*Editor's note*: The following note announced the periodic update of TEX and friends on March 18, 2008. (In this presentation, some of the verbatim code lines have been re-broken or otherwise adjusted to fit the narrow columns.) Work is underway to include the new versions in the next release of TEX Live. Unless you have experience in implementing the software, it's advisable to wait for that release. Please refrain from posing questions or reporting problems, to allow the implementors to pursue their work without interruptions. If you want to find out the details of the updates, those can be found on CTAN in the area `systems/knuth/errata/`.

# The TEX tuneup of 2008

Donald Knuth

I've written this note while going through the long, long file of bug reports and suggestions that were submitted during the years 2003–2007. You know that I am committed to keeping TEX and METAFONT as stable as possible, while also correcting serious blunders that are likely to be harmful if left as is. It is certainly not always obvious where to draw the line; I intend to keep drawing it as close to the existing implementations as I can, without feeling extremely guilty.

The index to *Digital Typography* lists eleven pages where the importance of stability is stressed, and I urge all maintainers of TEX and METAFONT to read them again every few years. Any object of nontrivial complexity is non-optimum, in the sense that it can be improved in some way (while still remaining non-optimum); therefore there's always a reason to change anything that isn't trivial. But one of TEX's principal advantages is the fact that it does not change — except for serious flaws whose correction is unlikely to affect more than a very tiny number of archival documents.

Let me give two examples. First, David Kastrup observes that TEX doesn't do the best possible rounding when it converts units. One inch is exactly 72.27 points, which is exactly 4736286.72 scaled points. When you say '`1in`', TEX converts it to 4736286sp; when you say '`72.27pt`', TEX converts it to 4736287sp, which is about 23.6 Ångstrom units closer to the truth. With a simple change to TEX §458, namely to add '`denom div 2`' before dividing by '`denom`', the rounding would be slightly better. But that would invalidate the line-break and page-break decisions of an enormous number of documents. It's unthinkable to change TEX in such a way today.

But of course the authors of other systems should adopt superior methods when they want to.

Second, I recently installed METAPOST version 0.993, which corrected a bug in the calculation of the bounding box of its outputs. I'm a user of METAPOST, not a developer; but I'm sort of glad that the developers had fixed this bug. On the other hand it was a tremendous headache for me, because it affected nearly 200 of the illustrations for *The Art of Computer Programming*, and caused severe changes to the layouts of more than a dozen pages, even though the individual corrections to the box sizes were typically 2pt or less! I spent three days going over everything so that I could once again typeset the volumes of my main life's work. I couldn't reasonably insist that the METAPOST developers retain such a serious bug as a "feature". With TEX, on the other hand, it's a different story, because people's accumulated investment in TEX documents is more than a million times the total current investment in METAPOST documents. If a comparable bug had showed up in TEX, I would *not* have changed it.

Let me also observe that I never intended TEX to be immune to vicious "cracker attacks"; I only wish it to be robust under reasonable use by people who are trying to get productive work done. Almost every limit can be abused in extreme cases, and I don't think it useful to go to extreme pain to prevent such things. Computers have general protection mechanisms to keep buggy software from inflicting serious damage; TEX and METAFONT are far less buggy than the software for which such mechanisms were designed. For instances of the philosophy that I had while writing these programs, see for instance TEX §9 and MF §9, which say that I expected the programs to be run with arithmetic overflow interrupt turned on; also TEX §104: "TEX does not check for overflow when dimensions are added or subtracted ... the chance of overflow is so remote that such tests do not seem worthwhile"; MF §369 says that the total weight in a picture "will be less than $2^{31}$ unless the edge structure has more than 174,762 edges"; MF §558, "we shall assume that the coordinates are sufficiently non-extreme"; MF §930, "users aren't supposed to be monkeying around with really big values."

## A proposal re file errors

I think the following change would be nice for the next versions of TEX, METAFONT, etc.: In place of the current message

```
Please type another %s file name:
```

produced by `prompt_file_name`, let's substitute

```
Please type another %s file name (or quit):
```

and then if the user's response is 'quit' we do the equivalent of control-C. If the response is null, let's give a help message.

This modification should be handled by change files, keeping the master files `tex.web` and `mf.web` and `whatever.web` as they are. I never have intended to control the aspects of user interaction on particular systems.

Maybe also introduce a finite loop, with '`(or quit)`' replaced by '`(or I'll quit)`' the third or fourth time. I agree that infinite loops are evil, and I'm sorry that `prompt_file_name` is invoked only within infinite loops in my own programs. If I had thought of this idea earlier, I'd have added a global variable like `max_prompt_repeats`, and initialized it to 3 or 4 just before those infinite loops; then `prompt_file_name` would decrement it, or give up if it's zero.

Another possibility is '`(or quit or retry)`', except the last time. That wording is a bit more suited to computer geeks, who have ideas about fixing things by repairing file permissions, etc.; if the user responds with either 'retry' or null, the intention is clearly to try again because of some reason to hope for success. Still, I prefer the non-geek version, because it reaches more people and enables the null-for-help option. Let the geeks type a few more keystrokes — they get satisfaction in other ways.

## TEX

TEX version 3.1415926 corrects a few minor bugs, following major studies by David Fuchs. A summary of the noteworthy changes to the Pascal code in `tex.web` can be found near the end of the (long) file `errata/tex82.bug`. Here are the most significant ones, in decreasing order of importance:

1. Leaders with `\mskip` glue never worked properly; this feature has now been disallowed.

2. Error recovery was incorrect when an extra right brace appeared within a macro parameter.

3. TEX's inner loop now runs a bit faster.

4. The size of insert boxes is now displayed more accurately by `\showlists`.

5. A restriction on TFM files enforced by TFtoPL (namely that there must be at least one entry in each of the width, height, depth, and italic correction tables) is now enforced also by TEX, since noncompliance could cause a mess.

6. TEX used to leak four words of memory if arithmetic overflow occurred when `\multiply` or `\divide` was applied to `glue` or `muglue`.

7. The old `iniTeX` could leak four words of memory in another way (but at most four total), if "`last_glue`" pointed to a glue specification when the format file was created.

There's an undocumented feature, which is inconvenient to explain anywhere in *The TeXbook*: `\pagedepth` is cleared to zero when the current page disappears into `\box255`; but `\pagetotal`, `\pagestretch`, `\pagefilstretch`, `\pagefillstretch`, `\pagefilllstretch`, and `\pageshrink` are zeroed later, when the current page becomes nonempty. (That's the time `\pagegoal` is set, and recorded in the log file with a `%%` line if you're tracing pages.) I don't recall why there is a discrepancy, but I certainly don't want to diddle with any of that logic at this late date.

Here are some other things that I don't want to touch:

i. David Kastrup found a glitch in plain TeX's footnote-splitting mechanism. Everything works according to the documentation in *The TeXbook* and I can't possibly make a change to such a sensitive part of TeX's logic at this late date. But his example is quite interesting, and I'd like to discuss it here for the benefit of people planning other systems.

Here's his construction (to be used with plain TeX):

```
\def\testpage#1{\dimen0=#1
  \vrule height .5\dimen0 depth .5\dimen0
  \quad #1\par
  Some text.\footnote*{A bigbreak follows...
                       \bigbreak
                       A bigbreak preceded.}
  \par\vfill\supereject}
\testpage{8.17in}
\testpage{8.23in}
\testpage{8.2in}
```

The first test page is an example where the entire footnote fits fine. In the second one, the footnote needs to be split; so two pages are generated, one with the first half of the footnote, as desired.

The third test page illustrates the problem: Plain TeX uses the worst of both strategies! Namely, it generates two pages, in which the first is underfull, while the second has the text and footnote that would have fit on the first page.

Why does plain TeX screw up here? Well, TeX knows that the footnote doesn't fit, when typeset at its natural height+depth of 36pt. So it tries to split it, by choosing a height threshold: It says to the `vsplit` routine, "Please give me your best break that doesn't exceed a height of 30.089pt." (That is what's left after we start with plain TeX's vsize of 8.9in and subtract the page-total-so-far, which is 8.2in for the `vrule`, plus 1pt of `lineskip`, plus 7.5pt for the height of 'Some text.', plus 12pt to separate the text from its first footnote.) The `vsplit` algorithm discovers two ways to break the footnote: One has height 8.5pt (the height of '* A bigbreak follows...'), depth 1.94444pt, and penalty −200 (at the `bigbreak`); the other has natural height 32.5pt, depth 3.5pt (which comes from a strut placed by plain TeX), and penalty −10000 (the force-out penalty at the very bottom of the footnote). This latter break is considered viable because 4pt of glue shrinkage is available to bring the height down to 30.089pt. Naturally `vsplit` chooses the latter alternative.

Then TeX does something dumb. It records the result of the split in the list of contributions to the current page, in such a way that the first part of the split will be included on the page only if there's room for its natural height+depth, namely 36pt in this case. (And in this case, the "first part of the split" actually turns out to be the whole footnote.) Therefore, when TeX next finds a legal breakpoint, the current page limit has been exceeded, and the line with its footnote is deemed not to be permissible. The previous break, which leaves an underfull vbox, is chosen instead of "overfilling" the page — even though there is really enough shrinkability to bring the page back to size.

As I said, it's too late now to correct my age-old faulty reasoning. If I'd known about the problem twenty years ago, I may well have decided to make the change that seems most appropriate to me today, which is this:

```
@x module 974
    best_height_plus_depth:=cur_height+prev_dp;
@y
    best_height_plus_depth:=cur_height+prev_dp;
    if (best_height_plus_depth>h+prev_dp)
      and (b<awful_bad) then
      best_height_plus_depth:=h+prev_dp;
@z
```

In other words, the log file (with `tracingpages=1`) now gets the line

```
% split254 to 30.08878,36.0 p=-10000
```

but after that patch it would instead say

```
% split254 to 30.08878,33.58878 p=-10000
```

and the footnote would wind up on the first page where it belongs.

When I made the mistake ages ago, I probably wasn't thinking of shrinkability inside the footnote, only in the "virtual" amount of space within `\skip254` that separates the text from its footnotes. Indeed, the present problem goes away if one sets `\skip254=12pt minus 8pt`. But that workaround would be appropriate only for this particular example.

ii. Section 798 could be made more robust with "`until q=cur_align`" moved down one line. Implementors can put this into a change file if they like.

iii. The format `plain.tex` leaves `\box0=\hbox {\tenex B}`; and it also defines `\\` to be a macro such that "`\\10pt`" expands to "`10`" (for example). I could have cleaned these up by saying something like

```
{\setbox1=\box0} \let\\=\undefined
```

but I decided not to change it, since `plain.tex` is so widely used as is.

iv. Frank Mittelbach reported a construction of Morten Høgholm Pedersen:

```
\parindent=0pt
\setbox0=\hbox{p} \hsize=\wd0
\discretionary{m-}{h}{p}\par
```

It gives an overfull box, because TEX doesn't see any feasible breakpoint. (More precisely, the pre-break part exceeds the line width, and TEX doesn't look ahead to see if some fairy godmother is going to save us.) Thus TEX is resigned to making an overfull box, and it takes the only legal breakpoint it knows.

This must be considered a feature of TEX's line-break algorithm. Namely, a discretionary break is normally never taken when the pre-break part would make an overfull box; but it is always taken in the unusual case that no other feasible break is possible (without looking ahead at the third, "unbroken" alternative of the discretionary). A problem can arise only if an unhyphenated word is actually shorter than its first hyphenated fragment. What, me worry?

Amusingly, if you put the line

```
\spaceskip=0pt plus 1fill
  \discretionary{p}{\kern-2em}{}
```

before the other discretionary, you get two p's and nothing overfull.

v. Jonathan Kew mentioned some of the surprising effects that occur when you try to do things in the command line (or in the very first line of TEX's input, at the `**` prompt). There are many, many such.

Before TEX knows the job name, it outputs just to the terminal. Log file output won't happen until an `\input` command has occurred, or input line one has been processed, whichever comes first, because the log file is given its name at that time.

For example,

```
**\showhyphens{whatever}
```

will show '`what-ever`' on the terminal, but not in the log file. Same for

```
**\showhyphens{whatever} \input foo
```

but in this case the log file is called `foo.log` instead of `texput.log`. With

```
**\input foo \showhyphens{whatever}
```

you see '`what-ever`' also in `foo.log`.

## plain TEX format

Version 3.141592653 of `plain.tex` is identical to version 3.14159265, except that `\errorstopmode` is no longer invoked by the `\tracingall` and `\loggingall` macros. (That mistake had been in `plain.tex` for more than 25 years, and I thank David Kastrup for the wakeup call.)

## METAFONT

Turning now to METAFONT, Thorsten Dahlheimer gave the whole program a much-needed scrutiny and came up with a number of bugs that have now been corrected in version 2.718281. (Incidentally, he has also given me invaluable help finding mistakes in the darker corners of *TAOCP*.) Only one of those bugs was serious enough to affect real programs with high probability; the others are the sorts of things that a good nitpicker will spot when reading code, although the actual misbehavior requires weird scenarios. As usual, you can find details of the significant changes to Pascal code in the file `errata/mf84.bug`. The complete source file `mf.web` shows many instances of improved commentary.

1. The serious bug arose from user input such as

```
boolean b[]; b1=true=b2;
```

earlier versions of METAFONT would go into an infinite loop from such constructions, so evidently nobody ever writes code like this. (Strings, paths, and pictures have similar problems, not just booleans.) No problem would occur if the statement had been "`b1=b2=true`" instead. I forgot to include one instruction in my program, and it's a glaring error in section 1003.

This bug is also in the METAPOST source, `mp.web`, which I assume somebody else will fix. Whoever does that should also look carefully at the other changes just made to `mf.web`, since so much of the code is common to both.

2. There also were problems in the TFM files when extremely large characters or dimensions were present. For example, from

```
mode:=lowres; mode_setup; designsize:=10pt#;
beginchar("!",160pt#,-160pt#,160pt#);
  endchar; end
```

you get a TFM file with a bad character width and depth, because of an off-by-one error in my code. (TFtoPL doesn't complain about the character height, which violates some but not all of the documentation of TFM files: A `fix_word` is supposed to lie between $-2048$ and $2048 - 2^{-20}$, inclusive, but *The METAFONTbook* says that no TFM dimension should result in the `fix_word` value $-2048$. TEX has no problem inputting that value.)

3. Another TFM problem was tweaked with ultralarge design sizes:

```
fontmaking:=1; designsize:=2000;
  fontdimen 2: 3000;
  shipout nullpicture; end
```

used to set `fontdimen 2` (the `SPACE` parameter) to be about 32000 points. The correct behavior is to reduce `fontdimen 2` to just less than 2048 points.

4. Weird behavior could previously occur with

```
transform T;
T=identity xscaled 4 yscaled 3 rotated 180;
pickup pencircle transformed T;
show currentpen;
```

which always came out correctly without the (redundant) rotation by 180.

5. Another bug arose in code fragments like

```
string a.b; a.b="lost"; outer a;
  numeric a.c; showvariable a;
```

the `string a.b` was indeed now lost. (METAPOST probably fails in the same way.)

6. METAFONT now checks that serial numbers don't overflow. Actually I had recommended that the program always be run with arithmetic integer overflow trapped; but this doesn't seem to be current practice. If a user creates $2^{25}$ distinct numeric variables, the "`METAFONT capacity exceeded`" error now occurs; formerly, this would have caused arithmetic overflow. (Well, this correction was actually made already in TEX-live change files some years ago; I've now introduced it into the master file `mf.web`, in a slightly different way.)

Not a bug: The `init_gf` procedure has an assignment to `str_start[str_ptr+1]` that looks like it could cause a segmentation fault if `str_ptr= max_strings`. Actually, however, that can't happen. (The test "`str_ptr+3>max_strings`" in `end_name`, together with the fact that `area_delimiter=0` in that procedure because `cur_area=""`, provides the extra breathing space.) But I changed `init_gf` anyway.

Anomalies that won't be changed: Autorounding does not work properly when filling certain nonconvex shapes, such as

```
pickup makepen((-.6,0)--(.6,0)--cycle);
filldraw (2,0){up}..(0,1){down}..%
  (1,0){down}..(0,-1){down}..cycle
```

at point `(1,0)`. Pens whose width and height are not integers are deprecated; there's no point cluttering up the code with stuff that benefits only them.

One of METAFONT's (and METAPOST's) most interesting algorithms is the way it chooses control points and directions for paths that are partially specified. I ran into a curious glitch some years ago when preparing an illustration for my book *Selected Papers on Computer Languages*: The two paths

`(0,0){dir45}...(15,0)...(0,0){dir150}`
and `(0,0){dir-45}...(15,0)...(0,0){dir-150}`
turn out to have amazingly different shapes. (The first one twists around almost unbelievably, while the second looks reasonable.) I tracked this down to the equations in METAFONT's "`solve_choices`" routine, which chooses the desired "turning angle" at the point `(15,0)`. In both cases this value, `psi[1]`, is set to `n_arg(-983040,0)`; here $-983040$ is the internal (scaled) representation of $-15$, and `n_arg` is supposed to determine the value of `angle(-15,0)`. [See page 67 of *The METAFONTbook*.] The answer is 180, which is appropriate in the second case, but the first case really wants the answer to be $-180$.

## Computer Modern

I made a noticeable change to the shape of one (and only one) letter in the CM family, namely the calligraphic F. The new one has a slightly different swash, which pleases me more when I look at it in *The Art of Computer Programming*. The change is small, yet it would be nice if people would remake the Type 1 versions of the fonts that use `calu.mf`, namely `cmsy*` and `cmbsy*`.

The lowercase Greek nu could develop a tiny notch at the bottom, especially at high resolutions of boldface versions (brought to my attention by Charles Duan, who conjectured its existence by reading the source code!). So I corrected that problem.

Duan also found a few other places where the source code was logically wrong in `greekl.mf`. I fixed those too. However, those changes don't actually show up in the generated font, since the differences in point positions are minuscule.

Karel Piška noticed that the bulbs of lowercase a and c are positioned rather differently when the "`blacker`" parameter of a mode varies. (He blamed it on varying resolution, but that's because my code was obscure.) In those characters I essentially try to move strokes apart so that there's twice as much white space as the thickness of the pen; therefore a `blacker` pen makes the strokes go further apart.

My logic was faulty, because the "`blacker`" setting was intended to compensate for differences in the device that make its apparent pen width too small, thereby making the actual appearance after printing only as black as it would have been on an ideal device; increasing "`blacker`" by 1 shouldn't make me reposition any strokes. Yet I do actually reposition them, on the lowercase a, by roughly 2 pixels per unit of `blacker`! And the bulb on c is positioned to be like that of a. Still, the repositioned bulbs look OK, and I'm happy to continue forever with this wart in the design.

## TeXware

TFtoPL version 3.2 is identical to version 3.1 except that a (missing) newline character now appears after one of the warning messages.

## *Computers & Typesetting*

Dozens of corrections were made to Volumes A, B, C, D, and E of the books *Computers & Typesetting*, bringing everything up to date with respect to the latest sources. (This includes *The TeXbook*, which is a paperback Volume A, and *The METAFONTbook*, which is a paperback Volume C.) Copies of the corrected books won't be available for sale until the publisher's stock of already-printed volumes is depleted; but I've prepared detailed errata from which you can make hardcopy inserts to paste into the books you have.

## Summary

All of the results of my changes appear in the following files:

- `tex/texbook.tex` % source file for *The TeXbook*
- `tex/tex.web` % complete master file for TeX in Pascal
- `tex/trip.fot` % torture test terminal output
- `tex/tripin.log` % torture test first log file
- `tex/trip.log` % torture test second log file
- `tex/trip.typ` % torture test output of DVItype
- `texware/tftopl.web` % complete master file for TFtoPL in Pascal
- `mf/mfbook.tex` % source file for *The METAFONTbook*
- `mf/mf.web` % complete master file for METAFONT in Pascal
- `mf/trap*` % (namely `trap.fot`, `trapin.log`, `trap.log`, `trap.typ`, `trap.pl`)
- `mf/trap.fot` % torture test terminal output
- `mf/trapin.log` % torture test first log file
- `mf/trap.log` % torture test second log file
- `mf/trap.typ` % torture test output of GFtype
- `mf/trap.pl` % torture test output of TFtoPL
- `cm/calu.mf` % master source file for calligraphic capital letters
- `cm/greekl.mf` % master source file for lowercase greek letters
- `cm/symbol.mf` % master source file for special symbols
- `errata/errata.ten` % changes to Volumes ABCDE before 2001
- `errata/errata.eleven` % changes to Volumes ABCDE in 2001
- `errata/errata.tex` % changes to Volumes ABCDE since the 2001 boxed set
- `errata/tex82.bug` % changes to `tex.web` since the beginning
- `errata/mf84.bug` % changes to `mf.web` since the beginning
- `errata/cm85.bug` % changes to Computer Modern metafont sources since 1985

These files are available in directory `pub/tex/dist` of the ftp server `cs.stanford.edu`, which accepts "`anonymous`" as a login name. They are a subset of the files in `pub/tex/dist/tex08.tar.gz`, which you can compare to `pub/tex/dist/tex03.tar.gz` if you like. Hopefully they will be easy to incorporate into the major distributions of TeX, and they will presumably soon be available on CTAN.

In general the changes can be characterized as a general cleanup, especially to the documentation. The new versions don't affect old documents, except when the existing behavior was seriously incorrect. (And except for the fact that TeX will often run a bit faster now.)

To do this revision I waded through more than 600 K bytes of text files, not counting the binary `.pdf` and `.png` files that were also submitted. Barbara Beeton faithfully compiled all of this material during the years 2003–2007, and organized it so that my task wasn't hopeless. She had many volunteers helping to separate wheat from chaff; needless to say, I'm extremely grateful for all of this assistance.

The total number of independent topics about which I had to make a decision, after they had come through the filtering process, was approximately 335. Some of these needed several days of thought and careful study; some of them needed only a few seconds. More than a hundred of them were nontrivial, and I did my best.

So now I send best wishes to the whole TeX community, as I leave for vacation to the land of *TAOCP* — until 31 December 2013. Au revoir!