The lwtverb package

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

July 18, 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`), extra fragile (cannot be used in section title, captions, etc.) 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.

lwtverb tries to provide *robust* (just as `\Verb` with the help of fvextra is) command for *breakable* and *justifyable* inline verbatim text. What follows next is side-by-side comparison of different `\lwtverb` variants and `\lwtverb` with other viable approaches.

Section 2 describes features lwtverb 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 lwtverb limitations.

2 Features

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

- Justifyable 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 thay 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 replaces with custom characters so they can be highlighted and thus preserved.
- The implementation higly relies on fvextra, so curly-braced versions of `\lwtverb` and `\lwtcode` commands are robust just as curly-braced version of `\Verb`. Other provided commands (`\lwttt`, `\justtt`) are initially robust.

3 Examples

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

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

`\lwttt[b]{...}`	`\lwttt[h]{...}`	`\lwttt[w]{...}`
`Lorem ipsum dolor sit amet, consectetuer adi piscing elit. Ut purus elit, vestibulum ut, p lacerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonumm y eget, consectetuer id, vulputate a, magna.`	`Lorem ipsum dolor sit amet, consectetuer ad- ipiscing elit. Ut purus elit, vestibulum ut, p- lacerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonu- mmy eget, consectetuer id, vulputate a, magna.`	`Lorem ipsum dolor sit amet, consectetuer adip ↪iscing elit. Ut purus elit, vestibulum ut, pl ↪acerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonum ↪my eget, consectetuer id, vulputate a, magna.`

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

\lwtverb|...|	\verb|...|	\Verb[breaklines, brea ↪kanywhere]|...|
Lorem ipsum dolor sit amet, consectetuer adi piscing elit. Ut purus elit, vestibulum ut, p lacerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonumm y eget, consectetuer id, vulputate a, magna.	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna.	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vesti␣ bulum ut, placerat ac, adipiscing vit␣ ae, felis. Curabitur dictum gravida maur␣ is. Nam arcu libero, nonummy eget, consec␣ tetuer id, vulputate a, magna.
With superduperveeerylo oong words and words_wi th_special_characters.	With superduperveeerylooong words and words_with_special_characters.	With superduperveeerylooong words and words_with_special_char␣ eryloooong words and words_with_special_c␣ haracters.

Table 3: Comparison to LaTeX's \verb and fancyvrb's \Verb (with breaklines option from fvextra)

\lwtttt{...}	\justtt{...}	\xurltt{...}
Use other text with superduperveeeryloooong words and words_with_special_characters, e.g. urls: http://example.com/hard-to-break-it-properly. Moreover, look at ‘‘<<’’ and ‘‘>>’’ symbols, they look different.	Use other text with superduperveeery-looong words and words_with_special_-characters, e.g. urls: http://example.com/hard-to-break-it-properly. Moreover, look at "«" and "»" symbols, they look different. *(needs manual escaping)*	Use other text with superduperveeeryloooong words and words$_{\text{w}}$ith$_{\text{s}}$pecial$_{\text{c}}$haracters, e.g. urls: http://example.com/hard-to-break-it-properly. Moreover, look at ‘‘<<’’ and ‘‘>>’’ symbols, they look different.
\commands , e.g. \LaTeX {}, may produce extra spaces.	\commands, e.g. \LaTeX{}, may produce extra spaces. *(needs manual escaping)*	\commands, e.g. \LaTeX{}, may produce extra spaces.
\lwtverb|...|	\justverb{...}	\xurltt|...|
\commands, e.g. \LaTeX{}, may produce extra spaces.	\commands , e.g. \LaTeX {}, may produce extra spaces.	\commands, e.g. \LaTeX{}, 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*

\lwttt[w,poskrn=0.5em]{...}	\lwttt[w,negkrn=0.5em]{...}	\lwttt[w,monospaced]{...}
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna.	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna.	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibul↪um ut, placerat ac, adipiscing vitae, fel↪is. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna.

Table 5: Same text typeset with lwtverb's \lwttt with interletter spacing adjusted differently

\lwttt[w,m,possp=6em]{↪...}	\lwttt[w,m,negsp=1em]{↪...}	\lwttt[w,monospaced]{↪..}
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, feli↪s. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna.	Loremipsumdolorsitamet, consectetueradipiscing elit. Ut purus elit, ve↪stibulum ut, placerat ac, adipiscing vitae, fe↪lis. Curabitur dictum gravida mauris. Nam arcu libero,nonummyeget,con↪sectetuerid,vulputate a, magna.	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibul↪um ut, placerat ac, adipiscing vitae, fel↪is. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna.

Table 6: Same text typeset with lwtverb's \lwttt with interword spacing adjusted differently

Lorem ipsu•m dolor s•it amet, c•onsectetu•er adipis•cing elit•. Ut purus elit, vest•ibulum ut•, placera•t ac, adi•piscing v•itae, fel•is. Curab•itur dict•um gravida mauris. Na•m arcu li•bero, non•ummy eget, consectetu•er id, vu•lputate a, magna.	Lorem ipsum d•olor sit amet•, consectetuer 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, consectetuer i•d, vulputate a, magna.	Lorem ipsum dolor sit amet, consec•tetuer adipiscing elit. Ut purus el•it, vestibulum ut, placerat ac, adipi•scing vitae, felis. Curabitur dictum g•ravida mauris. Nam arcu libero, nonummy eget, consectetue•r id, vulputate a, magna.	Lorem ipsum dolor sit amet, consectetuer adi•piscing elit. Ut purus elit, vestibulum ut, p•lacerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonum•my eget, consectetuer id, vulputate a, magna.

Table 7: Same text in columns of different width

`r1 = x;  // A`
`if (r1 != 0) // B`
`  y = 1;  // C`
	`r1 = x;  // A`
	`if (r1 != 0) // B`
	`  y = 1;  // C`
		`r1·=·x;·····//·A`
		`if·(r1·!=·0)·//·B`
		`····y·=·1;···//·C`

Table 8: This example demonstrates the power of `monospaced` option (note that `A`, `B` and `C` are properly aligned) and compares it with the result of default and `showspaces` options. The table itself is typeset using fixed-width `p{...}` columns. The issue with default options originates from different space widths used in default and `monospaced`-like styles

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}`.

`\lwtverb` is based on fvextra. It provides two versions of `\lwtverb` syntax:

```
\lwtverb[⟨opts⟩]⟨char⟩⟨input⟩⟨char⟩
\lwtverb[⟨opts⟩]{⟨input⟩}
```

The latter version should almost always be preferred because it much less fragile. It, however, has some limitations, e.g. may fail to properly typeset `\commands`, cannot have unpaired curly braces in argument, may gobble spaces. See section 7 and fvextra manual for more details.

`\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 typesetting inline code sensitive to line breaks.

`\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}`.

`\justttt` `\justttt` is like a normal `\texttt` but adds shrinking, stretching and (proper lexical) hyphenation support by default. Not a verbatim command. Doesn't take any options.

`\justverb` `\jusverb` is like a `\justttt` 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*⟩
`decoration2=`⟨*cmd2*⟩
`spacebox=`⟨*hbox*⟩

`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 characher. 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_act
↪ually_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. Spa
↪ces: 1[ ], 2[  ], 3[   ], 4[
 ], 5[    ]. Note that breaks at
 spaces are not marked with arrow
 sign, but_long_words_that_actual
↪ly_break_somewhere_still_are.
```

```
Text with a number of words. Spa
↪ces: 1[ ], 2[ ], 3[ ], 4[ ], 5[
]. Note that breaks at spaces a
↪re not marked with arrow sign,
but_long_words_that_actually_brea
↪k_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 <fille
↪r_text_that_should_wrap>|
\lwtverb[poskrn=1em, negkrn=0.3em]|This_text_must_be_very_tight <anoth
↪er_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 <anothe
↪r_filler>|
```

```
Spaces here must be reasonably wide
<filler_text_that_should_wrap>
Spaces here must be very short <another_filler>
```

spwidth=⟨*length*⟩ Specifies interword space width in normal and verbatim gobble mode.

hardspwidth=⟨*length*⟩ 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.

debug Synonym of decoration=\colorbox{lightgray}.

`breakline, b` Synonym of `breaksymr={}, breaksyml={}`.

`wrapline, w` Synonym of `breaksymr={\tiny$\hookrightarrow$}, breaksyml={}`.

`hyphenate, h` Synonym of `breaksymr={-}, breaksyml={}`.

`monospaced, m` All spacings are non-adjustable, all spaces are as in verbatim mode.

`showspaces` Installs `spacebox` with a small central dot.

`obeyspaces` Synonym of `gobble=verbatim`.

`nospaces` Synonym of `gobble=all`.

6 Comparison to other approaches

There are plenty of packages and/or tecniques 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.

LaTeX's `\verb`
: Does not allow line breaks inside words. May cause overfull hboxes. Very fragile.

fancyvrb's `\Verb`
: With the help of fvextra supports line breaks anywhere in the string and can indicate breaks appropriately. Not justifyable — produces too ragged result. Without `breakanywhere` option may cause overfull hboxes with long enough words. See examples above. fancyvrb and fvextra provide `\SaveVerb`, `\UseVerb` and many more useful commands to cope with plain `\Verb` limitations. lwtverb does not provide such mechanism.

minted's `\mintinline`
: Does not allow line breaks in 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 can 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 accout 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 an 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 tries to follow fvextra implementation in order to provide robust variant of `\lwtverb`.
- `\lwtverb|...|` may exibit a bit strange behavior when passed as an argument to other commands (e.g. `\id{\lwtverb|{\LaTeX  }|}` actually becomes "`\LaTeX`" instead of desired "`{\LaTeX  }`", where `\id` is defined as follows: `\def\id#1{#1}`. `\lwtverb{...}` survives and produces almost desired output "`{\LaTeX}`" but fails to preserve spaces after `\LaTeX`.

8 Historical notes

The package was originally implemented on top of newverbs. It provides very simple and straightforward approach to verbatim commands — one just needs to feed his command to `\Collectverb` which just passes collected input as an argument to provided command. It, however, is as fragile as `\verb` is, so it is hardly acceptible.

Implementation on top of plain fancyvrb was much less fragile, but it still was far from what it might be.

Current fvextra-based approach finally won and was adopted here.

9 Implementation

9.1 Package Options

Additional stretching to be added after each non-space char

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

Additional shrinking to be added after each non-space char

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

Additional stretching to be added after each space char

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

Additional shrinking to be added after each space char

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

Width of a regular space

```
\DeclareOptionX{spwidth}{%
    \DeclareDocumentCommand\@lwtverb@spwidth{}{#1}}
```

Width of space in verbatim mode

```
\DeclareOptionX{hardspwidth}{%
    \DeclareDocumentCommand\@lwtverb@hardspwidth{}{#1}}
```

Handle unknown options

```
\DeclareOptionX*{%
    \PackageWarning{lwtverb}{`\CurrentOption' ignored}}
```

Some reasonable defaults

```
\ExecuteOptionsX{
    poskrn=0.3pt,       % higher values result in 'a  b  c  d' (too loose)
    negkrn=0.3pt,       % higher values result in 'abcd' (too tight)
    possp=0.3em,        % higher values result in 'abcd   efgh'
    negsp=0.1em,        % higher values result in 'abcdefgh'
    spwidth=0.4em,
    hardspwidth=0.5em,
}
```

Done with package options

```
\ProcessOptionsX\relax
```

9.2 Command Options

```
\long\def\@firstoffour#1#2#3#4{#1}
\long\def\@fourthoffour#1#2#3#4{#4}

\options{
    /lwtverb/.new family,
    % options
    /lwtverb/decoration/.new value      = \relax,
    /lwtverb/decoration2/.new value     = \relax,
    /lwtverb/spacebox/.new value        = \relax,
```

```
    /lwtverb/breaksymr/.new value       = {},
    /lwtverb/breaksyml/.new value       = {},
    /lwtverb/poskrn/.new value          = \@lwtverb@poskrn,
    /lwtverb/negkrn/.new value          = \@lwtverb@negkrn,
    /lwtverb/possp/.new value           = \@lwtverb@possp,
    /lwtverb/negsp/.new value           = \@lwtverb@negsp,
    /lwtverb/spwidth/.new value         = \@lwtverb@spwidth,
    /lwtverb/hardspwidth/.new value     = \@lwtverb@hardspwidth,
    /lwtverb/gobble/.new choice         = { verbatim, no, extra, all },
    /lwtverb/gobble                     = no,
    /lwtverb/breakhandler/.new value    = \@fourthoffour\relax,
    % styles
    /lwtverb/debug/.new style* = {
        /lwtverb/decoration = \colorbox{lightgray},
    },
    /lwtverb/breakline/.new style* = {
        /lwtverb/breaksymr = {},
        /lwtverb/breaksyml = {},
    },
    /lwtverb/wrapline/.new style* = {
        /lwtverb/h,
        /lwtverb/breaksymr = {},
        /lwtverb/breaksyml = {\tiny$\hookrightarrow$},
    },
    /lwtverb/hyphenate/.new style* = {
        /lwtverb/breaksymr = {-},
        /lwtverb/breaksyml = {},
    },
    /lwtverb/monospaced/.new style* = {
        /lwtverb/poskrn     = 0pt,
        /lwtverb/negkrn     = 0pt,
        /lwtverb/possp      = 0pt,
        /lwtverb/negsp      = 0pt,
        /lwtverb/spwidth    = \@lwtverb@hardspwidth,
    },
    /lwtverb/showspaces/.new style* = {
        /lwtverb/spacebox=\hbox to \@lwtverb@hardspwidth {%
            \textcolor{gray}{$\hfill\cdot\hfill$}},
    },
    /lwtverb/obeyspaces/.new style* = {
        /lwtverb/gobble=verbatim,
    },
    /lwtverb/nospaces/.new style* = {
        /lwtverb/gobble=all,
    },
    /lwtverb/keepwords/.new style* = {
        /lwtverb/breakhandler = \@firstoffour\relax,
    },
    % shorthands
    /lwtverb/b/.new style* = { /lwtverb/breakline },
    /lwtverb/w/.new style* = { /lwtverb/wrapline },
    /lwtverb/h/.new style* = { /lwtverb/hyphenate },
    /lwtverb/m/.new style* = { /lwtverb/monospaced }
}
```

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.

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

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

```
    \hspace{\option{/lwtverb/spwidth}%
        plus \option{/lwtverb/possp}%
        minus \option{/lwtverb/negsp}}%
}
```

(*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.

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

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

```
    \hspace*{\option{/lwtverb/hardspwidth}%
        plus \option{/lwtverb/possp}%
        minus \option{/lwtverb/negsp}}%
}
```

(*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.

```
\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

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

(*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.

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

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

```
    \hspace{0pt%
        plus \option{/lwtverb/poskrn}%
        minus \option{/lwtverb/negkrn}}%
}
```

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

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

```
\DeclareDocumentCommand\@lwtverb@makediscretionary{}{%
    \discretionary{%
        \hbox{\option{/lwtverb/breaksymr}}%
    }{%
        \hbox{\option{/lwtverb/breaksyml}}%
    }{}%
}
```

(*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)
`#2` : The previous character

```
\DeclareDocumentCommand\@lwtverb@decoratedspace{m m}{%
    \ifoptionvoid{/lwtverb/spacebox}{%
        \ifnum\option{/lwtverb/gobble/@ord}=0% verbatim
```

Forbid break at first char as it may cause extra break in some rare cases (e.g. in `p{}` tabular cells).

```
            \ifx\relax#2%
            \else%
                \allowbreak%
            \fi%
            \@lwtverb@makehardspace%
        \else%
            \@lwtverb@makespace%
        \fi%
    }{%
```

Same trick here leads to the problem solved above... so leave it as is.

```
        \@lwtverb@makespacekern%
        \allowbreak%
        \option{/lwtverb/spacebox}%
    }%
}
```

(*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

```
\DeclareDocumentCommand\@lwtverb@decoratednonspace{m m}{%
    \ifoptionvoid{/lwtverb/decoration2}{%
        \option{/lwtverb/decoration}{#1}%
    }{%
        \option{/lwtverb/decoration2}{#1}{#2}%
    }%
}
```

(*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 characher (`\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.

```
\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.

```
    \ifx\@sptoken#1%
        \ifcase\option{/lwtverb/gobble/@ord}%
            % verbatim
            \@lwtverb@decoratedspace{#1}{#2}%
        \or% no
            \@lwtverb@decoratedspace{#1}{#2}%
        \or% extra
            \ifx\@sptoken#2%
            \else%
                \@lwtverb@decoratedspace{#1}{#2}%
            \fi%
        \else% all
        \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.

```
    \else%
        \ifx\@sptoken#2%
        \else%
            \@lwtverb@makekern%
```

```
            \option{/lwtverb/breakhandler}{#1}{#2}{\@lwtverb@makediscretionary}%
        \fi%
        \@lwtverb@decoratednonspace{#1}{#2}%
    \fi%
}
```

(*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 `\detokenize`d 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

```
\DeclareDocumentCommand\@lwtverb@process{m m m m}{%
    {\options{/lwtverb,#1}%
        \def\gobblechar{\let\xchar= }%
        \def\assignthencheck{%
            \afterassignment\xloop\gobblechar}%
        \let\xprevchar=\relax%
        \def\xloop{%
            \ifx\relax\xchar%
                \let\next=\relax%
            \else%
                \@lwtverb@breakablechar{\xchar}{\xprevchar}%
                \let\xprevchar= \xchar\relax%
                \let\next=\assignthencheck%
            \fi%
            \next}%
        {#2{\expandafter\assignthencheck\detokenize{#4}\relax}#3}}%
}
```

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

`\@lwtverb@fvtrampoline` Mimics `\Verb` implementation from fvextra but short-circuit it to our `\@lwtverb@process`. The implementation is just a thin (but important) wrapper around `\@lwtverb@fvextra`. This and the following macro are copy-paste from fvextra with all irrelevant code removed and arguments lwtverb needs added.

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

```
\def\@lwtverb@fvtrampoline#1#2#3{%
    \begingroup%
    \expandafter\endgroup\expandafter\@lwtverb@fvextra{#1}{#2}{#3}%
}
```

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

`\@lwtverb@fvextra`

```
\def\@lwtverb@fvextra#1#2#3{%
    \ifbool{FVExtraRobustCommandExpanded}{%
        \@ifnextchar\bgroup%
            {\@lwtverb@fvextra@i{#1}{#2}{#3}}%
            {\PackageError{lwtverb}%
                {\string\lwtverb\space delimiters must be paired curly braces in this context
                {Use curly braces as delimiters}}%
    }{%
        \@lwtverb@fvextra@i{#1}{#2}{#3}%
    }%
}
```

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

`\@lwtverb@fvextra@i`

```
\def\@lwtverb@fvextra@i#1#2#3{%
    \begingroup%
    \FVExtraReadVArg{%
        \FV@FormattingPrep%
        \FVExtraDetokenizeVArg{%
            \FVExtraRetokenizeVArg{\@lwtverb@fvextra@ii{#1}{#2}{#3}}{\FV@CatCodes}}%
    }%
}
```

(*End definition for* `\@lwtverb@fvextra@i`. *This function is documented on page* ??.)

`\@lwtverb@fvextra@ii` Receives already collected argument in the last parameter and passes all to `\@lwtverb@process`. This macro is the final point of `\@lwtverb@fvextra` implementation.

```
\def\@lwtverb@fvextra@ii#1#2#3#4{%
  \@lwtverb@process{#1}{#2}{#3}{#4}%
  \endgroup%
}
```

(*End definition for* `\@lwtverb@fvextra@ii`. *This function is documented on page* ??.)

`\@lwtverb@fvextra@robust` Following fvextra's `\Verb` implementation, define robust version of `\@lwtverb@fvtrampoline`.

```
\protected\def\@lwtverb@fvextra@robust#1#2#3{\@lwtverb@fvtrampoline{#1}{#2}{#3}}
\FVExtrapdfstringdefDisableCommands{%
    \def\@lwtverb@fvextra@robust#1#2#3{}}
```

(*End definition for* `\@lwtverb@fvextra@robust`. *This function is documented on page* ??.)

`\@lwtverb@fvtrampoline@robust` What should be used instead of just `\@lwtverb@fvtrampoline`.

```
\def\@lwtverb@fvtrampoline@robust#1#2#3{%
    \def\processor{\@lwtverb@fvextra@robust{#1}{#2}{#3}}%
    \FVExtraRobustCommand\processor\FVExtraUnexpandedReadStarOArgBVArg%
}
```

(*End definition for* `\@lwtverb@fvtrampoline@robust`. *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⟩
\lwtverb[⟨options⟩]{⟨input⟩}
```

`#1` : List of options
`#2` : The input to be typeset

The letter syntax should be preferred. See section 7 for details.

```
\DeclareDocumentCommand\lwtverb{O{}}{%
    \@lwtverb@fvtrampoline@robust{#1}{\tt}{}%
}
```

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

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

```
\lwtcode[⟨options⟩]⟨char⟩⟨input⟩⟨char⟩
\lwtcode[⟨options⟩]{⟨input⟩}
```

`#1` : List of options
`#2` : The input to be typeset

```
\DeclareDocumentCommand\lwtcode{O{}}{\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

```
\DeclareDocumentCommand\lwttt{O{} m}{%
    \@lwtverb@process{#1}{\tt}{}{#2}%
}
```

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

9.6 Other helpful commands

`\@lwtverb@justify` See this StackExchange answer for details.

```
\newcommand*\@lwtverb@justify{%
  \fontdimen2\font=0.4em% interword space
  \fontdimen3\font=0.2em% interword stretch
  \fontdimen4\font=0.1em% interword shrink
  \fontdimen7\font=0.1em% extra space
  % \hyphenchar\font=`\- fails in book documentclass for some reason
  \hyphenchar\font=`-% allowing hyphenation
}
```

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

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

`\justttt{`⟨*input*⟩`}`

`#1` : The input to be typeset

```
\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

```
\DeclareDocumentCommand\justverb{m}{{\tt\@lwtverb@justify{\detokenize{#1}}}}
```

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