What's new since TEX?

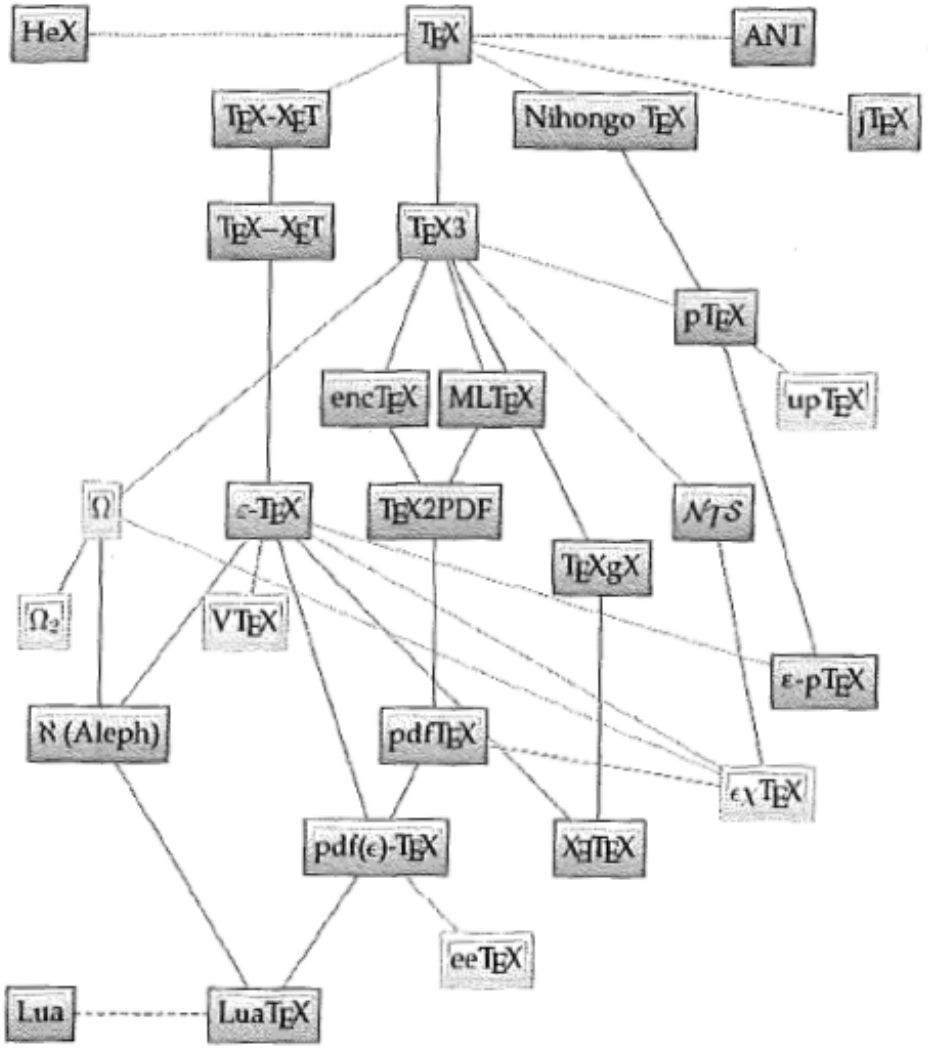
Based on
Frank Mittelbach
Guidelines for Future TEX Extensions — Revisited
TUGboat **34**:1, 2013

Raphael Finkel

CS Department, UK

November 20, 2013

All versions of TEX



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/LATEX 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.
- pTEX: 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: `\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/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: `\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.
- $\mathcal{NTS}$ (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.
- pTEX: 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.
- $\mathcal{NTS}$ (2000): re-implementation of TEX in Java. But not as modular or extensible as hoped.
- ϵ-TEX (1994): extensions now supported by all other engines: tracing facilities, mixed-direction typesetting, more registers, generalized `\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.
- pTEX: 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.
- $\mathcal{NTS}$ (2000): re-implementation of TEX in Java. But not as modular or extensible as hoped.
- ϵ-TEX (1994): extensions now supported by all other engines: tracing facilities, mixed-direction typesetting, more registers, generalized `\orphanpenalty`, spacing of last line in paragraph.
- Ω/Λ: Unicode input instead of 8-bit input.

Newer engines, still in use

- pdfTEX/pdfLATEX: embedded Type-1 fonts, virtual fonts, hyper-links, compression, micro-typography. Dominant in practical use.

Newer engines, still in use

- pdfTEX/pdfLATEX: embedded Type-1 fonts, virtual fonts, hyper-links, compression, micro-typography. Dominant in practical use.
- XƎTEX/XƎLATEX (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.
- XƎTEX/XƎLATEX (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.

Remaining problems

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): `\lastlinefit`
- 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 (ϵ-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.

Line breaking

- Last line of paragraph uses normal spacing, even if the previous line is loose or tight. Workaround (ϵ-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 breaking

- Last line of paragraph uses normal spacing, even if the previous line is loose or tight. Workaround (ϵ-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.

Line breaking

- Last line of paragraph uses normal spacing, even if the previous line is loose or tight. Workaround (ϵ-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 pdfTEX)
- Expansion: changing the width of glyphs. (in pdfTEX)

Micro-typography

- 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`

TEX engine			Micro-typographic features					
Engine	Version	Output	Protrusion	Expansion	(= auto)	Kerning	Spacing	Tracking
pdfTEX	< 0.14f	DVI/PDF	∅	∅	∅	∅	∅	∅
	≥ 0.14f	DVI/PDF	★	⊠	∅	∅	∅	∅
	≥ 1.20	DVI	★	⊠	∅	∅	∅	∅
		PDF	★	★	★	∅	∅	∅
	≥ 1.40	DVI	★	⊠	∅	⊠	⊠	∅
		PDF	★	★	★	⊠	⊠	⊠ [a]
LuaTEX	≥ 0.30	DVI	★	⊠	∅	∅	∅	∅
		PDF	★	★	★	∅	∅	∅
	≥ 0.62	DVI	★	⊠	∅	∅	∅	∅
		PDF	★	★	★	∅	∅	⊠
XETEX	≥ 0.9997	PDF	★	∅	∅	∅	∅	∅

★ = 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 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 XƎTEX 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 XƎTEX and luaTEX.
- 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, LuaTEX, XƎTEX).

Fonts

- Originally, TEX fonts provide few characteristics. Type-1 fonts do provide more information, and OpenType fonts do, too; these characteristics are used by XƎTEX and luaTEX.
- 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, LuaTEX, XƎTEX).
- Ligatures and kerning tables should be language-specific, not just font-specific. Example: ffl may not be ligated in German. Workaround (pdfTEX): suppress all ligatures starting with a given character.

Mathematics

- No double accents, under-accents, control over placement of equation numbers. Workaround: $\mathcal{AMS}$ macros.

Mathematics

- No double accents, under-accents, control over placement of equation numbers. Workaround: $\mathcal{AMS}$ macros.
- No way to adjust spacing rules. Solution: LuaTEX.

Mathematics

- No double accents, under-accents, control over placement of equation numbers. Workaround: $\mathcal{AMS}$ macros.
- No way to adjust spacing rules. Solution: LuaTEX.
- 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: $\mathcal{AMS}$ macros.
- No way to adjust spacing rules. Solution: LuaTEX.
- 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: `breqn` package for LATEX.

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: LuaTEX.
- 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: LuaTEX.

Advice

- Use the LATEX enhancements on whatever underlying TEX engine you choose.
- Use pdfLATEX unless you need Unicode input.
- If you need Unicode input, use luaLATEX unless you need bidirectional output.
- If you need bidirectional output, use XƎLATEX with the `bidi` package.
- Use the `microtype` package. No parameters needed.