PROGRAMMING

local control

context 2021 meeting

Expansion

TeX can be in several so called input reading modes:

Users mostly see it reading from the source file(s). Characters are picked up and interpreted. Depending on what token it becomes some action takes place.

```
1 \setbox0\hbox to 10pt{2} \count0=3 \the\count0 \multiply\count0 by 4
```

- The `1` gets typeset because characters like that are seen as text.
- The `\setbox` primitive triggers picking up a register number, then goes on scanning for a box specification and that itself will typeset a sequence of whatever until the group ends.
- The `count` primitive triggers scanning for a register number (or reference) and then scans for a number; the equal sign is optional.
- The `the` primitive injects some value into the current input stream (it does so by entering a new input level).
- The `multiply` primitive picks up a register specification and multiplies that by the next scanned number. The by is optional.
- Printing from Lua and scanning tokens with e.g. `\scantokens` is like reading (pseudo) files.

Expansion

```
\def\TestA {1 \setbox0\hbox{2} \count0=3 \the\count0}
```

```
\edef\TestB{1 \setbox0\hbox{2} \count0=3 \the\count0}
```

control sequence: TestA						
504049	12	49	other char	1	U+00031	
504048	10	32	spacer			
504023	116	0	set box			setbox
504022	12	48	other char	0	U+00030	
504057	30	10	make box			hbox
504054	1	123	left brace			
503395	12	50	other char	2	U+00032	
433129	2	125	right brace			
31071	10	32	spacer			
31139	109	0	register			count
298198	12	48	other char	0	U+00030	
490898	12	61	other char	=	U+0003D	
31106	12	51	other char	3	U+00033	
298197	10	32	spacer			
503309	129	0	the			the
31063	109	0	register			count
503400	12	48	other char	0	U+00030	

control sequence: TestB						
503444	12	49	other char	1	U+00031	
503442	10	32	spacer			
503250	116	0	set box			setbox
503452	12	48	other char	0	U+00030	
31110	30	10	make box			hbox
503469	1	123	left brace			
503441	12	50	other char	2	U+00032	
503480	2	125	right brace			
31046	10	32	spacer			
503523	109	0	register			count
503387	12	48	other char	0	U+00030	
503517	12	61	other char	=	U+0003D	
503466	12	51	other char	3	U+00033	
503557	10	32	spacer			
503558	12	49	other char	1	U+00031	

Local control

```
\edef\TestB{1 \setbox0\hbox{2} \count0=3 \the\count0}
```

```
\edef\TestC{1 \setbox0\hbox{2} \localcontrolled{\count0=3} \the\count0}
```

control sequence: TestB

503383	12	49	other char	1	U+00031	
503467	10	32	spacer			
503734	116	0	set box			setbox
503735	12	48	other char	0	U+00030	
290426	30	10	make box			hbox
503651	1	123	left brace			
503646	12	50	other char	2	U+00032	
503789	2	125	right brace			
503532	10	32	spacer			
503353	109	0	register			count
503473	12	48	other char	0	U+00030	
503533	12	61	other char	=	U+0003D	
503761	12	51	other char	3	U+00033	
503720	10	32	spacer			
31128	12	49	other char	1	U+00031	

control sequence: TestC

503814	12	49	other char	1	U+00031	
503797	10	32	spacer			
503816	116	0	set box			setbox
503367	12	48	other char	0	U+00030	
503791	30	10	make box			hbox
503614	1	123	left brace			
503803	12	50	other char	2	U+00032	
503519	2	125	right brace			
113605	10	32	spacer			
503521	10	32	spacer			
503776	12	51	other char	3	U+00033	

Side effects

```
\edef\TestB{1 \setbox0\hbox{2} \count0=3 \the\count0}
```

```
\edef\TestD{\localcontrolled{1 \setbox0\hbox{2} \count0=3 \the\count0}}
```

`1 3` ← Watch how the results end up here!

control sequence: TestB						
504618	12	49	other char	1	U+00031	
504617	10	32	spacer			
503839	116	0	set box			setbox
504584	12	48	other char	0	U+00030	
504608	30	10	make box			hbox
503582	1	123	left brace			
503650	12	50	other char	2	U+00032	
503247	2	125	right brace			
503202	10	32	spacer			
503512	109	0	register			count
503507	12	48	other char	0	U+00030	
31138	12	61	other char	=	U+0003D	
503638	12	51	other char	3	U+00033	
504046	10	32	spacer			
504072	12	51	other char	3	U+00033	

control sequence: TestD
<no tokens>

Usage

```
\def\WidthOf#1%
  {\beginlocalcontrol
   \setbox0\hbox{#1}%
   \endlocalcontrol
   \wd0 }
```

```
\scratchdimen\WidthOf{The Rite Of Spring}

\the\scratchdimen
```

105.38608pt

Not always pretty

```
\def\WidthOf#1%
  {\dimexpr
      \beginlocalcontrol
        \begingroup
          \setbox0\hbox{#1}%
          \expandafter
        \endgroup
      \expandafter
      \endlocalcontrol
      \the\wd0
   \relax}
```

```
\scratchdimen\WidthOf{The Rite Of Spring}

\the\scratchdimen
```

```
105.38608pt
```

The Lua end

Right from the start the way to get something into TEX from Lua has been the print functions. But we can also go local (immediate). There are several methods:

- via a set token register
- via a defined macro
- via a string

Among the things to keep in mind are catcodes, scope and expansion (especially in when the result itself ends up in macros).

Via a token register

```
\toks0={\setbox0\hbox{The Rite Of Spring (Igor Stravinsky)}}
\toks2={\setbox0\hbox{The Rite Of Spring (Joe Parrish)}}
```

```
\startluacode
tex.runlocal(0) context("[1: %p]",tex.box[0].width)
tex.runlocal(2) context("[2: %p]",tex.box[0].width)
\stopluacode
```

```
[1: 203.72003pt][2: 180.71667pt]
```

Via a token macro

```
\def\TestA{\setbox0\hbox{The Rite Of Spring (Igor Stravinsky)}}
\def\TestB{\setbox0\hbox{The Rite Of Spring (Joe Parrish)}}
```

```
\startluacode
tex.runlocal("TestA") context("[3: %p]",tex.box[0].width)
tex.runlocal("TestB") context("[4: %p]",tex.box[0].width)
\stopluacode
```

```
[3: 203.72003pt][4: 180.71667pt]
```

Via a string

```
\startluacode
tex.runstring([[\setbox0\hbox{The Rite Of Spring (Igor Stravinsky)}]])
context("[5: %p]",tex.box[0].width)
tex.runstring([[\setbox0\hbox{The Rite Of Spring (Joe Parrish)}]])
context("[6: %p]",tex.box[0].width)
\stopluacode
```

[5: 203.72003pt][6: 180.71667pt]

A bit more high level:

```
context.runstring([[\setbox0\hbox{(Here \bf 1.2345)}]])
context.runstring([[\setbox0\hbox{(Here \bf   %.3f)}]],1.2345)
```

Locked in Lua

```
\startluacode
token.setmacro("TestX",[[\setbox0\hbox{The Rite Of Spring (Igor)}]])
tex.runlocal("TestX")
context("[7: %p]",tex.box[0].width)
\stopluacode
```

[7: 139.38739pt]

```
\startluacode
tex.scantoks(0,tex.ctxcatcodes,[[\setbox0\hbox{The Rite Of Spring (Joe)}]])
tex.runlocal(0)
context("[8: %p]",tex.box[0].width)
\stopluacode
```

[8: 135.22568pt]

Order matters

A lot this relates to pushing stuff into the input which is stacked. Compare:

```
\startluacode
context("[HERE 1]")
context("[HERE 2]")
\stopluacode
```

[HERE 1][HERE 2]

with this:

```
\startluacode
tex.pushlocal() context("[HERE 1]") tex.poplocal()
tex.pushlocal() context("[HERE 2]") tex.poplocal()
\stopluacode
```

[HERE 2][HERE 1]