

# The `erw-l3` package\*

January 27, 2022

## Abstract

Provides utilities based on L<sup>A</sup>T<sub>E</sub>X3[1], such as `\erw_merge_sort:nNn`.

## Contents

<b>I</b>	<b>Usage</b>	<b>2</b>
1	<b>boilerplate</b>	2
2	<b>quark</b>	2
3	<b>predicate</b>	3
4	<b>keyval</b>	3
5	<b>op's on lists</b>	3
6	<b>algo</b>	3
7	<b>code</b>	4
<b>II</b>	<b>Other</b>	<b>4</b>
1	<b>Bibliography</b>	4
2	<b>Support</b>	4
<b>III</b>	<b>Implementation</b>	<b>4</b>
1	<b>kernel</b>	4
2	<b>boilerplate</b>	4
3	<b>quark</b>	5

---

\*This file describes version v4.1, last revised 2022-01-27.

<b>4</b>	<b>predicate</b>	<b>6</b>
<b>5</b>	<b>keyval</b>	<b>8</b>
<b>6</b>	<b>op's on list</b>	<b>9</b>
<b>7</b>	<b>algo</b>	<b>12</b>
7.1	split . . . . .	12
7.2	thread sort . . . . .	13
7.3	merge sort . . . . .	15
7.4	filter . . . . .	16
<b>8</b>	<b>code</b>	<b>17</b>
	<b>Index</b>	<b>20</b>

## Part I

# Usage

## 1 boilerplate

---

<code>\erw_keys_set:n</code>	<code>\erw_keys_set:n{⟨<i>keyval list</i>⟩}</code>
<code>\erw_keys_set:nn</code>	

---



---

<code>\erw_identity:n</code>	★
<code>\erw_int_incr:n</code>	★
<code>\erw_swap:nn</code>	★
<code>\erw_swap:ne</code>	★
<code>\erw_name_signature_cs:N</code>	★

---

## 2 quark

---

<code>\erw_all_q:w</code>	<code>\erw_remove_last_q:w⟨<i>tokenlist</i>⟩ \q_recursion_tail\q_recursion_stop</code>
<code>\erw_remove_first_q:w</code>	
<code>\erw_first_q:w</code>	
<code>\erw_remove_last_q:w</code>	
<code>\erw_last_q:w</code>	

---

### 3 predicate

---

<code>new_compare_p</code>	<code>\erw_keys_set:n{ new_compare_p = {&lt;name&gt;}{&lt;signature&gt;}{&lt;predicate&gt;} }</code>
	Instance
	<code>erw_compare_p:nNnNn</code>
	<code>erw_int_incr_p:nn</code>
	<code>erw_key_compare_p:nNn</code>
	<code>erw_key_compare_p:n</code>

### 4 keyval

---

<code>\erw_keyval_key:n</code>
<code>\erw_keyval_value:n</code>
<code>\erw_keyval:nn</code>
<code>\erw_keyval_dispatch:NNn</code>
<code>\erw_keyval_dispatch_protected:NNn</code>

---

### 5 op's on lists

---

<code>\erw_remove_first:n</code>
<code>\erw_remove_last:n</code>
<code>\erw_first:n</code>
<code>\erw_last:n</code>
<code>\erw_adjacent_insert:nn</code>
<code>\erw_adjacent_insert:en</code>

---

### 6 algo

---

<code>\erw_split_even:n</code>	<code>\erw_thread_sort:nnNn{&lt;first sorted list&gt;}{&lt;second sorted list&gt;}{&lt;compare</code>
<code>\erw_split_even:e</code>	<code>predicate name&gt;}&lt; &gt;</code>
<code>\erw_merge_sort:nNn</code>	<code>\erw_merge_sort:nNn{&lt;compare predicate name&gt;}&lt; &gt;{&lt;unsorted list&gt;}</code>
<code>\thread_sort:nnNn</code>	<code>\erw_filter_uniq:nn{&lt;compare predicate&gt;}{&lt;tokenlist&gt;}</code>
<code>\erw_filter_uniq:nn</code>	<code>\erw_filter_uniq:n{&lt;ascending intergers&gt;}</code>
<code>\erw_filter_uniq:n</code>	

---

## 7 code

---

<code>\erw_parameter:n</code>	<code>\erw_parameter:n{&lt;arity&gt;}</code>
<code>\erw_parameter:nn</code>	<code>\erw_parameter:nn{&lt;start pos&gt;}{&lt;arity&gt;}</code>
<code>\argument:nn</code>	<code>\erw_argument:nn{&lt;start pos&gt;}{&lt;signature&gt;}</code>
<code>\erw_code_analyze:n</code>	
<code>\erw_signature:n</code>	

---

## Part II Other

### 1 Bibliograhyy

- [1] The L<sup>A</sup>T<sub>E</sub>X3 Project Team. *The L<sup>A</sup>T<sub>E</sub>X3 interfaces*. <https://ctan.math.washington.edu/tex-archive/macros/latex/contrib/l3kernel/expl3.pdf>. 2019.

### 2 Support

This package is available from <https://github.com/rogard/erw-l3>.

## Part III Implementation

```
1 <*package>
2 <@@=erw>
3 %      \ExplSyntaxOn
```

### 1 kernel

```
4 \cs_generate_variant:Nn\int_compare_p:nNn{eNe}
5 \cs_generate_variant:Nn\int_eval:n{e}
6 \cs_generate_variant:Nn\prg_new_conditional:Nnn{c}
7 \cs_generate_variant:Nn\prg_replicate:nn{e}
8 \cs_generate_variant:Nn\regex_gset:Nn{c}
9 \cs_generate_variant:Nn\regex_log:N{c}
10 \cs_generate_variant:Nn\regex_match:NnTF{c}
11 \cs_generate_variant:Nn\tl_to_str:n{e}
12 \cs_generate_variant:Nn\prop_put:Nnn{Nne}
```

### 2 boilerplate

```
13 \msg_new:nnnn{__erw}{text}{text~is~not~loaded}{load~amsmath}
14 \cs_new:Npn \__erw_text:n #1
15 {\cs_if_exist:NTF\text{#1}}{\msg_error:nn{__erw}{text}}
16 \cs_new:Npn \__erw_empty:w #1 \q_recursion_stop {\c_empty_tl}
17 \cs_new_protected:Nn \erw_keys_set:n{ \keys_set:nn{__erw}{#1} }
```

```

18 \cs_new_protected:Nn\erw_keys_set:nn{ \keys_set:nn{__erw / #1}{#2} }
19 \cs_generate_variant:Nn\erw_apply:Nw{c}
20 \cs_new:Npn \erw_identity:n#1{#1}
21 \cs_new:Npn \erw_int_incr:n#1{\int_eval:n{#1+1}}
22 \cs_new:Npn \erw_swap:nn#1#2{#2#1}
23 \cs_generate_variant:Nn \erw_swap:nn{e}
24 \cs_new:Npn \erw_name_signature_cs:N #1
25 { \exp_last_unbraced:Ne
26   \__erw_name_signature_cs:nnn{\cs_split_function:N#1}}
27 \cs_new:Nn \__erw_name_signature_cs:nnn{{#1}{#2}}

```

### 3 quark

```

28 \msg_new:nnn{erw}{quark-only-tail}
29 {requires~tail;~got~'~#1';~\msg_line_context:}
30 \cs_new:Npn
31 \erw_all_q:w
32 #1
33 \q_recursion_stop
34 {%
35   \erw_remove_last_q:w#1\q_recursion_stop
36   \erw_last_q:w#1\q_recursion_stop
37 }
38 \cs_new:Npn
39 \erw_remove_first_q:w
40 #1 % <tokenlist ending with recursion tail>
41 \q_recursion_stop
42 {\quark_if_recursion_tail_stop:n{#1}
43   \__erw_remove_first_q:nw#1\q_recursion_stop}
44 \cs_new:Npn
45 \__erw_remove_first_q:nw
46 #1 % <head>
47 #2 % <rest>
48 \q_recursion_stop
49 {\erw_remove_last_q:w#2\q_recursion_stop
50   \erw_last_q:w#2\q_recursion_stop}
51 \cs_new:Npn
52 \erw_first_q:w
53 #1
54 \q_recursion_stop
55 {%
56   \quark_if_recursion_tail_stop:n{#1}
57   \__erw_first_q:enw{ \tl_if_head_is_group_p:n{#1}}#1\q_recursion_stop }
58 \cs_new:Npn
59 \__erw_first_q:nnw
60 #1 % <head is group>
61 #2 % <head>
62 #3 % <rest>
63 \q_recursion_stop
64 {%
65   \bool_if:nTF{#1}{#2}{#2}
66 }
67 \cs_generate_variant:Nn\__erw_first_q:nnw{e}
68 \cs_new:Npn

```

```

69 \erw_remove_last_q:w #1 \q_recursion_stop
70 {%
71   \quark_if_recursion_tail_stop:n{#1}
72   \__erw_remove_last_q:ew{\tl_if_head_is_group_p:n{#1}}#1\q_recursion_stop }
73 \cs_new:Npn
74 \__erw_remove_last_q:nw
75 #1 % <head is group>
76 #2 % <tokenlist>
77 \q_recursion_stop
78 { \__erw_remove_last_q:nnw{#1}#2\q_recursion_stop }
79 \cs_generate_variant:Nn\__erw_remove_last_q:nw{e}
80 \cs_new:Npn
81 \__erw_remove_last_q:nnw
82 #1 % <head is group>
83 #2 % <head>
84 #3 % <rest>
85 \q_recursion_stop
86 {%
87   \quark_if_recursion_tail_stop:n{#3}
88   \bool_if:nTF{#1}{#{#2}}{#2}
89   \__erw_remove_last_q:ew {\tl_if_head_is_group_p:n{#3}} #3 \q_recursion_stop
90 }
91 \cs_generate_variant:Nn\__erw_remove_last_q:nnw{e}
92 \cs_new:Npn
93 \erw_last_q:w #1 \q_recursion_stop
94 {\quark_if_recursion_tail_stop:n{#1}
95   \__erw_last_q:ew{\tl_if_head_is_group_p:n{#1}}#1\q_recursion_stop}
96 \cs_new:Npn
97 \__erw_last_q:nw
98 #1 % <head is group>
99 #2 % <tokenlist>
100 \q_recursion_stop
101 { \__erw_last_q:nnw{#1}#2\q_recursion_stop }
102 \cs_generate_variant:Nn\__erw_last_q:nw{e}
103 \cs_new:Npn
104 \__erw_last_q:nnw
105 #1 % <head is group>
106 #2 % <head>
107 #3 % <rest>
108 \q_recursion_stop
109 {%
110   \quark_if_recursion_tail_stop_do:nn{#3}{ \bool_if:nTF{#1}{#{#2}}{#2} }
111   \__erw_last_q:ew {\tl_if_head_is_group_p:n{#3}} #3 \q_recursion_stop
112 }
113 \cs_generate_variant:Nn\__erw_last_q:nnw{e}

```

## 4 predicate

```

114 \msg_new:nnn{\__erw}{predicate-empty}
115 {empty~expression~in~predicate}
116 \prg_new_conditional:Npnn
117 \erw_and_tl:nn
118 #1 % <predicate expression>
119 #2 % <tokens>

```

```

120 {p}
121 {%^~A
122   \__erw_and_tl:nw {#1}#2 \q_recursion_tail\q_recursion_stop
123 }
124 \cs_new:Npn
125   \__erw_and_tl:nw
126   #1 % <predicate expression>
127   #2 % <value>
128   \q_recursion_stop
129   {%
130     \quark_if_recursion_tail_stop_do:nn{#2}
131     { \prg_return_true: }
132     \__erw_and_tl:nnw
133     {#1} % <predicate expression>
134     #2 % <value>
135     \q_recursion_stop
136   }
137 \cs_new:Npn
138   \__erw_and_tl:nnw
139   #1 % <predicate expression>
140   #2 % <value>
141   #3 % <rest>
142   \q_recursion_stop
143   {%
144     \bool_if:nTF
145     {#1{#2}}
146     {\__erw_and_tl:nw{#1}#3\q_recursion_stop}
147     { \prg_return_false: }
148   }
149 \cs_new:Npn \__erw_new_compare_p:nnn
150   #1 % <name>
151   #2 % <signature>
152   #3 % <code>
153   {%
154     \prg_new_conditional:cnn{#1:#2}
155     {p}
156     {%
157       \bool_if:nTF
158       {#3}
159       {\prg_return_true:}
160       {\prg_return_false:}
161     }
162   }
163 \keys_define:nn{ __erw }
164 {
165   new_compare_p.code:n = {\__erw_new_compare_p:nnn#1}
166 }
167 \erw_keys_set:n
168 {%
169   new_compare_p =
170   {erw_compare} % <name>
171   {nNnNn}
172   { \__erw_compare:eecN{ #2{#3} }{ #2{#5} }{ #1:nNn }#4 }
173 }

```

```

174 \cs_new:Npn
175 \__erw_compare:nnNN
176 #1 % <first>
177 #2 % <second>
178 #3 % <predicate>
179 #4 % <operator>
180 { #3{ #1 }#4{ #2 } }
181 \cs_generate_variant:Nn\__erw_compare:nnNN{eec}
182 \erw_keys_set:n
183 {%
184   new_compare_p =
185   {erw_int_incr}
186   {nn}
187   {\exp_args:Ne
188     \int_compare_p:nNn{ \int_eval:n{#1+1} } = {#2} }
189 }

```

## 5 keyval

```

190 \cs_new:Npn\__erw_keyval_key:w #1 = #2 \q_recursion_stop{#1}
191 \cs_new:Npn\__erw_keyval_value:w #1 = #2 \q_recursion_stop{#2}
192 \cs_new:Npn \erw_keyval_key:n#1{\__erw_keyval_key:w #1 \q_recursion_stop}
193 \cs_new:Npn \erw_keyval_value:n#1{\__erw_keyval_value:w #1 \q_recursion_stop}
194 \cs_new:Npn \erw_keyval:nn#1#2{ #1 = #2 }
195 \erw_keys_set:n
196 {
197   new_compare_p = {erw_key_compare}
198   {nNn}{ \erw_compare_p:nNnNn
199     {int_compare_p}\erw_keyval_key:n{#1}#2{#3} },
200   new_compare_p = {erw_key_compare}
201   {n}{ \erw_compare_recurse_p:nnNN{#1}
202     {int_compare_p}\erw_keyval_key:n< }
203 }
204 \cs_new_protected:Npn
205 \__erw_keyval_dispatch_build:nn
206 #1 % <|_protected>
207 #2 % <ext>
208 {
209   \use:c{cs_new#1:cpn}
210   {erw_keyval_dispatch#2:NNn}
211   ##1 % <unary>
212   ##2 % <binary>
213   ##3 % <keyval list>
214   { \use:c{__erw_keyval_dispatch#2:NNw}##1##2##3=\q_recursion_tail\q_recursion_stop }
215   \use:c{cs_new#1:cpn}
216   {__erw_keyval_dispatch#2:NNw}##1##2##3=##4\q_recursion_stop
217   { \quark_if_recursion_tail_stop_do:nn{##4}{##1{##3}}
218     \use:c{__erw_keyval_dispatch#2:Nw}##2##3=##4\q_recursion_stop }
219   \use:c{cs_new#1:cpn}
220   {__erw_keyval_dispatch#2:Nw}##1##2=##3=\q_recursion_tail\q_recursion_stop
221   {##1{##2}{##3}}
222 }
223 \__erw_keyval_dispatch_build:nn{}{}
224 \__erw_keyval_dispatch_build:nn{_protected}{_protected}

```



## 6 op's on list

```

225 \cs_new:Npn
226 \erw_remove_first:n
227 #1 % <tokenlist>
228 {\erw_remove_first_q:w#1\q_recursion_tail\q_recursion_stop}
229 \cs_generate_variant:Nn\erw_remove_first:n{e}
230 \cs_new:Npn
231 \erw_remove_last:n
232 #1 % <tokenlist>
233 {\erw_remove_last_q:w#1\q_recursion_tail\q_recursion_stop}
234 \cs_generate_variant:Nn\erw_remove_last:n{e}
235 \cs_new:Npn
236 \erw_first:n
237 #1
238 {\erw_first_q:w#1\q_recursion_tail\q_recursion_stop}
239 \cs_generate_variant:Nn\erw_first:n{e}
240 \cs_new:Npn
241 \erw_last:n
242 #1 % <tokenlist>
243 {\erw_last_q:w#1\q_recursion_tail\q_recursion_stop}
244 \cs_generate_variant:Nn\erw_last:n{e}
245 \cs_new:Npn
246 \erw_adjacent_insert:nn
247 #1 % <list>
248 #2 % <separator>
249 {%
250   \erw_first:n{#1}
251   \erw_swap:en
252   { \erw_remove_first:n{#1} }
253   {%
254     \__erw_adjacent_insert:nw
255     {#2} % <separator>
256   }
257   \q_recursion_tail
258   \q_recursion_stop
259 }
260 \cs_generate_variant:Nn\erw_adjacent_insert:nn{e}
261 \cs_new:Npn
262 \__erw_adjacent_insert:nw
263 #1 % <separator>
264 #2 % <rest>
265 \q_recursion_stop
266 {%
267   \quark_if_recursion_tail_stop:n{#2}
268   \__erw_adjacent_insert:new {#1}{\tl_if_head_is_group_p:n{#2}}#2 \q_recursion_stop
269 }
270 \cs_new:Npn
271 \__erw_adjacent_insert:nnw
272 #1 % <separator>
273 #2 % <head is group>
274 #3 % <head>
275 #4 % <rest>
276 \q_recursion_stop

```

```

277 {%
278   #1\bool_if:nTF{#2}{#{3}}{#3}
279   \_erw_adjacent_insert:nw{#1}#4\q_recursion_stop
280 }
281 \cs_generate_variant:Nn\_erw_adjacent_insert:nnw{ne}

282 \cs_new:Npn
283 \erw_clist_tl:nn
284 #1 % <bool>
285 #2 % <list>
286 { \erw_clist_tl:nnw {#1} #2 \q_recursion_tail\q_recursion_stop }
287 \cs_new:Npn
288 \erw_clist_tl:nnw #1 #2\q_recursion_stop
289 {\quark_if_recursion_tail_stop:n{#2}
290  \erw_clist_tl:nenw {#1}
291  {\tl_if_head_is_group_p:n{#2}} #2 \q_recursion_stop}
292 \cs_generate_variant:Nn\erw_clist_tl:nnw{ne}
293 \cs_new:Npn
294 \erw_clist_tl:nnnw
295 #1 % <bool>
296 #2 % <head is group>
297 #3 % <head>
298 #4 % <rest>
299 \q_recursion_stop
300 {
301   \quark_if_recursion_tail_stop_do:nn{#4}
302   {%
303     \bool_if:nTF
304     {\bool_lazy_and_p:nn{#1}{#2}}
305     {#{3}}{#3}
306   }
307   \bool_if:nTF{\bool_lazy_and_p:nn{#1}{#2}}
308   {#{3}}{#3},
309   \erw_clist_tl:nnw {#1} #4 \q_recursion_stop
310 }
311 \cs_generate_variant:Nn\erw_clist_tl:nnnw{ne}
312 \prg_new_conditional:Npnn
313 \erw_if_in_clist:nn
314 #1 % <value>
315 #2 % <clist>
316 {p}
317 { \_erw_clist_if_in:nw {#1} #2, \q_recursion_tail \q_recursion_stop }
318 \cs_new:Npn
319 \_erw_clist_if_in:nw #1 #2 \q_recursion_stop
320 {%
321   \quark_if_recursion_tail_stop:n{#2}
322   \_erw_clist_if_in:nnw {#1} #2 \q_recursion_stop
323 }
324 \cs_new:Nn
325 \_erw_clist_if_in:nn
326 {\_erw_clist_if_in:nw{#1} #2 \q_recursion_stop}
327 \cs_new:Npn
328 \_erw_clist_if_in:nnw #1 #2, #3 \q_recursion_stop
329 {%
330   \quark_if_recursion_tail_stop_do:nn{#3}

```

```

331  {%
332    \str_if_eq:nnTF{#1}{#2}
333    {\prg_return_true:}{\prg_return_false:}
334  }
335  \str_if_eq:nnTF{#1}{#2}
336  {\prg_return_true:}
337  {\__erw_clist_if_in:nw {#1} #3 \q_recursion_stop}
338  \__erw_empty:w\q_recursion_stop
339 }

```

## 7 algo

### 7.1 split

```

340 \cs_new:Npn
341 \erw_split_even:n
342 #1 % <tokenlist>
343 {%
344   \tl_if_empty:nF{#1}
345   {%
346     \exp_last_unbraced:Ne
347     \__erw_split_even:nnnw
348     {%
349       {\__erw_split_even_threshold:n{#1}} % <count>
350       {\tl_if_head_is_group_p:n{#1}} % <head is group>
351     }
352     #1 % <tokenlist>
353     \q_recursion_tail
354     \q_recursion_stop
355   }
356 }
357 \cs_generate_variant:Nn\erw_split_even:n{e}
358 \cs_new:Npn
359 \__erw_split_even_threshold:n
360 #1 % <tokenlist>
361 {\exp_args:Ne
362   \int_div_round:nn{\tl_count:n{#1}}{2}}
363 \cs_new:Npn
364 \__erw_split_even:nnnw
365 #1 % <threshold>
366 #2 % <head is group>
367 #3 % <head>
368 #4 % <rest>
369 \q_recursion_stop
370 {%
371   \quark_if_recursion_tail_stop_do:nn{#4}
372   { { \bool_if:nTF{#2}{#{#3}}{#3} }{ } }
373   \exp_last_unbraced:Ne
374   \__erw_split_even:nnnw
375   {%
376     {1} % <left size>
377     { \tl_if_head_is_group_p:n{#4} }
378     {#1} % <threshold count>
379     { \bool_if:nTF{#2}{#{#3}}{#3} } % <left list>

```

```

380 }
381 #4 % <right list>
382 \q_recursion_stop
383 }
384 \cs_new:Npn
385 \__erw_split_even:nnnnw
386 #1 % <left size>
387 #2 % <right head is group>
388 #3 % <threshold count>
389 #4 % <left list>
390 #5 % <right head>
391 #6 % <right rest>
392 \q_recursion_stop
393 {%
394 \bool_if:nTF
395 { \int_compare_p:nNn {#1}<{#3} }
396 {%
397 \exp_last_unbraced:Ne
398 \__erw_split_even:nnnnw
399 {
400 { \int_eval:n{#1+1} } % <left size>
401 { \tl_if_head_is_group_p:n{#6} } % <right head is group>
402 {#3} % <threshold count>
403 {#4\bool_if:nTF{#2}{{#5}}{#5}} % <left list>
404 }
405 #6
406 \q_recursion_stop
407 }
408 {%
409 {#4}
410 {%
411 \bool_if:nTF{#2}{{#5}}{#5}
412 \erw_remove_last_q:w#6\q_recursion_stop\erw_last_q:w#6\q_recursion_stop}
413 }
414 }

```

## 7.2 thread sort

```

415 \cs_new:Npn
416 \erw_thread_sort:nnNn
417 #1 % <first sorted list>
418 #2 % <second sorted list>
419 #3 % <compare predicate name>
420 #4 % <compare operator>
421 {%
422 \__erw_thread_sort:nNnnn
423 {#3} % <compare predicate name>
424 #4 % <compare operator>
425 {\c_empty_tl} % <accum>
426 {#1}
427 {#2}
428 }
429 \cs_generate_variant:Nn\erw_thread_sort:nnNn{ee}
430 \cs_new:Npn
431 \__erw_thread_sort:nNnnn

```

```

432 #1 % <compare predicate name>
433 #2 % <compare operator>
434 #3 % <sorted>
435 #4 % <first>
436 #5 % <second>
437 {%
438   \__erw_thread_sort:nNnw
439   {#1} % <compare predicate name>
440   {#2} % <compare operator>
441   {#3} % <sorted>
442   #4 \q_recursion_tail% <first>
443   \q_stop
444   #5 \q_recursion_tail% <second>
445   \q_recursion_stop
446 }
447 \cs_generate_variant:Nn\__erw_thread_sort:nNnn{nNee}
448 \cs_new:Npn
449   \__erw_thread_sort:nNnw
450   #1 % <compare predicate name>
451   #2 % <compare operator>
452   #3 % <sorted>
453   #4 % <first>
454   \q_stop
455   #5 % <second>
456   \q_recursion_stop
457   {%
458     \quark_if_recursion_tail_stop_do:nn{#4}
459     { #3 \erw_all_q:w #5 \q_recursion_stop }
460     \quark_if_recursion_tail_stop_do:nn{#5}
461     { #3 \erw_all_q:w #4 \q_recursion_stop }
462     \__erw_thread_sort:nNneew
463     {#1}#2{#3}
464     { \tl_if_head_is_group_p:n{#4} }
465     { \tl_if_head_is_group_p:n{#5} }
466     #4\q_stop
467     #5\q_recursion_stop
468   }
469   \cs_new:Npn
470     \__erw_thread_sort:nNnnnw
471     #1 % <compare predicate name>
472     #2 % <compare operator>
473     #3 % <sorted>
474     #4 % <head is begin>
475     #5 % <head is begin>
476     #6 % <first head>
477     #7 % <first rest>
478     \q_stop
479     #8 % <second head>
480     #9 % <second rest>
481     \q_recursion_stop
482     {%
483       \bool_if:nTF
484       { \use:c{#1:nNn}{#6}#2{#8} }
485       {%

```

```

486 \__erw_thread_sort:nNeee
487 {#1}
488 #2
489 {#3\bool_if:nTF{#4}{#{#6}}{#6}}
490 {\erw_all_q:w#7\q_recursion_stop}
491 {\bool_if:nTF{#5}{#{#8}}{#8}\erw_all_q:w#9\q_recursion_stop}
492 }
493 {%
494 \__erw_thread_sort:nNeee
495 {#1}
496 #2
497 {#3\bool_if:nTF{#5}{#{#8}}{#8}}
498 {\bool_if:nTF{#4}{#{#6}}{#6}\erw_all_q:w#7\q_recursion_stop}
499 {\erw_all_q:w#9\q_recursion_stop}
500 }
501 }
502 \cs_generate_variant:Nn\__erw_thread_sort:nNnnnw{Nnee}

```

### 7.3 merge sort

```

503 \cs_new:Npn
504 \erw_merge_sort:nNn
505 #1 % <compare predicate name>
506 #2 % <compare operator>
507 #3 % <unsorted list>
508 {%
509 \tl_if_empty:nF{#3}
510 {%
511 \__erw_sort_merge:enNw
512 {\tl_if_head_is_group_p:n{#3}} % <head is group>
513 {#1} % <compare predicate name>
514 #2 % <compare operator>
515 #3 % <unsorted list>
516 \q_recursion_tail
517 \q_recursion_stop
518 }
519 }
520 \cs_generate_variant:Nn\erw_merge_sort:nNn{nNe}
521 \cs_new:Npn
522 \__erw_sort_merge:nnNw
523 #1 % <head is group>
524 #2 % <compare predicate name>
525 #3 % <compare operator>
526 #4 % <unsorted list head>
527 #5 % <unsorted list rest>
528 \q_recursion_stop
529 {%
530 \quark_if_recursion_tail_stop_do:nn{#5}
531 { \bool_if:nTF{#1}{#{#4}}{#4} }
532 \exp_last_unbraced:Ne
533 \__erw_sort_merge:nnnN
534 {%
535 \erw_split_even:e
536 {%
537 \bool_if:nTF{#1}{#{#4}}{#4}

```

```

538     \erw_all_q:w#5\q_recursion_stop
539   }
540   } % {<first sorted list>}{<second sorted list>}
541   {#2} % <compare predicate name>
542   #3 % <compare operator>
543   \__erw_empty:w \q_recursion_stop
544 }
545 \cs_generate_variant:Nn\__erw_sort_merge:nnNw{e}
546 \cs_new:Npn
547   \__erw_sort_merge:nnnN
548   #1 % <left unsorted list>
549   #2 % <right unsorted list>
550   #3 % <compare predicate name>
551   #4 % <compare operator>
552   {%
553     \erw_thread_sort:eeNn
554     {%
555       \__erw_sort_merge:enNw
556       {\tl_if_head_is_group_p:n{#1}}
557       {#3} % <compare predicate name>
558       #4 % <compare operator>
559       #1 % <unsorted list>
560       \q_recursion_tail
561       \q_recursion_stop
562     } % <first sorted list>
563     {%
564       \__erw_sort_merge:enNw
565       {\tl_if_head_is_group_p:n{#2}}
566       {#3} % <compare predicate name>
567       #4 % <compare operator>
568       #2 % <unsorted list>
569       \q_recursion_tail
570       \q_recursion_stop
571     } % <second sorted list>
572     {#3} % <compare predicate name>
573     #4 % <operator>
574   }

```

## 7.4 filter

```

575 \msg_new:nnn{\__erw}{tokenlist-incr}
576 {expecting~an~ascending~tokenlist~got~#1~followed~by~#2}
577 \cs_new:Npn
578   \__erw_filter_uniq:nnw
579   #1 % <compare predicate>
580   #2 % <greatest>
581   #3 % <tokenlist>
582   \q_recursion_stop
583   { %
584     \quark_if_recursion_tail_stop:n{#3}
585     \__erw_filter_uniq_aux:nnw{#1}{#2}#3\q_recursion_stop}
586 \cs_new:Npn
587   \__erw_filter_uniq_aux:nw
588   #1 % <compare predicate>
589   #2 % <tokenlist head>

```

```

590 #3 % <tokenlist rest>
591 \q_recursion_stop
592 {%
593   {#2}
594   \__erw_filter_uniq:nnw
595   {#1} % <compare predicate>
596   {#2} #3 % <tokenlist>
597   \q_recursion_stop }
598 \cs_new:Npn
599 \__erw_filter_uniq_aux:nnw
600 #1 % <compare predicate>
601 #2 % <last>
602 #3 % <head token>
603 #4 % <rest token>
604 \q_recursion_stop
605 { %
606   \bool_if:nTF{\use:c{#1:nNn}{#3}<{#2}}
607   {\msg_error:nnnn{\__erw}{tokenlist-incr}{#2}{#3}}
608   {%
609     \bool_if:nF
610     {\use:c{#1:nNn}{#3}={#2}}
611   % ^^A   {{#3}}
612   {\tl_if_single_token:nTF{#3}{#3}{{#3}}}
613   }
614   \quark_if_recursion_tail_stop:n{#4}
615   % ^^A   \__erw_filter_uniq:nnw{#1}{#3}#4\q_recursion_stop }
616   \__erw_filter_uniq:nnw{#1}{#3}#4\q_recursion_stop }
617   \cs_new:Npn
618   \__erw_filter_uniq:nw
619   #1 % <compare predicate>
620   #2 % <tokenlist>
621   {%
622     \quark_if_recursion_tail_stop_do:nn{#2}{\c_empty_tl}
623     \__erw_filter_uniq_aux:nw {#1}#2 \q_recursion_stop}
624   \cs_new:Npn
625   \erw_filter_uniq:nn
626   #1 % <compare predicate>
627   #2 % <tokenlist>
628   {%
629     \__erw_filter_uniq_aux:nw
630     {#1} % <compare predicate>
631     #2
632     \q_recursion_tail % <head token>
633     \q_recursion_stop}
634   \cs_new:Npn
635   \erw_filter_uniq:n
636   #1 % <ascending integers>
637   { \erw_filter_uniq:nn{int_compare_p}{#1} }
638   \cs_generate_variant:Nn\erw_filter_uniq:nn{ne}

```

## 8 code

```

639 \keys_define:nn{\__erw}
640 { clist_map_inline.code:n = \__erw_map_inline_clist:nnn#1 }

```



```

641 \cs_new_protected:Npn
642 \__erw_map_inline_clist:nnn
643 #1 % <clist>
644 #2 % <signature>
645 #3 % <code>
646 {
647   \cs_new_protected:cn
648   {__erw_do:#2}{#3}
649   \clist_map_inline:nn
650   {#1}
651   {\use:c{__erw_do:#2}##1}
652 }
653 \cs_new:Npn
654 \erw_parameter:n
655 #1 % ^^A <arity>
656 {## #1}
657 \cs_new:Npn
658 \__erw_parameter_aux:nn
659 #1 % <finish>
660 #2 % <start>
661 { \int_step_function:nnN {#2}{#1}\erw_parameter:n}
662 \cs_new:Npn
663 \erw_parameter:nn
664 #1 % <start>
665 #2 % <count>
666 {%
667   \exp_args:Ne
668   \__erw_parameter_aux:nn
669   {\int_eval:n{#1+#2-1}}{#1}}
670 \cs_new:Npn
671 \erw_argument:nn
672 #1 % <position>
673 #2 % <signature>
674 {\__erw_argument:nw{#1}#2\q_recursion_tail\q_recursion_stop}
675 \cs_new:Npn
676 \__erw_argument_unit:nn
677 #1 % <position>
678 #2 % <n|N>
679 {\use:c{__erw_argument_#2:w} #1 \q_recursion_stop}
680 \cs_new:Npn\__erw_argument_n:w #1 \q_recursion_stop{## #1}}
681 \cs_new:Npn\__erw_argument_N:w #1 \q_recursion_stop{## #1}
682 \cs_new:Npn
683 \__erw_argument:nw
684 #1 % <position>
685 #2 % <signature list>
686 \q_recursion_stop
687 { \quark_if_recursion_tail_stop:n{#2}
688   \__erw_argument:nnw{#1}#2\q_recursion_stop }
689 \cs_new:Npn
690 \__erw_argument:nnw
691 #1 % <position>
692 #2 % <n|N>
693 #3 % <signature rest>
694 \q_recursion_stop

```

```

695 {%
696   \__erw_argument_unit:nn{#1}{#2}
697   \exp_args:Ne
698   \__erw_argument:nw
699   {\erw_int_incr:n{#1}}#3\q_recursion_stop }
700 \ProcessKeysOptions{__erw}
701 \ExplSyntaxOff
702 \endpackage

```

# Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

## Symbols

\< ..... 376, 377, 384, 395, 405, 406, 412

## B

bool commands:

\bool\_if:nTF ..... 65,  
88, 110, 144, 157, 254, 363, 438, 442,  
507, 514, 529, 538, 546, 618, 624,  
626, 632, 633, 666, 672, 741, 744, 932  
\bool\_lazy\_and\_p:nn ..... 439, 442  
\c\_false\_bool ..... 265  
\c\_true\_bool ..... 223, 235

## C

clist commands:

\clist\_map\_inline:nn ..... 784

cs commands:

\cs\_generate\_variant:Nn ..... 4,  
5, 6, 7, 8, 9, 10, 11, 12, 19, 23,  
67, 79, 91, 102, 113, 181, 314, 319,  
324, 329, 345, 366, 427, 446, 492,  
564, 582, 637, 655, 680, 773, 912, 940  
\cs\_if\_exist:Nn ..... 15  
\cs\_new:Nn ..... 27, 459  
\cs\_new:Npn ..... 14,  
16, 20, 21, 22, 24, 30, 38, 44, 51, 58,  
68, 73, 80, 92, 96, 103, 124, 137, 149,  
174, 189, 208, 215, 226, 244, 275,  
276, 277, 278, 279, 310, 315, 320,  
325, 330, 346, 355, 368, 374, 382,  
389, 393, 399, 403, 410, 417, 422,  
428, 453, 462, 475, 493, 498, 519,  
550, 565, 583, 604, 638, 656, 681,  
712, 721, 733, 752, 759, 769, 788,  
792, 797, 805, 810, 815, 816, 817,  
824, 847, 876, 892, 904, 913, 928, 941  
\cs\_new\_protected:Nn ..... 17, 18, 782  
\cs\_new\_protected:Npn ..... 289, 776  
\cs\_split\_function:N ..... 26

## D

\def ..... 837, 839, 841

## E

erw commands:

\erw\_adjacent\_insert:nn ..... 331, 345  
\erw\_all\_q:w .....  
. 31, 594, 596, 625, 626, 633, 634, 673  
\erw\_and\_tl:nn ..... 117

\erw\_apply:Nw ..... 19  
\erw\_argument:nn ..... 806  
\erw\_clist\_tl:nn ..... 418  
\erw\_clist\_tl:nnnw ..... 425, 429, 446  
\erw\_clist\_tl:nnw .. 421, 423, 427, 444  
\erw\_code\_analyze:n ..... 914, 946  
\erw\_compare\_p:nNnNn ..... 255, 283  
\erw\_compare\_recurse\_p:nnN ..... 933  
\erw\_compare\_recurse\_p:nnNN ..... 195, 286  
\erw\_filter\_uniq:n ..... 770  
\erw\_filter\_uniq:nn ..... 760, 772, 773, 917  
\erw\_first:n ..... 321, 324, 335  
\erw\_first\_q:w ..... 52, 323  
\erw\_identity:n ..... 20, 198  
\erw\_if\_in\_clist:nn ..... 448  
\erw\_int\_incr:n ..... 21, 415, 834  
\erw\_keys\_set:n .....  
..... 17, 167, 182, 201, 267, 280  
\erw\_keys\_set:nn ..... 18  
\erw\_keyval:nn ..... 279  
\erw\_keyval\_key:n ..... 277, 284, 287  
\erw\_keyval\_value:n ..... 278, 934  
\erw\_last:n ..... 326, 329  
\erw\_last\_q:w ..... 36, 50, 93, 328, 547  
\erw\_map:NNnn ..... 404, 415  
\erw\_map\_index:Nnn ..... 411  
\erw\_merge\_sort:nNn ... 639, 655, 920  
\erw\_name\_signature\_cs:N ..... 24  
\erw\_parameter:n ..... 789, 796  
\erw\_parameter:nn ..... 798  
\erw\_remove\_first:n ... 311, 314, 337  
\erw\_remove\_first\_q:w ..... 39, 313  
\erw\_remove\_last:n ..... 316, 319  
\erw\_remove\_last\_q:w 35, 49, 69, 318, 547  
\erw\_signature:n ..... 942  
\erw\_split\_even:n ..... 476, 492, 670  
\erw\_swap:nn ..... 22, 23, 336  
\erw\_thread:Nnn ..... 394  
\erw\_thread:NNnnn ..... 375, 388  
\erw\_thread\_index:Nnnn ..... 383  
\erw\_thread\_sort:nnNn . 551, 564, 688

erw internal commands:

\\_\_erw\_adjacent\_insert:nnw .....  
..... 353, 356, 366  
\\_\_erw\_adjacent\_insert:nw .....  
..... 339, 347, 364  
\\_\_erw\_and\_tl:nnw ..... 132, 138  
\\_\_erw\_and\_tl:nw ..... 122, 125, 146



prop commands:		R	
\prop_put:Nnn	12	regex commands:	
		\regex_gset:Nn	8
		\regex_log:N	9
		\regex_match:NnTF	10
Q			
quark commands:		S	
\quark_if_recursion_tail_stop:n	42, 56, 71, 87, 94, 352, 371, 391, 401, 424, 456, 719, 749, 822	str commands:	
\quark_if_recursion_tail_stop_-do:nn	110, 130, 223, 235, 302, 436, 465, 506, 593, 595, 665, 757, 856	\str_if_eq:nnTF	467, 470
\q_recursion_stop	16, 33, 35, 36, 41, 43, 48, 49, 50, 54, 57, 63, 69, 72, 77, 78, 85, 89, 93, 95, 100, 101, 108, 111, 122, 128, 135, 142, 146, 214, 221, 224, 233, 242, 252, 263, 275, 276, 277, 278, 299, 301, 303, 305, 313, 318, 323, 328, 343, 350, 353, 361, 364, 369, 373, 381, 390, 392, 398, 400, 402, 409, 421, 423, 426, 434, 444, 452, 454, 457, 461, 463, 472, 473, 489, 504, 517, 527, 541, 547, 580, 591, 594, 596, 602, 616, 625, 626, 633, 634, 652, 663, 673, 678, 696, 705, 717, 720, 726, 732, 739, 750, 751, 758, 768, 809, 814, 815, 816, 821, 823, 829, 834, 854, 874, 890	T	
\q_recursion_tail	122, 214, 299, 305, 313, 318, 323, 328, 342, 381, 398, 409, 421, 452, 488, 577, 579, 651, 695, 704, 767, 809, 899, 900, 901	TeX and L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> commands:	
\q_stop	578, 589, 601, 613	\@gobbletwo	838
		\text	15
		tl commands:	
		\c_empty_tl	16, 560, 757, 897
		\tl_count:n	497
		\tl_if_empty:nTF	479, 644
		\tl_if_head_is_group_p:n	57, 72, 89, 95, 111, 353, 426, 485, 512, 536, 599, 600, 647, 691, 700
		\tl_if_single_token:nTF	747
		\tl_map_function:nN	934
		\tl_to_str:n	11
		token commands:	
		\token_if_group_begin:NTF	859
		\token_if_parameter:NTF	857
		U	
		use commands:	
		\use:N	294, 299, 300, 303, 304, 619, 741, 745, 786, 814