This is a list of all corrections made to *Computers & Typesetting*, Volumes A–E, between 20 February 1989 and 30 September 1989 (when TEX Version 3.0 and METAFONT Version 2.0 were fully defined). Corrections made to the softcover version of *The TEXbook* are the same as corrections to Volume A. Corrections to the softcover version of *The METAFONTbook* are the same as corrections to Volume C. Some of these corrections have already been made in reprintings of the books. Several minor changes to Volumes A and C are not shown here because they simply make room for the more substantive changes needed to describe the new features of TEX Version 3.0 and METAFONT Version 2.0. Hundreds of changes will soon be made to Volumes B and D because of the upgrades to TEX and METAFONT; it will unfortunately be impossible to document all of those changes. Therefore, readers who need up-to-date information on the TEX and METAFONT programs should refer to the `WEB` source files until new printings of Volumes B and D are issued.

Volume A, in general (9/23/89)

[Change '127' to '255' and '128' to '256' in contexts referring to character codes. This happens on pages 37(twice), 39, 41, 43, 44(twice), 48, 93, 154, 277, 305(twice), 308(twice), 313, and 343. Also change '7-bit' to '8-bit' on pages 214 and 277.]

Page A23, line 16 (9/23/89)

```
This is TeX, Version 3.0 (preloaded format=plain 89.7.15)
```

Page A34, new copy for bottom of page (9/23/89)

If you use TEX format packages designed by others, your error messages may involve many inscrutable two-line levels of macro context. By setting `\errorcontextlines=0` at the beginning of your file, you can reduce the amount of information that is reported; TEX will show only the top and bottom pairs of context lines together with up to `\errorcontextlines` additional two-line items. (If anything has thereby been omitted, you'll also see '`...`'.) Chances are good that you can spot the source of an error even when most of a large context has been suppressed; if not, you can say '`I\errorcontextlines=100\oops`' and try again. (That will usually give you an undefined control sequence error and plenty of context.) Plain TEX sets `\errorcontextlines=5`.

Page A45, lines 9–15 (9/23/89)

`^^` has an internal code between 64 and 127, TEX subtracts 64 from the code; if the code is between 0 and 63, TEX adds 64. Hence code 127 can be typed `^^?`, and the dangerous bend sign can be obtained by saying `{\manual^^?}`. However, you must change the category code of character 127 before using it, since this character ordinarily has category 15 (invalid); say, e.g., `\catcode`\^^?=12`. The `^^` notation is different from `\char`, because `^^` combinations are like single characters; for example, it would not be permissible to say `\catcode`\char127`, but `^^` symbols can even be used as letters within control words.

Page A45, new copy before line 20 (9/23/89)

There's also a special convention in which `^^` is followed by *two* "lowercase hexadecimal digits," `0`–`9` or `a`–`f`. With this convention, all 256 characters are obtainable in a uniform way, from `^^00` to `^^ff`. Character 127 is `^^7f`.

[Also remove one of the two dangerous bend signs on line 20.]

Page A45, bottom paragraph and footnote (9/23/89)

People who install TEX systems for use with non-American alphabets can make TEX conform to any desired standard. For example, suppose you have a Norwegian keyboard containing the letter æ, which comes in as code 241 (say). Your local format package should define `\catcode‘æ=11`; then you could have control sequences like `\særtrykk`. Your TEX input files could be made readable by American installations of TEX that don't have your keyboard, by substituting `^^f1` for character 241. (For example, the stated control sequence would appear as `\s^^f1rtrykk` in the file; your American friends should also be provided with the format that you used, with its `\catcode‘^^f1=11`.) Of course you should also arrange your fonts so that TEX's character 241 will print as æ; and you should change TEX's hyphenation algorithm so that it will do correct Norwegian hyphenation. The main point is that such changes are not extremely difficult; nothing in the design of TEX limits it to the American alphabet. Fine printing is obtained by fine tuning to the language or languages being used.

European languages can also be accommodated effectively with only a limited character set. For example, let's consider Norwegian again, but suppose that

[Now continue with the text on line 11 of page 46.]

Page A47, lines 9–21 (9/23/89)

If TEX sees a superscript character (category 7) in any state, and if that character is followed by another identical character, and if those two equal characters are followed by a character of code $c < 128$, then they are deleted and 64 is added to or subtracted from the code c. (Thus, `^^A` is replaced by a single character whose code is 1, etc., as explained earlier.) However, if the two superscript characters are immediately followed by two of the lowercase hexadecimal digits `0123456789abcdef`, the four-character sequence is replaced by a single character having the specified hexadecimal code. The replacement is carried out also if such a trio or quartet of characters is encountered during steps (b) or (c) of the control-sequence-name scanning procedure described above. After the replacement is made, TEX begins again as if the new character had been present all the time. If a superscript character is not the first of such a trio or quartet, it is handled by the following rule.

If TEX sees a character of categories 1, 2, 3, 4, 6, 8, 11, 12, or 13, or a character of category 7 that is not the first of a special sequence as just described, it converts the character to a token by attaching the category code, and goes into state M. This is the normal case; almost every nonblank character is handled by this rule.

Page A48, line 15 (9/23/89)

the input line ‘ `$x^2$~ \TeX ^^62^^6`’?

Page A54, third line from the bottom (9/23/89)

For example, a well-designed TEX font for French might well treat accents as lig-

Page A76, lines 3–5 from the bottom (9/23/89)

TEX does not assign any value to `\sfcode'042`.

Page A107, new copy for top of page (9/23/89)

If you want to avoid overfull boxes at all costs without trying to fix them manually, you might be tempted to set `tolerance=10000`; this allows arbitrarily bad lines to be acceptable in tough situations. But infinite tolerance is a bad idea, because TEX doesn't distinguish between terribly bad and preposterously horrible lines. Indeed, a tolerance of 10000 encourages TEX to concentrate all the badness in one place, making one truly unsightly line instead of two moderately bad ones, because a single “write-off” produces fewest total demerits according to the rules. There's a much better way to get the desired effect: TEX has a parameter called `\emergencystretch` that is added to the assumed stretchability of every line when badness and demerits are computed, in cases where overfull boxes are otherwise unavoidable. If `\emergencystretch` is positive, TEX will make a third pass over a paragraph before choosing the line breaks, when the first passes did not find a way to satisfy the `\pretolerance` and `\tolerance`. The effect of `\emergencystretch` is to scale down the badnesses so that large infinities are distinguishable from smaller ones. By setting `\emergencystretch` high enough (based on `\hsize`) you can be sure that the `\tolerance` is never exceeded; hence overfull boxes will never occur unless the line-breaking task is truly impossible.

Page A116, lines 11–15 (6/7/89)

If you have two or more `\topinsert` or `\pageinsert` commands in quick succession, TEX may need to carry them over to several subsequent pages; but they will retain their relative order when they are carried over. For example, suppose you have pages that are nine inches tall, and suppose you have already specified 4 inches of text for some page, say page 25. Then suppose you make seven topinserts in a row, of

Page A125, lines 13–29 (9/23/89)

When the best page break is finally chosen, TEX removes everything after the chosen breakpoint from the bottom of the “current page,” and puts it all back at the top of the “recent contributions.” The chosen breakpoint itself is placed at the very top of the recent contributions. If it is a penalty item, the value of the penalty is recorded in `\outputpenalty` and the penalty in the contribution list is changed to 10000; otherwise `\outputpenalty` is set to 10000. The insertions that remain on the current page are of three kinds: For each class n there are unsplit insertions, followed possibly by a single split insertion, followed possibly by others. If `\holdinginserts` > 0,

all insertions remain in place (so that they might be contributed again); otherwise they are all removed from the current page list as follows: The unsplit insertions are appended to `\box`n, with no interline glue between them. (Struts should be used, as in the `\vfootnote` macro of Appendix B.) If a split insertion is present, it is effectively `\vsplit` to the size that was computed previously in Step 4; the top part is treated as an unsplit insertion, and the remainder (if any) is converted to an insertion as if it had not been split. This remainder, followed by any other floating insertions of the same class, is held over in a separate place. (They will show up on the "current page" if `\showlists` is used while an `\output` routine is active; the total number of such insertions appears in `\insertpenalties` during an `\output` routine.) Finally, the remaining items before the best break on the current page are put together in a `\vbox`

Page A131, line 12 (9/22/89)

work fine; but sometimes you want to have uniformity between different members of a

Page A155, lines 3–5 (9/23/89)

when it encounters a character that is given explicitly as `\char`⟨number⟩.

Page A214, lines 19–24 (9/23/89)

- `\the`⟨special register⟩, where ⟨special register⟩ is one of the integer quantities `\prevgraf`, `\deadcycles`, `\insertpenalties`, `\inputlineno`, `\badness`, or `\parshape` (denoting only the number of lines of `\parshape`); or one of the dimensions `\pagetotal`, `\pagegoal`, `\pagestretch`, `\pagefilstretch`, `\pagefillstretch`, `\pagefilllstretch`, `\pageshrink`, `\pagedepth`. In horizontal modes you can also refer to a special integer, `\the\spacefactor`; in vertical modes there's a special dimension, `\the\prevdepth`.

Page A229, new copy after line 11 (9/23/89)

TEX will report the badness of glue setting in a box if you ask for the numeric quantity `\badness` after making a box. For example, you might say

```
\setbox0=\line{\trialtexta}
\ifnum\badness>250 \setbox0=\line{\trialtextb}\fi
```

The badness is between 0 and 10000 unless the box is overfull, when `\badness=1000000`.

Page A271, lines 17–20 (9/23/89)

| ⟨countdef token⟩ | `\count`⟨8-bit number⟩ | ⟨codename⟩⟨8-bit number⟩
| ⟨chardef token⟩ | ⟨mathchardef token⟩ | `\parshape` | `\inputlineno`
| `\hyphenchar`⟨font⟩ | `\skewchar`⟨font⟩ | `\badness`

Page A272, lines 3–4 (9/23/89)

value is between 0 and $2^8 - 1 = 255$; a ⟨4-bit number⟩ is similar.

Page A273, insert after lines 11, 20, 21, 21, 38 (9/23/89)

`\holdinginserts` (positive if insertions remain dormant in output box)
`\language` (the current set of hyphenation rules)
`\lefthyphenmin` (smallest fragment at beginning of hyphenated word)
`\righthyphenmin` (smallest fragment at end of hyphenated word)
`\errorcontextlines` (maximum extra context shown when errors occur)

Page A274, insert after line 4 (9/23/89)

`\emergencystretch` (reduces badnesses on final pass of line-breaking)

Page A275, line 13 (9/23/89)

That makes a total of 103 parameters of all five kinds.

Page A283, line 14 (9/23/89)

| `\noboundary` | `\unhbox` | `\unhcopy` | `\valign` | `\vrule`

Page A286, lines 3–12 from the bottom (9/23/89)

■ ⟨letter⟩, ⟨otherchar⟩, `\char`⟨8-bit number⟩, ⟨chardef token⟩, `\noboundary`. The most common commands of all are the character commands that tell TEX to append a character to the current horizontal list, using the current font. If two or more commands of this type occur in succession, TEX processes them all as a unit, converting to ligatures and/or inserting kerns as directed by the font information. (Ligatures and kerns may be influenced by invisible "boundary" characters at the left and right, unless `\noboundary` appears.) Each character command adjusts `\spacefactor`, using the `\sfcode` table as described in Chapter 12. In unrestricted horizontal mode, a '`\discretionary{}{}{}`' item is appended after a character whose code is the `\hyphenchar` of its font, or after a ligature formed from a sequence that ends with such a character.

Page A287, insert after line 19 (9/23/89)

■ `\setlanguage`⟨number⟩. See the conclusion of Appendix H.

Page A289, lines 9–14 from the bottom (9/23/89)

$2^{15} - 1$. This is done by replacing the character number by its `\mathcode` value. If the `\mathcode` value turns out to be 32768 = ″`8000`, however, the ⟨character⟩ is replaced by an active character token having the original character code (0 to 255); TEX forgets the original ⟨character⟩ and expands this active character according to the rules of Chapter 20.

Page A290, insert before 13th line from bottom (9/23/89)

▪ `\noboundary`. This command is redundant and therefore has no effect; boundary ligatures are automatically disabled in math modes.

Page A296, line 16 from the bottom (9/22/89)

[There should be a '`^`' just above the '`3`' in the line below. This was mistakenly dropped by the printer some time during 1985; it was correct in the first two printings and it has always been correct inside the computer!]

Page A309, lines 3–5 (9/23/89)

8.4. `$`$_3$ `x`$_{11}$ `^`$_7$ `2`$_{12}$ `$`$_3$ `~`$_{13}$ ␣$_{10}$ `TeX` `b`$_{12}$ `v`$_{12}$ ␣$_{10}$. The final space comes from the ⟨return⟩ placed at the end of the line. Code `^^6` yields `v` only when not followed by `0`–`9` or `a`–`f`. The initial space is ignored, because state N governs the beginning of the line.

Page A314, line 27 (9/23/89)

The English word 'eighteen' might deserve similar treatment. TEX's hyphenation algorithm will not make such spelling changes automatically.

Page A318, line 19 (3/3/89)

```
\def\clearnotenumber{\notenumber=0\relax}
```

Page A330, line 3 (8/25/89)

20.10. `\def\overpaid{{\count0=\balance`

Page A336, lines 4–8 from the bottom (9/23/89)

badness rating of a box is at most 10000, except that the `\badness` of an overfull box is 1000000. `INITEX` initializes `\tolerance` to 10000, thereby making all line breaks feasible. Penalties of 10000 or more prohibit breaks; penalties of −10000 or less make breaks mandatory. The cost of a page break is 100000, if the badness is 10000 and if the associated penalties are less than 10000 in magnitude (see Chapter 15).

Page A337, lines 2–16 (9/23/89)

ifies characters whose codes differ by 64 from the codes of `?`, `@`, `A`; this convention applies only to characters with ASCII codes less than 128. There are 256 possible characters, hence 256 entries in each of the `\catcode`, `\mathcode`, `\lccode`, `\uccode`, `\sfcode`, and `\delcode` tables. All `\lccode`, `\uccode`, and `\char` values must be less than 256. A font has at most 256 characters. There are 256 `\box` registers, 256 `\count` registers, 256 `\dimen` registers, 256 `\skip` registers, 256 `\muskip` registers, 256 `\toks` registers, 256 hyphenation tables. The "at size" of a font must be less than 2048 pt, i.e., 2^{11} pt. Math delimiters are encoded by multiplying the math code of the "small character" by 2^{12}. The magnitude of a ⟨dimen⟩ value must be less than 16384 pt, i.e., 2^{14} pt; similarly, the ⟨factor⟩ in a ⟨fil dimen⟩ must be less than 2^{14}. A `\mathchar`

or `\spacefactor` or `\sfcode` value must be less than 2^{15}; a `\mathcode` or `\mag` value must be less than or equal to 2^{15}, and 2^{15} denotes an "active" math character. There are 2^{16} sp per pt. A `\delcode` value must be less than 2^{24}; a `\delimiter`, less than 2^{27}. The `\end` command sometimes contributes a penalty of -2^{30} to the current page. A ⟨dimen⟩ must be less than 2^{30} sp in absolute value; a ⟨number⟩ must be less than 2^{31} in absolute value.

Page A348, line 12 from the bottom (9/23/89)

```
\showboxbreadth=5 \showboxdepth=3 \errorcontextlines=5
```

Page A364, insert before line 18 from the bottom (9/23/89)

```
\lefthyphenmin=2 \righthyphenmin=3 % disallow x- or -xx breaks
```

Page A364, line 5 from the bottom (9/23/89)

```
\def\fmtname{plain}\def\fmtversion{3.0} % identifies the current format
```

Page A369, insert before line 5 from the bottom (9/23/89)

Modern keyboards allow 256 codes to be input, not just 128; so TEX represents characters internally as numbers in the range 0–255 (i.e., *´000–´377*, or *"*`00`*–"*`FF`). Implementations of TEX differ in which characters they will accept in input files and which they will transmit to output files; these subsets can be specified independently. A completely permissive version of TEX allows full 256-character input and output; other versions might ignore all but the visible characters of ASCII; still other versions might distinguish the tab character (code *´011*) from a space on input, but might output each tab as a sequence of three characters `^^I`.

Page A370, lines 3–7 (9/23/89)

close as possible to the ASCII conventions. (b) Make sure that codes *´041–´046*, *´060–´071*, *´141–´146*, and *´160–´171* are present and that each unrepresentable internal code $<$ *´200* leads to a representable code when *´100* is added or subtracted; then all 256 codes can be input and output. (c) Cooperate with everyone else who shares the same constraints, so that you all adopt the same policy. (See Appendix J for information about the TEX Users Group.)

Page A370, bottom line (9/23/89)

doesn't matter if these symbols have their plain TEX meanings or not. (6) There is a special convention for representing characters 0–255 in the hexadecimal forms `^^00`–`^^ff`, explained in Chapter 8. This convention is always acceptable as input, when `^` is any character of catcode 7. Text output is produced with this convention only when representing characters of code $\geq$ 128 that a TEX installer has chosen not to output directly.

Page A385, line 8 (5/14/89)

```
\def\beginbox{\setbox0=\hbox\bgroup}
```

Page A400, line 18 from the bottom (9/23/89)

page prematurely if you want to pass a signal. (Set `\holdinginserts` positive to pass a signal when the contents of `\box255` will be sent back through the page builder again, if any insertions are present.)

Page A419, lines 4–6 (9/23/89)

shortened or lengthened anyway; book preparation with TEX, as with type, encourages interaction between humans and machines.) The lines of the quotations are set flush right by using `\obeylines` together with a stretchable `\leftskip`:

Page A444, lines 21–26 (9/23/89)

following one, using the specified family and the current size, then insert the ligature character and continue as specified by the font; two characters may collapse into one, or a new character may appear. Otherwise if the font information shows a kern between the current symbol and the next, insert a kern item after the current Ord atom and move to the next item after that. Otherwise (i.e., if no ligature or kern is specified between the present text symbol and the following character), go to Rule 17.

Page A453, lines 12–14 from the bottom (9/23/89)

Exception: The character '`.`' is treated as if it were a ⟨letter⟩ of code 0 when it appears in a pattern. Code 0 (which obviously cannot match a nonzero `\lccode`) is used by TEX to represent the left or right edge of a word when it is being hyphenated.

Page A454, lines 7–15 from the bottom (9/23/89)

If a trial word $l_1 \ldots l_n$ has been found by this process, hyphenation will still be abandoned unless $n \geq \lambda + \rho$, where $\lambda = \max(1, \texttt{\lefthyphenmin})$ and $\rho = \max(1, \texttt{\righthyphenmin})$. (Plain TEX takes $\lambda = 2$ and $\rho = 3$.) Furthermore, the items immediately following the trial word must consist of zero or more characters, ligatures, and implicit kerns, followed immediately by either glue or an explicit kern or a penalty item or a whatsit or an item of vertical mode material from `\mark`, `\insert`, or `\vadjust`. Thus, a box or rule or math formula or discretionary following too closely upon the trial word will inhibit hyphenation. (Since TEX inserts empty discretionaries after explicit hyphens, these rules imply that already-hyphenated compound words will not be further hyphenated by the algorithm.)

Page A455, new copy after line 13 (9/23/89)

So far we have assumed that TEX knows only one style of hyphenation at a time; but in fact TEX can remember up to 256 distinct sets of rules, if you have enough memory in your computer. An integer parameter called `\language` selects the rules actually used; every `\hyphenation` and `\patterns` specification appends new rules to those previously given for the current value of `\language`. (If `\language` is negative or greater than 255, TEX acts as if `\language` = 0.) All `\patterns` for all languages must be given before a paragraph is typeset, if `INITEX` is used for typesetting.

TEX is able to work with several languages in the same paragraph, because it operates as follows. At the beginning of a paragraph the "current language" is defined to be 0. Whenever a character is added to the current paragraph (i.e., in unrestricted horizontal mode), the current language is compared to `\language`; if they differ, the current language is reset and a whatsit node specifying the new current language is inserted before the character. Thus, if you say '`\def\french{\language1...}`' and '`mix {\french franc/ais} with English`', TEX will put whatsits before the `f` and the `w`; hence it will use language 1 rules when hyphenating `franc/ais`, after which it will revert to language 0. You can insert the whatsit yourself (even in restricted horizontal mode) by saying `\setlanguage`⟨number⟩; this changes the current language but it does not change `\language`.

Page A459, right column (9/23/89)

*`\badness`, 214, *229*, 271.

Page A461, right column (9/23/89)

caron, *see* háček.

Page A464, line 10 (5/15/89)

displays, 87, 103, *139–145*, *166–167*,

Page A464, right column (9/23/89)

*`\emergencystretch`, 107, 274.

Page A465, left column (9/23/89)

*`\errorcontextlines`, 34, 273, *348*.

Page A466, entry for 'fractions' (9/23/89)

[Add page 332 to this entry.]

Page A466, entry for 'French" (9/23/89)

[Add page 455 to this entry.]

Page A467, entry for 'hexadecimal' (9/23/89)

[Add pages 45, 47–48 to this entry.]

Page A467, right column (9/23/89)

*`\holdinginserts`, 125, 273, 400.

Page A467, bottom line (9/23/89)

*`\hyphenation`, 277, *419*, 452–453, 455.

Page A468, right column (9/23/89)

infinite badness, 97, 107, 111, 229, 317.

Page A468, right column (9/23/89)

*`\inputlineno`, 214, 271.

Page A469, entry for kerns (9/23/89)

[Add pages 286 and 444 to this entry.]

Page A469, left column (9/23/89)

*`\language` (hyphenation method), 273, 455.

Page A469, right column (9/23/89)

*`\lefthyphenmin`, 273, *364*, 454.

Page A470, entry for ligatures (9/23/89)

[Add pages 286 and 444 to this entry.]

Page A472, left column (9/23/89)

*`\noboundary`, 283, 286, 290.

Page A473, right column (9/23/89)

overfull boxes, 27–30, 94, 229, 238,
302–303, 307, 400.
avoiding, 107.

Page A474, left column (9/23/89)

*`\patterns`, 277, 453, 455.

Page A476, left column (9/23/89)

*`\righthyphenmin`, 273, *364*, 454.

Page A476, right column (9/23/89)

*\setlanguage, 287, 455.

Page A476, right column (9/23/89)

*\showboxbreadth, 273, 302, 303, *348*.
*\showboxdepth, 79, 273, 302, 303, *348*.

Page A479, left column (9/23/89)

*\tolerance, *29–30*, 91, 94, 96, 107, 272,
317, *333*, *342*, *348*, *364*, *451*.

Page A481, right column, last six entries (9/23/89)

1/2, 67, 332.
1/2, in unslashed form, 141, 186.
⟨4-bit number⟩, 271.
⟨8-bit number⟩, 271, 276–278.
⟨15-bit number⟩, 271, 277, 289, 291.
⟨27-bit number⟩, 271, 289, 291.

Page A483, lines 15 and 21 (9/23/89)

[Delete these two lines, as TUG's address is no longer c/o AMS.]

Page Bvii, top two lines (4/21/89)

WEB documentation for four utility programs that are often used in conjunction with TEX: POOLtype, TFtoPL, PLtoTF, and DVItype.

Page B2, line 32 (6/20/89)

```
define banner ≡ 'This␣is␣TeX,␣Version␣2.991'   { printed when TeX starts }
```

Page B118, lines 2–4 (3/2/89)

```
begin if cur_level > level_one then
  begin check_full_save_stack; save_type(save_ptr) ← insert_token;
  save_level(save_ptr) ← level_zero; save_index(save_ptr) ← t; incr(save_ptr);
  end;
```

Page B182, line 13 becomes two lines (6/20/89)

```
k, kk: small_number;   { number of digits in a decimal fraction }
p, q: pointer;   { top of decimal digit stack }
```

Page B182, line 15 from the bottom (6/20/89)

```
begin k ← 0; p ← null; get_token;   { point_token is being re-scanned }
```

Page B182, line 11 from the bottom (6/20/89)

begin $q \leftarrow get\_avail$; $link(q) \leftarrow p$; $info(q) \leftarrow cur\_tok - zero\_token$; $p \leftarrow q$; $incr(k)$;

Page B182, line 8 from the bottom (6/20/89)

done1: **for** $kk \leftarrow k$ **downto** 1 **do**
begin $dig[kk-1] \leftarrow info(p)$; $q \leftarrow p$; $p \leftarrow link(p)$; $free\_avail(q)$;
end;
$f \leftarrow round\_decimals(k)$;

Page B332, lines 11 and 12 from the bottom (4/8/89)

begin if $cur\_align = null$ **then** $confusion(\text{`endv'})$;
$q \leftarrow link(cur\_align)$; **if** $q = null$ **then** $confusion(\text{`endv'})$;

Page B466, line 5 becomes three lines (6/7/89)

mmode + *halign*: **if** *privileged* **then**
if $cur\_group = math\_shift\_group$ **then** *init_align*
else *off_save*;

Page B518, line 25 (8/31/89)

$undump(lo\_mem\_stat\_max + 1)(lo\_mem\_max)(rover)$; $p \leftarrow mem\_bot$; $q \leftarrow rover$;

Volume C, in general (9/23/89)

[Change '127' to '255' and '128' to '256' in contexts referring to character codes. This happens on pages 188(thrice) and 251.]

Page C91, lines 12 and 13 (8/31/89)

```
\mode=cheapo; input newface
```

and the same file should also produce a high-resolution font if we start with

Page C204, line 4 (8/18/89)

so that *currenttransform* multiplies all y coordinates by *aspect_ratio*, when paths are

Page C212, lines 24–27 (9/30/89)

boundarychar the right boundary character for ligatures and kerns

All of these quantities are numeric. They are initially zero at the start of a job, except for *year*, *month*, *day*, and *time*, which are initialized to the time the run began; furthermore, *boundarychar* is initially -1. A *granularity* of zero is equivalent to *granularity* = 1. A preloaded base file like plain METAFONT will usually give nonzero values to several other internal quantities on this list.

Page C259, lines 16 and 17 from the bottom (5/14/89)

`screenchars`; `screenstrokes`; `imagerules`; `gfcorners`; `nodisplays`; `notransforms`; `input` ⟨filename⟩.

Page C282, the three lines following the chart (9/30/89)

METAFONT can also be configured to accept any or all of the character codes 128–255. However, METAFONT programs that make use of anything in addition to the 95 standard ASCII characters cannot be expected to run on other systems, so the use of extended character sets is discouraged.

Page C316, bottom 14 lines and top 30 of page C317 (9/30/89)

Ligature information and kerning information is specified in short "ligtable programs" of a particularly simple form. Here's an example that illustrates most of the features (although it is not a serious example of typographic practice):

```
ligtable "f": "f" =: oct"013", "i" |=: oct"020", skipto 1;
ligtable "o": "b": "p": "e" kern .5u#, "o" kern .5u#, "x" kern-.5u#,
          1:: "!" kern u#;
```

This sequence of instructions can be paraphrased as follows:

> Dear TEX, when you're typesetting an 'f' with this font, and when the following character also belongs to this font, look at it closely because you might need to do something special: If that following character is another 'f', replace the two f's by character code `oct"013"` [namely 'ff']; if it's an 'i', retain the 'f' but replace the 'i' by character code `oct"020"` [a dotless 'ı']; otherwise skip down to label '`1::`' for further instructions. When you're typesetting an 'o' or 'b' or 'p', if the next input to TEX is 'e' or 'o', add a half unit of space between the letters; if it's an 'x', subtract a half unit; if it's an exclamation point, add a full unit. The last instruction applies also to exclamation points following 'f' (because of the label '`1::`').

When a character code appears in front of a colon, the colon "labels" the starting place for that character's ligature and kerning program, which continues to the end of the ligtable statement. A double colon denotes a "local label"; a `skipto` instruction advances to the next matching local label, which must appear before 128 ligtable steps intervene. The special label `||:` can be used to initiate ligtable instructions for an invisible "left boundary character" that is implicitly present just before every word; an invisible "right boundary character" equal to *boundarychar* is also implicitly present just after every word, if *boundarychar* lies between 0 and 255.

The general syntax for ligtable programs is pretty easy to guess from these examples, but we ought to exhibit it for completeness:

> ⟨ligtable command⟩ ⟶ `ligtable` ⟨ligtable program⟩⟨optional skip⟩
> ⟨ligtable program⟩ ⟶ ⟨ligtable step⟩ | ⟨ligtable program⟩ `,` ⟨ligtable step⟩
> ⟨optional skip⟩ ⟶ `,` `skipto` ⟨code⟩ | ⟨empty⟩

⟨ligtable step⟩ ⟶ ⟨code⟩⟨ligature op⟩⟨code⟩
| ⟨code⟩ `kern` ⟨numeric expression⟩
| ⟨label⟩⟨ligtable step⟩
⟨ligature op⟩ ⟶ `=:` | `|=:` | `|=:>` | `=:|` | `=:|>` | `|=:|` | `|=:|>` | `|=:|>>`
⟨label⟩ ⟶ ⟨code⟩ `:` | ⟨code⟩ `::` | `||:`
⟨code⟩ ⟶ ⟨numeric expression⟩ | ⟨string expression⟩

A ⟨code⟩ should have a numeric value between 0 and 255, inclusive, after having been rounded to the nearest integer; or it should be a string of length 1, in which case it denotes the corresponding ASCII code (Appendix C). For example, `"A"` and `64.61` both specify the code value 65. Vertical bars to the left or right of '`=:`' tell TEX to retain the original left and/or right character that invoked a ligature. Additional '`>`' signs tell TEX to advance its focus of attention instead of doing any further ligtable operations at the current character position.

Page C338, lines 21 and 22 (9/30/89)

---

and 127–255 have to be specified with the '`#`' option, on non-fancy installations of TEX, and so does code 35 (which is the ASCII code of '`#`' itself).

Page C346, left column, after line 14 (9/30/89)

---

*`|=:`, *316*, 317.
*`|=:>`, 317.
*`=:|`, 317.
*`=:|>`, 317.
*`|=:|`, 317.
*`|=:|>`, 317.
*`|=:|>>`, 317.

Page C346, left column, after line 31 (9/30/89)

---

*`::` (local label), 317.
*`||:` (left boundary label), 317.

Page C347, left column (9/30/89)

---

*`boundarychar`, 212, 317.

Page C352, left column (9/30/89)

[Change '⟨ligature replacement⟩' to '⟨ligature op⟩'.]

Page C354, left column (9/30/89)

⟨optional skip⟩, 217.

Page C356, left column (9/30/89)

---

*`skipto`, *316*, 317.

Page Dvi, bottom two lines, and top lines of page vii (4/21/89)

■ "METAFONTware" by Donald E. Knuth, Tomas G. Rokicki, and Arthur L. Samuel, Stanford Computer Science Report 1255 (Stanford, California, April 1989), 207 pp. *The* `WEB` *programs for four utility programs that are often used in conjunction with METAFONT:* `GFtype`*,* `GFtoPK`*,* `GFtoDVI`*, and* `MFT`*.*

Page D63, line 9 (8/31/89)

mem, so we allow pointers to assume any *halfword* value. The minimum memory index represents

Page D63, line 28 (8/31/89)

$$null = mem\_min < lo\_mem\_max < hi\_mem\_min < mem\_top \leq mem\_end \leq mem\_max.$$

Page D67, in the July 1987 printing (4/7/89)

[Delete line 7, which has a redundant '**if** $r = p$ **then**'; move line 8 to the left 10 points for alignment; and restore the following line (which was deleted by mistake after line 8):

$node\_size(p) \leftarrow q - p$ { reset the size in case it grew }

These corrections are needed only in the reprinting made July, 1987.]

Page D228, in the July 1987 printing (4/7/89)

[Delete lines 14–15, which were inserted erroneously from a previous errata list; and restore the following lines (which were deleted by mistake):

begin $double(max\_coef)$; $double(x0)$; $double(x1)$; $double(x2)$;
$double(y0)$; $double(y1)$; $double(y2)$;
end

These corrections are needed only in the reprinting made July, 1987.]

Page D248, in the July 1987 printing (4/7/89)

[Delete line 16, which begins with '$d \leftarrow take\_fraction$'; and restore the following line (which was deleted by mistake after line 22):

if $d < alpha$ **then** $d \leftarrow alpha$

These corrections are needed only in the reprinting made July, 1987.]

Page D389, line 10 (6/20/89)

help1 (´`The␣expression␣above␣should␣have␣been␣a␣number␣>=3/4.`´);

Page D504, line 25 (8/31/89)

$undump(lo\_mem\_stat\_max + 1)(lo\_mem\_max)(rover)$; $p \leftarrow mem\_min$; $q \leftarrow rover$;

Page D510, in the July 1987 printing (4/7/89)

[Move the 7th-to-last line, which begins with '*internal*[*fontmaking*]', one line down, and indent it to the right by 10 more points. This correction is needed only in the reprinting made July, 1987.]

Page Exiii, bottom four lines (5/5/89)

▪ "Metamarks: Preliminary studies for a Pandora's Box of shapes" by Neenie Billawala, Stanford Computer Science Report 1256 (Stanford, California, May 1989), 132 pp. *Lavishly illustrated studies in parameter variation, leading to the design of a new family of typefaces called Pandora.*

Page E401, bottom line (5/16/89)

math_fit$(-.3cap\_height\# * slant - .5u\#, ic\#)$;
penlabels$(1, 2, 3, 4, 5, 6, 7, 8)$; **endchar**;

[some points and labels are missing at the tip of the tail on page 400]