What’s new since \TeX?

Based on
Frank Mittelbach
Guidelines for Future \TeX\ Extensions — Revisited
TUGboat 34:1, 2013

Raphael Finkel

CS Department, UK

November 20, 2013
Original TeX and early successors

- TeX 82: 7-bit fonts, English-based hyphenation
Original \TeX{} and early successors

- \TeX{} 82: 7-bit fonts, English-based hyphenation
- \TeX/LAT\TeX{} 3.0 (1989): 8-bit fonts, minimal adjustments. Currently at version 3.1415926.
Original \TeX\ and early successors

- \TeX\ 82: 7-bit fonts, English-based hyphenation
- \TeX/LATEX 3.0 (1989): 8-bit fonts, minimal adjustments. Currently at version 3.1415926.
- p\TeX: Japanese (Kanji: 16-bit fonts) and vertical typesetting
Original TeX and early successors

- TeX 82: 7-bit fonts, English-based hyphenation
- TeX/LaTeX 3.0 (1989): 8-bit fonts, minimal adjustments. Currently at version 3.1415926.
- pTeX: Japanese (Kanji: 16-bit fonts) and vertical typesetting
- ML-Tex: \texttt{charsubdef} lets characters with diacritics to be treated as a single character, solving bad hyphenation. No longer needed given newer fonts containing most accented characters.
Original \TeX and early successors

- \TeX 82: 7-bit fonts, English-based hyphenation
- \TeX/La\TeX 3.0 (1989): 8-bit fonts, minimal adjustments. Currently at version 3.1415926.
- p\TeX: Japanese (Kanji: 16-bit fonts) and vertical typesetting
- ML-\TeX: \texttt{\char24\char31\textbackslash charsTube} lets characters with diacritics to be treated as a single character, solving bad hyphenation. No longer needed given newer fonts containing most accented characters.
- \texttt{\NCTS} (2000): re-implementation of \TeX in Java. But not as modular or extensible as hoped.
Original \TeX{} and early successors

- \TeX{} 82: 7-bit fonts, English-based hyphenation
- \TeX{}/\LaTeX{} 3.0 (1989): 8-bit fonts, minimal adjustments. Currently at version 3.1415926.
- p\TeX{}: Japanese (Kanji: 16-bit fonts) and vertical typesetting
- ML-\TeX{}: \texttt{\backslash chars sub def} lets characters with diacritics to be treated as a single character, solving bad hyphenation. No longer needed given newer fonts containing most accented characters.
- N\TeX{}S (2000): re-implementation of \TeX{} in Java. But not as modular or extensible as hoped.
- \epsilon-\TeX{} (1994): extensions now supported by all other engines: tracing facilities, mixed-direction typesetting, more registers, generalized \texttt{\backslash orphanpenalty}, spacing of last line in paragraph.
Original \TeX{} and early successors

- \TeX{} 82: 7-bit fonts, English-based hyphenation
- \TeX{}/\LaTeX{} 3.0 (1989): 8-bit fonts, minimal adjustments. Currently at version 3.1415926.
- p\TeX{}: Japanese (Kanji: 16-bit fonts) and vertical typesetting
- ML-\TeX{}: `\charsubdef` lets characters with diacritics to be treated as a single character, solving bad hyphenation. No longer needed given newer fonts containing most accented characters.
- \N\LaTeX{} (2000): re-implementation of \TeX{} in Java. But not as modular or extensible as hoped.
- \epsilon-\TeX{} (1994): extensions now supported by all other engines: tracing facilities, mixed-direction typesetting, more registers, generalized `\orphanpenalty`, spacing of last line in paragraph.
- \Omega/\Lambda: Unicode input instead of 8-bit input.
Newer engines, still in use

- pdf\TeX{}/pdf\LaTeX{}: embedded Type-1 fonts, virtual fonts, hyper-links, compression, micro-typography. Dominant in practical use.
Newer engines, still in use

- pdf\TeX/ pdf\LaTeX: embedded Type-1 fonts, virtual fonts, hyper-links, compression, micro-typography. Dominant in practical use.
- \XML/\LaTeX\ (2012): more font technologies (OpenType, Graphite, Apple) without configuring \TeX\ font metrics. Unicode (UTF-8) input and font-glyph references. Supports the bidirectional (bidi) algorithm.
Newer engines, still in use

- `pdfTeX/pdfLaTeX`: embedded Type-1 fonts, virtual fonts, hyper-links, compression, micro-typography. Dominant in practical use.

- `XeTeX/XeLaTeX` (2012): more font technologies (OpenType, Graphite, Apple) without configuring `TeX` font metrics. Unicode (UTF-8) input and font-glyph references. Supports the bidirectional (bidi) algorithm.

- `LuaTeX/LuaLaTeX` (2007): `pdfTeX` with embedded Lua scripting engine. Provides callbacks to hook into or replace underlying `TeX` typesetting engines.
Researchers have identified various problems that require resolution; the rest of this talk covers some of them.
Line breaking

- Last line of paragraph uses normal spacing, even if the previous line is loose or tight. Workaround (ε-TEX and successors): \lastlinefit
Line breaking

- Last line of paragraph uses normal spacing, even if the previous line is loose or tight. Workaround (ε-TEX and successors): \lastlinelight
- Tallest object in a line determines line height or depth, spreading lines even if they would fit. Theoretical workaround: LuaTEX.
Line breaking

- Last line of paragraph uses normal spacing, even if the previous line is loose or tight. Workaround ($\epsilon$-\TeX{} and successors): `\lastlinefit`
- Tallest object in a line determines line height or depth, spreading lines even if they would fit. Theoretical workaround: Lua\TeX{}.
- Cannot adjust paragraph shape based on position on the page, because of the order in which \TeX{} processes text. Theoretical workaround: multi-pass formatting.
Line breaking

- Last line of paragraph uses normal spacing, even if the previous line is loose or tight. Workaround ($\varepsilon$-\TeX{} and successors): `\lastlinefit`
- Tallest object in a line determines line height or depth, spreading lines even if they would fit. Theoretical workaround: Lua\TeX{}.
- Cannot adjust paragraph shape based on position on the page, because of the order in which \TeX{} processes text. Theoretical workaround: multi-pass formatting.
- Consecutive hyphenated lines: can discourage two, but not 3 or more.
Line breaking

- Last line of paragraph uses normal spacing, even if the previous line is loose or tight. Workaround ($\epsilon$-\TeX{} and successors): `\lastlinefit`
- Tallest object in a line determines line height or depth, spreading lines even if they would fit. Theoretical workaround: Lua\TeX{}.
- Cannot adjust paragraph shape based on position on the page, because of the order in which \TeX{} processes text. Theoretical workaround: multi-pass formatting.
- Consecutive hyphenated lines: can discourage two, but not 3 or more.
- Line quality is tight, decent, loose, or very loose, and \TeX{} tries not to change much from line to line. But only 4 categories are insufficient.
Line breaking

- Last line of paragraph uses normal spacing, even if the previous line is loose or tight. Workaround ($\varepsilon$-TEX and successors): `\lastlinefit`
- Tallest object in a line determines line height or depth, spreading lines even if they would fit. Theoretical workaround: LuaTEX.
- Cannot adjust paragraph shape based on position on the page, because of the order in which TeX processes text. Theoretical workaround: multi-pass formatting.
- Consecutive hyphenated lines: can discourage two, but not 3 or more.
- Line quality is tight, decent, loose, or very loose, and TeX tries not to change much from line to line. But only 4 categories are insufficient.
- Rivers of white, identical word repeated in same place on successive lines.
Micro-typography

- Characters bounding white space should affect inter-word spacing. Unresolved.
Micro-typography

- Characters bounding white space should affect inter-word spacing. Unresolved.
- Tracking: increasing/decreasing inter-letter spaces within a word. (in pdfTEX)
Micro-typography

- Characters bounding white space should affect inter-word spacing. Unresolved.
- Tracking: increasing/decreasing inter-letter spaces within a word. (in pdf\TeX)
- Expansion: changing the width of glyphs. (in pdf\TeX)
Characters bounding white space should affect inter-word spacing. Unresolved.

Tracking: increasing/decreasing inter-letter spaces within a word. (in pdfTEX)

Expansion: changing the width of glyphs. (in pdfTEX)

Hanging punctuation: punctuation at the end of a line should protrude slightly. (in pdfTEX and luaTEX)
Microtypography

Features of package microtype

<table>
<thead>
<tr>
<th>TeX engine</th>
<th>Version</th>
<th>Output</th>
<th>Micro-typographic features</th>
</tr>
</thead>
<tbody>
<tr>
<td>pdfTeX</td>
<td>&lt; 0.14f</td>
<td>DVI/PDF</td>
<td>✓</td>
</tr>
<tr>
<td>pdfTeX</td>
<td>≥ 0.14f</td>
<td>DVI/PDF</td>
<td>✓</td>
</tr>
<tr>
<td>pdfTeX</td>
<td>≥ 1.20</td>
<td>DVI</td>
<td>✓</td>
</tr>
<tr>
<td>pdfTeX</td>
<td>≥ 1.40</td>
<td>PDF</td>
<td>✓</td>
</tr>
<tr>
<td>LuaTeX</td>
<td>≥ 0.30</td>
<td>DVI</td>
<td>✓</td>
</tr>
<tr>
<td>LuaTeX</td>
<td>≥ 0.62</td>
<td>PDF</td>
<td>✓</td>
</tr>
<tr>
<td>XeTeX</td>
<td>≥ 0.9997</td>
<td>PDF</td>
<td>✓</td>
</tr>
</tbody>
</table>

- ✓ = enabled
- ☒ = not enabled
- ☺ = not available

- a = ≥ 1.40.4 recommended
Page layout

- No support for complex float management, such as floats across columns or formatting depending on environment. Not well understood.
Page layout

- No support for complex float management, such as floats across columns or formatting depending on environment. Not well understood.

- No support for baseline-to-baseline spacing, because T\(\TeX\) uses instead a concept of “interline glue”.
Page layout

- No support for complex float management, such as floats across columns or formatting depending on environment. Not well understood.
- No support for baseline-to-baseline spacing, because TeX uses instead a concept of “interline glue”.
- No support for grid-based design, which would require that baseline positions be predictable.
Page layout

- No support for complex float management, such as floats across columns or formatting depending on environment. Not well understood.
- No support for baseline-to-baseline spacing, because TeX uses instead a concept of “interline glue”.
- No support for grid-based design, which would require that baseline positions be predictable.
- Consecutive penalties treated as $\min$, so adding an explicit penalty to prevent a break doesn’t work if there is an implicit small penalty (for instance, to discourage an orphan).
Fonts

- Originally, \TeX\ fonts provide few characteristics. Type-1 fonts do provide more information, and OpenType fonts do, too; these characteristics are used by \XeTeX\ and \luaTeX. 
Fonts

- Originally, \TeX\ fonts provide few characteristics. Type-1 fonts do provide more information, and OpenType fonts do, too; these characteristics are used by \LaTeX\ and \lua\TeX.  

- Encoding standardization: \TeX\ requires that every font place the same glyph at each code point. OK for 7 bits. For 8 bits, \LaTeX\ has internal character encoding. Now one should use Unicode (Ω, \Lua\TeX, \LaTeX).
Originally, \TeX fonts provide few characteristics. Type-1 fonts do provide more information, and OpenType fonts do, too; these characteristics are used by \LaTeX{} and lua\TeX{}.

Encoding standardization: \TeX{} requires that every font place the same glyph at each code point. OK for 7 bits. For 8 bits, \LaTeX{} has internal character encoding. Now one should use Unicode (\Ω, Lua\TeX{}, \XeTeX{}).

Ligatures and kerning tables should be language-specific, not just font-specific. Example: ffl may not be ligated in German. Workaround (pdf\TeX{}): suppress all ligatures starting with a given character.
No double accents, under-accent, control over placement of equation numbers. Workaround: \textit{AMS} macros.
Mathematics

- No double accents, under-accents, control over placement of equation numbers. Workaround: *AMS* macros.
- No way to adjust spacing rules. Solution: LuaTEX.
Mathematics

- No double accents, under-accents, control over placement of equation numbers. Workaround: \texttt{AMS} macros.
- No way to adjust spacing rules. Solution: \texttt{Lua\LaTeX}.
- Sub-formulas are typeset (boxed) in natural width even if contained in a stretchable formula, and they cannot cross lines.
Mathematics

- No double accents, under-accents, control over placement of equation numbers. Workaround: \texttt{AMS} macros.
- No way to adjust spacing rules. Solution: \texttt{Lua\TeX}.
- Sub-formulas are typeset (boxed) in natural width even if contained in a stretchable formula, and they cannot cross lines.
- No control over line-breaking in formulas. Solution: \texttt{breqn} package for \texttt{La\TeX}. 
\TeX as a programming language

- \TeX has limited programming constructs and it works by expansion (it’s a “macro” language), making it very difficult to program. Solution: the expl3 package provides a comfortable programming environment.

- It’s impossible to access and manipulate internal data structures. Solution: \LaTeX.

- The “mouth” leads to the “stomach”, not the reverse.
  - Mouth: token parsing and manipulation
  - Stomach: box generation and manipulation

  Once tokens are in boxes and glue, they cannot become tokens again. It would be better to have an intermediate data structure: character data plus attributes, like an abstract syntax tree in a compiler. Solution: \LaTeX.
Advice

- Use the \LaTeX{} enhancements on whatever underlying \TeX{} engine you choose.
- Use pdf\LaTeX{} unless you need Unicode input.
- If you need Unicode input, use lua\LaTeX{} unless you need bidirectional output.
- If you need bidirectional output, use Xe\LaTeX{} with the bidi package.
- Use the microtype package. No parameters needed.