

The `lwttverb` package

Alexander Vasilevsky (kalaider)
a.kalaider@yandex.ru

July 17, 2020

1 Introduction

There are many approaches to `text written in typewriter font`. Basic approach of using just `\tt` or `\texttt` does not support hyphenation and cannot be hard-wrapped, potentially causing overfull hbox warnings. Moreover, it is almost inappropriate for typesetting code as all special characters must be escaped manually.

Verbatim-like commands help here but they fail to break properly (`\verb`) or produce too ragged results (`\Verb` from `fancyvrb` with `breaklines` and `breakanywhere` options from `fvextra`). `minted` does not support breaking inline code at all. `url` or `xurl`-based solutions handle special characters inconsistently and have some limitations.

`lwttverb` tries to provide breakable and justifiable inline verbatim command. What follows next is side-by-side comparison of different `\lwttverb` variants and `\lwttverb` with other viable approaches.

Section 2 describes features `lwttverb` provides. Section 3 shows a number of usage examples, some spacing tweaks “in action”, comparison to alternative approaches. Section 4 and section 5 describe command usage and available options. See section 6 for textual comparison to other approaches and section 7 to learn about `lwttverb` limitations.

2 Features

`lwttverb` provides a number of features useful for typesetting code and regular text in typewriter font. This section discusses what it can. See section 7 for what it cannot.

- Justifiable and breakable text in typewriter font.
- Line break can be marked with `hyphen` or any pair of user-provided symbols.
- If line break occurs at the space character, line break is not indicated.
- Subsequent spaces can be leaved as is, collapsed to a single space or removed entirely. If line break occurs between two space characters, it is configuration dependent whether they are removed or retained verbatim.

- Individual characters can be decorated. A variant of decoration function takes two arguments: the current character being typeset and its predecessor.
- Spaces can be replaced with custom characters so they may be highlighted and preserved.

3 Examples

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna.	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna.	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna.
---	---	---

Table 1: Here is how normal L^AT_EX renders the same text three times (just for comparison)

<code>\lwttt[b]{...}</code>	<code>\lwttt[h]{...}</code>	<code>\lwttt[w]{...}</code>
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna.	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna.	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna.

Table 2: Here is how `lwttt` renders the same text with different options applied: `breakline`, `hyphenate` and `wrapline`

<code>\lwtttt{...}</code>	<code>\justtt{...}</code>	<code>\xurltt{...}</code>
Use other text with superduperveeerylooong words and words _{with} special_characters, e.g. urls: <code>http://example.com/hard-to-break-it-properly</code> . Moreover, look at “<<” and “>>” symbols, they look different.	Use other text with superduperveeerylooong words and words _{with} special_characters, e.g. urls: <code>http://example.com/hard-to-break-it-properly</code> . Moreover, look at “<<” and “>>” symbols, they look different. <i>(needs manual escaping)</i>	Use other text with superduperveeerylooong words and words _{with} special_characters, e.g. urls: <code>http://example.com/hard-to-break-it-properly</code> . Moreover, look at “<<” and “>>” symbols, they look different.
<code>\commands</code> , e.g. <code>\LaTeX{}</code> , may produce extra spaces.	<code>\commands</code> , e.g. <code>\LaTeX{}</code> , may produce extra spaces. <i>(needs manual escaping)</i>	<code>\commands</code> , e.g. <code>\LaTeX{}</code> , may produce extra spaces.
<code>\lwttverb ... </code>	<code>\justverb{...}</code>	<code>\xurltt ... </code>
<code>\commands</code> , e.g. <code>\LaTeX{}</code> , may produce extra spaces.	<code>\commands</code> , e.g. <code>\LaTeX{}</code> , may produce extra spaces.	<code>\commands</code> , e.g. <code>\LaTeX{}</code> , may produce extra spaces.

Table 4: Comparison to `\justtt`, `\justverb` and `xurl`’s `\url` with the `obeyspaces` package option. *Note also that `hyperref` does actually interfere with `url`, so the result shown above is somewhat far from what it should/may be. See section 6 for discussion*

<code>\lwttt[w,poskrn=0.5em]↔{...}</code>	<code>\lwttt[w,negkrn=0.5em]↔{...}</code>	<code>\lwttt[w,monospaced]{...}</code>
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna.	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna.	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna.

Table 5: Same text typeset with `lwttverb`’s `\lwttt` with interletter spacing adjusted differently

<code>\lwttt[w,m,poss=6em]{</code> <code>↪...}</code>	<code>\lwttt[w,m,negsp=1em]{</code> <code>↪...}</code>	<code>\lwttt[w,monospaced]{</code> <code>↪...}</code>
<p> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, feli ↪s. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. </p>	<p> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vesti ↪bulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum grava mauris. Nam arcu libero, nonummy eget, consecte ↪tuer id, vulputate a, magna. </p>	<p> Lorem ipsum dolor sit amet, consectetur adi ↪piscing elit. Ut puru ↪s elit, vestibulum ut, placerat ac, adipiscin ↪g vitae, felis. Cura ↪bitur dictum grava mauris. Nam arcu libero ↪, nonummy eget, conse ↪ctetuer id, vulputate a, magna. </p>

Table 6: Same text typeset with `lwttt` with interword spacing adjusted differently

<p> Lorem ipsu •m dolor s •it amet, c •onsectetu •er adipis •cing elit •. Ut purus elit, vesti •ibulum ut •, placera •t ac, ad •piscing v •itae, fel •is. Curab •itur dict •um grava mauris. Na •m arcu li •bero, non •ummy eget, consectetur •er id, vu •lputate a, magna. </p>	<p> Lorem ipsum d •olor sit amet •, consectetur •adipiscing eli •t. Ut purus el •it, vestibulum ut, placerat ac, adipiscing vit •ae, felis. Cu •rabitur dictum gravida mauris. Nam arcu libero •, nonummy eget, consectetur i •d, vulputate a, magna. </p>	<p> Lorem ipsum dolor sit amet, consec •tetuer adipiscin elit. Ut purus el •it, vestibulum ut, placerat ac, ad •scing vitae, felis. Curabitur dictum g •ravitida mauris. Nam arcu libero, nonummy eget, consectetur •r id, vulputate a, magna. </p>	<p> Lorem ipsum dolor sit amet, consectetur adi •piscing elit. Ut purus elit, vestibulum ut, p •lacerat ac, adipiscin vitae, felis. Curabitur dictum grava mauris. Nam arcu libero, nonum •my eget, consectetur id, vulputate a, magna. </p>
---	---	---	--

Table 7: Same text in columns of different width

4 Usage

`\lwtverb` Using `\lwtverb` is just as simple as regular `\verb`. Here is an example of `\lwtverb`:

While `{\LaTeX}` provides `\verb+\verb+` allowing us to output things verbatim, e.g. `\verb|{\LaTeX}|`, `{\thispkg}` provides `\lwtverb+\lwtverb+` allowing the same, e.g. `\lwtverb|{\LaTeX}|`.

While `LATEX` provides `\verb` allowing us to output things verbatim, e.g. `{\LaTeX}`, `\lwtverb` provides `\lwtverb` allowing the same, e.g. `{\LaTeX}`.

`\lwtcode` `\lwtcode|...|` is just a shorthand for `\lwtverb[w]|...|`, i.e. it is a version of `\lwtverb` defaulted to `wrapline` style. It may be useful for inline code typesetting because it marks wrapped words.

`\lwttt` `\lwttt` is a limited version of `\lwtverb` with syntax of a regular command. It may be used in place of `\lwtverb` when it is just needed to typeset text without any special characters or commands in it. `\lwttt` may introduce extra space after commands. Note additional space after `\LaTeX` here: `{\LaTeX}`. `\lwtverb` handles this properly: `{\LaTeX}`.

`\justtt` `\justtt` is like a normal `\texttt` but allows shrinking, stretching and hyphenation by default. Not a verbatim command. Doesn't take any options.

`\jusverb` `\jusverb` is like a `\justtt` but does its best to handle special characters. Internally highly relies on `\detokenize`. Doesn't take any options.

5 Options

All `\lwtxxx` commands take the same options.

`decoration=<cmd1>` `decoration`, `decoration2` and `spacebox` options allow altering style of individual symbols. The first one takes only a single parameter `<cur>`, the current character. `cmd2` takes additionally `<prev>`, the previous character. Consider the example:

`\lwtverb[decoration=\colorbox{lightgray}, spacebox=\framebox{}]|Text with spaces.|`

`T e x t w i t h s p a c e s .`

Another example allows different behavior for the first letter of the word.

`\makeatletter\def\highlight#1#2{\ifx\@sptoken#2\textcolor{red}{#1}\els
↪e#1\fi}\makeatother`
`\lwtverb[decoration2=\highlight]|Text with spaces.|`

`Text with spaces.`

`gobble=<verbatim, no,
extra, all>` The `gobble` option allows one to alter the behavior of `\lwtverb` regarding spaces.

The `verbatim` option value forces `\lwtverb` to treat all spaces as normal characters except that line breaks near the spaces are not marked with `breaksymr` and `breaksyml`.

The default `no` value allows spaces at the beginning and at the end of the line to be gobbled. It is the default behavior \LaTeX itself has. If `spacebox` is provided, `no` behaves exactly as `verbatim`.

The `extra` value leaves only a single space removing all subsequent spaces (may be useful in combination with `spacebox`).

The `all` value removes all spaces completely (even if `spacebox` is provided).

Here is how one may get all spaces obeyed and highlighted with dots.

```
\lwtcode[spacebox=\hbox to 0.5em {\$\hfill\cdot\hfill$}]|Text with a
number of words. Spaces: 1[ ], 2[ ], 3[ ], 4[ ], 5[ ]. Note that
breaks at spaces are not marked with arrow sign, but_long_words_that_ac
tually_break_somewhere_still_are.|
```

```
Text·with·a·number·of·wo
↳rds·.·Spaces:·1[·],·2[·
·],·3[···],·4[····],·5[·
····]·.·Note·that·breaks·a
↳t·spaces·are·not·marked
·with·arrow·sign,·but_lon
↳g_words_that_actually_br
↳eak_somewhere_still_are.
```

As this behavior is quite common, `showspaces` option is a shorthand for the desired behavior above. Here is how each option affects the outcome. `<value>` takes the `no`, `verbatim`, `extra` and `all` value correspondingly.

```
\lwtcode[gobble=<value>]|Text with a number of words. Spaces: 1[ ], 2[
], 3[ ], 4[ ], 5[ ]. Note that breaks at spaces are not marked with
arrow sign, but_long_words_that_actually_break_somewhere_still_are.|
```

```
Text with a number of words. Sp
↳aces: 1[ ], 2[ ], 3[ ], 4[
], 5[ ]. Note that breaks at
spaces are not marked with arrow
sign, but_long_words_that_actuall
↳y_break_somewhere_still_are.
```

```
Text with a number of words. Spaces:
1[ ], 2[ ], 3[ ], 4[ ], 5[
]. Note that breaks at spaces
are not marked with arrow sign,
but_long_words_that_actually_bre
↳ak_somewhere_still_are.
```

```
Text with a number of words. Spaces: 1[], 2[], 3[], 4[], 5[
]. Note that breaks at spaces are not marked with arrow sign,
but long words that actually break somewhere still are.
```

```
Textwithanumberofwords.Spaces:1[
,2[],3[],4[],5[] .Notethatbreaksa
tspacesarenotmarkedwitharrowsig
n,but_long_words_that_actually_
break_somewhere_still_are.
```

Here is how each option behaves on the input consisting almost entirely of spaces:

```
no      “    !    ”
verbatim “    !    ”
extra   “    !    ”
all      “!”
```

`breaksymr=<hbox>` `breaksyml=<hbox>` `breaksymr` and `breaksyml` allow to specify line break indicators, e.g. hyphen, arrow, etc. `lwtverb` effectively inserts `\discretionary{<breaksymr>}{<breaksyml>}{}` after each non-space character of the input.

```
\lwtverb[breaksyml=$\triangleleft$, breaksymr=$\triangleright$] |Long_w
ord_that_should_be_hyphenated. |
```

```
Long_word_that_▷
◁should_be_hyph▷
◁enated.
```

`poskrn=<length>` `negkrn=<length>` Specifies interletter stretching (`poskrn`) and shrinking (`negkrn`) boundaries.

```
\lwtverb[poskrn=1em, negkrn=0.3em] |This_text_must_be_very_loose <filler
_r_text_that_should_wrap>|
\lwtverb[poskrn=1em, negkrn=0.3em] |This_text_must_be_very_tight <another_filler>|
```

```
This_text_must_be_very_loose
<filler_text_that_should_wrap>
This_text_must_be_very_tight <another_filler>
```

`possp=<length>` `negsp=<length>` Specifies interword stretching (`possp`) and shrinking (`negsp`) boundaries.

```
\lwtverb[possp=1em, negsp=0.3em] |Spaces here must be very wide <filler_
text_that_should_wrap>|
\lwtverb[possp=1em, negsp=0.3em] |Spaces here must be very short <another_filler>|
```



```
Spaces here must be reasonably wide
<filler_text_that_should_wrap>
Spacesheremustbeveryshort<another_filler>
```

`breakhandler=<cmd3>` Specifies a command with a three parameters, namely *<cur>*, *<prev>*, *<discretionary>*, i.e. current char, previous char and discretionary box. One should return *<discretionary>* in order to allow break and nothing to prevent it. E.g. here is how **keepwords**-like behavior may be achieved.

```
\def\ignorethree#1#2#3{}
\lwtverb[breakhandler=\ignorethree]|Will not be able to break words.
Long-word-that-is-to-be-normally-hyphenated will be left as is.|
```

```
Will not be able
to break words.
Long-word-that-is-to-be-normally-hyphenated
will be left as is.
```

One may imagine a requirement to break only at a certain symbol, e.g. hyphen. Here is how it may be achieved.

```
\def\breakhyph#1#2#3{\if-#2#3\fi}
\lwtverb[breakhandler=\breakhyph]|Will not be able to break words.
Long-word-that-is-to-be-normally-hyphenated will break at ‘-’.|
```

```
Will not be able to break
words. Long-word-that-is-to-
be-normally-hyphenated will
break at ‘-’.
```

Here we test if the previous character was a hyphen and if so insert discretionary.

<code>debug</code>	Synonym of <code>decoration=\colorbox{lightgray}</code> .
<code>breakline, b</code>	Synonym of <code>breaksymr={}</code> , <code>breaksyml={}</code> .
<code>wrapline, w</code>	Synonym of <code>breaksymr={\tiny\$\hookrightarrow\$}</code> , <code>breaksyml={}</code> .
<code>hyphenate, h</code>	Synonym of <code>breaksymr={-}</code> , <code>breaksyml={}</code> .
<code>monospaced, m</code>	Synonym of <code>poskrn=0pt</code> , <code>negkrn=0pt</code> .
<code>showspaces</code>	Synonym of <code>spacebox=\hbox to 0.5em {\textcolor{gray}{\$\hfill\cdot\hfill\$}}</code> .
<code>obeyspaces</code>	Synonym of <code>gobble=verbatim</code> .
<code>nospaces</code>	Synonym of <code>gobble=all</code> .

6 Comparison to other approaches

There are plenty of packages and/or techniques that provide similar functionality. Why then `lwtverb`? Here is a brief overview of alternatives to `lwtverb` that reveals some subtleties of each of them.

`\tt`, `\texttt`

Requires manual escaping of special characters. Does not support hyphenation by default.

`\justtt`

Simple extension of the previous approach. See [this StackExchange answer](#) for details. `\detokenize` handles most of the special characters automatically, but fails with commands, e.g. `\detokenize{\LaTeX{}}` introduces unwanted extra space after the command name: `\LaTeX {}`. `\detokenize` also fails to preserve `<<` and `>>` resulting in `«` and `»` correspondingly. `lwtverb` provides `\justtt` (without `\detokenize`) and `\justverb` (`\detokenize`-based) just for completeness.

L^AT_EX's `\verb`

Does not allow line breaks inside words. May cause overfull hboxes.

fancyvrb's `\Verb`

With the help of `fvextra` supports line breaks at space only. May cause overfull hboxes with long enough words. See examples above. **fancyvrb** does not allow to insert line break marks.

minted's `\mintinline`

Does not allow to break inline code at all.

url's or **xurl**'s `\url`

Does its job well but handles some characters inconsistently (e.g. `<<` is typeset in typewriter font, but `>>` is not). Suffers from the same problems as `\detokenize` does. `hyperref` may interfere with `url`. Actually, it may be seen on table 4. `\xurltt` from the example above is defined as follows: `\DeclareUrlCommand\xurltt{\urlstyle{tt}}`.

7 Limitations

There are many limitations the author is currently aware of:

- Missing support for inline math.
- Hyphenation does not take into account whether it is semantically allowed to break at certain position. True hyphenation is desirable for typesetting text but in such case simple `\justtt`-based approach would be enough.
- Only some of the command options have their package option equivalents. The user may, however, simply define his own command as `lwtverb` does with `\lwtcode`.
- There is no option to trim leading and trailing spaces. But is such option really necessary?
- As with all other inline verbatim commands, `\lwtverb` and others are fragile. It means that `\lwtverb` (but not `\lwttt`) cannot be used in section names, captions, it may conflict with some tabular environments. `fvextra` fixes a lot of **fancyvrb**'s `\Verb` fragility issues, but not all and at some cost (whitespace preservation, alternative syntax with its own limitations, etc.).

- `\lwtverb` may behave strangely when passed as an argument to other commands (e.g. `\id{\lwtverb|{\LaTeX}|}` becomes “`\LaTeX` ” instead of “`{\LaTeX}`”, where `\id` is defined as follows: `\def\id#1{#1}`).

8 Plans

There currently are no benefits of migration from `newverbs` to `fancyvrb`. It had actually been implemented as a case study but then was rejected. Adopting `fvextra` solution to robustness requires some care. Straightforward approach fails immediately because of the way `fvextra` handles its input — it conflicts with how `xparse` parses its optional arguments. `fvextra` has some limitations too regarding whitespace preservation, requirement of balanced curly braces in curly-braced input, etc. but it provides robust `\Verb`. `lwtverb` differentiates between `\lwtverb` and `\lwttt` (the latter is actually robust since it is not a true verbatim command) and `\lwttt` is supposed to be used in titles, but as always it fails to typeset `\commands` properly. One may solve it using options (such as `nospaces`) having something like `\lwttt{my ‘}\lwttt[nospaces]{\command}\lwttt{’ command}` resulting in “my ‘`\command`’ command” but it is too complex and verbose to be used in practice.

9 Implementation

9.1 Package Options

Additional stretching to be added after each non-space char

```
1 \DeclareOptionX{poskrn}{%
2   \DeclareDocumentCommand\@lwtverb@poskrn{}{#1}}
```

Additional shrinking to be added after each non-space char

```
3 \DeclareOptionX{negkrn}{%
4   \DeclareDocumentCommand\@lwtverb@negkrn{}{#1}}
```

Additional stretching to be added after each space char

```
5 \DeclareOptionX{possp}{%
6   \DeclareDocumentCommand\@lwtverb@possp{}{#1}}
```

Additional shrinking to be added after each space char

```
7 \DeclareOptionX{negsp}{%
8   \DeclareDocumentCommand\@lwtverb@negsp{}{#1}}
```

Handle unknown options

```
9 \DeclareOptionX*{%
10   \PackageWarning{lwtverb}{‘\CurrentOption’ ignored}}
```

Some reasonable defaults

```
11 \ExecuteOptionsX{
12     poskrn=0.3pt,    % higher values result in 'a b c d' (too loose)
13     negkrn=0.3pt,    % higher values result in 'abcd' (too tight)
14     possp=0.3em,     % higher values result in 'abcd e fgh'
15     negsp=0.1em,     % higher values result in 'abcdefgh'
16 }
```

Done with package options

```
17 \ProcessOptionsX\relax
```

9.2 Command Options

```
18 \long\def\@firstoffour#1#2#3#4#{#1}
19 \long\def\@fourthoffour#1#2#3#4{#4}
20
21 \options{
22     /lwtverb/.new family,
23     % options
24     /lwtverb/decoration/.new value = \relax,
25     /lwtverb/decoration2/.new value = \relax,
26     /lwtverb/spacebox/.new value = \relax,
27     /lwtverb/breaksymr/.new value = {},
28     /lwtverb/breaksyml/.new value = {},
29     /lwtverb/poskrn/.new length = \@lwtverb@poskrn,
30     /lwtverb/negkrn/.new length = \@lwtverb@negkrn,
31     /lwtverb/possp/.new length = \@lwtverb@possp,
32     /lwtverb/negsp/.new length = \@lwtverb@negsp,
33     /lwtverb/gobble/.new choice = { verbatim, no, extra, all },
34     /lwtverb/gobble = no,
35     /lwtverb/breakhandler/.new value = \@fourthoffour\relax,
36     % styles
37     /lwtverb/debug/.new style* = {
38         /lwtverb/decoration = \colorbox{lightgray},
39     },
40     /lwtverb/breakline/.new style* = {
41         /lwtverb/breaksymr = {},
42         /lwtverb/breaksyml = {},
43     },
44     /lwtverb/wrapline/.new style* = {
45         /lwtverb/h,
46         /lwtverb/breaksymr = {},
47         /lwtverb/breaksyml = {\tiny$\hookrightarrow$},
48     },
49     /lwtverb/hyphenate/.new style* = {
50         /lwtverb/breaksymr = {-},
51         /lwtverb/breaksyml = {},
52     },
53     /lwtverb/monospaced/.new style* = {
54         /lwtverb/poskrn = 0pt,
55         /lwtverb/negkrn = 0pt,
56     },
57 }
```

```

57 /lwtverb/showspaces/.new style* = {
58   /lwtverb/spacebox=\hbox to 0.5em {%
59     \textcolor{gray}{\hfill\cdot\hfill}},
60 },
61 /lwtverb/obeyspaces/.new style* = {
62   /lwtverb/gobble=verbatim,
63 },
64 /lwtverb/nospaces/.new style* = {
65   /lwtverb/gobble=all,
66 },
67 /lwtverb/keepwords/.new style* = {
68   /lwtverb/breakhandler = \@firstoffour\relax,
69 },
70 % shorthands
71 /lwtverb/b/.new style* = { /lwtverb/breakline },
72 /lwtverb/w/.new style* = { /lwtverb/wrapline },
73 /lwtverb/h/.new style* = { /lwtverb/hyphenate },
74 /lwtverb/m/.new style* = { /lwtverb/monospaced }
75 }

```

9.3 Character Decorator

`\@lwtverb@makespace` Creates properly-sized `\hspace` from (implicitly passed) options. This space replaces the actual space character of the input string.

```

76 \DeclareDocumentCommand\@lwtverb@makespace{}{%

```

Allow L^AT_EX to remove spaces automatically at the beginning and at the end of the line; use unstarred `\hspace` version for this purpose.

```

77   \hspace{0.4em%
78     plus \option{/lwtverb/possp}%
79     minus \option{/lwtverb/negsp}}%
80 }

```

(End definition for \@lwtverb@makespace. This function is documented on page ??.)

`\@lwtverb@makehardspace` Creates properly-sized `\hspace` from (implicitly passed) options. This space replaces the actual space character of the input string and cannot be gobbled.

```

81 \DeclareDocumentCommand\@lwtverb@makehardspace{}{%

```

Prevent L^AT_EX from removing spaces automatically at the beginning and at the end of the line; use starred `\hspace` version for this purpose.

```

82   \hspace*{0.4em%
83     plus \option{/lwtverb/possp}%
84     minus \option{/lwtverb/negsp}}%
85 }

```

(End definition for \@lwtverb@makehardspace. This function is documented on page ??.)

`\@lwtverb@makekern` Creates properly-sized `\hspace` from (implicitly passed) options. This space is put between letters in words to make them shirnkable and stretchable enough.

```
86 \DeclareDocumentCommand\@lwtverb@makekern{}{%
```

Allow LaTeX to remove spaces automatically at the beginning and at the end of the line; use unstarred `\hspace` version for this purpose

```
87 \hspace*{0pt%
88     plus \option{/lwtverb/poskrn}%
89     minus \option{/lwtverb/negkrn}}%
90 }
```

(End definition for \@lwtverb@makekern. This function is documented on page ??.)

`\@lwtverb@makespacekern` Creates properly-sized `\hspace` from (implicitly passed) options. This space is put between letter and decorated space to make them shirnkable and stretchable.

```
91 \DeclareDocumentCommand\@lwtverb@makespacekern{}{%
```

Use starred (non-breakable) version of `\hspace` to allow the following discretionary to do its job.

```
92 \hspace{0pt%
93     plus \option{/lwtverb/poskrn}%
94     minus \option{/lwtverb/negkrn}}%
95 }
```

(End definition for \@lwtverb@makespacekern. This function is documented on page ??.)

`\@lwtverb@makediscretionary` Creates discrtectionary from (implicitly passed) options that is used to indicate line break.

```
96 \DeclareDocumentCommand\@lwtverb@makediscretionary{}{%
97     \discretionary{%
98         \hbox{\option{/lwtverb/breaksymr}}}%
99     }{%
100        \hbox{\option{/lwtverb/breaksyml}}}%
101     }{}%
102 }
```

(End definition for \@lwtverb@makediscretionary. This function is documented on page ??.)

`\@lwtverb@decoratedspace` Creates a box to be used in place of the actual space character of the input. Unlike `\@lwtverb@makespace`, takes `spacebox` option into account.

#1 : The space character (ignored)

```
103 \DeclareDocumentCommand\@lwtverb@decoratedspace{m}{%
104     \ifoptionvoid{/lwtverb/spacebox}{%
105         \ifnum\option{/lwtverb/gobble/@ord}=0% verbatim
106             \allowbreak%
107             \@lwtverb@makehardspace%
108         \else%
```

```

109         \@lwtverb@makespace%
110     \fi%
111 }{%
112     \@lwtverb@makespacekern%
113     \allowbreak%
114     \option{/lwtverb/spacebox}%
115 }%
116 }

```

(End definition for \@lwtverb@decoratedspace. This function is documented on page ??.)

\@lwtverb@decoratednonspace Creates a box to be used in place of the actual non-space character of the input. Takes decoration option into account and does not apply any interletter spacing.

#1 : The non-space character

```

117 \DeclareDocumentCommand\@lwtverb@decoratednonspace{m m}{%
118     \ifoptionvoid{/lwtverb/decoration2}{%
119         \option{/lwtverb/decoration}{#1}%
120     }{%
121         \option{/lwtverb/decoration2}{#1}{#2}%
122     }%
123 }

```

(End definition for \@lwtverb@decoratednonspace. This function is documented on page ??.)

\@lwtverb@breakablechar This macro actually does the job of inserting the interletter glue, interword spacing and potential line breaks. Almost all options passed to `\lwtverb` and other commands of `lwtverb` are handled here.

#1 : Current character to be typeset and decorated

#2 : Previous character (`\relax` for the first)

The previous character is used to detect word boundaries in order not to insert discretionary right before the space. The motivation for this is simple — otherwise hyphen (or in general a pair of any two symbols) may be inserted right before the space which is certainly unpleasant in most cases.

```

124 \DeclareDocumentCommand\@lwtverb@breakablechar{m m}{%

```

Handle spaces separately. If there is a `spacebox` option set, we treat space as a regular character except we should not insert hyphenation marks.

```

125     \ifx\@sptoken#1%
126         \ifcase\option{/lwtverb/gobble/@ord}%
127             % verbatim
128             \@lwtverb@decoratedspace{#1}%
129         \or% no
130             \@lwtverb@decoratedspace{#1}%
131         \or% extra
132             \ifx\@sptoken#2%
133                 \else%
134                     \@lwtverb@decoratedspace{#1}%
135                 \fi%

```

```

136         \else% all
137         \fi%

```

Handle all other non-space characters. Insert additional space between two non-space characters (i.e. if the previous character is not a space) and add discretionary to allow line break.

```

138     \else%
139         \ifx\sptoken#2%
140         \else%
141             \@lwtverb@makekern%
142             \option{/lwtverb/breakhandler}{#1}{#2}{\@lwtverb@make discretionary}%
143         \fi%
144         \@lwtverb@decoratednonspace{#1}{#2}%
145     \fi%
146 }

```

(End definition for \@lwtverb@breakablechar. This function is documented on page ??.)

9.4 Input Processing Routine

\@lwtverb@process Highly inspired by [this StackExchange answer](#). Defines internal input processing routine, which effectively iterates over the input and calls \@lwtverb@breakablechar for each character read. The macro itself specifies necessary styles and feeds the \detokenized input to that routine. This macro is intended to be used in tandem with \Collectverb from newverbs (or its equivalent). See \lwtverb for example of such usage.

#1 : List of options
#2 : Introductory command to be issued (e.g. \tt or \texttt)
#3 : Finalization command
#4 : The input to be typeset

```

147 \DeclareDocumentCommand\@lwtverb@process{m m m m}{%
148     {\options{/lwtverb,#1}%
149     \def\gobblechar{\let\xchar= }%
150     \def\assignthencheck{%
151         \afterassignment\xloop\gobblechar}%
152     \let\xprevchar=\relax%
153     \def\xloop{%
154         \ifx\relax\xchar%
155             \let\next=\relax%
156         \else%
157             \@lwtverb@breakablechar{\xchar}{\xprevchar}%
158             \let\xprevchar= \xchar\relax%
159             \let\next=\assignthencheck%
160         \fi%
161         \next}%
162     {#2{\expandafter\assignthencheck\detokenize{#4}\relax}#3}}%
163 }

```

(End definition for \@lwtverb@process. This function is documented on page ??.)

9.5 User-exposed commands

`\lwtverb` The main command provided to the end user. Takes optional list of arguments. The command is fully analogous to `\verb` in a sense it takes its (last) argument. See usage examples above.

```
\lwtverb[options](char)(input)(char)
```

#1 : List of options

#2 : The input to be typeset

```
164 \DeclareDocumentCommand\lwtverb{0{}}{%
165   \Collectverb{\@lwtverb@process{#1}{\tt}{}}%
166 }
```

(End definition for \lwtverb. This function is documented on page ??.)

`\lwtcode` Same as `\lwtverb` but defaulted to mark wrapped lines instead of just breaking them.

```
\lwtcode[options](char)(input)(char)
```

#1 : List of options

#2 : The input to be typeset

```
167 \DeclareDocumentCommand\lwtcode{0{}}{\lwtverb[w,#1]}
```

(End definition for \lwtcode. This function is documented on page ??.)

`\lwttt` Same as `\lwtverb` but takes its (last) argument as a normal command does (i.e. in `{...}`). It may be very useful for typesetting standalone words without special characters because of its natural syntax.

```
\lwtverb[options]{(input)}
```

#1 : List of options

#2 : The input to be typeset

```
168 \DeclareDocumentCommand\lwttt{0{}} m{%
169   \@lwtverb@process{#1}{\tt}{#2}%
170 }
```

(End definition for \lwttt. This function is documented on page ??.)

9.6 Other helpful commands

`\@lwtverb@justify` See [this StackExchange answer](#) for details.

```

171 \newcommand*\@lwtverb@justify{%
172   \fontdimen2\font=0.4em% interword space
173   \fontdimen3\font=0.2em% interword stretch
174   \fontdimen4\font=0.1em% interword shrink
175   \fontdimen7\font=0.1em% extra space
176   % \hyphenchar\font='- fails in book documentclass for some reason
177   \hyphenchar\font='- % allowing hyphenation
178 }

```

(End definition for \@lwtverb@justify. This function is documented on page ??.)

`\justtt` Alternative approaches to justified typewriter text based on just `\lwtverb@justify`.

```

\justtt{<input>}
#1 : The input to be typeset
179 \DeclareDocumentCommand\justtt{m}{\tt\@lwtverb@justify{#1}}

```

(End definition for \justtt. This function is documented on page ??.)

`\jusverb` Alternative approaches to justified typewriter text based on `\lwtverb@justify` + `\detokenize`.

```

\jusverb{<input>}
#1 : The input to be typeset
180 \DeclareDocumentCommand\jusverb{m}{\tt\@lwtverb@justify{\detokenize{#1}}}

```

(End definition for \jusverb. This function is documented on page ??.)