁₈₉, 11₁₉₀, 11₁₉₁, 11₁₉₂,
11₁₉₃, 11₁₉₄, 11₁₉₅, 11₁₉₆,
11₁₉₇, 11₁₉₈, 11₁₉₉, 11₂₀₀,
11₂₀₁, 11₂₀₂, 11₂₀₃, 11₂₀₄,
11₂₀₅, 11₂₀₆, 11₂₀₇, 11₂₀₈,
11₂₀₉, 11₂₁₀, 11₂₁₁, 11₂₁₂,
11₂₁₃, 11₂₁₄
`\mathhexbox` 7₁₀₃, 7₁₀₄, 7₁₀₅,
7₁₀₆, 7₁₁₇
`\mathinner` 31₇₃₄, 31₇₃₅, 31₇₃₈
`\mathop` 31₇₅₈, 31₇₆₁, 34₈₄₈,
34₈₅₃, 34₈₅₄, 34₈₅₅, 34₈₅₆,
34₈₅₇, 34₈₅₈, 34₈₅₉, 34₈₆₀,
34₈₆₁, 34₈₆₂, 34₈₆₃, 34₈₆₄,
34₈₆₅, 34₈₆₆, 34₈₆₇, 34₈₆₈,
34₈₆₉, 34₈₇₀, 34₈₇₁, 34₈₇₂,
34₈₇₃, 34₈₇₄, 34₈₇₅, 34₈₇₆,
34₈₇₇, 34₈₇₈, 34₈₇₉, 34₈₈₀,
34₈₈₁, 34₈₈₂, 34₈₈₃, 34₈₈₄
`\mathopen` 33₇₉₅, 33₇₉₈, 33₈₀₁,
33₈₀₄
`\mathord` 25₅₃₂, 25₅₃₃
`\mathpalette` 33₈₁₆, 33₈₂₄,
33₈₃₃, 34₈₃₉, 34₈₄₂, 34₈₄₄
`\mathph@nt` 33₈₂₄
`\mathrel` 28₆₀₈, 30₇₁₆, 30₇₁₇,
30₇₁₈, 30₇₂₁, 30₇₂₂, 33₇₉₆,
33₇₉₉, 33₈₀₂, 33₈₀₅, 34₈₃₉,
34₈₄₂, 34₈₄₄, 34₈₄₈
`\mathsm@sh` 33₈₃₃
`\mathstrut` 35₈₉₃, 35₈₉₄
`\mathsurround` 22₄₆₂
`\matrix` 35₈₉₅
`\maxdepth` 19₃₅₅, 39₁₀₀₇
`\maxdimen` 7₁₀₉, 15₂₈₉, 19₃₅₆,
19₃₅₇, 21₄₂₆, 26₅₄₄, 34₈₄₀,
38₉₈₉, 38₉₉₈, 40₁₀₃₈
`\medbreak` 24₅₁₇
`\medmuskip` 9₁₅₄, 27₅₄₈, 35₈₈₆,
35₈₈₇
`\medskip` 22₄₅₁, 24₅₁₉
`\medskipamount` 20₃₈₆, 21₄₂₁,
22₄₅₀, 24₅₁₉
`\message` 24₅₁₆
`\mkern` 25₅₂₉, 25₅₃₀, 25₅₃₁,
25₅₃₂, 25₅₃₃, 25₅₃₄, 25₅₃₅,
28₅₉₂, 28₆₀₈, 28₆₁₀, 30₇₁₆,
31₇₃₈, 31₇₃₉, 31₇₆₄, 31₇₆₅,
33₈₁₉, 34₈₄₃, 35₈₈₆, 35₈₈₇,
35₈₈₈
`\mscount` 23₄₇₁, 23₄₇₂, 23₄₇₃
`\mskip` 27₅₄₇, 27₅₄₈, 27₅₄₉,
27₅₅₀, 35₈₈₆, 35₈₈₇
`\multiply` 7₁₁₀
`\muskip` 14₂₆₅, 31₇₆₄, 31₇₆₅
`\n@space` 33₈₀₇, 33₈₀₈, 33₈₀₉,
33₈₁₀
`\ne` 30₆₈₃
`\neg` 28₆₁₄
`\neq` 30₆₈₃
`\newfam` 13₂₃₂, 13₂₃₄, 13₂₃₆,
13₂₃₉
`\newinsert` 37₉₅₉, 38₉₈₂
`\newlinechar` 19₃₄₁
`\next` 7₁₃₂, 23₄₇₈, 23₄₈₀, 23₄₈₁,
33₈₂₅, 33₈₃₃, 33₈₃₄, 36₉₃₇,
36₉₃₈, 37₉₆₈, 37₉₆₉, 37₉₇₀,
37₉₇₁, 40₁₀₂₄
`\ni` 30₆₉₆
`\noalign` 28₆₀₉, 31₇₅₃, 31₇₅₆,
31₇₅₈, 31₇₅₉, 31₇₆₂, 31₇₆₃,
35₈₉₃, 35₈₉₄, 35₈₉₉, 35₉₀₂,
35₉₁₉
`\nobreak` 22₄₃₁, 22₄₃₄, 24₅₁₆,
38₁₀₀₁, 39₁₀₁₀
`\noexpand` 37₉₆₉, 40₁₀₂₉, 40₁₀₃₁
`\noindent` 24₅₁₆, 24₅₁₈
`\nointerlineskip` 21₄₂₈, 28₆₀₉,
31₇₅₃, 31₇₅₆, 31₇₅₉, 31₇₆₂,
36₉₄₉
`\nolimits` 28₆₂₈, 28₆₃₄, 34₈₅₃,
34₈₅₄, 34₈₅₅, 34₈₅₉, 34₈₆₀,
34₈₆₁, 34₈₆₂, 34₈₆₃, 34₈₆₄,
34₈₆₅, 34₈₆₆, 34₈₆₇, 34₈₆₈,
34₈₆₉, 34₈₇₀, 34₈₇₁, 34₈₇₆,
34₈₇₇, 34₈₇₈, 34₈₇₉, 34₈₈₁,
34₈₈₄
`\nonfrenchspacing` 39₁₀₂₂
`\nonscript` 35₈₈₆, 35₈₈₇
`\normalbaselines` 35₈₈₉, 35₈₉₁,
39₁₀₂₁
`\normalbaselineskip` 20₃₈₈,
20₃₉₅
`\normallineskip` 20₃₈₉, 20₃₉₄
`\normallineskiplimit` 20₃₉₀,
20₃₉₅, 35₉₂₀
`\not` 28₆₀₈, 30₆₈₃, 36₉₄₁

`\null` 23₄₇₇, 23₄₇₈, 23₄₉₃,
33₈₂₈, 35₈₉₁, 35₉₀₉, 35₉₁₄
`\nulldelimiterspace` 19₃₆₀,
33₈₁₁
`\number` 37₉₅₂
`\nxt` 23₄₈₀, 23₄₈₁, 23₄₈₂, 36₉₃₈,
36₉₃₉
`\oalign` 7₁₁₂, 7₁₁₃
`\oalign` 7₁₀₈, 7₁₀₉
`\obeylines` 21₄₀₆
`\obeyspaces` 21₄₀₉
`\of` 33₈₁₅
`\ointop` 28₆₃₄
`\omit` 23₄₇₁, 23₄₇₃, 35₉₀₂,
35₉₀₃
`\oalign` 7₁₁₆, 7₁₁₇, 34₈₄₃,
34₈₄₅
`\openup` 35₉₁₄, 35₉₁₈
`\or` 6₇₆
`\outer` 4₁₃, 14₂₆₂, 14₂₆₃, 14₂₆₄,
14₂₆₅, 14₂₆₆, 14₂₆₈, 14₂₆₉,
14₂₇₀, 14₂₇₁, 14₂₇₂, 14₂₇₃,
14₂₇₉, 23₄₈₄, 24₅₁₄, 24₅₁₇,
26₅₄₆, 40₁₀₂₇
`\output` 38₁₀₀₂
`\outputpenalty` 38₁₀₀₅
`\over` 34₈₄₈, 34₈₄₉
`\overfullrule` 19₃₅₂
`\owns` 30₆₉₆
`\p@` 15₂₉₂, 21₄₂₄, 21₄₂₅, 31₇₃₆,
31₇₃₇, 31₇₃₈, 31₇₃₉, 31₇₅₃,
31₇₅₆, 31₇₅₈, 31₇₅₉, 31₇₆₂,
31₇₆₃, 33₈₀₇, 33₈₀₈, 33₈₀₉,
33₈₁₀, 33₈₄₀, 35₈₉₉, 35₉₀₀,
35₉₀₇, 35₉₁₉, 36₉₄₈, 36₉₄₉,
37₉₅₀, 37₉₅₇, 37₉₅₈, 38₉₈₀,
38₉₈₁, 38₉₉₂
`\p@gefalse` 38₉₈₄, 38₉₉₄
`\p@getrue` 38₉₈₆
`\p@renwd` 35₈₉₇, 35₉₀₀, 35₉₀₆
`\pagebody` 38₁₀₀₃
`\pagecontents` 39₁₀₀₇
`\pagegoal` 38₉₉₄
`\pageno` 36₉₄₃, 37₉₅₂, 37₉₅₄,
37₉₅₅
`\pageshrink` 38₉₉₃
`\pagetotal` 38₉₉₃
`\par` 20₃₉₈, 21₄₀₆, 21₄₀₇, 22₄₃₀,
22₄₄₂, 22₄₄₃, 22₄₄₄, 22₄₄₅,
22₄₄₈, 22₄₅₀, 22₄₅₂, 24₅₁₀,
24₅₁₁, 24₅₁₄, 24₅₁₇, 24₅₁₈,
26₅₄₆, 38₉₉₀
`\parfillskip` 19₃₈₄, 26₅₄₄
`\parindent` 19₃₆₆, 24₅₀₈, 24₅₁₁,
24₅₁₂, 24₅₁₃
`\parskip` 19₃₇₂, 24₅₁₅
`\penalty` 22₄₃₆, 22₄₃₈, 22₄₃₉,
22₄₄₀, 22₄₄₂, 22₄₄₃, 22₄₄₅,
22₄₄₉, 22₄₅₁, 22₄₅₃, 24₅₁₄,
24₅₁₉, 35₈₈₇, 35₉₂₁, 38₉₉₆
`\phant` 33₈₂₁, 33₈₂₂, 33₈₂₃
`\plainoutput` 38₁₀₀₂
`\pr@@s` 36₉₃₈
`\pr@@@t` 36₉₃₈
`\pr@m@s` 36₉₃₇
`\predisplaypenalty` 18₃₀₉
`\preloaded` 5₂₇, 5₂₈, 5₃₀, 5₃₃,
5₃₄, 5₃₆, 5₃₇, 5₄₀, 5₄₁,
5₄₃, 5₄₇, 5₄₈, 5₅₀, 5₅₃,
5₅₄, 5₅₇, 5₅₈, 5₅₉, 5₆₁,
6₆₃, 6₆₅, 6₆₇, 6₆₉, 6₇₀,
6₇₁, 6₇₃, 6₇₄, 8₁₃₆, 8₁₃₇,
8₁₃₉, 8₁₄₃, 8₁₄₄, 8₁₄₆,
9₁₅₁, 9₁₅₂
`\pretolerance` 18₂₉₆, 26₅₄₅
`\prevdepth` 21₄₂₄, 22₄₃₀, 22₄₃₁,
35₉₁₉
`\prim@s` 36₉₃₆, 36₉₄₀
`\prime` 36₉₃₇
`\quad` 35₈₉₀, 35₈₉₂, 35₉₀₁
`\r@@t` 33₈₁₆
`\r@ggedbottomfalse` 37₉₅₈
`\r@ggedbottomtrue` 37₉₅₇
`\radical` 32₇₉₄
`\raise` 7₁₁₇, 7₉₉, 31₇₃₈, 31₇₃₉,
33₈₁₉, 34₈₄₅
`\rbrace` 32₇₈₄
`\read` 14₂₇₀
`\relax` 6₇₆, 7₁₁₂, 7₁₁₃, 7₁₁₉,
21₄₁₆, 21₄₁₇, 21₄₁₈, 23₄₆₇,
23₄₇₁, 23₄₇₉, 23₄₈₁, 23₄₈₂,
25₅₂₁, 25₅₂₃, 33₈₃₂, 40₁₀₂₄
`\Relbar` 30₇₂₃, 30₇₂₆
`\relbar` 30₇₂₄, 30₇₂₅
`\relpenalty` 18₃₀₄
`\removelastskip` 22₄₄₉, 22₄₅₁,
22₄₅₃, 24₅₁₉
`\repeat` 23₄₇₂, 23₄₈₆, 40₁₀₂₃,
40₁₀₂₅
`\rhook` 30₇₂₀
`\right` 33₈₀₇, 33₈₀₈, 33₈₀₉,
33₈₁₀, 35₈₉₀, 35₈₉₅, 35₉₀₈
`\Rightarrow` 30₇₂₃, 30₇₂₉
`\rightarrow` 25₅₃₂, 30₇₀₂, 30₇₀₃,
30₇₁₉, 30₇₂₄, 30₇₂₈
`\rightarrowfill` 31₇₅₃
`\rightharpoonup` 34₈₄₆
`\righthypenmin` 39₁₀₁₉
`\rightskip` 24₅₁₃, 25₅₂₁, 25₅₂₃,
37₉₆₇
`\rlap` 7₉₉, 36₉₃₄
`\rlh@` 34₈₄₄
`\rm` 34₈₅₃, 34₈₅₄, 34₈₅₅, 34₈₅₆,
34₈₅₇, 34₈₅₈, 34₈₅₉, 34₈₆₀,
34₈₆₁, 34₈₆₂, 34₈₆₃, 34₈₆₄,
34₈₆₅, 34₈₆₆, 34₈₆₇, 34₈₆₈,
34₈₆₉, 34₈₇₀, 34₈₇₁, 34₈₇₂,
34₈₇₃, 34₈₇₄, 34₈₇₅, 34₈₇₆,
34₈₇₇, 34₈₇₈, 34₈₇₉, 34₈₈₀,
34₈₈₁, 34₈₈₂, 34₈₈₃, 34₈₈₄,
35₈₈₇, 35₈₈₈, 39₁₀₂₁
`\romannumeral` 37₉₅₂
`\rootbox` 33₈₁₅, 33₈₁₉
`\s@tcols` 23₄₈₁
`\s@tt@b` 23₄₈₀
`\sb` 34₈₅₁
`\scriptfont` 12₂₂₅, 12₂₂₇, 12₂₂₉,
12₂₃₁, 13₂₃₇
`\scriptscriptfont` 12₂₂₅, 12₂₂₇,
12₂₂₉, 12₂₃₁, 13₂₃₈
`\scriptscriptstyle` 33₈₁₃,
33₈₁₅
`\scriptspace` 19₃₆₁
`\scriptstyle` 28₆₀₇, 33₈₁₃
`\setbox` 7₁₁₅, 7₉₄, 7₉₈, 22₄₆₃,
23₄₆₆, 23₄₇₇, 23₄₇₈, 23₄₈₈,
23₄₉₂, 23₄₉₃, 23₄₉₆, 24₄₉₇,
24₅₀₀, 24₅₀₂, 24₅₀₃, 24₅₀₄,
24₅₀₅, 24₅₀₆, 25₅₃₈, 26₅₄₁,
26₅₄₄, 33₈₁₅, 33₈₁₇, 33₈₂₆,
33₈₂₇, 33₈₂₈, 33₈₃₅, 33₈₃₆,
35₈₉₇, 35₈₉₉, 35₉₀₄, 35₉₀₅,
35₉₀₆, 35₉₀₇, 38₉₉₀
`\sett@b` 23₄₇₈
`\sevenbf` 5₄₂, 13₂₃₇
`\seveni` 8₁₃₈, 10₁₅₆, 12₂₂₇
`\sevenrm` 5₂₉, 12₂₂₅
`\sevensy` 8₁₄₅, 10₁₅₇, 12₂₂₉
`\sfcode` 8₁₃₄, 25₅₂₄, 25₅₂₅,
25₅₂₆, 25₅₂₇
`\sh@t` 7₁₁₂, 7₁₁₃
`\shipout` 38₁₀₀₃
`\showboxbreadth` 19₃₄₇, 40₁₀₃₈
`\showboxdepth` 19₃₄₈, 26₅₄₅,
40₁₀₃₈
`\sim` 34₈₃₉
`\skewchar` 10₁₅₆, 10₁₅₇
`\skip` 14₂₆₄, 15₂₈₂, 37₉₇₆,
38₉₈₇, 39₁₀₁₅
`\skip@` 21₄₂₉, 22₄₃₁, 22₄₃₂,
22₄₃₄
`\sl` 24₅₁₈
`\slfam` 13₂₃₄, 13₂₃₅
`\smallskip` 22₄₄₉, 24₅₁₆
`\smallskipamount` 20₃₈₅, 21₄₂₀,
22₄₄₈
`\smash` 25₅₃₀, 25₅₃₁, 25₅₃₄,
25₅₃₅, 30₇₁₇
`\sp` 34₈₅₁
`\sp@n` 23₄₇₂
`\space` 21₄₀₉
`\spacefactor` 22₄₃₃, 22₄₃₄,
37₉₆₁
`\spaceskip` 25₅₂₁, 37₉₆₇
`\span` 23₄₇₃
`\splitmaxdepth` 19₃₅₆, 37₉₆₆,
38₉₉₈

`\splittopskip` 19₃₈₀, 37₉₆₅,
37₉₇₄, 38₉₉₇
`\sqrt` 33₈₁₇
`\string` 14₂₇₈, 15₂₈₆, 40₁₀₃₃
`\strut` 35₉₀₂, 35₉₀₃, 35₉₁₅,
37₉₇₃
`\strutbox` 23₄₆₆, 23₄₆₇, 37₉₆₅,
37₉₆₆
`\supereject` 26₅₄₆, 39₁₀₁₀
`\t@bb@x` 24₄₉₈
`\t@b@box` 24₄₉₈
`\tabalign` 23₄₈₄
`\tabs` 23₄₇₇, 23₄₇₈, 23₄₈₈,
23₄₉₂, 23₄₉₆
`\tabdone` 23₄₉₃, 23₄₉₆, 24₅₀₆
`\tabskip` 23₄₆₉, 35₉₂₂, 35₉₂₃,
36₉₂₆, 36₉₂₇, 36₉₂₈, 36₉₂₉,
36₉₃₁, 36₉₃₂, 36₉₃₃, 36₉₃₄
`\tabsyet` 23₄₇₇, 23₄₉₂, 23₄₉₆,
24₅₀₂
`\tenbf` 5₃₉, 13₂₃₆, 13₂₃₇
`\tenex` 8₁₄₉, 12₂₃₁, 35₈₉₇
`\teni` 8₁₃₅, 10₁₅₆, 12₂₂₇, 12₂₂₈
`\tenit` 5₅₆, 13₂₃₂, 13₂₃₃
`\tenrm` 5₂₆, 12₂₂₅, 12₂₂₆, 26₅₄₄,
36₉₄₅
`\tensl` 5₅₂, 13₂₃₄, 13₂₃₅
`\tensy` 8₁₄₂, 10₁₅₇, 12₂₂₉
`\tentt` 5₄₆, 13₂₃₉, 13₂₄₀
`\textfont` 7₁₃₂, 12₂₂₅, 12₂₂₇,
12₂₂₉, 12₂₃₁, 13₂₃₃, 13₂₃₅,
13₂₃₇, 13₂₄₀, 34₈₅₂
`\textintend` 24₅₁₀, 24₅₁₁, 37₉₆₈
`\textstyle` 33₈₁₃, 34₈₄₉
`\the` 7₁₃₂, 14₂₇₈, 15₂₈₆, 34₈₅₂,
36₉₄₉, 37₉₅₀, 37₉₆₁
`\thickmuskip` 9₁₅₅, 27₅₄₉
`\thinmuskip` 9₁₅₃, 27₅₄₇, 27₅₅₀
`\thinspace` 34₈₅₂, 35₉₀₀
`\to` 30₇₀₂
`\toks` 14₂₆₉
`\tolerance` 18₂₉₇, 26₅₄₅
`\topins` 38₉₈₂, 38₉₈₇, 38₉₈₈,
38₉₈₉, 38₉₉₆, 39₁₀₁₂
`\topskip` 19₃₇₉, 21₄₂₈, 37₉₅₇,
37₉₅₈, 39₁₀₁₀
`\tracingcommands` 40₁₀₃₅
`\tracinglostchars` 18₃₂₆, 40₁₀₃₆
`\tracingmacros` 40₁₀₃₇
`\tracingonline` 40₁₀₃₅
`\tracingoutput` 40₁₀₃₆
`\tracingpages` 40₁₀₃₆
`\tracingparagraphs` 40₁₀₃₇
`\tracingrestores` 40₁₀₃₇
`\tracingstats` 40₁₀₃₅
`\triangleleft` 30₇₂₁
`\triangleright` 30₇₂₁
`\tt` 25₅₂₃
`\ttfam` 13₂₃₉, 13₂₄₀
`\tw@` 12₂₃₀, 31₇₆₄, 33₈₂₈, 33₈₂₉,
33₈₃₀, 35₉₀₄, 35₉₀₅, 35₉₀₆,
35₉₀₉, 40₁₀₃₅, 40₁₀₃₇
`\u` 7₁₂₅
`\uccode` 40₁₀₃₄
`\uchyph` 18₃₃₀
`\undefined` 6₇₄
`\underline` 22₄₆₄
`\unhbox` 7₁₁₆, 7₉₆, 23₄₈₈, 23₄₉₆,
24₅₀₂, 24₅₀₅, 24₅₀₆, 35₉₀₅
`\unhcopy` 23₄₆₇
`\unskip` 35₉₀₅
`\unvbox` 23₄₉₅, 35₉₀₈, 38₁₀₀₀,
39₁₀₁₂, 39₁₀₁₃, 39₁₀₁₇
`\unvcopy` 35₉₀₄
`\upbracefill` 31₇₆₃
`\uppercase` 40₁₀₃₄
`\us@false` 23₄₈₂
`\us@true` 23₄₈₃
`\v` 7₁₂₄
`\v@false` 33₈₂₂
`\v@true` 33₈₂₁, 33₈₂₃
`\vbadness` 18₂₉₉
`\vbox` 7₁₁₄, 7₉₇, 24₄₉₇, 26₅₄₄,
28₆₀₇, 31₇₃₆, 31₇₃₈, 31₇₅₂,
31₇₅₅, 31₇₅₈, 33₈₀₇, 33₈₀₈,
33₈₀₉, 33₈₁₀, 34₈₄₀, 35₈₉₉,
35₉₀₄, 35₉₀₇, 35₉₀₉, 36₉₄₈,
36₉₄₉, 37₉₇₄, 38₁₀₀₀, 38₁₀₀₃,
38₉₉₀, 39₁₀₀₇
`\vcenter` 34₈₄₅, 35₈₈₉, 35₈₉₁,
35₉₀₈, 35₉₁₄
`\vee` 29₆₄₂
`\Vert` 32₇₇₃
`\vfil` 22₄₄₂, 39₁₀₁₈
`\vfill` 26₅₄₆, 39₁₀₁₀
`\vfilneg` 22₄₄₂
`\vfootnote` 37₉₆₂
`\vfuzz` 19₃₅₁
`\vgl@` 21₄₂₉
`\vglue` 21₄₂₈
`\voidb@x` 7₉₆
`\vphantom` 33₈₃₁
`\vrule` 22₄₃₃, 23₄₆₆, 25₅₃₉,
26₅₄₀, 26₅₄₂, 26₅₄₃
`\vsize` 6₇₉, 19₃₅₄, 24₅₁₄, 24₅₁₅,
38₁₀₀₀, 39₁₀₀₇
`\vskip` 21₄₂₀, 21₄₂₁, 21₄₂₂,
22₄₃₁, 22₄₄₇, 24₅₁₄, 24₅₁₅,
35₉₂₀, 36₉₄₈, 39₁₀₁₅
`\vss` 7₁₁₄, 36₉₄₉
`\vtop` 7₁₀₇, 31₇₆₁
`\wd` 7₉₄, 24₅₀₄, 24₅₀₅, 33₈₃₀,
35₈₉₇, 35₉₀₆
`\wedge` 29₆₄₁
`\widowpenalty` 18₃₀₆
`\wlog` 14₂₇₈, 15₂₈₆
`\write` 14₂₅₅, 14₂₇₁
`\xspaceskip` 25₅₂₁, 37₉₆₇
`\z@` 7₁₀₈, 7₁₁₀, 7₁₁₁, 7₁₁₅,
7₁₁₆, 12₂₂₆, 15₂₉₃, 21₄₂₆,
22₄₃₀, 22₄₃₃, 22₄₄₀, 22₄₄₇,
22₄₅₉, 22₄₆₀, 22₄₆₂, 22₄₆₃,
22₄₆₄, 23₄₆₆, 23₄₈₆, 23₄₉₅,
24₄₉₇, 24₅₀₀, 24₅₀₄, 24₅₀₅,
24₅₀₇, 24₅₁₄, 24₅₁₅, 25₅₂₁,
25₅₂₃, 25₅₃₈, 25₅₃₉, 26₅₄₀,
26₅₄₁, 26₅₄₂, 26₅₄₃, 31₇₃₆,
31₇₆₄, 31₇₆₅, 33₈₁₁, 33₈₁₇,
33₈₁₈, 33₈₁₉, 33₈₂₆, 33₈₂₇,
33₈₂₉, 33₈₃₀, 33₈₃₅, 33₈₃₆,
34₈₃₇, 34₈₄₈, 35₈₉₉, 35₉₀₄,
35₉₀₈, 36₉₄₈, 37₉₅₂, 37₉₅₄,
38₁₀₀₀, 38₁₀₀₁, 38₉₉₀, 38₉₉₂,
38₉₉₅, 38₉₉₈, 38₉₉₉, 39₁₀₀₉
`\z@skip` 7₁₀₇, 15₂₉₄, 23₄₆₉,
26₅₄₄, 35₉₂₂, 35₉₂₃, 36₉₂₇,
36₉₂₉, 36₉₃₂, 37₉₆₇, 38₉₈₇,
38₉₉₇

Chunks

<code>< * ></code>	3
<code>< Allocate registers ></code>	3, 13
<code>< Allocate scratch registers ></code>	14
<code>< Assign delimiter codes ></code>	8, 12
<code>< Assign initial values to parameters ></code>	3, 18
<code>< Assign math codes ></code>	8, 10
<code>< Assign space factor codes ></code>	5, 8
<code>< Assign uppercase and lowercase code values ></code>	5, 8
<code>< Assign values to dimen parameters ></code>	18, 19
<code>< Assign values to glue parameters ></code>	18, 19
<code>< Assign values to integer parameters ></code>	18
<code>< Assign values to special registers ></code>	18, 20
<code>< Completing the job ></code>	20, 26
<code>< Define Greek letters ></code>	26, 27
<code>< Define \showhyphens macro ></code>	20, 26
<code>< Define \tracingall macro ></code>	40
<code>< Define commonly used constants ></code>	3, 4
<code>< Define font families ></code>	8, 12
<code>< Define implementation-level register allocation macros ></code>	14
<code>< Define math fonts ></code>	8
<code>< Define math space macros ></code>	26, 27
<code>< Define math symbols ></code>	26, 28
<code>< Define sectioning macros ></code>	20, 24
<code>< Define space macros ></code>	20–22
<code>< Define text fonts ></code>	5, 6
<code>< Encode binary operations ></code>	26, 29
<code>< Encode large operators ></code>	26, 28
<code>< Encode math accents ></code>	8, 10
<code>< Encode relations ></code>	26, 29
<code>< Encode special characters, and characters not available on the keyboard ></code>	5, 6
<code>< Establish spacing after punctuation characters ></code>	20, 25
<code>< Establish spacing around mathematical objects ></code>	8, 9
<code>< Establish standard category code values ></code>	3, 4
<code>< Identify the format ></code>	3, 40
<code>< Initialize register constants ></code>	14, 15
<code>< Initialize the layout ></code>	3, 39
<code>< Prepare page for output ></code>	3, 36
<code>< Provide \cases and \matrix macros ></code>	26, 35
<code>< Provide access to delimiters of various sizes ></code>	26, 33
<code>< Provide alignment macros ></code>	20, 23
<code>< Provide macros for math formatting ></code>	3, 26, 33
<code>< Provide macros for text formatting ></code>	3, 20
<code>< Provide programming constructs ></code>	3, 40
<code>< Provide support for accented characters ></code>	5, 7
<code>< Provide support for font scaling ></code>	5, 6
<code>< Provide support to typeset displayed equations ></code>	26, 35
<code>< Provide user-level register allocation macros ></code>	14
<code>< Read hyphenation patterns ></code>	3, 39

⟨ <i>Set up math fonts</i> ⟩	3, 8
⟨ <i>Set up text fonts</i> ⟩	3, 5
⟨ <i>Set up the output routine</i> ⟩	36, 38
⟨ <i>Supply ‘boxing’ macros</i> ⟩	20, 22
⟨ <i>Supply basic macros for text formatting</i> ⟩	20, 21
⟨ <i>Supply common math functions</i> ⟩	26, 34, 36
⟨ <i>Supply extensible delimiters</i> ⟩	26, 32
⟨ <i>Supply floating insertions</i> ⟩	36, 38
⟨ <i>Supply footnotes</i> ⟩	36, 37
⟨ <i>Supply headers and footers</i> ⟩	36
⟨ <i>Supply loops and conditionals</i> ⟩	40
⟨ <i>Supply ragged setting</i> ⟩	20, 25
⟨ <i>Supply raggedbottom setting</i> ⟩	36, 37
⟨ <i>Supply strut</i> ⟩	20, 23
⟨ <i>Supply variable-width math accents</i> ⟩	26, 31
⟨ <i>Supply various paragraph shapes</i> ⟩	20, 24
⟨ <i>Supply various ways to fill space</i> ⟩	20, 25
⟨ <i>Supply vertical and diagonal dots</i> ⟩	26, 31