

CONCEPTS

experiments turned features

context 2020 meeting

Experiments

There have been quite some experiments. Some results were rejected, some kept. Here are a few (that come to mind). This talk is a mix of summary, discussion and some demos.

Math

There are a couple of additional features in the math engine. Most concern a bit more control over hard coded behavior, but some are sort of new:

```
test $a = b \discretionary class 3 {$<$}{$>$}{$\neq$} c$ test
```

When there is enough room this will give

```
test  $a = b \neq c$  test
```

When `\hsize` is limited we get:

```
test  
 $a =$   
 $b <$   
 $> c$   
test
```

$$\begin{aligned}
& x_1 + x_2 - x_3 + x_4 - x_5 + x_6 - x_7 + x_8 - x_9 + x_{10} - x_{11} + x_{12} - x_{13} + x_{14} - x_{15} + x_{16} - x_{17} + x_{18} - x_{19} + \\
& + x_{20} - x_{21} + x_{22} - x_{23} + x_{24} - x_{25} + x_{26} - x_{27} + x_{28} - x_{29} + x_{30} - x_{31} + x_{32} - x_{33} + x_{34} - x_{35} + x_{36} - \\
& - x_{37} + x_{38} - x_{39} + x_{40} - x_{41} + x_{42} - x_{43} + x_{44} - x_{45} + x_{46} - x_{47} + x_{48} - x_{49} + x_{50} - x_{51} + x_{52} - x_{53} + \\
& + x_{54} - x_{55} + x_{56} - x_{57} + x_{58} - x_{59} + x_{60} - x_{61} + x_{62} - x_{63} + x_{64} - x_{65} + x_{66} - x_{67} + x_{68} - x_{69} + x_{70} - \\
& - x_{71} + x_{72} - x_{73} + x_{74} - x_{75} + x_{76} - x_{77} + x_{78} - x_{79} + x_{80} - x_{81} + x_{82} - x_{83} + x_{84} - x_{85} + x_{86} - x_{87} + \\
& + x_{88} - x_{89} + x_{90} - x_{91} + x_{92} - x_{93} + x_{94} - x_{95} + x_{96} - x_{97} + x_{98} - x_{99} + x_{100} - x_{101} + x_{102} - x_{103} + \\
& + x_{104} - x_{105} + x_{106} - x_{107} + x_{108} - x_{109} + x_{110} - x_{111} + x_{112} - x_{113} + x_{114} - x_{115} + x_{116} - x_{117} + \\
& + x_{118} - x_{119} + x_{120} - x_{121} + x_{122} - x_{123} + x_{124} - x_{125} + x_{126} - x_{127} + x_{128} - x_{129} + x_{130} - x_{131} + \\
& + x_{132} - x_{133} + x_{134} - x_{135} + x_{136} - x_{137} + x_{138} - x_{139} + x_{140} - x_{141} + x_{142} - x_{143} + x_{144} - x_{145} + \\
& + x_{146} - x_{147} + x_{148} - x_{149} + x_{150} - x_{151} + x_{152} - x_{153} + x_{154} - x_{155} + x_{156} - x_{157} + x_{158} - x_{159} + \\
& + x_{160} - x_{161} + x_{162} - x_{163} + x_{164} - x_{165} + x_{166} - x_{167} + x_{168} - x_{169} + x_{170} - x_{171} + x_{172} - x_{173} + \\
& + x_{174} - x_{175} + x_{176} - x_{177} + x_{178} - x_{179} + x_{180} - x_{181} + x_{182} - x_{183} + x_{184} - x_{185} + x_{186} - x_{187} + \\
& + x_{188} - x_{189} + x_{190} - x_{191} + x_{192} - x_{193} + x_{194} - x_{195} + x_{196} - x_{197} + x_{198} - x_{199} + x_{200} = n
\end{aligned}$$

test wel \sqrt{x} come test test wel \sqrt{x} come test test wel \sqrt{x} come test test wel \sqrt{x} come test test hel \sqrt{y} lo
good \sqrt{z} bye test test wel \sqrt{x} come test test wel \sqrt{x} come test test wel \sqrt{x} come test test hel \sqrt{y} lo
good \sqrt{z} bye test test wel \sqrt{x} come test

test $1x + 2x + \dots + nx$ test test $2x + 2x + \dots + nx$ test test $3x + 2x + \dots + nx$ test test $4x + 2x + \dots$
 $\dots + nx$ test test $5x + 2x + \dots + nx$ test test $6x + 2x + \dots + nx$ test test $7x + 2x + \dots + nx$ test test
 $8x + 2x + \dots + nx$ test test $9x + 2x + \dots + nx$ test test $10x + 2x + \dots + nx$ test

More math

In traditional T_EX the last setting wins:

```
1 \def\whatevera
2   {\Umathordrelspacing \textstyle=50mu
3   \Umathopenbinspacing\textstyle=50mu}
4
5 \def\whateverb
6   {\Umathordrelspacing \textstyle=25mu
7   \Umathopenbinspacing\textstyle=25mu}
8
9 $\whatevera a = (-2)$ \par
10 $\whateverb a = (-2)$ \par
11 $\whatevera a = (-2) \quad \whateverb a = (-2)$ \par
12
13  $a = (-2)$ 
14  $a = (-2)$ 
15  $a = (-2) \quad a = (-2)$ 
```

In LuaMetaTeX we can freeze settings on the spot:

```
1 \def\whatevera
2   {\frozen\Umathordrelspacing \textstyle=50mu
3   \frozen\Umathopenbinspacing\textstyle=50mu}
4
5 \def\whateverb
6   {\frozen\Umathordrelspacing \textstyle=25mu
7   \frozen\Umathopenbinspacing\textstyle=25mu}
8
9 $\whatevera a = (-2)$ \par
10 $\whateverb a = (-2)$ \par
11 $\whatevera a = (-2) \quad \whateverb a = (-2)$ \par
12
13  $a = (-2)$ 
14  $a = (-2)$ 
15  $a = (-2) \quad a = (-2)$ 
```

We can now also enable and disable specific features in the engine that control traditional or OpenType approaches. This is only there for experimental and educational purposes.

Macros

Not storing arguments:

```
1 \def\foo#1#0#3{...}
```

```
2 \foo{11}{22}{33}
```

```
3 \foo #1#0#3->....
```

```
4 #1<-11
```

```
5 #2<-
```

```
6 #3<-33
```

Ignoring arguments:

```
1 \def\foo#1#-#2{#1#2}
```

```
2 \foo{1}{2}{3}
```

```
3 13
```

Normal behaviour:

```
1 \def\foo#1#2#3{\#1#2#3}
```

```
2 \foo{1}{\{2\}}{3}
```

```
3 \foo #1#2#3->#1#2#3
```

```
4 #1<-1
```

```
5 #2<-\{2\}
```

```
6 #3<-3
```

Special behaviour:

```
\def\foo#1#+#3{\#1#2#3}
```

```
2 \foo #1#2#3->#1#2#3
```

```
3 #1<-1
```

```
4 #2<-\{\{2\}\}
```

```
5 #3<-3
```

There are more specifiers and I admit that they are hard to remember. But they are mostly used in low level macros anyway.

Optional tokens (we also show some T_EX-expansion-horror here):

```
1 \edef\!space{\expandtoken \ignorecatcode \spaceasciicode}
2
3 \normalexpanded {
4     \protected \def \noexpand \doifelseinset#1#2%
5         {\noexpand\ifhasxtoks{\!space#1,}{\!space#2,}%
6         \noexpand\expandafter\noexpand\firstoftwoarguments
7         \noexpand\else
8         \noexpand\expandafter\noexpand\secondoftwoarguments
9         \noexpand\fi}
10 }
```

or as tokens (`\showluatokens\doifelseinset`) on the next page:

There are some expansion related extensions that are discussed in the low level expansion manual.

1	591504	13	1	argument	
2	643771	13	2	argument	
3	595596	14	0	end match	
4	633535	120	48	if test	ifhasxtoks
5	643789	1	123	left brace	
6	643793	12	44	other char	
7	643741	9	32	ignore	
8	185919	5	1	parameter	
9	633495	12	44	other char	
10	57752	2	125	right brace	
11	167619	1	123	left brace	
12	643686	12	44	other char	
13	228803	5	2	parameter	
14	643434	12	44	other char	
15	643792	2	125	right brace	
16	643788	114	0	expand after	expandafter
17	643775	125	0	call	firstoftwoarguments
18	590609	120	3	if test	else
19	643628	114	0	expand after	expandafter
20	643754	125	0	call	secondoftwoarguments
21	643763	120	2	if test	fi

Cheating with arguments:

```
1 \def\foo#1=#2,{(#1/#2)}
```

```
2 \foo 1=2,\ignorearguments
```

```
3 \foo 1=2\ignorearguments
```

```
4 \foo 1\ignorearguments
```

```
5 \foo \ignorearguments
```

```
(1/2)(1/2)(1/)
```

As in:

```
1 \def\foo#1=#2,{\ifarguments\or(#1)\or(#1/#2)\fi}
```

```
2 \foo 1=2,\ignorearguments
```

```
3 \foo 1=2\ignorearguments
```

```
4 \foo 1\ignorearguments
```

```
5 \foo \ignorearguments
```

```
(1/2)(1/2)(1)
```

Hyphenation

Hyphenation at work:

NED- ER- LANDS	Ned- er- lands	ned- er- lands	Con- T _E Xt	text- test	test- test
NEDERLANDS	Nederlands	nederlands	\CONTEXT	text\ -test	test-test

Controlling hyphenation:

```
\nohyphens NEDERLANDS {\dohyphens Nederlands} nederlands
```

and

```
NEDERLANDS {\nohyphens Nederlands} nederlands
```

NEDERLANDS	NE- DER- LANDS
Ne- der- lands	Nederlands
nederlands	ne- der- lands

There are several ways to implement this:

- choose a language with no patterns:
 - it's quite efficient
 - we loose language specifics
- set the left and right hyphen min values high:
 - it works okay
 - it is a hack
 - we still enter the routine
- block the mechanism:
 - it provides detailed control
 - it is conceptually clean

The last method is what we use in LMTX:

```
\dohyphens: protected macro:\hyphenationmode \completehyphenationcode
```

```
\nohyphens: protected macro:\hyphenationmode \partialhyphenationcode
```

For the moment we have this (it might evolve):

```
1 \chardef \completehyphenationmodecode \numexpr
2   \normalhyphenationmodecode           % \discretionary
3   + \automatichyphenationmodecode       % -
4   + \explicitlyhyphenationmodecode      % \-
5   + \syllablehyphenationmodecode        % pattern driven
6   + \uppercasehyphenationmodecode       % replaces \uchyph
7   + \compoundhyphenationmodecode        % replaces \compoundhyphenmode
8   % \strictstarthyphenationmodecode     % replaces \hyphenationbounds (strict = original tex)
9   % \strictendhyphenationmodecode       % replaces \hyphenationbounds (strict = original tex)
10  + \automaticpenaltyhyphenationmodecode % replaces \hyphenpenaltymode (otherwise use \exhyphenpenalty)
11  + \explicitpenaltyhyphenationmodecode  % replaces \hyphenpenaltymode (otherwise use \exhyphenpenalty)
12  + \permitgluehyphenationmodecode      % turn glue into kern in \discretionary
13  + \permitallhyphenationmodecode       % okay, let's be even more tolerant
14  + \permitmathreplacehyphenationmodecode % and again we're more permissive
15 \relax
```

This replaces some Lua_T_EX mode variables and adds some more which is why we now use a bitset instead of multiple parameters.

In addition we have more detailed discretionary control:

```
1 nederlands\discretionary          {\!}\!}\!}\!}\!}\!}\!}\!}\!}\!}\!}\!}\nederlands
2 nederlands\discretionary options 1 {\!}\!}\!}\!}\!}\!}\!}\!}\!}\!}\!}\nederlands
3 nederlands\discretionary options 2 {\!}\!}\!}\!}\!}\!}\!}\!}\!}\!}\!}\nederlands
4 nederlands\discretionary options 3 {\!}\!}\!}\!}\!}\!}\!}\!}\!}\!}\!}\nederlands
```

nederlands!	ne-	nederlands!	ne-
!nederlands	der-	!ne-	der-
	lands!	der-	lands!
	!nederlands	lands	!ne-
			der-
			lands

At some point it will become ‘frozen’ functionality and that's when it gets documented (first we need to integrate and play a bit more with it in ConT_EXt).

There is now a plugin mechanism that provides more control over language specific hyphenation, e.g. compound words combined with ligatures.

Local control

In LuaTeX we have experimental (kind of ugly) immediate assignments that can be used in expansions without blocking (resulting in tokens that is).

But now we now have local control:

```
1 \newcount\foocounter
```

```
2 \def\foo
```

```
3   {\advance\foocounter\plusone
```

```
4   \the\foocounter}
```

```
5 \edef\oof{(\foo)(\foo)(\foo)(\foo)}
```

```
6 \meaning\oof
```

```
macro:(\advance \foocounter \plusone 0)(\advance \foocounter \plusone  
0)(\advance \foocounter \plusone 0)(\advance \foocounter \plusone 0)
```


Immediate expansion:

```
1 \def\foo
2   {\beginlocalcontrol
3     \advance\foocounter\plusone
4     \endlocalcontrol
5     \the\foocounter}
6
7 \edef\oof{(\foo)(\foo)(\foo)(\foo)}
```

\meaning\oof

macro:(1)(2)(3)(4)

Hidden assignments:

```
1 \scratchcounterone \beginlocalcontrol
2   \scratchcountertwo 100
3   \multiply \scratchcountertwo by 4
4 \endlocalcontrol \scratchcountertwo
5 \the\scratchcounterone
400
```

Fancy expansion:

```
1 \protected\def\foo
2   {\beginlocalcontrol
3     \advance\foocounter\plusone
4     \endlocalcontrol
5     \the\foocounter}
6
7 \edef\oof{(\foo)(\foo)(\foo)(\foo)}
8 \edef\ofo{(\expand\foo)(\expand\foo)(\expand\foo)(\expand\foo)}
9
10 \meaning\oof \par \meaning\ofo
11
12 macro:(\foo )(\foo )(\foo )(\foo )
13
14 macro:(1)(2)(3)(4)
```

And a teaser:

```
1 \protected\def\widthofcontent#1{\beginlocalcontrol
2   \setbox\scratchbox\hbox{#1}\endlocalcontrol \wd\scratchbox}
```

These mechanisms can have surprising side effects due to input stacking. There is some more info in the low level expansion manual.

Conditionals

We can get nicer code that this:

```
1 \ifdim\scratchdimen=10pt
2   \expandafter\one
3 \else\ifnum\scratchcounter=20
4   \expandafter\expandafter\expandafter\two
5 \else
6   \expandafter\expandafter\expandafter\three
7 \fi\fi
```

This becomes:

```
1 \ifdim\scratchdimen=10pt
2   \expandafter\one
3 \orelse\ifnum\scratchcounter=20
4   \expandafter\two
5 \else
6   \expandafter\three
7 \fi
```

There is a bunch of extra conditions like the generic:

`\ifcondition`

some token testers like:

`\iftok` and `\ifhas(x)tok(s)`

some specific for math:

`\ifmathstyle` and `\ifmathparameter`

macro helpers:

`\ifarguments`, `\ifboolean` and `\ifempty`

robust number and dimension interception:

`\ifchknum`, `\ifchkdim`, `\ifcmpnum`, `\ifcmpdim`, `\ifnumval` and `\ifdimval`

bonus checks:

`\iffrozen`, `\ifprotected` and `\ifusercmd`

and the mentioned:

`\orelse` and `\orunless`

Migration

```
1 h: \setbox0\hbox{box} \footnote{h: box}}\setbox2\hbox{\box 0}\box2\par
2 h: \setbox0\hbox{copy} \footnote{h: copy}}\setbox2\hbox{\copy 0}\box2\par
3 h: \setbox0\hbox{unbox} \footnote{h: unhbox}}\setbox2\hbox{\unhbox 0}\box2\par
4 h: \setbox0\hbox{uncopy} \footnote{h: unhcopy}}\setbox2\hbox{\unhcopy0}\box2\par
5 v: \setbox0\hbox{box} \footnote{v: box}}\setbox2\vbox{\box 0}\box2\par
6 v: \setbox0\hbox{copy} \footnote{v: copy}}\setbox2\vbox{\copy 0}\box2\par
7 v: \setbox0\hbox{unbox} \footnote{v: unhbox}}\setbox2\vbox{\unhbox 0}\box2\par
8 v: \setbox0\hbox{uncopy} \footnote{v: unhcopy}}\setbox2\vbox{\unhcopy0}\box2\par
9 \starttabulate[|]
10 \NC tabulate \footnote{tabulate} \NC \NR
11 \stoptabulate
```

h: box¹
h: copy²
h: unbox³
h: uncopy⁴
v: box⁵
v: copy⁶
v: unbox⁷
v: uncopy⁸
tabulate⁹

Everything insert related will always have side effects. It's complicated by the fact that the page flow interferes with expectations of where notes break cq. end up.

-
- ¹ h: box
 - ² h: copy
 - ³ h: unhbox
 - ⁴ h: unhcopy
 - ⁵ v: box
 - ⁶ v: copy
 - ⁷ v: unhbox
 - ⁸ v: unhcopy
 - ⁹ tabulate

Normalizing lines

We can have predictable lines:

```
\hangindent3cm \hangafter 2 \leftskip1cm \rightskip1cm \input ward \par
```

Standard (but already with left skips):

The Earth, as a habitat for animal life, is in old age and has a fatal illness. Several, in fact,
It would be happening whether humans had ever evolved or not. But our presence is like
the effect of an old-age patient who smokes many packs of cigarettes per
day—and we humans are the cigarettes.

Normalized (enhanced, no shifts, indent skip):

The Earth, as a habitat for animal life, is in old age and has a fatal illness. Several, in fact.
It would be happening whether humans had ever evolved or not. But our presence is like
the effect of an old-age patient who smokes many packs of cigarettes per
day—and we humans are the cigarettes.

```
\parshape 2 1cm 10cm 2cm 15cm \leftskip1cm \rightskip1cm \input ward \par
```

Standard:

The Earth, as a habitat for animal life,
is in old age and has a fatal illness. Several, in fact. It would
be happening whether humans had ever evolved or not. But our
presence is like the effect of an old-age patient who smokes many
packs of cigarettes per day—and we humans are the cigarettes.


Normalized:

The Earth, as a habitat for animal life,
is in old age and has a fatal illness. Several, in fact. It would
be happening whether humans had ever evolved or not. But our
presence is like the effect of an old-age patient who smokes many
packs of cigarettes per day—and we humans are the cigarettes.

There might be some more normalization in the future in other subsystems of the engine. One should be aware of this when manipulating node lists after they come out such subsystems.

Freezing paragraph properties

```
1 \forgetparagraphfreezing \getbuffer[sample]
```

 David Stork: With most science fiction films, the more science you understand, the *less* you admire the film or respect its makers. An evil interstellar spaceship careens across the screen. The hero's ship fires off a laser blast, demolishing the enemy ship—the audience cheers at the explosion. But why is the laser beam visible? There is nothing in space to scatter the light back to the viewer. And what slowed the beam a billionfold to render its advance toward the enemy ship perceptible? Why, after the moment of the explosion, does the debris remain centered in the screen instead of continuing forward as dictated by the laws of inertia? What could possibly drag and slow down the expanding debris (and cause the smoke to billow) in the vacuum of outer space? Note too the graceful, falling curve of the debris. Have the cinematographers forgotten that there is no gravity—no ‘downward’—in outer space? Of course the scene is accompanied by the obligatory deafening boom. But isn't outer space eternally silent? And even if there were some magical way to hear the explosion, doesn't light travel faster than sound? Shouldn't we *see* the explosion long before we *hear* it, just as we do with lightning and thunder? Finally, isn't all this moot? Shouldn't the enemy ship be invisible anyway, as there are no nearby stars to provide illumination?

```
1 \setparagraphfreezing \getbuffer[sample]
```



David Stork: With most science fiction films, the more science you understand, the *less* you admire the film or respect its makers. An evil interstellar spaceship careens across the screen. The hero's ship fires off a laser blast, demolishing the enemy ship—the audience cheers at the explosion. But why is the laser beam visible? There is nothing in space to scatter the light back to the viewer. And what slowed the beam a billionfold to render its advance toward the enemy ship perceptible? Why, after the moment of the explosion, does the debris remain centered in the screen instead of continuing forward as dictated by the laws of inertia? What could possibly drag and slow down the expanding debris (and cause the smoke to billow) in the vacuum of outer space? Note too the graceful, falling curve of the debris. Have the cinematographers forgotten that there is no gravity—no ‘downward’—in outer space? Of course the scene is accompanied by the obligatory deafening boom. But isn't outer space eternally silent? And even if there were some magical way to hear the explosion, doesn't light travel faster than sound? Shouldn't we *see* the explosion long before we *hear* it, just as we do with lightning and thunder? Finally, isn't all this moot? Shouldn't the enemy ship be invisible anyway, as there are no nearby stars to provide illumination?

Sample:

```
1 \startplacefigure[location=left,number=no] \externalfigure[halslegacy.jpg][width=30pt] \stopplacefigure
```

```
2 {\bf David Stork:} \samplefile{stork}
```

This feature will stepwise be applied to mechanism and might have side effects when users have their own hacks around T_EX's limitations (and side effects).

Wrapping up paragraphs

We have `\wrapuppar` as new hook. An experimental mechanism has been build around it so that Wolfgang and I can freak out on this.

```
1 \def\TestA{\registerparwrapper
2   {A}
3   {[\ignorespaces}
4   {\removeunwantedspaces}\showparwrapperstate{A}}}}
5
6 \def\TestB#1{\registerparwrapper
7   {B#1}
8   {(\ignorespaces}
9   {\removeunwantedspaces)\showparwrapperstate{B#1}}}}
10
11 \def\TestC{\registerparwrapper
12   {C}
13   {<\ignorespaces}
14   {\removeunwantedspaces>\showparwrapperstate{C}\forgetparwrapper}}
15
16 \def\TestR{\registerparwrapperreverse
17   {R}
18   {<\ignorespaces}
19   {\removeunwantedspaces>\showparwrapperstate{R}}}}
```

Example 1:

```
1 \TestA
2 \dorecurse{3}
3   {1.#1 before \ruledvbox{\hsize2em\raggedcenter\TestB1 !\par} after\par}
4 \dorecurse{3}
5   {2.#1 before \ruledvbox{\hsize3em\raggedcenter          !\par} after\par}
6 \dorecurse{3}
7   {3.#1 before \ruledvbox{\hsize4em\raggedcenter\TestB2 !}      after\par}
8 \forgetparwrapper
9 \dorecurse{3}
10  {4.#1 before \ruledvbox{\hsize5em\raggedcenter\TestB3 !}      after\par}
11 \TestC
12 \dorecurse{3}
13  {5.#1 before \ruledvbox{\hsize2em\raggedcenter\TestA  !}      after\par}
```

[1.1 before (!)_{1B1,1} after]_{1A,1}

[1.2 before !₁ after]_{1A,2}

[1.3 before !₁ after]_{1A,3}

[2.1 before !₂ after]_{1A,4}

[2.2 before !₂ after]_{1A,5}

[2.3 before !₂ after]_{1A,6}

[3.1 before (!)_{1B2,1} after]_{1A,7}

[3.2 before !₃ after]_{1A,8}

[3.3 before !₃ after]_{1A,9}

4.1 before (!)_{1B3,1} after

4.2 before !₄ after

4.3 before !₄ after

<5.1 before !₅ after>_{1C,1}

5.2 before !₅ after

5.3 before !₅ after

Example 2:

```
1 \TestA
2 \dorecurse{3}{6.#1 before after\par} \blank
3 \TestB4
4 \dorecurse{3}{7.#1 before after\par} \blank
5 \TestB5
6 \TestR
7 \dorecurse{3}{8.#1 before after\par} \blank
```

6.1 before after

6.2 before after

6.3 before after

(7.1 before after)_{B4¹}

(7.2 before after)_{B4²}

(7.3 before after)_{B4³}

<((8.1 before after)_{B5¹})_{B4⁴}>_{R3¹}

<((8.2 before after)_{B5²})_{B4⁵}>_{R3²}

<((8.3 before after)_{B5³})_{B4⁶}>_{R3³}

These are just weird examples, but you can expect more interesting features to show up. Beware of stacking because order matters.