This is a list of all corrections made to *Computers & Typesetting*, Volumes A, B, C, and D, between 1 January 1991 and 15 March 1992. 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 the corrections below have already been made in reprintings of the books. Changes to Volume B refer to the fourth printing (1991), which differs markedly from earlier printings because it includes all the revisions for TEX3.0. Changes to Volume D refer to the third printing (1991), which differs markedly from earlier printings because it includes all the revisions for METAFONT2.0. Changes to the mini-indexes and master indexes of Volumes B and D are not shown here unless they are not obviously derivable from what has been shown. Dozens of changes, too many to list here, have been made to Volume E because of recent upgrades to the Computer Modern font source files. Those changes, which affect only the digitization at low resolution and the appearance of lowercase delta and some characters in the math symbols fonts (but not the `TFM` files), are documented at the end of file `cm85.bug`.

Page A96, lines 9–11 (9/18/91)

Some German words traditionally change their spelling when they are split between lines. For example, 'backen' becomes 'bak-ken' and 'Bettuch' becomes 'Bett-tuch'. How can you instruct TEX to produce such effects?

Page A178, line 17 (11/19/91)

If you say '`\phantom{`⟨subformula⟩`}`' in any formula, plain TEX will do its

---

Page A286, bottom two lines and continuing into A287 (11/21/91)

---

stands for zero or more ⟨assignment⟩ commands other than `\setbox`. If the assignments are not followed by a ⟨character⟩, where ⟨character⟩ stands for any of the commands just discussed in the previous paragraph, TEX treats `\accent` as if it were `\char`, except that the space factor is set to 1000. Otherwise the character that follows the assignment is accented by the character that corresponds to the ⟨8-bit number⟩. (The purpose of the intervening assignments is to allow the accenter and accentee to be in different fonts.) If the accent must be moved up or down, it is put into an hbox that is raised or lowered. Then the accent is effectively superposed on the character by means of kerns, in such a way that the width of the accent does not influence the width of the resulting horizontal list. Finally, TEX sets `\spacefactor=1000`.

Page A291, lines 6–8 (11/21/91)

'`}`' may be followed by optional ⟨assignment⟩ commands other than `\setbox`, after which '`$$`' must conclude the display. TEX will insert the `\abovedisplayskip` and `\belowdisplayskip` glue before and after the result of the alignment.

Page A293, line 14 (9/18/91)

explained in Appendix G. TEX scans ⟨one optional space⟩ after completing a displayed formula; this is usually the implicit space at the end of a line in the input file.

Page A311, bottom four lines (9/18/91)

12.7. 1000, except: 999 after `O`, `B`, `S`, `D`, and `J`; 1250 after the comma; 3000 after the exclamation point, the right-quote marks, and the periods. If a period had come just after the `B` (i.e., if the text had said '`B. Sally`'), the space factor after that period would have been 1000, not 3000.

Page A314, lines 16–18 from the bottom (1/10/92)

14.8. `ba\ck/en` and `Be\ttt/uch`, where the macros `\ck/` and `\ttt/` are defined by

```
\def\ck/{\discretionary{k-}{k}{ck}}
\def\ttt/{tt\discretionary{-}{t}{}}
```

Page A354, line 8 (9/18/91)

```
\def\multispan#1{\omit\mscount=#1\relax\loop\ifnum\mscount>1 \sp@n\repeat}
```

Page A356, line 11 from the bottom (9/23/91)

```
  \else{\ooalign{\unhbox0\crcr\hidewidth\char'30\hidewidth}}\fi}
```

Page A358, line 8 from the bottom (9/18/91)

```
\mathchardef\mapstochar="3237 \def\mapsto{\mapstochar\rightarrow}
```

Page A359, line 13 (11/4/91)

```
\def\overrightarrow#1{\vbox{\m@th\ialign{##\crcr
```

Page A359, line 16 (11/4/91)

```
\def\overleftarrow#1{\vbox{\m@th\ialign{##\crcr
```

Page A359, line 19 (11/4/91)

```
\def\overbrace#1{\mathop{\vbox{\m@th\ialign{##\crcr\noalign{\kern3pt}
```

Page A359, line 22 (11/4/91)

```
\def\underbrace#1{\mathop{\vtop{\m@th\ialign{##\crcr
```

Page A359, lines 7–14 from the bottom (1/11/92)

```
\def\lgroup{\delimiter"462833A }      \def\rgroup{\delimiter"562933B }
\def\lmoustache{\delimiter"437A340 }  \def\rmoustache{\delimiter"537B341 }
\def\uparrow{\delimiter"3222378 }     \def\Uparrow{\delimiter"322A37E }
\def\downarrow{\delimiter"3223379 }   \def\Downarrow{\delimiter"322B37F }
\def\updownarrow{\delimiter"326C33F } \def\arrowvert{\delimiter"026A33C }
\def\Updownarrow{\delimiter"326D377 } \def\Arrowvert{\delimiter"026B33D }
\def\vert{\delimiter"026A30C }        \def\Vert{\delimiter"026B30D }
\def\backslash{\delimiter"026E30F }   \def\bracevert{\delimiter"077C33E }
```

Page A360, line 13 (11/19/91)

`\phantom`, `\smash`, `\root`, and other operations. (Actually `\phantom` and `\smash` are not perfect: They assume that the current style is uncramped.)

Page A360, line 2 from the bottom (11/4/91)

```
\def\c@ncel#1#2{\m@th\ooalign{$\hfil#1\mkern1mu/\hfil$\crcr$#1#2$}}
```

Page A361, top line (11/4/91)

```
\def\rlh@#1{\vcenter{\m@th\hbox{\ooalign{\raise2pt
```

Page A364, line 5 from the bottom (11/4/91)

```
\def\fmtname{plain}\def\fmtversion{3.141}
```

Page A377, the bottom 17 lines (9/18/91)

story: Macro `\stest` decides whether or not a given token list register begins with a ⟨space token⟩ as defined in Chapter 24. If so, the macro decides whether the token is explicit and/or funny and/or active.

```
\newif\ifspace \newif\iffunny \newif\ifexplicit \newif\ifactive
\def\stest#1{\funnyfalse \expandafter\s\the#1! \stest}
\def\s{\global\explicitfalse \global\activefalse \futurelet\next\ss}
\def\ss{\ifcat\noexpand\next\stoken\let\nxt\sx\else\let\nxt\ns\fi\nxt}
\def\sx{\spacetrue\ifx\next\stoken\let\nxt\sss\else\let\nxt=\ssss\fi\nxt}
\long\def\sss#1 #2\stest{\def\next{#1}%
  \ifx\next\empty \global\explicittrue \else\testactive#1\s\fi}
\long\def\ssss#1#2\stest{\funnytrue{\escapechar=\if*#1‘?\else‘*\fi\relax
  \if#1\string#1\uccode‘#1=‘~ % we assume that ~ is an active character
    \uppercase{\ifcat\noexpand#1}\noexpand~\global\activetrue
    \else\global\explicittrue\fi
  \else\testactive#1\s\fi}}
\long\def\ns#1\stest{\spacefalse}
\long\def\testactive#1#2\s{\expandafter\tact\string#1\s\tact}
\long\def\tact#1#2\tact{\def\next{#2}\ifx\next\xs\global\activetrue
  \else\ifx\next\empty \global\activetrue\fi\fi} \def\xs{\s}
```

Page A444, lines 15–26 (3/26/91)

14. If the current item is an Ord atom, go directly to Rule 17 unless all of the following are true: The nucleus is a symbol; the subscript and superscript are both empty; the very next item in the math list is an atom of type Ord, Op, Bin, Rel, Open, Close, or Punct; and the nucleus of the next item is a symbol whose family is the same as the family in the present Ord atom. In such cases the present symbol is marked as a text symbol. If the font information shows a ligature between this symbol and the following one, using the specified family and the current size, then insert the ligature character and continue as specified by the font; in this process, two characters may collapse into a single Ord text symbol, and/or new Ord text characters may appear. If the font information shows a kern between the current symbol and the next, insert a kern item following the current atom. As soon as an Ord atom has been fully processed for ligatures and kerns, go to Rule 17.

Page A446, lines 5 and 6 from the bottom (1/13/92)

are used to change the current style just as in the first pass, so that both passes have the same value of C when they work on any particular atom.

Page A447, in the parameter usage table (1/13/92)

[Delete the entry for 'σ_2'; the entry for 'σ_{17}' moves down to the bottom of the left column.]

Page A447, line 2 after the parameter usage table (1/13/92)

to parameters in arbitrary families: Rule 17 uses `\fontdimen` parameter 2 (space) to de-

Page A467, entry for `\hss` (9/18/91)

*`\hss`, 71–72, *82–83*, 233, 283, 285, 290, 442.

Page A467, new subentry under hyphenation (9/18/91)

suppressing, 93, 414, 424, 454.

Page A476, right column (11/21/91)

*`\setbox`, 66–67, 77, 81, 120, 276, 279, 286, 291, *386–392*.

Page B2, line 10 from the bottom (1/11/92)

define *banner* ≡ ´`This␣is␣TeX,␣Version␣3.141`´ { printed when TEX starts }

Page B18, lines 21 and 22 (10/12/91)

must have an *xchr* equivalent in the local character set. (This restriction applies only to preloaded strings, not to those generated dynamically by the user.)

Page B26, new line before fourth line from bottom (1/24/92)

```
nl: integer;  { new-line character to restore }
```

Page B26, bottom line and top 3 lines of B27 (1/24/92)

```
else begin if selector > pseudo then
    begin print_char(s); return;  { internal strings are not expanded }
    end;
  if (⟨ Character s is the current new-line character 244 ⟩) then
    if selector < pseudo then
      begin print_ln; return; end;
  nl ← new_line_char; new_line_char ← −1;  { temporarily disable new-line character }
  j ← str_start[s];
  while j < str_start[s + 1] do
    begin print_char(so(str_pool[j])); incr(j); end;
  new_line_char ← nl; return;
  end;
```

Page B27, lines 9 and 10 (9/19/91)

60. Control sequence names, file names, and strings constructed with `\string` might contain *ASCII_code* values that can't be printed using *print_char*. Therefore we use *slow_print* for them:

Page B27, lines 13–26 (1/24/92)

```
var j: pool_pointer;  { current character code position }
begin if (s ≥ str_ptr) ∨ (s < 256) then print(s)
else begin j ← str_start[s];
  while j < str_start[s + 1] do
    begin print(so(str_pool[j])); incr(j);
    end;
  end;
end;
```

Page B28, line 8 (9/19/91)

```
else begin slow_print(format_ident); print_ln;
```

Page B33, line 3 (1/11/92)

recursively. A similar interlock is provided by *set_box_allowed*.

Page B33, new line to come after line 14 (1/11/92)

```
set_box_allowed: boolean;  { is it safe to do a \setbox assignment? }
```

Page B33, new line to come after line 20 (1/11/92)

```
set_box_allowed ← true;
```

Page B36, line 12 (9/19/91)

```
begin print_nl("You␣want␣to␣edit␣file␣"); slow_print(input_stack[base_ptr].name_field);
```

Page B46, lines 9 and 10 (5/24/91)

arithmetic; see *TUGboat* **3**,1 (March 1982), 10–27. (But the routines cited there must be modified to allow negative glue ratios.)

Page B47, lines 2 and 3 (5/24/91)

structures on a *memory_word*, which contains either a (signed) integer, possibly scaled, or a (signed) *glue_ratio*, or a small number of fields that are one half or one quarter of the size used

Page B177, lines 10 and 11 (9/19/91)

```
begin print_err("Bad␣mathchar");
help2("A␣mathchar␣number␣must␣be␣between␣0␣and␣32767.")
```

Page B196, new lines after line 11 (1/13/92)

```
if align_state < 1000000 then   { unmatched '}' aborts the line }
  begin repeat get_token; until cur_tok = 0;
  align_state ← 1000000; goto done;
  end;
```

Page B208, line 21 (9/19/91)

```
begin slow_print(a); slow_print(n); slow_print(e);
```

Page B214, line 14 (9/19/91)

```
begin wlog(banner); slow_print(format_ident); print("␣␣"); print_int(day); print_char("␣");
```

Page B214, line 2 from the bottom (9/19/91)

```
print_char("("); incr(open_parens); slow_print(name); update_terminal; state ← new_line;
```

Page B234, line 22 (9/19/91)

```
print("␣in␣font␣"); slow_print(font_name[f]); print_char("!"); end_diagnostic(false);
```

Page B267, lines 7 and 8 (9/19/91)

```
print_nl("Output␣written␣on␣"); slow_print(output_file_name);
print("␣("); print_int(total_pages); print("␣page");
```

Page B296, new lines after line 8 of section 716 (1/11/92)

```
if f < 0 then
  begin decr(n); f ← f + ´200000;
  end;
```

Page B297, new lines after line 7 of section 717 (1/11/92)

```
if f < 0 then
  begin decr(n); f ← f + ´200000;
  end;
```

Page B348, bottom two lines (1/3/92)

Up to three passes might be made through the paragraph in an attempt to find at least one set of feasible breakpoints. On the first pass, we have *threshold* = *pretolerance* and *second_pass* =

Page B364, line 20 (1/3/92)

863. The '**loop**' in the following code is performed at most thrice per call of *line_break*, since

Page B377, insert new line after line 12 (9/19/91)

```
hyf_bchar: halfword;  { boundary character after c_n }
```

Page B378, line 12 from the bottom (9/19/91)

```
hyf_bchar ← character(s); c ← qo(hyf_bchar);
```

Page B378, line 9 from the bottom (1/10/92)

```
hb ← s; incr(hn); hu[hn] ← c; hc[hn] ← lc_code(c); hyf_bchar ← non_char;
```

Page B378, line 5 from the bottom (9/19/91)

```
else if (type(s) = kern_node) ∧ (subtype(s) = normal) then hb ← s
  else goto done3;
```

Page B379, line 6 (9/19/91)

```
j ← hn; q ← lig_ptr(s); if q > null then hyf_bchar ← character(q);
```

Page B379, new line between lines 14 and 15 (1/10/92)

```
if odd(subtype(s)) then hyf_bchar ← font_bchar[hf] else hyf_bchar ← non_char;
```

Page B379, line 19 (9/19/91)

```
if hn < l_hyf + r_hyf then goto done1;    { l_hyf and r_hyf are always ≥ 1 }
```

Page B380, lines 9–11 from the bottom reduce to a single line (1/10/92)

```
q ← link(hb); link(hb) ← null; r ← link(ha); link(ha) ← null; bchar ← hyf_bchar;
```

Page B436, lines 9 and 10 (3/15/92)

$$cur\_r = \begin{cases} character(lig\_stack), & \text{if } lig\_stack > null; \\ font\_bchar[cur\_font], & \text{otherwise;} \end{cases}$$

except when $character(lig\_stack) = font\_false\_bchar[cur\_font]$. Several additional global variables are needed.

Page B438, line 13 from the bottom (3/15/92)

cur_q ← *tail*; *cur_l* ← *character*(*lig_stack*);

Page B507, line 6 of section 1241 (1/11/92)

```
scan_optional_equals;
if set_box_allowed then scan_box(box_flag + n)
else begin print_err("Improper␣"); print_esc("setbox");
  help2("Sorry,␣\setbox␣is␣not␣allowed␣after␣\halign␣in␣a␣display,")
  ("or␣between␣\accent␣and␣an␣accented␣character."); error;
  end;
```

Page B511, new line inserted after line 3 (1/24/92)

flushable_string: *str_number*; { string not yet referenced }

Page B512, new line inserted after line 3 of section 1260 (1/24/92)

flushable_string ← *str_ptr* − 1;

Page B512, the former line 6 of section 1260 (1/24/92)

```
    begin if cur_name = flushable_string then
      begin flush_string; cur_name ← font_name[f]; end;
    if s > 0 then
```

Page B512, line 10 from the bottom (9/19/91)

set_font: **begin** *print*("select␣font␣"); *slow_print*(*font_name*[*chr_code*]);

Page B514, line 9 (1/11/92)

set_box_allowed ← *false*; *prefixed_command*; *set_box_allowed* ← *true*;

Page B515, line 19 (9/19/91)

slow_print(*s*); *update_terminal*;

Page B516, line 2 (9/19/91)

begin *print_err*(""); *slow_print*(*s*);

Page B531, lines 19 and 20 (9/19/91)

print_nl(`"Beginning␣to␣dump␣on␣file␣"`); *slow_print*(*w_make_name_string*(*fmt_file*)); *flush_string*; *print_nl*(`""`); *slow_print*(*format_ident*)

Page B533, line 29 (9/19/91)

begin *print_nl*(`"Transcript␣written␣on␣"`); *slow_print*(*log_name*); *print_char*(`"."`);

Page B538, line 13 (9/19/91)

10: *slow_print*(*n*);

Page B577, left column (12/23/91)

[Add 798 to the index entries for 'system dependencies'.]

Page C262, line 15 (3/26/91)

```
string base_name, base_version; base_name="plain"; base_version="2.7";
```

Page C271, line 17 from the bottom (3/26/91)

```
  currentpen_path shifted (z.t_) withpen penspeck enddef;
```

Page C347, Bront"e entry (1/29/91)

[The accent was clobbered; her name should, of course, be Brontë. Fix the entries for Dürer, Möbius, and Stravinsky in the same way.]

Page C348, left column (1/11/92)

Page C353, right column (1/11/92)

Page C354, miscellaneous entries in both columns (1/11/92)

Page C356, right column (1/11/92)

Page C357, right column (1/11/92)

Page D2, last line of section 2 (1/24/92)

define *banner* ≡ ´This␣is␣METAFONT,␣Version␣2.71´ { printed when METAFONT starts }

Page D102, line 15 from the bottom (11/1/91)

Then $eq\_type(h(x)) = tag\_token$ and $equiv(h(x)) = p$, where p is a two-word value node with

Page D188, lines 16 and 17 (1/24/92)

errors. Our subroutines also obey the identity $t[a, b] + t[b, a] = a + b$.

Page D190, new copy before bottom four lines (1/24/92)

if $x\_coord(r) < x\_coord(pp)$ **then** $x\_coord(r) \leftarrow x\_coord(pp)$
else if $x\_coord(r) > dest\_x$ **then** $x\_coord(r) \leftarrow dest\_x$;
if $left\_x(r) > x\_coord(r)$ **then**
  begin $left\_x(r) \leftarrow x\_coord(r)$; **if** $right\_x(pp) > x\_coord(r)$ **then** $right\_x(pp) \leftarrow x\_coord(r)$; **end**;
if $right\_x(r) < x\_coord(r)$ **then**
  begin $right\_x(r) \leftarrow x\_coord(r)$; **if** $left\_x(qq) < x\_coord(r)$ **then** $left\_x(qq) \leftarrow x\_coord(r)$; **end**;

Page D191, new copy before bottom two lines of section 416 (1/24/92)

if $x\_coord(s) < x\_coord(r)$ **then** $x\_coord(s) \leftarrow x\_coord(r)$
else if $x\_coord(s) > dest\_x$ **then** $x\_coord(s) \leftarrow dest\_x$;
if $left\_x(s) > x\_coord(s)$ **then**
  begin $left\_x(s) \leftarrow x\_coord(s)$; **if** $right\_x(r) > x\_coord(s)$ **then** $right\_x(r) \leftarrow x\_coord(s)$; **end**;
if $right\_x(s) < x\_coord(s)$ **then**
  begin $right\_x(s) \leftarrow x\_coord(s)$; **if** $left\_x(qq) < x\_coord(s)$ **then** $left\_x(qq) \leftarrow x\_coord(s)$; **end**;

Page D194, lines 4 and 5 (1/24/92)

[Delete those two lines; I no longer believe that the assertion has been proved (although it might be true).]

Page D194, lines 7–13 of section 424 (1/24/92)

if $y\_coord(r) < y\_coord(p)$ **then** $y\_coord(r) \leftarrow y\_coord(p)$
else if $y\_coord(r) > dest\_y$ **then** $y\_coord(r) \leftarrow dest\_y$;
if $x\_coord(p) + y\_coord(r) > dest\_x + dest\_y$ **then** $y\_coord(r) \leftarrow dest\_x + dest\_y - x\_coord(p)$;
if $left\_y(r) > y\_coord(r)$ **then**
  begin $left\_y(r) \leftarrow y\_coord(r)$; **if** $right\_y(p) > y\_coord(r)$ **then** $right\_y(p) \leftarrow y\_coord(r)$; **end**;
if $right\_y(r) < y\_coord(r)$ **then**
  begin $right\_y(r) \leftarrow y\_coord(r)$; **if** $left\_y(q) < y\_coord(r)$ **then** $left\_y(q) \leftarrow y\_coord(r)$; **end**;

Page D194, lines 8–11 from the bottom (1/24/92)

if $right\_y(r) < y\_coord(r)$ **then**
  begin $right\_y(r) \leftarrow y\_coord(r)$; **if** $left\_y(q) < y\_coord(r)$ **then** $left\_y(q) \leftarrow y\_coord(r)$; **end**;

Page D195, lines 3–9 of section 425 (1/24/92)

```
if y_coord(s) < y_coord(r) then y_coord(s) ← y_coord(r)
else if y_coord(s) > dest_y then y_coord(s) ← dest_y;
if x_coord(r) + y_coord(s) > dest_x + dest_y then y_coord(s) ← dest_x + dest_y − x_coord(r);
if left_y(s) > y_coord(s) then
  begin left_y(s) ← y_coord(s); if right_y(r) > y_coord(s) then right_y(r) ← y_coord(s); end;
if right_y(s) < y_coord(s) then
  begin right_y(s) ← y_coord(s); if left_y(q) < y_coord(s) then left_y(q) ← y_coord(s); end;
```

Page D195, lines 3–7 from the bottom if section 425 (1/24/92)

```
if right_y(s) < y_coord(s) then
  begin right_y(s) ← y_coord(s); if left_y(q) < y_coord(s) then left_y(q) ← y_coord(s); end;
```

Page D289, lines 9 and 10 (11/1/91)

```
  p ← dep_list(p); r ← inf_val;
  repeat if value(info(p)) ≥ value(r) then
```

Page D486, line 18 (11/1/91)

The *label_loc* and *label_char* arrays have been set up to record all the starting addresses; we have