TOKENS

tokens as I see them

context 2020 meeting

About tokens

- Like nodes, it's a common term used in programming.
- In TEX The Program tokens and nodes are therefore omni-present.
- For most users they are irrelevant concepts.
- But we will explain them anyway.
- Let's try to avoid the snobbish token-speak sometimes heard in the community.
- So . . . I won't correct you as long as you don't correct me.
- Let's now enter the world of tokens in the naïve way.

What are tokens

- It is an internal data structure, effectively a (32 bit) integer.
- This integer encodes a command (opcode) and an char code (operand).
- But often it's not a character but more a sub command.
- Input is converted into tokens.
- Tokens are either expanded (interpreted) or stored.
- When they are stored they are part of a larger data structure, a memory word.
- Token memory is an array of such memory words.
- The token memory ‘word’ has two integers: a token value and an index into token memory.
- That way TEX can have forward linked lists of tokens.
- A hash table maps control sequences onto indices into token memory.

Some implementation details

- Sometimes there is special head token at the start.
- A head token makes for easier appending of extra tokens.
- Shared lists use the head node for a reference count.
- Original TEX uses global temporary lists.
- This is needed when we expand (nested) and need to report issues.
- This is not needed when we just serialize (which we do a lot in LuaTEX).
- So, this is all optimized for performance and memory consumption.
- Freed tokens are collected in a cache so tokens can get scattered.
- In LuaMetaTEX we stay as close to original TEX as possible.
- But the Lua interfaces force us to occasionally divert.

A schematic view of tokens

A token value:

cmd	chr

Token memory:

1	info	link
2	info	link
3	info	link
n	info	link

Looking up control sequences

- A very visible to-be-token is a `\controlsequence`.
- When read, the name will be looked up in the hash table.
- When found its value will point to the table of equivalents.
- That table keeps track of:
  - the type (cmd)
  - the current level (grouping)
  - the current meaning (token list)

The (big) table of equivalents (simplified)

main hash	null control sequence
	128K hash entries
	frozen control sequences
	special sequences (undefined)
registers	17 internal & 64K user glues
	4 internal & 64K user mu glues
	12 internal & 64K user tokens
	2 internal & 64K user boxes
	116 internal & 64K user integers
	0 internal & 64K user attribute
	22 internal & 64K user dimensions
specifications	5 internal & 0 user
extra hash	additional entries (grows dynamic)

The hash table (simplified)

The hash table runs parallel to the main hash. On the todo list is is to move the registers to its own tables and make them dynamic.

1	string index	equivalents or (next > n) index
2	string index	equivalents or (next > n) index
n	string index	equivalents or (next > n) index
n + 1	string index	equivalents or (next > n) index
n + 2	string index	equivalents or (next > n) index
n + m	string index	equivalents or (next > n) index

Equivalents (registers direct, macros indirect i.e. token lists):

1	level	type	value
2	level	type	value
3	level	type	value
n	level	type	value

Other data management

- Grouping is handled by a nesting stack.
- Nested conditionals (`\if...`) have their own stack.
- The values before assignments are saved on the save stack.
- Also other local changes (housekeeping) ends up in the save stack.
- Token lists and macro aliases have references pointers (reuse).
- Attributes, being linked node lists, have their own management.

Example 1: in the input

```
1 \luatokentable{1 \bf{2} 3\what {!}}
```

given token list:						
6578	12	49	other char	1	U+00031	
185661	10	32	spacer			
347410	132	0	protected call			bf
385334	1	123	left brace			
324202	12	50	other char	2	U+00032	
502211	2	125	right brace			
502063	10	32	spacer			
501987	12	51	other char	3	U+00033	
502048	119	0	undefined cs			what
502096	1	123	left brace			
502144	12	33	other char	!	U+00021	
502038	2	125	right brace			

Example 2: in the input

```
1 \luatokentable{a \the\scratchcounter b \the\parindent \hbox to 10pt{x}}
```

given token list:

502074	11	97	letter	a	U+00061	
501806	10	32	spacer			
113	129	0	the			the
502090	85	257	register int			scratchcounter
30818	11	98	letter	b	U+00062	
114	10	32	spacer			
30868	129	0	the			the
502180	88	0	internal dimen			parindent
501787	30	10	make box			hbox
385316	11	116	letter	t	U+00074	
430626	11	111	letter	o	U+0006F	
501947	10	32	spacer			
501356	12	49	other char	1	U+00031	
489426	12	48	other char	0	U+00030	
501931	11	112	letter	p	U+00070	
501878	11	116	letter	t	U+00074	
489420	1	123	left brace			
502055	11	120	letter	x	U+00078	
187885	2	125	right brace			

Example 3: user registers

```
1 \scratchtoks{foo \framed{\red 123}456}

2 \luatokentable\scratchtoks
```

token register: scratchtoks

502299	11	102	letter	f	U+00066	
501953	11	111	letter	o	U+0006F	
502146	11	111	letter	o	U+0006F	
501976	10	32	spacer			
501432	134	0	tolerant protected call			framed
489505	1	123	left brace			
502469	132	0	protected call			red
502278	12	49	other char	1	U+00031	
501810	12	50	other char	2	U+00032	
502308	12	51	other char	3	U+00033	
501968	2	125	right brace			
178149	12	52	other char	4	U+00034	
30805	12	53	other char	5	U+00035	
297103	12	54	other char	6	U+00036	

Example 4: internal variables

```
1 \luatokentable\everypar
```

internal token variable: everypar				
43775	132	0	protected call	dotagsetparcounter
501786	132	0	protected call	page_otr_command_synchronize_side_floats
502218	132	0	protected call	checkindentation
502137	131	0	call	showparagraphnumber
501921	132	0	protected call	restoreinterlinepenalty
502028	131	0	call	flushnotes
30846	132	0	protected call	registerparoptions
502290	131	0	call	flushpostponednodedata
297101	131	0	call	typo_delimited_repeat
501933	131	0	call	spac_paragraphs_flush_intro
502267	131	0	call	typo_initial_handle
502319	131	0	call	typo_firstline_handle
177106	131	0	call	spac_paragraph_wrap
30848	132	0	protected call	spac_paragraph_freeze

Example 5: macro definitions

```
1 \protected\def\whatever#1[#2](#3)\relax{oeps #1 and #2 & #3 done ## error}

2 \luatokentable\whatever
```

protected control sequence: whatever						
502270	19	49	match		argument 1	
502623	12	91	other char	[	U+0005B	
503012	19	50	match		argument 2	
502217	12	93	other char	]	U+0005D	
503220	12	40	other char	(	U+00028	
512246	19	51	match		argument 3	
289579	12	41	other char	)	U+00029	
501883	16	0	relax			relax
502166	20	0	end match			
502214	11	111	letter	o	U+0006F	
512333	11	101	letter	e	U+00065	
30871	11	112	letter	p	U+00070	
512107	11	115	letter	s	U+00073	
385364	10	32	spacer			
502251	21	1	parameter reference			
502236	10	32	spacer			
489423	11	97	letter	a	U+00061	
385335	11	110	letter	n	U+0006E	
503038	11	100	letter	d	U+00064	
502073	10	32	spacer			
503048	21	2	parameter reference			
502068	10	32	spacer			
6583	12	38	other char	&	U+00026	
502660	10	32	spacer			
6579	21	3	parameter reference			
502087	10	32	spacer			
449001	11	100	letter	d	U+00064	
385366	11	111	letter	o	U+0006F	
264636	11	110	letter	n	U+0006E	
502737	11	101	letter	e	U+00065	
501957	10	32	spacer			
512405	6	35	parameter			
512359	10	32	spacer			
491825	11	101	letter	e	U+00065	
512498	11	114	letter	r	U+00072	
30806	11	114	letter	r	U+00072	
501986	11	111	letter	o	U+0006F	
491719	11	114	letter	r	U+00072	

Example 6: commands

```
1 \luatokentable\startitemize
```

frozen instance protected control sequence: startitemize						
30795	134	0	tolerant protected call			startitemgroup
502989	12	91	other char	[	U+0005B	
502692	11	105	letter	i	U+00069	
502228	11	116	letter	t	U+00074	
501877	11	101	letter	e	U+00065	
503221	11	109	letter	m	U+0006D	
503088	11	105	letter	i	U+00069	
501895	11	122	letter	z	U+0007A	
501975	11	101	letter	e	U+00065	
502011	12	93	other char	]	U+0005D	

Example 7: commands

```
1 \luatokentable\doifelse
```

permanent protected control sequence: doifelse					
512213	19	49	match	argument 1	
502620	19	50	match	argument 2	
378581	20	0	end match		
30870	126	21	if test		iftok
502157	1	123	left brace		
502151	21	1	parameter reference		
502497	2	125	right brace		
501798	1	123	left brace		
512103	21	2	parameter reference		
501913	2	125	right brace		
502275	120	0	expand after		expandafter
501784	131	0	call		firstoftwoarguments
154218	126	3	if test		else
30844	120	0	expand after		expandafter
501790	131	0	call		secondoftwoarguments
112070	126	2	if test		fi

Example 8: nothing

```
1 \luatokentable\relax
```

primitive control sequence: relax				
512342	16	0	relax	relax

Example 9: hashes

```
1 \edef\foo#1#2{(#1)(\letterhash)(#2)}  \luatokentable\foo
```

control sequence: foo					
501793	19	49	match		argument 1
512348	19	50	match		argument 2
30857	20	0	end match		
501845	12	40	other char	(	U+00028
502702	21	1	parameter reference		
512929	12	41	other char	)	U+00029
503014	12	40	other char	(	U+00028
297109	12	35	other char	#	U+00023
512094	12	41	other char	)	U+00029
385333	12	40	other char	(	U+00028
502514	21	2	parameter reference		
512324	12	41	other char	)	U+00029

Example 10: nesting

```
1 \def\foo#1{\def\foo##1{(#1)(##1)}}  \luatokentable\foo
```

control sequence: foo

512482	19	49	match	argument 1	
503189	20	0	end match		
503184	115	1	def		def
501797	131	0	call		foo
30839	6	35	parameter		
501596	12	49	other char	1 U+00031	
502929	1	123	left brace		
502795	12	40	other char	(U+00028	
503103	21	1	parameter reference		
502026	12	41	other char	) U+00029	
503005	12	40	other char	(U+00028	
502277	6	35	parameter		
501909	12	49	other char	1 U+00031	
503015	12	41	other char	) U+00029	
489360	2	125	right brace		