

# The `l3sort` package

## Sorting lists\*

The L<sup>A</sup>T<sub>E</sub>X3 Project<sup>†</sup>

Released 2015/06/15

## 1 `l3sort` documentation

L<sup>A</sup>T<sub>E</sub>X3 comes with a facility to sort list variables (sequences, token lists, or comma-lists) according to some user-defined comparison. For instance,

```
\clist_set:Nn \l_foo_clist { 3 , 01 , -2 , 5 , +1 }
\clist_sort:Nn \l_foo_clist
{
  \int_compare:nNnTF { #1 } > { #2 }
  { \sort_reversed: }
  { \sort_ordered: }
}
```

will result in `\l_foo_clist` holding the values `{ -2 , 01 , +1 , 3 , 5 }` sorted in non-decreasing order.

The code defining the comparison should perform `\sort_reversed:` if the two items given as `#1` and `#2` are not in the correct order, and otherwise it should call `\sort_ordered:` to indicate that the order of this pair of items should not be changed.

For instance, a *comparison code* consisting only of `\sort_ordered:` with no test will yield a trivial sort: the final order is identical to the original order. Conversely, using a *comparison code* consisting only of `\sort_reversed:` will reverse the list (in a fairly inefficient way).

**T<sub>E</sub>Xhackers note:** Internally, the code from `l3sort` stores items in `\toks`. Thus, the *comparison code* should not alter the contents of any `\toks`, nor assume that they hold a given value.

---

<code>\seq_sort:Nn</code>	<code>\seq_sort:Nn &lt;sequence&gt; {&lt;comparison code&gt;}</code>
---------------------------	--

<code>\seq_gsort:Nn</code>	Sorts the items in the <i>&lt;sequence&gt;</i> according to the <i>&lt;comparison code&gt;</i> , and assigns the result to <i>&lt;sequence&gt;</i> .
----------------------------	--

---

\*This file describes v5630, last revised 2015/06/15.

<sup>†</sup>E-mail: [latex-team@latex-project.org](mailto:latex-team@latex-project.org)

<hr/> <code>\tl_sort:Nn</code> <hr/>	<code>\tl_sort:Nn &lt;tl var&gt; {&lt;comparison code&gt;}</code>
<code>\tl_gsort:Nn</code>	Sorts the items in the <code>&lt;tl var&gt;</code> according to the <code>&lt;comparison code&gt;</code> , and assigns the result to <code>&lt;tl var&gt;</code> .
<hr/> <code>\clist_sort:Nn</code> <hr/>	<code>\clist_sort:Nn &lt;clist var&gt; {&lt;comparison code&gt;}</code>
<code>\clist_gsort:Nn</code>	Sorts the items in the <code>&lt;clist var&gt;</code> according to the <code>&lt;comparison code&gt;</code> , and assigns the result to <code>&lt;clist var&gt;</code> .
<hr/> <code>\tl_sort:nN</code> ★ <hr/>	<code>\tl_sort:nN {&lt;token list&gt;} &lt;conditional&gt;</code>
	Sorts the items in the <code>&lt;token list&gt;</code> , using the <code>&lt;conditional&gt;</code> to compare items, and leaves the result in the input stream. The <code>&lt;conditional&gt;</code> should have signature <code>:nnTF</code> , and return <b>true</b> if the two items being compared should be left in the same order, and <b>false</b> if the items should be swapped.
	<b>TeXhackers note:</b> The result is returned within <code>\exp_not:n</code> , which means that the token list will not expand further when appearing in an <code>x</code> -type argument expansion.

## 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.

	<b>C</b>		<code>\seq_sort:Nn</code> ..... <i>1, 1</i>
clist commands:		sort commands:	
<code>\clist_gsort:Nn</code> ..... <i>2</i>		<code>\sort_ordered:</code> ..... <i>1, 1</i>	
<code>\clist_sort:Nn</code> ..... <i>2, 2</i>		<code>\sort_reversed:</code> ..... <i>1, 1</i>	
	<b>E</b>		
exp commands:			<b>T</b>
<code>\exp_not:n</code> ..... <i>2</i>		TeX and L <sup>A</sup> T <sub>ε</sub> X commands:	
	<b>F</b>	<code>\toks</code> ..... <i>1, 1</i>	
foo commands:		tl commands:	
<code>\l_foo_clist</code> ..... <i>1</i>		<code>\tl_gsort:Nn</code> ..... <i>2</i>	
	<b>S</b>	<code>\tl_sort:Nn</code> ..... <i>2, 2</i>	
seq commands:		<code>\tl_sort:nN</code> ..... <i>2, 2</i>	
<code>\seq_gsort:Nn</code> ..... <i>1</i>			