

TypoG – Typographic Fine-Tuning

Ch. L. Spiel*

v0.2 2024/03/29

Abstract

Package `typog` provides macros and environments for (micro-)typographic enhancements. It also supplies some means to avoid common typographic problems as, for example, orphan or widow lines. Moreover it supplies a high-level front-end of package `microtype`.



This package is copyright © 2024 Ch. L. Spiel. It may be distributed and/or modified under the conditions of the L^AT_EX Project Public License (LPPL), either version 1.3c of this license or – at your option – any later version. This work has the LPPL maintenance status »author-maintained«.

* cspiel@users.sourceforge.org

Contents

List of Tables	iii
1 Introduction	1
1.1 Overview	1
1.2 Prerequisites	1
2 Package Options/Configuration	2
3 Macros and Environments	4
3.1 Information	5
3.1.1 Font Information	5
3.1.2 Paragraph- and Page-Breaking Trace	5
3.2 Hyphenation	9
3.3 Disable/Break Ligatures	12
3.4 Manual Italic Correction	13
3.5 Apply Extra Kerning	14
3.5.1 Slash	14
3.5.2 Hyphen	15
3.6 Raise Selected Characters	15
3.6.1 Capital Hyphen	16
3.6.2 Capital Dash	16
3.6.3 Number Dash (Figure Dash)	17
3.6.4 Multiplication Sign	18
3.6.5 Guillemets	18
3.7 Align Last Line	20
3.8 Fill Last Line	21
3.8.1 Problem Definition	21
3.8.2 Manual Changes	22
3.8.3 Multi-Purpose Environments	24
3.8.4 Specialized Environments	24
3.9 Spacing	25
3.9.1 Looser/Tighter	25
3.9.2 Wide Space	26
3.9.3 Narrow Space	27
3.10 Microtype Front-End	28
3.10.1 Tracking	28
3.10.2 Font Expansion	29
3.10.3 Character Protrusion	31
3.11 Sloppy Paragraphs	31
3.12 Vertically Partially-Tied Paragraphs	33
3.13 Breakable Displayed Equations	36
3.14 Smooth Ragged	38

Table of Contents continued on next page.

4 Other Packages for Fine L^AT_EX Typography	41
A Package Code	42
A.1 Information	46
A.2 Hyphenation	47
A.3 Disable/Break Ligatures	48
A.4 Manual Italic Correction	49
A.5 Apply Extra Kerning	50
A.6 Raise Selected Characters	51
A.7 Align Last Line	55
A.8 Fill Last Line	55
A.9 Spacing	57
A.10 Microtype Front-End	59
A.11 Sloppy Paragraphs	62
A.12 Vert. Tie Paragraphs	64
A.13 Breakable Disp. Eqs.	69
A.14 Smooth Ragged	69
B typog-grep	75
Change History	80
References	81
Index	83

List of Tables

1	Hyphens and automatic hyphenation	10
2	Suggested raise amounts for <code>\figuredash</code>	17
3	Suggested raise amounts for guillemets	19
4	Spacing changes made by <code>loosespacing</code>	25
5	Spacing changes made by <code>tightspacing</code>	26
6	<code>\fontdimen<number></code> parameters	26
7	Comparison of some space sizes	28
8	Shrink values of <code>setfontshrink</code>	29
9	Stretch values of <code>setfontstretch</code>	29
10	Shrink and stretch values of <code>setfontexpand</code>	30
11	Parameter adjustments of <code>\slightlyloppy</code>	32
12	Partial paragraph line counts	35
13	Env. <code>breakabledisplay</code> and <code>\interdisplaylinepenalty</code>	36

1 Introduction

»Good typography« is the minimum acceptable solution;
 »fine typography« is what we aspire to.
 — Ilene Strizver

L^AT_EX is the start of good typesetting not the end. This package provides some tools for even better looking documents. When applied correctly its effects appear subtle and inconspicuous.

1.1 Overview

Package typog focuses on (micro-)typographic improvements.

The first section (Sec. 3.1) tends to the wish for more information in the typesetting process whether during the draft phase or in the final printed manuscript. The second part, Sec. 3.2, expands the hyphenation facilities of L^AT_EX. The third part, Secs. 3.3 to 3.6, deals with positioning glyphs in a more pleasant way. The fourth part discusses dearly missed macros for better control of the last line of a paragraph (Secs. 3.7 and 3.8). Part five covers the manipulation of the length of a paragraph: spacing (Sec. 3.9), font tracking (Sec. 3.10.1), and font expansion (Sec. 3.10.2). In the sixth part we address some shortcomings of spacing control, in particular a replacement for the macro `\sloppy` (Sec. 3.11). Section 3.12 presents special functions to avoid club or widow lines in a paragraph. As a simple extension of displayed mathematical equations we define a breakable variant in the seventh part (Sec. 3.13).

In the last part, Sec. 3.14, we introduce a novel way of generating ragged paragraphs, which still is experimental.

1.2 Prerequisites

Package typog requires ϵ -T_EX; it relies on the L^AT_EX3 interface. Parts of it are based on package microtype. However, if the respective functionality is not used, typog can be applied without microtype.

The package was tested with pdfTeX 3.141592653-2.6-1.40.24 from the TeX Live distribution of 2022 as shipped by Debian.

Throughout the whole document we indicate actual uses of the package's features in the margin. All these notes are examples themselves as they are typeset with `slightly-sloppy`, `loosespacing`, and `smoothraggedrightpar`. ¶ The title page has already demonstrated the effect of `last-linecenteredpar` in justified paragraphs for the abstract and the copyright notice.

2 Package Options/Configuration

Package `typog` does not override any existing macros or environments when loaded, unless explicitly told by a package option.

```
\usepackage[...]{microtype} % Only required for macros and
                             % environments in Secs. 3.10.1 and 3.10.2.

\usepackage[⟨OPTION⟩...]{typog}
```

Package `⟨OPTIONS⟩`/Configuration `⟨key⟩`s

The package `⟨OPTIONS⟩` serve as configuration `⟨key⟩`s, too. This means they can be set with `typogsetup` and their values can be retrieved with `\typogget`. Options that rely on package `microtype` are indicated with »microtype req.«.

`breakpenalty=⟨penalty⟩`

Penalty for a line break at various points. Default value: 50; initialized by the current `\exhyphenpenalty`: 50.

`debug, nodebug`

Write package-specific debug information to the log file. Opposite: `nodebug`. The default is not to log debug information.

`ligaturekern=⟨dim⟩`

Set `⟨dim⟩` of the kern that is inserted to split a ligature in macro `\nolig`. See Sec. 3.3. Default value: $\frac{3}{1000}$ em.

`mathitalicscorrection=⟨dim⟩`

Italics correction in math mode. See Sec. 3.4 and also the complementary configuration option `textitalicscorrection`. Default value: 0.4mu.¹

`raise*=⟨dim⟩`

Set the length by which selected characters (dash, hyphen, times, and number dash) are raised. Default value: 0pt.

Only the raise amounts for guillemets are unaffected by this option.

`raisecapitaldash=⟨dim⟩`

Set the length that the `\textendash` is raised in `\capitaldash`. See Sec. 3.6.2. Default value: 0.0pt.

`raisecapitalhyphen=⟨dim⟩`

Set the length that the hyphen character `⁂` is raised in `\capitalhyphen`. See Sec. 3.6.1. Default value: 0.0pt.

`raisecapitaltimes=⟨dim⟩`

Set the length that the multiplication symbol `⋈` is raised in `\capitaltimes`. See Sec. 3.6.4. Default value: 0.0pt.

¹ Note that 1 mu is $\frac{1}{18}$ em of the mathematical font's em.

This sub-section is type-set with all typog parameters reset to their defaults by wrapping it in a `typogsetup` environment with an empty argument.

We access the configuration values with `\typogget`.

This description list is protected against breaking items across pages within the first three lines by `vtietop`.

`raisecapitalguillemets=<dim>`

Set the length that single and double guillemets are raised in the uppercase versions of the guillemet macros. See Sec. 3.6.5. Default value: 0.0pt.

`raiseguillemets=<dim>`

Set the length that single and double guillemets are raised in the lowercase versions of the guillemet macros. See Sec. 3.6.5. Default value: 0.0pt.

`raisefiguredash=<dim>`

Set the length that the `\textendash` is raised in `\figuredash`. See Sec. 3.6.3. Default value: 0.0pt.

`shrinklimits={<limit-1>, <limit-2>, <limit-3>}` microtype req.

`stretchlimits={<limit-1>, <limit-2>, <limit-3>}` microtype req.

Set the three limits, given in $\frac{1}{1000}$ em, of shrinkability and stretchability for the respective levels. They are used in `setfontshrink` (`shrinklimits` triple only), `setfontstretch` (`stretchlimits` triple only), and `setfontexpand` (both triples of limits). See Sec. 3.10.2.

New `<limit-#>` values replace old ones. If one or more limits of the triple should remain unchanged pass a $\frac{*}{1000}$ instead of a number.

Defaults for `shrinklimits` are 5, 10, 20 and those for `stretchlimits` are 5, 10, 20.

Both options can be used when loading the package and in the document preamble, but *not* in the document body.

`slashkern=<dim>`

Set the size of the kerns before and after `\kernedslash`. See Sec. 3.5.1. Default value: $\frac{50}{1000}$ em.

`textitalicscorrection=<dim>`

Italics correction fallback-value; used if `\fontdimen1` is zero. See Sec. 3.4 on manual italic correction and also the complementary configuration option `mathitalicscorrection`. Default value: $\frac{20}{1000}$ em.

`trackingttspacing={<outer-spacing>}` microtype req.

Set the outer spacing of all typewriter fonts if used in environment `set-tracking` as described in Sec. 3.10.1.

The argument `<outer-spacing>` gets passed to microtype's `\SetTracking` option `outer spacing` [19, Sec. 5.3]. If it contains commas, enclose the whole argument in curly braces. Default argument value: 300, 90, 60.

The option can be used when loading the package and in the document preamble, but *not* in the document body.

By default this option is unset.

3 Macros and Environments

Easy things should be easy, and
hard things should be possible.
— Larry Wall

This is the »User Manual« section of the documentation, where we describe all user-relevant macros and environments that are defined in package `typog`.

We follow the naming convention that every environment whose name ends with `...par` issues a `\par` at its end. Environments with different name suffixes never close with `\par`.

`typogsetup (env.)`

Configure the package with the given $\langle keys \rangle$. An empty argument of `typogsetup` resets all $\langle keys \rangle$ to their default values.

```
typogsetup{ $\langle keys \rangle$ }
```

The package can be (re-)configured at any point with `\typogsetup{ $\langle keys \rangle$ }`, or – for localized changes – as

```
\begin{typogsetup}{ $\langle keys \rangle$ }
...
\end{typogsetup}
```

where $\langle keys \rangle$ have the same format as the package options described in Sec. 2.

Use Cases

`\typogsetup` can substitute configuring the package at load-time or serve as an addition. ¶ Using the `typogsetup` environment allows to fine-tune the parameters for a specific use, e. g., display-sized text. ¶ It even is conceivable that a well-established typog-configuration gets attached to font-changing macros like `\rm`, `\sf`, etc. ■

`\typogget`

Sometimes the user needs to access internal configuration values of package `typog`. This can be done in a safe way without resorting to code that is bracketed by `\makeatletter/\makeatother` with the help of the following macro.

```
\typogget{ $\langle key \rangle$ }
```

Retrieve the configuration value that is associated with $\langle key \rangle$. For a list of available $\langle key \rangle$ s see Sec. 2.

Use Case

Raise glyphs by the same amount as configured with `typog`.

```
\newcommand*{\seesubst}
{\raisebox{\typogget{raisecapitalguillemets}}%
{\rightarrowhead}}
\renewcommand*{\labelitemi}
{\raisebox{\typogget{raisecapitaldash}}{\cdot}}
```

The latter only is useful inside of an `itemize` environment of course. ■

3.1 Information

Never forget: The visual output counts;
it must always be checked, [...].
— Udo Wermuth [25]

The em-dash at then end of
the quote is height-adjusted with
`\capitalem dash*`.

We define some functions for introspection of the typesetting process.

3.1.1 Font Information

`\fontsizeinfo` Capture the font size² and line spacing³ at the point where `\fontsizeinfo` is called in macro `<cs-name>`. Both dimensions are measured in points (pt) and the results are rounded to tenths.

```
\fontsizeinfo{<cs-name>}
```

The call to `\fontsizeinfo` introduces a pair of macros to access the stored values. The unstarred version `\cs-name` expands to the lengths including their units (i. e., pt), the starred version `\cs-name*` omits the units. The separating slash is `\kerned slash`, which is introduced in Sec. 3.5.1.

Use Cases

Colophon. ¶ Font test pages. ■

3.1.2 Paragraph- and Page-Breaking Trace

`typoginspect (env.)` The environments `typoginspect` and `typoginspectpar` turn on the tracing of paragraphs and pages; optionally they display the parbox' contents. These environments can assist the user in identifying typographic problems in a quantitative way without getting distracted by unrelated information in the trace or the *log*-file.

```
typoginspect[<option>]{<id>}
typoginspectpar[<option>]{<id>}
```

The `<id>` is an arbitrary string that identifies the results in the *log*-file. If the mandatory argument is empty, `typog` constructs a unique value.

² We use `\fontdimen6`, the em-height as the font size.

³ The line spacing simply is `\baselineskip`.

Option**tracingboxes**[=*size*]

Specify the maximum box breadth and box depth reported in the log. If *size* is omitted the maximum values are assumed; this is similar to the `\tracingboxes` macro [1, p. 312].

Caution

The end-of-trace marker sometimes gets placed too early and the trace seems truncated. \LaTeX reliably logs the requested the trace information, but the write operations for trace data and `\immediate\write` which is used to print the end-tag are not synchronized. ■

\LaTeX log-file and trace. The trace data in the *log*-file is bracketed by XML-tags.

```
<typog-inspect_id="⟨id⟩" _job="⟨jobname⟩" _line="⟨line-number⟩" _page="⟨page-number⟩">
...
</typog-inspect>
```

where the *⟨id⟩* is the user-supplied, unique⁴ identifier of the group, *⟨jobname⟩* is the value of `\jobname`, *⟨line-number⟩* records the `\inputlineno` of the `\begin` of the group, and *⟨page-number⟩* gets replaced with the current value of the page counter.

- Any text tool can be used to ferret out the tags. EMACS users will find `(occur <regexp>)` to be useful.
- As long as the tags are not nested `sed` or `perl` extract the information gathered by `typoginspect`, for example:

```
sed -ne '/<typog-inspect_id="..."/,/#</typog-inspect>#p'
    < jobname.log
```

or

```
perl -ne '$a=0 if /<\/typog-inspect>/; \
    print $_ if $a; \
    $a=1 if /<typog-inspect_id="..."/' \
    < jobname.log
```

- The companion program **typog-grep** is tailored to extract the information marked up by `typoginspect` and `typoginspectpar` even if the environments are nested.

We reproduce the complete manual page of **typog-grep** in Appendix B.

⁴ It has turned out advantageous to use unique *⟨id⟩*s. However, *⟨id⟩*s are *not required* to be distinct.

Tips

- It may be necessary to run whatever L^AT_EX engine with a larger log-file line length, to prevent wrapped lines. With short lines the wannabe XML opening tags can get wrapped and thus become unrecognizable to dumb post-processors. To avoid wrapped lines prepend

```
/usr/bin/env max_print_line=2147483647
```

to the command-line. The value $2147483647 = 2^{31} - 1$ effectively disables all line wrapping by L^AT_EX.

As both **pdf_latex** and **lua_latex** support changing their configuration on a by-call basis with option `-cnf-line=⟨STRING⟩` an alternative to the above example is to add

```
-cnf-line=max_print_line=2147483647
```

to the respective command-line.

- If more trace information is needed just add `\tracing...` calls right after `\begin{typoginspect}` or `\begin{typoginspectpar}`. ■

Investigating the badness of a paragraph. It is generally unnecessary to determine the *exact* classification of a paragraph's badness [13, p. 97n], though the curious user can switch on logging of T_EX's line-break information with `\tracing-paragraphs=1`⁵ or simply use the `typoginspect` environment and check the suffixes

```
@@⟨breakpoint-number⟩ line ⟨line-number⟩.⟨suffix⟩
```

of each line in the paragraph, where for `⟨suffix⟩` the following mapping holds [13, p. 99]:

$0 \mapsto$ very loose, $1 \mapsto$ loose, $2 \mapsto$ decent, and $3 \mapsto$ tight.

Example

```
@@17: line 15.1- t=142289 s=93.58414 a=2.86073 -> @@16
```

1. The feasible breakpoint @@ number 17 in the paragraph leads to
2. line 15, which is the loose .1 last line of the paragraph.
3. Up to this breakpoint the paragraph has picked up total demerits t of 142289.
4. The following two values only show up if `\lastlinefit ≠ 0`:
 - (a) The shortfall s and
 - (b) glue a or g.⁶
5. The best⁷ way to get here, i. e., @@17 is via -> breakpoint @@ 16. ■

⁵ Reference 24 provides an exceptionally detailed discussion of the output of `\tracingparagraphs`.

⁶ The author is unaware of any descriptions of s, a, or g. The interested reader is referred to the source code, e. g., *pdf_ltex.web*; search for `print("_s=")`. In the weaved documentation the first relevant section is §1851.

⁷ ›Best‹ means the minimum-demerits path in the graph of the feasible breakpoints, which has been constructed for the paragraph.

Note

When package microtype's font expansion feature jumps in the reports on »Loose \hbox (badness ...)« and »Tight \hbox (badness ...)« contain the amount of shrinking or expansion as parenthesized values (units are thousandths of the current font's em) like, e. g.,

```
\T1/erewhon-LF/m/n/9/@/@ (-13) ...
```

or

```
\T1/erewhon-LF/m/n/9/@/@/10ls (+7) ...
```

An `ls` appended to the font name specification indicates that microtype's letter spacing feature is active and changed the tracking by that many thousands on an em as indicated before `ls`. ■

Investigating page-breaks. Use `\tracingpages=1` or the `typoginspect` environment to switch on tracing of T_EX's page-break information [13, p. 112n].⁸

The first time vertical material enters a new page, T_EX logs

```
%% goal height=<text-height>, max depth=<max-depth>
```

where `<text-height>` is the total height T_EX wants to achieve and `<max-depth>` is the maximum depth of the hbox in the last line of the page is allowed to have without considering `<text-height>` to be exceeded. For example:

```
%% goal height=598.0, max depth=5.0
```

For every vertical breakpoint T_EX records

```
% t=<total-height> g=<goal-height> b=<badness> p=<penalty>
    c=<cost>
```

Here, `<total-height>` and `<goal-height>` are the current total height of the page and the current goal height to achieve with respect to this vertical breakpoint.

The value of `<penalty>` and `<cost>` can be infinite, which would be indicated with an asterisk `*` instead of a numerical value. The best vertical breakpoint found so far on the current page is indicated by a trailing sharp-sign `#`.

Example

```
% t=351.3 plus 11.0 minus 1.0 g=553.9 b=10000 p=-300 c=100000#
```

1. At this vertical breakpoint the total page height `t` is 351.3 pt. We have picked up glue with 11 pt stretchability and 1 pt shrinkability along the way.
2. The current goal height `g` is 553.9 pt. If the initial goal height was 598 pt we can deduce that some space for other vertical material was subtracted.
3. The badness `b` of this vertical break is horrendous which is expected for the first lines on a page since breaks so early are rightfully considered infinitely bad.
4. The penalty `p` at this point actually is a bonus.
5. As the badness is 10000 the cost for a break is calculated to 100000. ■

⁸ See also the discussion of the T_EX output routines by SOLOMON [21].

All of our guillemets were raised by $\frac{33}{1000}$ em.

3.2 Hyphenation

\TeX 's and thus \LaTeX 's hyphenation algorithm is highly sophisticated, yet the document author sometimes lacks convenient macros to solve seemingly trivial typographic tasks. For example, to hyphenate a compound word connected by a hyphen.

`\allowhyphenation`

\TeX inhibits breaks of the component words by default. The following macro rectifies the problem.

```
\allowhyphenation
```

Macro `\allowhyphenation` re-enables automatic hyphenation after \TeX has turned it off, for example, in the innocuous case of a hyphenated compound.

The admittedly simple rules when \TeX auto-hyphenates and when not give rise to so many different, yet interesting cases that we devote Tab. 1 to them. The seemingly special cases shown there are not that uncommon, e. g., consider ›spin-½‹ which is coded as `\mbox{spin-\textfrac{1}{2}}`. A line break between the text and the fraction would garble the term.

Use Cases

All examples from the bottom of Tab. 1 on p. 10. ¶

Fix line breaks of index-entries in a narrow index:

```
Halbgruppe, Transformations\allowhyphenation\mbox{-}\,---
```

The first part, ›Transformations‹ is allowed to be hyphenated, but a break after the hyphen is prohibited as it results in a prowling em-dash at the beginning of the next line. ¶

Re-enable hyphenation when a macro decays into a `\hbox`:

```
Einselement\allowhyphenation\rlap{,}\footnote{...}
```

where `\rlap` is equivalent to something like `\makebox[0pt]{#1\hss}`. ¶

Use `\allowhyphenation` to turn on hyphenation of the first word of a paragraph as, e. g., in a narrow index or a `\marginpar`:

```
\marginpar{\allowhyphenation Kontakttransformationen}
```

A common trick to sweet-talk \TeX into hyphenating the first word of a paragraph is to put `\hskip0pt` in front of it. ■

Whenever using `\-`, the short-hand form of `\discretionary{-}{ }{ }`, authors writing in a foreign language should reconsider whether it really beats `\hyphenation` or `\babelhyphenation`⁹. in the particular situation. However, sometimes `\-` actually *is* the way to go.

Let us assume we mark up proper names with

```
\DeclareRobustCommand*{\propername}[1]
{\mbox{\textsc{#1}}}
```

⁹ `\babelhyphenation` is the multi-lingual extension of \TeX 's `\hyphenation` and it is defined in package `babel` [5]

TABLE 1: \TeX offers plenty of possibilities to hyphenate a compound. ¶
 We use the sample ›hyphenated-compound‹ to show various code exam-
 ples and the results that they produce. The parts are automatically hy-
 phenated like this: ›hyphenated‹ \rightarrow ›hy-phen-ated‹ and ›compound‹ \rightarrow
 ›com-pound‹.

\LaTeX -Code	Result	Note
hyphenated-compound	hyphenated- compound	Most frequently used code; the <code>hyphen</code> expands to <code>\discretionary{-}{-}{-}</code> rendering the parts un-breakable
hyphenated\mbox{-}% compound	hyphenated-compound	Suppress hyphenation with the <code>\mbox</code> in the compound
\mbox{hyphenated-% compound}	hyphenated-compound	Avoid line break and thus hyphenation
hyphenated\hyp compound	hy- phen- ated- com- pound	Macro <code>\hyp</code> defined in package <code>hyphenat</code> [31]
hyphenated% \allowhyphenation-% compound	hy- phen- ated- compound	Macro <code>\allowhyphenation</code> of package <code>typog</code> ; only unblock hyphenation of the first part
hyphenated-% \allowhyphenation compound	hyphenated- com- pound	Macro <code>\allowhyphenation</code> of package <code>typog</code> ; only unblock hyphenation of the second part
hyphenated% \allowhyphenation \mbox{-}% compound	hy- phen- ated-compound	Macro <code>\allowhyphenation</code> of package <code>typog</code> ; hyphenate first part and keep the original <code>hyphen</code> unbreakable
hyphenated% \allowhyphenation-% \allowhyphenation compound	hy- phen- ated- com- pound	Macro <code>\allowhyphenation</code> of package <code>typog</code> ; hyphenate both parts, similar to <code>\hyp</code> shown above

and we want to have hyphenatable »ABELsche Gruppe« or »EUKLIDischer Vektorraum« without dropping the markup. To that end we define commands that insert a hyphenation point at the right place:

```
\newcommand*\abelsche{
    {\propername{Abel}\-sche}
\newcommand*\euklidischer{
    {\propername{Euklid}i\-scher}
```

which are impossible to encode with `\hyphenation` or `\babelhyphenation` as these expect only letters and dashes as their arguments with spaces separating the words.

Tip — Typewriter Fonts

Sometimes it is desired to get a hyphenatable typewriter font. L^AT_EX suppresses any hyphenation for fonts in `\ttfamily` by un-defining their `\hyphenchars`. If these are reassigned, the usual hyphenation occurs again.

So, a fictitious macro ‘`\code`’ to typeset short pieces of code could look like this:

```
\newcommand*\code{[1]
    {\ttfamily
    \hyphenchar\font='-\relax #1}} ■
```

`\breakpoint`
`\breakpoint*`

The empty discretionary construct [13, p. 95], `\discretionary{}{}{}{}`, is so helpful that it deserves its own macro – with a descriptive name.

```
\breakpoint
\breakpoint*
```

The starred form inserts an empty discretionary, which disables automatic hyphenation. The unstarred form inserts an empty discretionary and immediately re-enables automatic hyphenation.

The difference between `\breakpoint` and the L^AT_EX macro `\allowbreak` is not only that the former has a starred form, but the penalty associated with `\breakpoint` is the current¹⁰ `\exhyphenpenalty`, whereas `\allowbreak` statically assigns a zero penalty.

Use Case

Prefixes that end in a hyphen inside of a pair of parenthesis:

```
\mbox{(pre-)}\breakpoint* \propername{Hilbert} space ■
```

¹⁰ At this point in the document `\exhyphenpenalty=50` holds.

3.3 Disable/Break Ligatures

`\nolig*` Break a ligature without introducing a hyphenation opportunity.

```
\nolig*[\langle kerning \rangle]
```

Inserting `\nolig*` disables a ligature at the given point by a kern. Set the size of the kern with `ligaturekern` or override this value with `\langle kerning \rangle` as thousandths of the current font's em.

Use Cases

`\nolig*` can be useful in headings, where additional hyphenation points are unwelcome. ¶ In fonts with an overly rich set of ligatures `\nolig*` offers a straightforward means to suppress unwanted ligatures at non-hyphenatable positions. ¶ Rectify the appearance of a pseudo ligature, i. e., two adjacent characters that look like a ligature, but actually are not. ■

`\nolig` Break a ligature and introduce a hyphenation opportunity.

```
\nolig[\langle kerning \rangle]
```

Inserting `\nolig` disables a ligature at the given point as `\nolig*` does and introduces a hyphenation opportunity with penalty `breakpenalty`.

Important — hyperref bookmarks

If a `\nolig` – whether starred or un-starred – occurs in an argument that is processed with package `hyperref` for inclusion into the document's PDF-bookmarks an additional argument is necessary to parse the macro. This argument either is `\relax` or the empty group `{}`.

```
\nolig*[\langle kerning \rangle]\relax    \nolig[\langle kerning \rangle]\relax
\nolig*[\langle kerning \rangle]{}    \nolig[\langle kerning \rangle]{}

```

The prototypical places where this processing-for-PDF-bookmarks happens are the sectioning macros, e. g., `\chapter`, `\section`, `\subsection`, etc.

L^AT_EX will bail out with an error if the extra argument is not passed to `\nolig` in these situations.

Alternatively use `\texorpdfstring` [18, Sec. 4.1.2, p. 22]. ■

Use Cases

`\nolig` can be used with just about any ligature that needs to be split into its parts. ¶ It also has proven beneficial in separating pairs of characters that are kerned to tightly (e. g. the `ij`, as in `bijection`, which is particularly distractive here, for it occurs at the boundary of two syllables). ■

3.4 Manual Italic Correction

`\itcorr`
`\itcorr*`

The italic correction offered by T_EX or L^AT_EX sometimes needs a helping hand.

```
\itcorr{⟨strength⟩}
\itcorr*{⟨strength⟩}
```

In text mode macro `\itcorr` inserts a kern whose width is proportional to `\fontdim1`, which is the font's italic correction. If `\fontdim1` happens to be zero (e.g. for an upright font), `\itcorr` uses the value set with `textitalicscorrection` instead of `\fontdim1`. The starred version always uses `textitalicscorrection`. In math mode macro `\itcorr` uses the value set with `mathitalicscorrection`¹¹ in both the starred and the unstarred form.

Typical slant angles of serif italics fonts range from 8° to 18° and thus values for `textitalicscorrection` from .14 to .32. Note: `⟨strength⟩` can be negative and fractional `⟨strength⟩`s are allowed.

Use Cases

Stronger or weaker correction than `\/.` ¶ Correct a non-slanted or non-italicized font. ¶ Negative correction at the left-hand side¹² of italics, i. e., compensate »shift-to-the-right effect« of italics. ¶ Positive correction at the left-hand side of italics, e.g., an opening parenthesis or square bracket followed by an italic *f* (before: 8, after: 7) or *y* (before: 4, after: 1) reaching far to the left below the baseline. ■

The `⟨strength⟩` parameter explained. T_EX records the slant angle α of a font in `\fontdim1` as $1\text{ pt} \times \sin \alpha$. Rephrased the formula means: *How much horizontal space is required for a letter slanted with α that is 1 pt high?* So, `\itcorr{⟨strength⟩}` calculates

$$\langle strength \rangle \times 1\text{ pt} \times \sin \alpha.$$

A well-chosen `⟨strength⟩` should be the absolute minimum value which avoids that the glyphs typeset in italics collide with other – usually non-italics – letters or symbols unless this disturbs the consistency of the overall tracking.

Correction of the right-hand side and $\alpha > 0$: A reasonable first guess of `⟨strength⟩` is the highest point where the rightmost part of the letter would touch a rule angled at α with respect to the baseline. The correction of the left-hand side and $\alpha > 0$ considers the lowest »touching« point below the baseline on the left-hand side of the letter. Negative values of α exchange the reference points.

Figure 1 shows how `⟨strength⟩` and α are related. Moreover, it demonstrates how intricate italics correction is.

¹¹ Separate adjustments may be desirable if the math font's italics have markedly different slants.

¹² Groff has the machinery for left-italic-correction. Its font-metrics files support per glyph left-italic-correction values and users can access them conveniently via `\,^`.

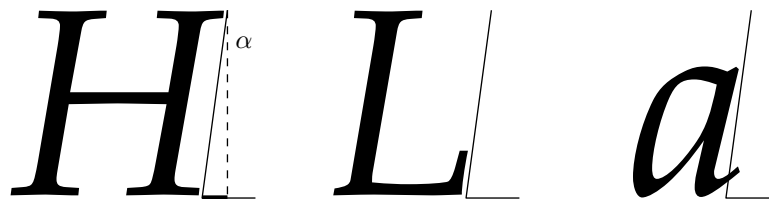


FIGURE 1: Some letters of an italics font. We use the capital `H` to measure the angle α between the plumb-line (drawn dashed) and a tangent to the rightmost parts of the glyph. The length of the plumb-line is proportional to `<strength>` and the short, thick part of the baseline symbolizes the resulting italics correction. ¶ The middle example, the capital `L`, shares α with `H` but obviously needs a far smaller `<strength>` or even no correction at all. ¶ The `a` at the right-hand side is an example of why T_EX allows to assign an italic correction to each individual character of a font. Not only features the lowercase `a` a larger α – despite being a member of the same font – but its serif adds as much to the width as the slanted stem.

We center the last lines of each figure and table caption with the help of `lastlinecentered-par`.

3.5 Apply Extra Kerning

Package `typog` supplies two sets of macros to kern some of the punctuation symbols. One is for forward slashes the other, more extensive one, for hyphens.

3.5.1 Slash

`\kernedslash`
`\kernedslash*`

Macro `\kernedslash` expands to a forward slash (`/`) with some extra space around it.

```
\kernedslash
\kernedslash*
```

The starred form is unbreakable, the non-starred version introduces a break point with penalty `breakpenalty` after the slash. Configure the kerning around the slash with `slashkern`.

If the word following the slash should not be hyphenated append `\nobreak` after `\kernedslash*`.

Use Cases

`\kernedslash` improves the appearance of pairs of years typeset in lining numerals: `<year1>/<year2>`. ¶ The macro has proven helpful in many cases where the right hand side of the slash starts with a capital as, for example, `<city>/<state-code>` (US-specific) or `<anything>/<noun>` (any language that capitalizes `<noun>`). ■

3.5.2 Hyphen

`\kernedhyphen`
`\kernedhyphen*` Macros `\kernedhyphen*` and `\kernedhyphen` expand to a hyphen (·) with given kerning to its left and to its right.

```
\kernedhyphen[⟨raise⟩]{⟨left-kerning⟩}{⟨right-kerning⟩}
\kernedhyphen*[⟨raise⟩]{⟨left-kerning⟩}{⟨right-kerning⟩}
```

Typeset an unbreakable hyphen with `\kernedhyphen*` or a breakable hyphen (like `\hyp` of package `hyphenat` [31]) with `\kernedhyphen` and apply some kerning to left and to the right of it. The values `⟨left-kerning⟩` and `⟨right-kerning⟩` are multiplied with one thousandth of the current font's em to get the size of the kern.

The optional argument `⟨raise⟩`, also given in $\frac{1}{1000}$ em, allows to adjust the height of the hyphen similar to the macros described in Sec. 3.6. In text mode the special argument `_*` for `⟨raise⟩` transfers the current value of `raisecapitalhyphen`. The default for `⟨raise⟩` is zero.

We also define specialized versions for kerning on the left-hand side or the right-hand side only. These macros work like their two-argument counterparts and set the appropriate other kerning to zero.

```
\leftkernedhyphen[⟨raise⟩]{⟨left-kerning⟩}
\leftkernedhyphen*[⟨raise⟩]{⟨left-kerning⟩}
\rightkernedhyphen[⟨raise⟩]{⟨right-kerning⟩}
\rightkernedhyphen*[⟨raise⟩]{⟨right-kerning⟩}
```

Use Cases

Composites in the form `⟨math⟩-⟨noun⟩` in languages where nouns are capitalized. ¶
 Composites where one or both sides of the hyphen are typeset in different fonts, like, `⟨small-caps⟩-⟨roman⟩`. ■

3.6 Raise Selected Characters

Usually all hyphens and dashes of a font are designed to join lowercase letters. This holds also true for most of our `\labelitem⟨N⟩` markers, bullets, stars, and even fancy dingbats. If these hyphens and dashes connect uppercase letters (or lining numerals) they sometimes appear to low; they disrespect the glyphs' symmetry axis. A similar situation arises if `itemize` list markers precede an uppercase letter, a lining numeral, or a big mathematical operator.

We introduce a set of macros for the most common cases that allow typesetting these characters at a user definable, adjusted height above the baseline. Users can base their own definitions of raised characters on their associated dimensions.¹³

¹³ Also compare with Ex. 12 in Ref. 30 for an attempt to automate vertical alignment.

Caution

The height adjustment disables a font's built-in kerning. ■

General note for all raised hyphen-like macros: Prefer the starred version if applied in front of any punctuation.

3.6.1 Capital Hyphen

`\capitalhyphen`
`\capitalhyphen*`

In many fonts the height of the hyphen character `␣` above the baseline is optimized for lowercase letters. In languages that capitalize their nouns as, e. g., German, this may be too low for compounds involving capitals.

```
\capitalhyphen
\capitalhyphen*
```

The unstarred version introduces a hyphenation opportunity right after the hyphen character (with penalty `breakpenalty`) whereas the starred version does not. The actual amount the hyphen gets raised in `\capitalhyphen` is determined by `raisecapitalhyphen`.

Use Cases

In languages that capitalize their nouns, the typical use-case is between an *⟨abbreviation⟩* and a *⟨noun⟩* when *⟨abbreviation⟩* is a string of uppercase letters. The same holds true for a connection of an uppercase variable in mathematical mode and a *⟨noun⟩* starting with a capital letter. ¶ Abbreviated compound first names (e. g., A.-M. Legendre) can be joined with the starred version. ¶ Also, the starred form is suited for ISO 8601-formatted dates if they are composed with lining-style numerals. ■

3.6.2 Capital Dash

`\capitalendash`
`\capitalendash*`
`\capitaldash`
`\capitaldash*`

The situation of the en-dash `—` is almost identical to the one of the hyphen character `␣` described in the previous section or the number dash to be introduced in the next section.

```
\capitalendash    \capitaldash (alias)
\capitalendash*   \capitaldash* (alias)
```

The unstarred version introduces a hyphenation opportunity right after the dash (with penalty `breakpenalty`) whereas the starred version does not. The actual amount the hyphen gets raised in `\capitaldash` is determined by `raisecapitaldash`.

Use Cases

Letter ranges as used in the title of an index. ¶ Any mixed letter-digit ranges (of capital letters and lining-style numerals) as in e. g., Sec. B-2. ■

`\capitalemdash`
`\capitalemdash*`

For completeness we also introduce a raised em-dash `—`. It behaves just like its en-dash sibling.

```
\capitalemdash
\capitalemdash*
```

Use Cases

Item symbols in `itemized` lists if the item text starts with an uppercase letter. ¶
 Theorem headings, like, e. g., Definition 6.2 — LIE Algebra. ■

3.6.3 Number Dash (Figure Dash)

```
\figuredash
\figuredash*
```

`\figuredash` yields 12–34–56–78 for sans-serif and 12–34–56–78 for the roman typeface.

The en-dash often gets used as separator for numerical ranges. In most fonts it has the correct height above baseline for oldstyle numerals, e. g. 12–34–56–78, but with lining numerals – depending on the font – it may look like it suffers from »broken suspenders«: 12–34–56–78. The situation is similar to `\capitaldash` and `\capitalhyphen` discussed in Secs. 3.6.1 and 3.6.2.

```
\figuredash
\figuredash*
```

The unstarred version introduces a hyphenation opportunity right after the en-dash with penalty `breakpenalty` whereas the starred version does not. The actual amount the en-dash gets raised in `\figuredash` is determined by `raisefiguredash`.

Values of .05em to .1em are typical for fonts that need this kind of correction and .1em is a good starting point. Table 2 summarizes some findings.

TABLE 2: Suggested values for raising the en-dash between lining numerals of some selected fonts.

Raise em	Font Name
0	Alegreya, Arvo, Bitter, Clara, EB Garamond, Gentium, Ibarra Real Nova, INRIA Serif, Libertine, Libertinus, Merriweather, PT Serif, Roboto Slab, Spectral, STIX, and many more
.05	fbb, Source Serif Pro
.0667	Libre Baskerville, Crimson Pro, Erewhon, Droid Serif
.1	GFS Artemisia, Libre Caslon, Coelacanth, Crimson Pro, Crimson Text, T _E X Gyre Pagella, Quattrocento, TX Fonts, ADF Venturis, and many more

Other macros may be redefined with `\figuredash` for a consistent appearance of the copy, like, for example, `\citedash` (package `cite` [3]), or `\crefrangeconjunction` (package `cleveref` [9]).

Use Case

The key customers of `\figuredash` are the PAGES entries of bibliography databases. ¶
 In an index generated with `makeindex` the range delimiter `delim_r` is a candidate for `\figuredash*`. ■

3.6.4 Multiplication Sign – Times \times `\capitaltimes`

The `\capitaltimes` macro is a variation of the `\capitalhyphen` theme.

```
\capitaltimes
```

In text mode it expands to an appropriately raised `\texttimes`, and in math mode to a raised `\times` binary operator, where `raisecapitaltimes` determines the amount of upward-shifting applied; it never inserts any break points.

Use Case

Prime use are two- or higher-dimensional shape specifications with lining numerals or uppercase letters in mathematical mode as, for example, matrix or tensor sizes. ■

3.6.5 Guillemets

Another possible typographic problem this package addresses is that both sets – single and double quotes – of guillemets may suffer from a too small distance to the baseline.

For the implementation typog relies on the T1¹⁴ font encoding not on package babel.

```
\singleguillemetleft
\singleguillemetright
\doubleguillemetleft
\doubleguillemetright
```

Lowercase Versions.

```
\singleguillemetleft  \singleguillemetright
\doubleguillemetleft  \doubleguillemetright
```

For consistency and easy accessibility we define height-adjusted left and right single guillemets as `\singleguillemetleft` and `\singleguillemetright`; double guillemets are available with `\doubleguillemetleft` and `\doubleguillemetright`. Their heights above the baseline are collectively adjusted with `raiseguillemets`.

14 Font encoding T1 can be forced via `\usepackage[T1]{fontenc}` in the document preamble.

\Singleguillemetleft
 \Singleguillemetright
 \Doubleguillemetleft
 \Doubleguillemetright

Uppercase Versions.

\Singleguillemetleft \Singleguillemetright
 \Doubleguillemetleft \Doubleguillemetright

The companion set of single, double, left, and right quotes corrected for uppercase letters or lining numerals is \Singleguillemetleft and \Singleguillemetright and \Doubleguillemetleft and \doubleguillemetright. Mnemonic: These macros start with an uppercase letter. Their height above the baseline is adjusted with `raisecapitalguillemets`. Values of .025em to .075em are typical for fonts that need this kind of correction. Table 3 summarizes some findings.

TABLE 3: Suggested values for raising guillemets of some selected fonts.

Raise		Font Name
Lowercase em	Uppercase em	
0	.05	EB Garamond, Libertinus, Merriweather, and many more
.025	.05	Gentium
.04	.0667	ADF Baskervald
.05	.0625	GFS Artemisia, GFS Didot

Tip

Define shorthand macros that simplify the application of guillemets, like, e. g.,

```
\newcommand*{\singlequotes}[1]
    {\singleguillemetright #1%
     \singleguillemetleft}
\let\sq=\singlequotes
```

and similar definitions for \Singlequotes, \doublequotes, and \Doublequotes.

Users working according to the French typesetting conventions will want to add extra spacing between the guillemets and the macro argument already in these macros. ■

Whether the guillemets must be height-adjusted for lowercase letters depends on the font. Careful judgment at various magnifications with a variety of samples is necessary.

Interaction with package `csquotes`. The users of package `csquotes` can hook up the guillemets as defined by `typog` with `\DeclareQuoteStyle`:

```
\DeclareQuoteStyle{typog-guillemets}
{\doubleguillemetright}% opening outer mark
{\doubleguillemetleft}% closing outer mark
{\singleguillemetright}% opening inner mark
{\singleguillemetleft}% closing inner mark
```

As always, the influence of package `babel` on `csquotes` has to be put into consideration. See Sec. 8 of the `csquotes` manual for a description of its configuration possibilities.

Use Case

All-capital words as for example acronyms put in guillemets that are raised somewhat almost always look better, whether using the French typographic convention (guillemets pointing outward plus some extra kerning) or the other way round (guillemets pointing inward). ■

Anticipated Changes & Possible Extensions

A correction in the other direction, i. e., lowering certain characters may also be desirable, to visually align them to the surrounding copy. Parentheses and in particular square brackets around all-lowercase text come into mind. ■

3.7 Align Last Line of a Paragraph

The usual algorithms of L^AT_EX typeset the last line of a paragraph flush with the left margin unless `center`, `raggedleft` or `Centering`, `FlushRight` (package `ragged2e` [20]) are in effect. For an instructive discussion consult Ch. 17, »Paragraph End«, of Ref. 10. The following environments allow to adjust the last lines of paragraphs in different ways.

The environment `lastlinerraggedleftpar` adjusts the various skips such that the last lines of the paragraphs gets typeset flush with the right margin.

`lastlinerraggedleftpar`
(env.)
`lastlineflushrightpar`
(env.)

```
lastlinerraggedleftpar
lastlineflushrightpar (alias)
```

The name `lastlineflushrightpar` is an alias for `lastlinerraggedleftpar`.

Center the last lines of the paragraphs enclosed by this environment.¹⁵

`lastlinecenteredpar`
(env.)

```
lastlinecenteredpar
```

Use Cases

`lastlineflushrightpar`: Narrow, justified parts of the text put flush against the right margin. ¶ `lastlinecenteredpar`: Table or figure captions typeset justified as centered boxes. ■

¹⁵ Also compare the approach taken in Ref. 27.

3.8 Fill Last Line of a Paragraph

The problem of when and how to ›fill‹ the last line of a paragraph is quite intricate. We first define the problem then we proceed to general purpose functions and we close the section with specific environments to control the length of the last line.

3.8.1 Problem Definition

Depending on the value of `\parindent`, either zero or nonzero, there may be the desire to control the length of the last line of a paragraph.

1. `\parindent > 0` [27, O1]

If the last line of a paragraph is shorter than the `\parindent` of the following paragraph a visual gap tears open.



The same problem arises with displayed math in a flush-left¹⁶ setting, e. g., `amsmath` [2] and option `fleqn`.¹⁷

A possible remedy is to reflow the paragraph in a way that its last line is clearly wider than `\parindent`; a typical suggestion being twice the `\parindent`.



2. `\parindent = 0` [27, O2]

If the last line of a paragraph is completely filled with text, i. e., flush with the right margin, it may become hard to spot the start of the following paragraph unless `\parskip` is large.¹⁸



¹⁶ The common practice of centering displayed equations does not call for the manipulations of a paragraph's last line discussed here.

¹⁷ For displayed equations and `amsmath` the relevant parameter is `\mathindent`.

¹⁸ Package `parskip` defines `\parskip` as 6pt plus 2pt for a base size of 10pt.

A possible, more legible solution is to reformat the paragraph in a way such that its last line leaves a marked gap with respect to the right margin.



The suggestions for the gap-width vary from two em to twice the width of a »typical« `\parindent`¹⁹ for the gap [7].

Tip

In theory both problems, O1 and O2 can be resolved by either shortening or prolonging the last line of the paragraph. For the concrete case it is up to the user to decide which direction to go and to choose the method that yields the most pleasing typographic results.

\TeX always considers the paragraph in its entirety. Thus any change the user demands »just for the last line« will permeate the whole paragraph and in unfortunate cases botch it.

Prudent users check the appearance of the problematic, original paragraph against one or more corrected versions of it – at least visually. Quantitative comparisons can be performed with the help of `\tracingparagraphs`. ■

Important

For the techniques in the following two subsections to work the paragraphs treated with them should have certain advantageous properties.

- Technically, the paragraphs need to contain enough glue (see e. g. Sec. 3.11) to achieve a low badness such that the desired paragraph end is deemed feasible by \TeX .
- Aesthetically, the paragraphs must be long enough to absorb the change in last-line fill level otherwise their gray-values visibly deviate from the average. ■

3.8.2 Manual Changes

Most O1 or O2 situations can be navigated with do-it-yourself methods. Here are some common recipes.

1. End-of-paragraph intervention.

(a) Tie `\sim`

Tie the last words.

The problem with the tie may be a hyphenation of one of the words that participates in the tie. The next item avoids this disadvantage.

¹⁹ For example, \LaTeX 's class `article` uses a `\parindent` of 25pt.

(b) `\mbox`

Join the last words or inline equation at the end of the paragraph with an `\mbox`.

(c) `\linebreak`

Add a `\linebreak` to the back part of the paragraph (approximately where the `\mbox` of item 1b would start) in a way that the last line receives the desired length [29]. In turn the next-to-last lines may become unsightly. Counteract this degradation e. g. with recipes 2a to 2c.

Tying and `\mbox`ing lend themselves to generalizations. We need not only tie at end of a paragraph but fuse logical units of sentences or inline equations so that the relevant information literally stays in the reader's focus. Cementing together text of course finds an end when overfull lines start to show up.

2. Uniform paragraph change.

(a) Vary spacing.

Modify the inter-word spacing, for example, with the macros introduced in Sec. 3.9.1.

Enclose the paragraph in either `loosespacing` or `tightspacing`. Increase the spacing $\langle level \rangle$ until the last line gets the desired length.

(b) Vary font tracking.

Enclose the paragraph in a `setfonttracking` group. See Sec. 3.10.1. Increase or decrease the tracking in steps of $\frac{1}{4000}$ em until the last line looks good.

(c) Vary font expansion.

Enclose the paragraph in a `setfontexpand` group. See Sec. 3.10.2.

3. A combination of any of the above items.

4. Some curveballs.

(a) If the paragraph already suffers from one of the problems that \TeX addresses with `\doublehyphendemerits`, `\finalhyphendemerits`, or `\adjdemerits`, crank up one or all of these values to 10000 and observe whether the length of last line changes in the desired direction.

(b) If any influential microtype features have been enabled try with one more more of them *disabled*. See, e. g., environment `nofontexpansion` in Sec. 3.10.2.

3.8.3 Multi-Purpose Environments

`shortenpar` (*env.*) The two environments `shortenpar` and `prolongpar` can be employed in quite general situations when a paragraph should be typeset one line longer or shorter, e. g., to avoid a widow line²⁰ or a club line²¹ [13, p. 104 and 16]. (See also Sec. 3.12 for special functions to avoid clubs or widows.) ›Accidentally‹, they also change the length of the last line of the paragraph.

`shortenpar`

Environment `shortenpar` decreases the `\looseness` of the paragraph.²² It performs well if the last line of the paragraph is short or the whole paragraph is loose.

`prolongpar`

This environment increases the `\looseness` of the paragraph, which is why it works best with decent or tight last lines that are almost full.

3.8.4 Specialized Environments

We introduce environments not just skips to get the correct behavior – set up all paragraph parameters *before* the paragraph ends – and, at the same time, limit the range of this parameter change.

`covernextindentpar` (*env.*) Environment `covernextindentpar` can be helpful for [case O1](#), i. e., a too short last line.

`covernextindentpar` [*<dim>*]

The environment asks \TeX to extend the last line of a paragraph such that it takes at least `2\parindent` (if `\parindent` $\neq 0$), `2em` (if `\parindent` $= 0$), or *<dim>* if called with an optional argument.

`openlastlinepar` (*env.*) The next environment, `openlastlinepar`, takes care of [case O2](#), i. e., a last line in a paragraph that is almost full or completely filled.

`openlastlinepar` [*<dim>*]

It may resolve [case O2](#) as it attempts to prevent a completely filled line by introducing a partly unshrinkable `\parfillskip`. Without optional argument the threshold of unused last-line length is either `2\parindent` (if `\parindent` \neq

²⁰ The last line of a paragraph becomes a ›widow‹ (ger. *Hurenkind*) if it starts the following page or column.

²¹ The first line of a paragraph is called ›club‹ or ›orphan‹ (ger. *Schusterjunge*) if it appears at the bottom of the page or column.

²² Command `\looseness` is a \TeX primitive [13, p. 103n]. A thorough discussion of the interaction of `\linepenalty` and `\looseness` can be found in Ref. 26.

0) or 2em (if `\parindent = 0`). The optional argument $\langle dim \rangle$ directly sets the gap threshold.

Note that the application of this environment can be successful, this is, a completely filled last line is avoided, but the result may be of [type O1](#) nonetheless.

3.9 Spacing

90 % of design is typography.
And the other 90 % is whitespace.
— Jeffrey Zeldman

The functions described in this section rely only on plain L^AT_EX. No extra packages are required. Compare to the microtype-based functionality of [Sec. 3.10](#).

3.9.1 Looser or Tighter Spacing

Never try to adjust lines by squeezing or stretching the tracking.
Go for the subtle solution: adjust word spacing instead.
— Jan Middendorp [[15](#), p. 119]

The environments in this section directly influence the spacing, this is, they change the width and stretchability of the horizontal space.

They at the one hand act gently by adjusting the spacing only by a small amount. On the other hand they operate decidedly in controlling the glue associated with the adjusted space. The latter also being important to ensure the monotonicity of the different $\langle level \rangle$ s. However, the strictly managed stretchability/shrinkability may lead to many overfull boxes with `\fussy` or when applied to short lines.

`loosespacing (env.)`
`tightspacing (env.)`

Environments `loosespacing` and `tightspacing` introduce four $\langle level \rangle$ s of `>looseness<` or `>tightness<`, where $\langle level \rangle = 0$ disables the functionalities. The higher the $\langle level \rangle$ the looser or tighter the text will be typeset, respectively.

`loosespacing[$\langle level \rangle$]`

Environment `loosespacing` increases the width of a space by the percentages given in the [Tab. 4](#).

$\langle level \rangle$	Adjustment %	Comment
0	n/a	neutral
1	+5	default
2	+10	
3	+20	
≥ 4	+30	

TABLE 4: Adjustments made by environment `loosespacing` to `\spaceskip`. The mapping of $\langle level \rangle$ to the exact skip definitions are $1 \mapsto 1.05^{+.5}_{-.1}$, $2 \mapsto 1.1^{+.5}_{-.1}$, $3 \mapsto 1.2^{+.6}_{-.2}$, and $\geq 4 \mapsto 1.3^{+.8}_{-.3}$, where all factors scale with `\dimen2`, the current font's space-width.

The default level of `loosespacing` is 1.

`tightspacing[⟨level⟩]`

Environment `tightspacing` decreases the width of a space by the percentages given in Tab. 5.

⟨level⟩	Adjustment %	Comment
0	n/a	neutral
1	-1.25	default
2	-2.5	
3	-5	
≥ 4	-10	

TABLE 5: Adjustments made by environment `tightspacing` to `\spaceskip`. The mapping of ⟨level⟩ to the exact skip definitions are $1 \mapsto .9875^{+.0125}_{-.5}$, $2 \mapsto .975^{+.025}_{-.5}$, $3 \mapsto .95^{+.05}_{-.5}$, and $\geq 4 \mapsto .9^{+.1}_{-.5}$, where all factors scale with `\dimen2`, the current font’s space-width.

The default level of `tightspacing` is 1.

Note

At a given ⟨level⟩ the changes of `loosespacing` are much larger than those of `tightspacing`. ■

Use Cases

Nudge line breaks or hyphenation points. ¶ Separate clashing descenders and ascenders. ¶ Eliminate rivers. ■

3.9.2 Wide Space

The `\widespace` macro and its companion `\narrowospace` derive their appearances from several of the current font’s `\fontdimen⟨number⟩`s. T_EX addresses the latter by integers, which is totally non-memnonic. Therefore, we play softball by first presenting Tab. 6 that associates the `\fontdimen⟨number⟩`s with their meanings and also reports on their current values (for this document).²³

#	Description	Value %
1	Slant per 1 pt height	0
2	Interword space width	23.3
3	Interword stretch	11.6
4	Interword shrink	7.8
5	<code>\x</code> height	47.5
6	<code>\quad</code> height	100
7	Extra space width	3.9

TABLE 6: The first column # states the index of the `\fontdimen` parameter: ⟨number⟩. Column 2 presents short descriptions of the `\fontdimen⟨number⟩` parameters. As examples, the values for the current font are shown in column 3; they are normalized to the quad-size.

`\widespace`
`\widespace*`

STARRED FORM SINCE V0.2

Typeset a wide, sentence-ending space as if in `\nonfrenchspacing` mode.

²³ The association is given in Appendix F (p. 433) of Ref. 13. For a concise and understandable explanation of the T_EX `\fontdimen` parameters consult Ref. 8.

Consult Table 7 for a comparison of the various sizes.

```
\widespace
\widespace*
```

The unstarred macro `\widespace` inserts a space that is as wide as the font's sentence-ending space in `\nonfrenchspacing` mode, this is

$$\text{\fontdimen2} + \text{\widespacestrength} \times \text{\fontdimen7}.$$

Its width is independent of any `\frenchspacing` or `\nonfrenchspacing` settings, but depends on `\widespacestrength` which defaults to 1. The latter can be overridden by the user to get a more or less pronounced effect.

If `\fontdimen7` happens to be zero `\widespace` uses

$$\text{\widespacescale} \times \text{\fontdimen2}$$

as width instead, where `\widespacescale` defaults to 1.125. The stretchability and shrinkability of `\widespace` always are scaled with `\widespacescale`. The `\widespacescale` too can be redefined by the user to achieve different effects.

The starred form, `\widespace*`, unconditionally uses the `\fontdimen7 = 0` code-path.

Use Case

Useful as a sentence-ending space if, for example, the sentence ends in an abbreviation with a period or decimal number without trailing digits *and* the next sentence should be delimited in a clearer way. ¶ Open tight lines with a series of `\widespaces`.²⁴ ■

3.9.3 Narrow Space

```
\narrowespace
\narrowespace*
SINCE V0.2
```

Typeset a narrow space. Consult Table 7 for a comparison of the various sizes.

```
\narrowespace
\narrowespace*
```

The unstarred macro `\narrowespace` inserts a narrow space with the width

$$\text{\fontdimen2} - \text{\narrowespacestrength} \times \text{\fontdimen7}$$

if `\fontdimen7` is different from zero or otherwise

$$\text{\narrowspacescale} \times \text{\fontdimen2}.$$

The starred version, `\narrowespace*`, unconditionally uses the `\fontdimen7 = 0` code-path. Refer to Table 6 for the meanings of the various `\fontdimen` parameters.

²⁴ See also »Investigating the badness of a paragraph« on Page 7.

The sentence that ends with ›1.« uses `\widespace` after the period.

The stretchability and shrinkability of `\narrowospace` always get scaled with `\narrowospace scale`. Both factors, `\narrowospace strength` and `\narrowospace scale` can be redefined by the user; their defaults are .5 and .9375, respectively.

Use Case

Tighten loose lines with a series of `\narrowospace`.²⁵ ■

TABLE 7: Exemplary comparison of standard `\space` versus `\narrowospace` and `\widespace`. All values are relative to the size of the current font's quad size. `\narrowospace` and `\widespace` use the package's defaults. ■ The upper values in the Width-column for `\narrowospace`, and `\widespace` refer to the `\fontdimen7 ≠ 0` case and the lower ones to the `\fontdimen7 = 0` code-path.

Name	Width %	Stretch %	Shrink %
<code>\narrowospace</code>	21.4 21.8	10.9	7.3
<code>\space</code>	23.3	11.6	7.8
<code>\widespace</code>	27.2 26.2	13.1	8.7

3.10 Microtype Front-End

The functionalities are just front-ends of selected macros in package `microtype`. All macros and environments introduced in this section require that package `microtype` [19] has been loaded.

3.10.1 Tracking

Caution

The tracking changes may interfere with implicit changes of tracking declared with `\SetTracking`. Explicit calls to `\textls` remain in effect. ■

`setfonttracking (env.)`

Override the default tracking for all fonts.

```
setfonttracking{⟨delta⟩}
```

The environment `setfonttracking` manages a group for `\lsstyle` of package `microtype`. The change `⟨delta⟩` in tracking is given as multiples of $\frac{1}{1000}$ em. Positive as well as negative values of `⟨delta⟩` are allowed.

See Sec. 5.3, »Tracking«, and 7, »Letterspacing revisited«, in the documentation of `microtype` [19] for a detailed explanation.

²⁵ Footnote 24 again applies.

For font combinations involving monospaced fonts (T_EX lingo: typewriter) an overly large spacing may show up at the borders where fonts change. This is caused by the calculation of the »outer spacing« described in Sec. 5.3 of the microtype manual.

Use configuration variable `trackingttspacing` to reduce the outer spacing to a reasonable value either directly at package-load time

```
\usepackage[trackingttspacing={250, 75, 50}]{typog}
```

or with the help of `\typogsetup` in the document *preamble* (after loading microtype and typog)

```
\typogsetup{trackingttspacing={250, 75, 50}}
```

If the argument of option `trackingttspacing` is omitted the outer spacing defaults to 300, 90, 60.

Use Cases

Nudge line breaks or hyphenation points. ¶ Avoid clashes of descenders and ascenders, e. g., for `\smashed` symbols of inline math. – Think of integrals. ¶ Control the length of the last line in a paragraph. ■

3.10.2 Font Expansion

`setfontshrink` (*env.*)
`setfontstretch` (*env.*)

Adjust the limits of either only stretchability or only shrinkability and zero the other component, i. e., shrinkability and stretchability, respectively.

```
setfontshrink{<level>}  
setfontstretch{<level>}
```

A `<level>` of zero is a no-op. Tables 8 and 9 summarize the values for stretch and shrink in these environments.

<code><level></code>	stretch $\frac{1}{1000}$ em	shrink $\frac{1}{1000}$ em	Comment
0	n/a	n/a	no operation
1	0	5	default
2	0	10	
3	0	20	

TABLE 8: Preconfigured values for shrink inside of environment `setfontshrink`. Note that all stretch values are zero, so the fonts only can shrink.

<code><level></code>	stretch $\frac{1}{1000}$ em	shrink $\frac{1}{1000}$ em	Comment
0	n/a	n/a	no operation
1	5	0	default
2	10	0	
3	20	0	

TABLE 9: Preconfigured values for stretch inside of environment `setfontstretch`. Note that all shrink values are zero, so the fonts only can stretch.

The three (nonzero) shrink limits of `setfontshrink` can be configured with package option `shrinklimits` and – in the same way – the three (nonzero) stretch limits of `setfontstretch` with package option `stretchlimits`.

Use Cases

Nudge line breaks or hyphenation points. ¶ Control the length of the last line in a paragraph. ■

`setfontexpand (env.)`

Manipulate both, stretch and shrink values at the same time.

```
setfontexpand{<level>}
```

Table 10 gives an overview of the values associated with `<level>`.

<code><level></code>	stretch 1/1000 em	shrink 1/1000 em	Comment
0	n/a	n/a	no operation
1	5	5	default
2	10	10	
3	20	20	

TABLE 10: Preconfigured values for shrink and stretch inside of environment `setfontexpand`. Note that both shrink and stretch values are nonzero, so the fonts can shrink or expand.

The six shrink and stretch limits of `setfontexpand` can be configured with package options `shrinklimits` and `stretchlimits`.

Notes

- Environment `setfontexpand` shares its `shrinklimits` with `setfontshrink` and its `stretchlimits` with `setfontstretch`.
 - These environments do not nail down any font’s expansion but only set up its available range. See Sec. 3.3, »Font Expansion«, in the microtype documentation [19].
- Moreover, a text may not »respond« neither to `setfontshrink`, `setfontstretch`, nor `setfontexpand` because \TeX already considers it optimal without expansion or within the previous expansion limits, e. g., those set at microtype load time as opposed to typog’s load time. ■

Use Cases

Nudge line breaks or hyphenation points. ¶ Control the length of a paragraph, e. g., to avoid a widow. ■

`nofontexpansion (env.)`

Disable the microtype feature »expansion« inside of the environment.

```
nofontexpansion
nofontexpand (alias)
```

The name `nofontexpand` is an alias for `nofontexpansion`.

Use Cases

Nudge line breaks or hyphenation points. ¶ Prevent severe scaling effects in paragraphs strongly manipulated by other means, e.g., `shortenpar` or `prolongpar`. ■

3.10.3 Character Protrusion

`nocharprotrusion`
(*env.*)

Disable the microtype feature ›protrusion‹ inside of the environment.

```
nocharprotrusion
```

Use Cases

Table of Contents or similar tables with aligned section numbers. ¶ Any table with left- or right-aligned numerals in particular tabular numerals. ¶ Index. ■

3.11 Sloppy Paragraphs

Experienced L^AT_EX users know that `\sloppy` is more of a problem by itself and not really a viable solution of the ›overfull box‹ syndrome.

`\slightlyloppy`
`slightlyloppypar`
(*env.*)

We define the macro `\slightlyloppy` and the associated environment, `slightlyloppypar`, with a user-selectable `⟨sloppiness⟩` parameter. The constructions recover the known settings `\fussy` (`⟨sloppiness⟩ = 0`) and `\sloppy` (`⟨sloppiness⟩ ≥ 8`), and introduce seven intermediate `⟨sloppiness⟩` levels.²⁶ The default `⟨sloppiness⟩` is 1.

```
\slightlyloppy[⟨sloppiness⟩]
slightlyloppypar[⟨sloppiness⟩]
```

Table 11 summarizes the adjustments that `\slightlyloppy` makes depending on the `⟨sloppiness⟩` level.

Environment `slightlyloppypar[⟨sloppiness⟩]` mimics L^AT_EX's `sloppy-par`, while offering the flexibility of `\slightlyloppy`.

Use Cases

Drop-in replacement for `\sloppy`, whether explicit or implicit (think of `\parbox`). ¶ Initial paragraphs in theorem environments (e.g., as defined by `amsmath` or `amsthm`), where the theorem head already takes a lot of space. ¶ Bibliographies as environment `thebibliography` sets `\sloppy`. ■

26 Also compare the findings for `\emergencystretch` in Ref. 25.

TABLE 11: Adjustments made by `\slightlyloppy` to various $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ parameters at different levels of $\langle sloppiness \rangle$.

$\langle sloppiness \rangle$	<code>\toler-</code> ance	<code>\hfuzz</code> <code>\vfuzz</code> pt	<code>\emergency-</code> stretch G em	Comment
0	200	.1	0	$\mathrm{T}_{\mathrm{E}}\mathrm{X}$: <code>\fussy</code> default
1	330 [†]	.15	.375 [‡]	
2	530 [†]	.2	.75 [‡]	
3	870 [†]	.25	1.125 [‡]	
4	1410 [†]	.3	1.5 [‡]	
5	2310 [†]	.35	1.875 [‡]	
6	3760 [†]	.4	2.25 [‡]	
7	6130 [†]	.45	2.625 [‡]	$\mathrm{T}_{\mathrm{E}}\mathrm{X}$: <code>\sloppy</code>
≥ 8	9999	.5	3	

[†] All intermediate levels set `\pretolerance = \tolerance/2`.

[‡] The intermediate levels scale the amount of available glue G (indicated in column 4 of the table) for `\emergencystretch` with the actual line length, this means, in these levels

$$\text{\emergencystretch} = G \times \frac{\text{\linewidth}}{\text{\textwidth}}.$$

to prevent excessive stretchability in narrow lines.

3.12 Vertically Partially-Tied Paragraphs

L^AT_EX provides several macros and environments to tie material vertically – most prominently `samepage` and `minipage`.²⁷ Typog’s macros and environments constitute more sophisticated but weaker forms of these. They tie only the first or last couple of lines in a paragraph while the rest of the paragraph gets broken into pages by T_EX in the usual way.

The macros and environments described in this section locally set ϵ -T_EX penalty arrays [6, Sec. 3.8]. In addition the environments `vtietoppar`, `vtiebotpar`, and `vtiebotdisptoppar` explicitly issue a `\par` at the end of the group.

Avoid a club line in each partial paragraph.

`\vtietop`
`vtietoppar (env.)`

```
\vtietop[⟨number-of-lines⟩]
vtietoppar[⟨number-of-lines⟩]
```

Vertically tie the first *⟨number-of-lines⟩* in a paragraph. Zero or one for *⟨number-of-lines⟩* are no-ops. Up to nine lines can be fused. The default is to link three lines.

Use Cases

String together the first paragraph right after a sectioning command. ¶ Tie the first line of an itemized, enumerated, or a description list with the paragraph following `\item`. ■

`\splicevtietop`

Inside of a `list` a one-off solution simply concatenates `\item[...]\vtietop` to fuse the line with the `item#`, the representation of the `enum#`, or the description term with the first paragraph. For a systematic use prefer `\splicevtietop` and apply it as the first thing in the `list` body.

```
\splicevtietop[⟨number-of-lines⟩]
```

Use this macro *inside* of a `list`-like environment to equip each `\item` with `\vtietop[⟨number-of-lines⟩]`. The default *⟨number-of-lines⟩* is three as for any of the `vtie...` functions.

Example for a description list and plain L^AT_EX:

```
\begin{description}
  \splicevtietop[2]
  \item[...]
\end{description}
```

Alternatively with package `enumitem` [4]:

```
\begin{description}[first=\splicevtietop[2]]
  \item[...]
\end{description}
```

or shorter and with the default *⟨number-of-lines⟩*, 3, using the `enumitem` style `vtietop`:

`enumitem style vtietop`

²⁷ A valuable complement to these is package `needspace` [33] which takes a different approach and reliably works in *mixed* horizontal and vertical mode situations.

```

\usepackage{enumitem}
\begin{description}[vtietop]
  \item[...]
\end{description}

```

`\vtiebot`
`vtiebotpar (env.)`

Avoid a widow line in each partial paragraph.

```

\vtiebot[⟨number-of-lines⟩]
vtiebotpar[⟨number-of-lines⟩]

```

Vertically tie the last *⟨number-of-lines⟩* in a paragraph. Zero or one for *⟨number-of-lines⟩* are no-ops. Up to nine lines can be fused. The default is to link three lines. Avoid a display widow line in each partial paragraph.

`vtiebotdisp (env.)`

```

vtiebotdisp[⟨before-disp-number-of-lines⟩]

```

Vertically tie the last *⟨before-disp-number-of-lines⟩* in a paragraph before a display. Zero or one for *⟨before-disp-number-of-lines⟩* are no-ops. Up to nine lines can be fused. The default is to link three lines.

To use the function bracket the paragraph before the display (the one that needs protection) and the associated displayed math:

```

\begin{vtiebotdisp}
  % vertically tied paragraph before the math display
  \begin{equation}
    % math
  \end{equation}
\end{vtiebotdisp}

```

`vtiebotdisptoppar`
`(env.)`

Avoid a display widow, compound the display with its preceding *and* following paragraph, and avoid a club line in the paragraph right after the display.

```

vtiebotdisptoppar[⟨before-disp-number-of-lines⟩]
                  [⟨after-disp-number-of-lines⟩]

```

Vertically tie the last *⟨before-disp-number-of-lines⟩* in the paragraph before a display and the first *⟨after-disp-number-of-lines⟩* in the paragraph after the display. Moreover, turn the paragraphs and the display into an un-breakable unit.²⁸

Zero or one for *⟨before-disp-number-of-lines⟩* as well as *⟨after-disp-number-of-lines⟩* are no-ops for the respective paragraph. Up to nine lines each can be fused.

Both optional arguments default to three. If only the first argument is given the second acquires the same value.

To use the function bracket the paragraphs before and after the display:

²⁸ The paragraphs and the display are concreted together by setting both `\predisplaypenalty` and `\postdisplaypenalty` to 10000.

```

\begin{vtiebotdisptoppar}
% vertically tied paragraph before the math display
\begin{equation}
% math
\end{equation}
% vertically tied paragraph after the math display
\end{vtiebotdisptoppar}

```

See also Sec. 3.8.3 for other methods to avoid club or widow lines.

Partial Paragraphs And Counting Lines. The top-of-paragraph ties, `\vtietop` and `vtietoppar` count *<number-of-lines>* from the beginning of every partial paragraph. Each displayed math in the paragraph resets the count. The bottom-paragraph ties, `\vtiebot`, `vtiebotpar`, `\vtiebotdisp`, and `vtiebotdisppar` count backward from the end of each partial paragraph. Again, each displayed math in the paragraph resets the count. According to \TeX 's rules, a displayed math formula always is counted as *three* lines no matter its contents. Table 12 summarizes these rules with the help of an example.

TABLE 12: Exemplary, eight-line paragraph compounded of two partial paragraphs of three and two lines and a displayed math formula of arbitrary size sandwiched in between.

Continuous Line Number	Example Contents	<code>\vtietop</code> [†] Count	<code>\vtiebot</code> [‡] Count
1	Text line ₁	1	3
2	Text line ₂	2	2
3	Text line ₃	3	1
4	} Display math		
5			
6			
7	Text line ₄	1	2
8	Text line ₅	2	1

[†] This is $\epsilon\text{-}\text{\TeX}$'s counting scheme of `\clubpenalties`; it also holds for `vtietoppar`.

[‡] The same counting scheme also holds for `vtiebotpar`, `\vtiebotdisp`, and `vtiebotdisppar`. It is implied by $\epsilon\text{-}\text{\TeX}$'s line counts of `\widowpenalties` and `\displaywidowpenalties` on which the functions of this package are based.

Tips

- The environments can be combined to arrive at paragraphs that simultaneously are protected against club lines and (display) widow lines.
- For very long derivations that are not interrupted and thus made break-

able with the help of `\intertext`²⁹ or `\shortintertext`³⁰ it is desirable to make the display breakable. This is achieved with `\allowdisplaybreaks` or the environment `breakabledisplay` which will be described in Sec. 3.13. ■

Use Cases

Fix widows and orphans, e. g., those turned up by package `widows-and-orphans` [17]. ¶
Extend the typographic convention of »three to four lines instead of a single club or widow line« to a context-dependent number of lines that tries to keep all (well, dream on) the information together the reader needs at that particular point. ■

3.13 Breakable Displayed Equations

`breakabledisplay`
(*env.*)

Package `amsmath` offers `\allowdisplaybreaks` to render displayed equations breakable at each of their lines. Environment `breakabledisplay` is a wrapper around it which limits the macro's influence to the environment. Furthermore, the default $\langle level \rangle$ of `breakabledisplay` is 3 whereas that of `\allowdisplaybreaks` is 4. This makes `breakabledisplay` less eager to break a displayed equation and thus better suited to full automation of the page-breaking process.

```
breakabledisplay[ $\langle level \rangle$ ]
```

Environment `breakabledisplay` simply passes on $\langle level \rangle$ to `\allowdisplaybreaks`. Table 13 shows the default penalties that `amsmath` associated with each of the $\langle level \rangle$ s.

TABLE 13: Penalties `\interdisplaylinepenalty` associated with different $\langle level \rangle$ s of environment `breakabledisplay`. Depending on the version of package `amsmath` the actual penalties may differ.

$\langle level \rangle$	<code>\interdisplaylinepenalty</code>	Comment
0	10000	no operation
1	9999	
2	6999	
3	2999	default
4	0 [†]	

[†] This is the default of `\allowdisplaybreaks`.

Tips

- Terminating a line with `*` inhibits a break after this line.

²⁹ Introduced in package `amsmath` [2].

³⁰ Defined in package `mathtools` [11].

- A `\displaybreak[⟨level⟩]` can be set for *each* line of the displayed equation separately. L^AT_EX resumes with the original value of `\interdisplaylinepenalty` in the following lines.
- If a discretionary break of the displayed equation is to be accompanied with some aid for the reader, team `\intertext` (or `\shortintertext`) with `\displaybreak` as, e.g.,

```
\newcommand*{\discretionarydisplaybreak}
{\intertext{\hfill Eq.~cont.~on next page.}%
\displaybreak
\intertext{Eq.~cont.~from prev.~page.\hfill}} ■
```

Use Cases

Extremely long derivations without interspersed `\intertext` or `\shortintertext`. ■ Draft phase of a document. ■

3.14 Smooth Ragged

The attention someone gives
to what he or she makes
is reflected in the end result,
whether it is obvious or not.
— Erik Spiekermann

Package `typog` implements a novel approach to typeset ragged paragraphs. Instead of setting the glue inside of a paragraph to zero and letting the line-widths vary accordingly [28] we prescribe the line-widths with the `\parshape` primitive and leave alone the stretchability or shrinkability of the glue.

We introduce three environments that allow for setting three, five, or seven different line-lengths: `smoothraggedrightshapetriplet`, `smoothraggedrightshapequintuplet`, and `smoothraggedrightshapeseptuplet`; they work for paragraphs up to 99, 95, or 98 lines, respectively.

```
smoothraggedrightshapetriplet[⟨option...⟩]{⟨width1⟩}{⟨width2⟩}{⟨width3⟩}
smoothraggedrightshapequintuplet[⟨option...⟩]{⟨width1⟩}{⟨width2⟩}...{⟨width5⟩}
smoothraggedrightshapeseptuplet[⟨option...⟩]{⟨width1⟩}{⟨width2⟩}...{⟨width7⟩}
```

The environments take $N = 3, 5, \text{ or } 7$ mandatory line-width parameters, where each $\langle width I \rangle$, $I = 1, \dots, N$ is a skip, i. e., a `dimen` that can include some glue.

Options

leftskip=⟨*dim*⟩

Set the left margin for the smooth ragged paragraph to ⟨*dim*⟩. Similar to the `TeX` parameter `\leftskip`.

parindent=⟨*dim*⟩

Set the first-line indent for the smooth ragged paragraph to ⟨*dim*⟩. Similar to the `TeX` parameter `\parindent`.

Environment `smoothraggedrightpar` builds upon the three generators. It typesets a single paragraph with a given ⟨*ragwidth*⟩ of the ragged, right margin, where the rag width is the length-difference of the longest and the shortest lines.

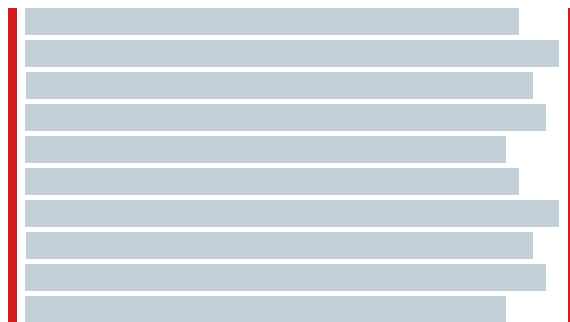
```
smoothraggedrightpar[⟨option...⟩]
```

The line lengths equally divide the ragged margin, i. e., they are arithmetic means with respect to the generator size.

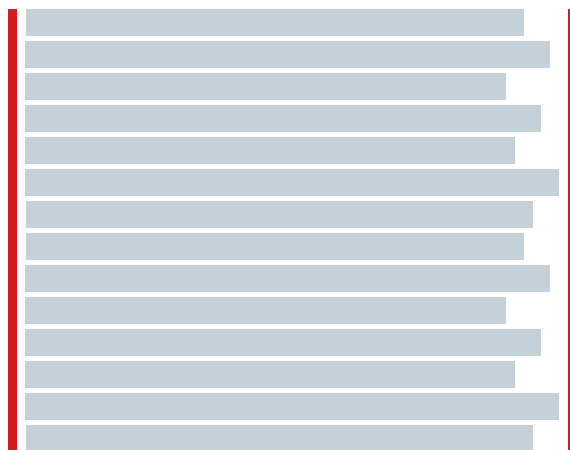
- The triplet generator repeats a *short line – long line – middle-length line* sequence. Shown below are two complete cycles.



- The quintuplet generator varies the theme of the triplets and avoids the ›ladder‹ of lines 2–3–4 (or, if numbered by cycle: 1.2–1.3–2.1) there. Shown here are two cycles.



- The septuplet generator uses a permutation that looks ›random‹. At least it hides the boundaries of cycles well. Shown here are two of them.



`smoothraggedright`
(*env.*)

Environment `smoothraggedright` is the multi-paragraph version of `smoothraggedrightpar`. It takes the same optional arguments.

```
smoothraggedright[⟨option...⟩]
```

Options

`\linewidth=<dim>`

Override the length of the longest line. The default line-width is `\linewidth`.

Global Parameters

`\smoothraggedrightfuzzfactor=<factor>`

The environment adds glue to every line-width³¹ to achieve a more convincing »ragged appearance« and to reduce the number of overfull lines. The algorithm divides the smooth margin into 3, 5, or 7 parts depending on the chosen `\smoothraggedrightgenerator` (see below). The `\smoothraggedrightfuzzfactor` is the amount of glue of each line expressed as a multiple of the distance between the division points. The default of 1.0 means to add as much glue such that the lines just do not overlap (assuming justification is feasible).

`\smoothraggedrightgenerator`

Select a generator to use. Valid generator names:

- triplet,
- quintuplet,
- septuplet.

The default generator is triplet.

`\smoothraggedrightleftskip=<dim>`

Value for `leftskip` to pass to the generator. Default: 0pt.

`\smoothraggedrightparindent=<dim>`

Value for `parindent` to pass to the generator. Default: 0pt.

`\smoothraggedrightragwidth=<dim>`

Value for the width of the ragged right margin. Default: 2em.

Use Cases

Replacement for `\RaggedRight` [20].  Design alternative for fully justified paragraphs if used with a small rag-width. 

31 The shortest line only gets stretchability, the longest only receives shrinkability. All other lines are both stretchable and shrinkable.

4 Other Packages for Fine L^AT_EX Typography

Many other packages help with getting better output from L^AT_EX. Here is a list – in alphabetical order – of the ones the author considers particularly valuable.

- | | |
|-----------|--|
| enumitem | Flexible and consistent definition of all basic L ^A T _E X-list types plus in-line lists [4]. |
| geometry | Powerful and sophisticated setup of the page layout [23]. Best accompanied by layout [14] to visualize the page geometries. |
| hyphenat | Hyphens that do not inhibit further auto-hyphenation of a compound word [31]. |
| microtype | Fine control of spacing, tracking, sidebearings, character protrusion into the margins, font expansion, and much more [19].
See also KHIREVICH's discussion [12]. |
| ragged2e | Improved versions of environments <code>raggedleft</code> , <code>raggedright</code> , and <code>center</code> [20]. |
| setspace | Consistently set the document's line-spacing, i. e., <code>\baselineskip</code> [22]. |

A Package Code

This is the »Reference Manual« section of the documentation where we describe the package's code and explain its implementation details.

```

1%<*package>
2\NeedsTeXFormat{LaTeX2e}[2005/12/01]
3\ProvidesPackage{typog}[2024/03/29 v0.2 Typographic extensions]
4
5\RequirePackage{everyhook}
6\RequirePackage{xkeyval}
7

```

Coding Conventions

1. All package-related names start with `typog@`.
2. Persistent values get a single `@`, temporary ones a double-`@`.

Declarations of Lengths, Skips, etc.

`\typog@TYPOG` Define a macro that unequivocally identifies this very package.

```
8\newcommand*\typog@TYPOG{}
```

`\typoglogo` We have our own, low-key logo.

```
9\newcommand*\typoglogo{\textsf{T\itcorr*{-5}\textsl{y}poG}}
```

`\iftypog@debug` Our switch for debug information.

```
10\newif\iftypog@debug
```

`\typog@typeout` Our debug information printer.

```

11\newcommand*\typog@typeout[1]
12  {\iftypog@debug
13   \typeout{typog: #1}%
14   \fi}
15

```

`\typog@trim@spaces` Pull `\tl_trim_spaces` into the `>classic<` namespace.

```

16\ExplSyntaxOn
17\let\typog@trim@spaces=\tl_trim_spaces:o
18\ExplSyntaxOff
19

```

`pog@register@pdfsubstitute` We often need to register (simple) substitute commands suitable for PDF bookmarks. This is a convenient abbreviation for that task.

```

20\newcommand{\typog@register@pdfsubstitute}[1]{%
21  \AtBeginDocument{%
22    \ifdefined\pdfstringdefDisableCommands
23      \pdfstringdefDisableCommands{#1}%
24    \fi}}
25

```

Some functionality depends on package microtype. To complicate matters for certain setup operations, e. g., `\SetExpansion`, microtype must be loaded *before* package typog, a fact that we encode in `\iftypog@microtype@preloaded`.

`ftypog@microtype@preloaded`

```
26 \newif\iftypog@microtype@preloaded
27
```

`require@preloaded@microtype`

It is easy to determine whether microtype has been sourced. We raise to the occasion and define a pair of check macros which simplify the test for the correct microtype load state.

```
28 \ifdefined\MT@MT
29   \typog@typeout{package microtype preloaded}%
30   \typog@microtype@preloadedtrue
31   \def\typog@require@preloaded@microtype{\relax}
32 \else
33   \typog@microtype@preloadedfalse
34   \def\typog@require@preloaded@microtype
35     {\PackageError{typog}%
36               {package microtype not (pre-)loaded}%
37               {package microtype must be loaded before pack-
38                 age typog}}
39 \fi
```

`\iftypog@microtype@loaded`

```
40 \newif\iftypog@microtype@loaded
41
```

`\typog@require@microtype`

This code duplicates `\typog@require@preloaded@microtype`; the only difference is that we call the test *after* the preamble was processed.

```
42 \AtBeginDocument{
43   \ifdefined\MT@MT
44     \typog@typeout{package microtype loaded}%
45     \typog@microtype@loadedtrue
46     \def\typog@require@microtype{\relax}
47   \else
48     \typog@microtype@loadedfalse
49     \def\typog@require@microtype
50       {\PackageError{typog}%
51               {package microtype not loaded}%
52               {require package microtype before package ty-
53                 pog}}%
54   \fi
55 }
```

Our own state...

`ypog@mathitalicscorrection`

```
56 \newmuskip\typog@mathitalicscorrection
```

```

ypog@textitalicscorrection
    57 \newlength{\typog@textitalicscorrection}

\typog@ligaturekern
    58 \newlength{\typog@ligaturekern}

\typog@raisecapitaldash
    59 \newlength{\typog@raisecapitaldash}

ypog@raisecapitalguillemets
    60 \newlength{\typog@raisecapitalguillemets}

\typog@raisecapitalhyphen
    61 \newlength{\typog@raisecapitalhyphen}

\typog@raisecapitaltimes
    62 \newlength{\typog@raisecapitaltimes}

\typog@raiseguillemets
    63 \newlength{\typog@raiseguillemets}

\typog@raisefiguredash
    64 \newlength{\typog@raisefiguredash}

\typog@slashkern
    65 \newlength{\typog@slashkern}

\typog@breakpenalty
    66 \newcommand*{\typog@breakpenalty}{\exhyphenpenalty}

\typog@dim@unit We would like to express the argument values for example of \kernedhyphen*
and \kernedhyphen as multiples of a thousandth of an em. Therefore, we define
a dimen as »base unit« which simplifies matters greatly.
    67 \newlength{\typog@dim@unit}
    68 \setlength{\typog@dim@unit}{.001em}

\typog@trackingttspacing
    69 \newcommand*{\typog@trackingttspacing}{300, 90, 60}

\typog@default@shrink@i The default configuration for shrink values.
    70 \newcommand*{\typog@default@shrink@i}{5}

\typog@default@shrink@ii
    71 \newcommand*{\typog@default@shrink@ii}{10}

\typog@default@shrink@iii
    72 \newcommand*{\typog@default@shrink@iii}{20}

\typog@shrink@i Configurable shrink values. Initialized from the typog@default@shrink@ set.
    73 \newcommand*{\typog@shrink@i}{}

```

`\typog@shrink@ii`

74 `\newcommand*{\typog@shrink@ii}{}`

`\typog@shrink@iii`

75 `\newcommand*{\typog@shrink@iii}{}`

`\typog@default@stretch@i` The default configuration for stretch values.

76 `\newcommand*{\typog@default@stretch@i}{5}`

`\typog@default@stretch@ii`

77 `\newcommand*{\typog@default@stretch@ii}{10}`

`\typog@default@stretch@iii`

78 `\newcommand*{\typog@default@stretch@iii}{20}`

`\typog@stretch@i` Configurable stretch values. Initialized from the `\typog@default@stretch` set.

79 `\newcommand*{\typog@stretch@i}{}`

`\typog@stretch@ii`

80 `\newcommand*{\typog@stretch@ii}{}`

`\typog@stretch@iii`

81 `\newcommand*{\typog@stretch@iii}{}`

Setup

`\typogsetup` (*env.*) An empty argument list resets all initialized values to their defaults.

82 `\NewDocumentEnvironment{typogsetup}{m}`

83 `{\def\typog@@arg{#1}%`

84 `\ifx\typog@@arg\empty`

85 `\typog@initialize@options`

86 `\else`

87 `\setkeys{typog}{#1}%`

88 `\fi`

89 `\ignorespaces}`

90 `{\ignorespacesafterend}`

`\typogget`

91 `\NewDocumentCommand{\typogget}{m}{\csname typog@#1\endcsname}`

92

A.1 Information

`\typog@round@dim@to@tenths`

```

93 \ExplSyntaxOn
94 \newcommand*{\typog@round@dim@to@tenths}[1]
95   {\fp_to_decimal:n {round(10 * \dim_to_fp:n{#1} / 1\p@) / 10}}
96 \ExplSyntaxOff
97

```

`\typog@formatsizeinfo` Arguments 1 and 2 are the font size and the line spacing. The third parameter adds (decorative) units to both numbers.

```

98 \newcommand*{\typog@formatsizeinfo}[3]
99   {#1#3\kernedslash #2#3}
100

```

`\fontsizeinfo` All macros defined inside of `\fontsizeinfo` must be global because the call can occur inside of a group.

The two `\edefs` at the beginning capture the desired values at the point where the macro *is called*. The user-macro is tricky for we need a global macro with a constructed name and an associated starred version.

Implementation Note

`\@ifstar` caused too many problems which `\@ifnextchar` in combination with `\@gobble` avoid.

```

101 \NewDocumentCommand{\fontsizeinfo}{s m}
102   {\global\expandafter\edef\csname typog@fontsize@#2\endcsname
103     {\typog@round@dim@to@tenths{\fontdimen6\font}}}%
104   \global\expandafter\edef\csname typog@linespacing@#2\endcsname
105     {\typog@round@dim@to@tenths{\baselineskip}}}%
106   \protected\expandafter\gdef\csname #2\endcsname
107     {\@ifnextchar*{\typog@formatsizeinfo
108                   {\csname typog@fontsize@#2\endcsname}%
109                   {\csname typog@linespacing@#2\endcsname}%
110                   }% no unit
111                   \ignorespaces % eat spaces after star
112                   \@gobble} % consume the star itself
113   {\typog@formatsizeinfo
114     {\csname typog@fontsize@#2\endcsname}%
115     {\csname typog@linespacing@#2\endcsname}%
116     {\,pt}% decorative unit 'pt'
117   }}
118

```

`@default@inspect@id@prefix` Id-prefix for those `typoinspect` environments that were not identified by the user.

```

119 \newcommand*{\typog@default@inspect@id@prefix}{a-}

```

`typog@inspect@count` Counter to supply unique number and in turn *<id>* for those `typoinspect` environments that were not identified by the user.

```

120 \newcounter{typog@inspect@count}

```

`typoginspect` (*env.*)

```

121 \define@key[typog]{typoginspect}{tracingboxes}[\maxdimen]%
122     {\def\typog@typoginspect@tracingboxes{#1}}
123 \NewDocumentEnvironment{typoginspect}{0}{ m}
124     {\def\typog@typoginspect@tracingboxes{\m@ne}%
125     \setkeys[typog]{typoginspect}{#1}%

```

If the user does not supply an $\langle id \rangle$, we fall back to our own counter and construct a hopefully unique $\langle id \rangle$ from that.

```

126     \edef\typog@@arg{#2}%
127     \ifx\typog@@arg\empty
128         \stepcounter{typoginspect@count}%
129         \edef\typog@id{\typog@default@inspect@id@prefix\arabic{typoginspect@count}}
130     \else
131         \edef\typog@id{\typog@trim@spaces{\typog@@arg}}%
132     \fi
133     \typeout{<typog-inspect id="\typog@id" job="\jobname" line="\the\inputlineno"

```

Set both badness thresholds to absurdly low values as to activate \TeX 's reports.

```

134     \hbadness=\m@ne
135     \vbadness=\m@ne

```

Carefully select the tracing functionality we want (to improve our typography). Too much trace data distracts and the user always can turn on more tracing at the beginning of the environment.

```

136     \tracingnone
137     \tracingpages=\@ne
138     \tracingparagraphs=\@ne
139     \showboxbreadth=\typog@typoginspect@tracingboxes
140     \showboxdepth=\typog@typoginspect@tracingboxes}
141     {\typeout{</typog-inspect>}}%
142     \ignorespacesafterend}

```

`typoginspectpar` (*env.*) Companion environment to `typoginspect` which adds a `\par` before the end of the group.

```

143 \NewDocumentEnvironment{typoginspectpar}{m}
144     {\typoginspect{#1}}
145     {\par\endtypoginspect}
146

```

A.2 Hyphenation

`\typog@allowhyphenation` Re-enable automatic hyphenation.

The same or almost the same implementation can be found in `babel` as macro `\bbl@allowhyphens` and `hyphenat` as macro `\prw@zbreak`.

```

147 \newcommand*{\typog@allowhyphenation}
148     {\ifvmode

```

```

149     \relax
150   \else
151     \nobreak
152     \hskip\z@skip
153   \fi}
154

```

`\allowhyphenation` Define a user-visible alias unless the name is already used.

```

155 \unless\ifdefined\allowhyphenation
156   \let\allowhyphenation=\typog@allowhyphenation
157 \fi
158

```

`\breakpoint` The starred form inhibits hyphenation of the right-hand component.

```

159 \NewDocumentCommand{\breakpoint}{s}
160   {\discretionary{}{}{}}%
161   \IfBooleanTF{#1}%
162     {\ignorespaces}%
163     {\typog@allowhyphenation}}
164

```

PDF-substitute definition

```

165 \typog@register@pdfsubstitute{
166   \def\breakpoint#1{\if*\detokenize{#1}\ignorespaces\fi}%
167 }
168

```

A.3 Disable/Break Ligatures

`\typog@hyphen` We define our own hyphen so the user can override the definition in a pinch.

```

169 \newcommand*{\typog@hyphen}{\char‘-}
170

```

`\nolig`

```

171 \NewDocumentCommand{\nolig}{s o}
172   {\dimen0=\IfNoValueTF{#2}{\typog@ligaturekern}{#2\typog@dim@unit}%
173   \IfBooleanTF{#1}%
174     {\kern\dimen0\ignorespaces}%
175     {\discretionary{\typog@hyphen}{}{\kern\dimen0}%
176     \typog@allowhyphenation
177     \IfNoValueTF{#2}{\ignorespaces}}}
178

```

The PDF-ready version of `\nolig` cannot be implemented with `\futurelet`.
Doh!

```

179 \typog@register@pdfsubstitute{
180   \RenewExpandableDocumentCommand{\nolig}{s o m}{%
181     \ifx\typog@TYPOG#3\typog@TYPOG

```

```

182     \relax
183   \else
184     \ifx\relax#3\relax
185       \relax
186     \else
187       \PackageError{typog}
188         {Missing third argument of \nolig}
189         {Append empty group or \relax after macro in-
vocation}
190     \fi
191   \fi}
192 }
193

```

A.4 Manual Italic Correction

`@itcorr@text@unconditional` Fallback italics correction for text mode.

```

194 \newcommand*{\typog@itcorr@text@unconditional}[1]
195   {\kern#1\typog@textitalicscorrection}

```

`\typog@itcorr@text` Conditional italics correction depending on the current font's own italics correction, i. e., `\fontdimen1`.

```

196 \newcommand*{\typog@itcorr@text}[1]
197   {\def\typog@@strength{#1}%
198     \dimen0=\fontdimen1\font
199     \ifdim\dimen0=\z@
200       \typog@itcorr@text@unconditional{\typog@@strength}%
201     \else
202       \kern\typog@@strength\dimen0
203     \fi}

```

`\typog@itcorr@math` Italics correction for math mode.

```

204 \newcommand*{\typog@itcorr@math}[1]
205   {\mkern#1\typog@mathitalicscorrection}

```

`\itcorr` If the font has no italics correction we fall back to our own length. In text mode the starred version always uses the fallback. The star is a no-op in math mode.

```

206 \NewDocumentCommand{\itcorr}{s m}
207   {\ifmmode
208     \typog@itcorr@math{#2}%
209   \else
210     \IfBooleanTF{#1}%
211       {\typog@itcorr@text{#2}}%
212       {\typog@itcorr@text@unconditional{#2}}%
213     \fi}

```

PDF-substitute definition

```

214 \typog@register@pdfsubstitute{
215   \RenewExpandableDocumentCommand{\itcorr}{s m}{}
216 }
217

```

A.5 Apply Extra Kerning

Slash

`\typog@forwardslash` We define our own forward-slash so the user can override the definition in a pinch.

```
218 \newcommand*{\typog@forwardslash}{\char‘/}
```

`\kernedslash` Macro `\kernedslash` introduces a hyphenation possibility right after the dash, whereas the starred version does not.

By the way, `\slash` expands to `‘/\penalty\exhyphenpenalty’`.

```
219 \NewDocumentCommand{\kernedslash}{s}
220   {\hspace*{\typog@slashkern}%
221    \typog@forwardslash
222    \IfBooleanTF{#1}%
223     {\hspace*{\typog@slashkern}\ignorespaces}%
224     {\typog@breakpoint\typog@allowhyphenation\hspace*{\typog@slashkern}}}
```

PDF-substitute definition

```
225 \typog@register@pdfsubstitute{
226   \def\kernedslash#1{\if*\detokenize{#1}/\ignorespaces\else/#1\fi}%
227 }
228
```

Hyphen

`\kernedhyphen`

```
229 \NewDocumentCommand{\kernedhyphen}{s O{0} m m}
230   {\ifmmode
231     \mspace{\muexpr(#3 mu) * 18 / 1000}%
232     \raisebox{#2\typog@dim@unit}{\mathord{-}}%
233     \mspace{\muexpr(#4 mu) * 18 / 1000}%
234   \else
235     \def\typog@@auto{*}%
236     \def\typog@@optarg{#2}%
237     \hspace*{#3\typog@dim@unit}%
238     \raisebox{\ifx\typog@@optarg\typog@@auto
239               \typog@raisecapitalhyphen
240               \else
241                 \typog@@optarg\typog@dim@unit
242               \fi}{\typog@hyphen}%
243     \hspace{#4\typog@dim@unit}%
244     \IfBooleanT{#1}{\nobreak}%
245   \fi}
```

PDF-substitute definition

```
246 \typog@register@pdfsubstitute{
247   \RenewExpandableDocumentCommand{\kernedhyphen}{s o m m}{-}
248 }
```

One-argument shorthands.

`\leftkernedhyphen` Apply kerning on the left-hand side of the hyphen only.

```
249 \NewDocumentCommand{\leftkernedhyphen}{s O{0} m}
250   {\IfBooleanTF{#1}%
251     {\kernedhyphen*[#2]{#3}{0}\ignorespaces}%
252     {\kernedhyphen[#2]{#3}{0}}}
```

PDF-substitute definition

```
253 \typog@register@pdfsubstitute{
254   \RenewExpandableDocumentCommand{\leftkernedhyphen}{s o m}{-}
255 }
256
```

`\rightkernedhyphen` Apply kerning on the right-hand side of the hyphen only.

```
257 \NewDocumentCommand{\rightkernedhyphen}{s O{0} m}
258   {\IfBooleanTF{#1}%
259     {\kernedhyphen*[#2]{0}{#3}\ignorespaces}%
260     {\kernedhyphen[#2]{0}{#3}}}
```

PDF-substitute definition

```
261 \typog@register@pdfsubstitute{
262   \RenewExpandableDocumentCommand{\rightkernedhyphen}{s o m}{-}
263 }
264
```

A.6 Raise Selected Characters

`\typog@breakpoint` We want our own penalty for a line-break at a particular point. The predefined `\allowbreak` is too eager. A package-private, user-configurable penalty fits best.

```
265 \newcommand*{\typog@breakpoint}
266   {\penalty\typog@breakpenalty}
```

`\capitalhyphen` Macro `\capitalhyphen` introduces a hyphenation possibility right after the dash, whereas the starred version does not.

```
267 \NewDocumentCommand{\capitalhyphen}{s}
268   {\raisebox{\typog@raisecapitalhyphen}{\typog@hyphen}%
269     \IfBooleanTF{#1}%
270       {\ignorespaces}%
271       {\typog@breakpoint\typog@allowhyphenation}}}
```

The non-hyperref version's code is straightforward. The `\pdfstringdef-DisableCommands` version must be expandable and must match the other version's signature. Yikes! We exploit the fact that conditions are expandable. However, we cannot use `\typog@hyphen` in the expansion as `\char` gets in the way. So, we fall back to the least common denominator and use a bare dash.

```

272 \typog@register@pdfsubstitute{
273   \def\capitalhyphen#1{%
274     \if*\detokenize{#1}%
275       -\ignorespaces
276     \else
277       -#1%
278     \fi}
279 }
280

```

`\capitalendash` Macro `\capitalendash` introduces a hyphenation possibility right after the dash; its starred version does not.

```

281 \NewDocumentCommand{\capitalendash}{s}
282   {\raisebox{\typog@raisecapitaldash}{\textendash}%
283    \IfBooleanTF{#1}%
284      {\ignorespaces}%
285      {\typog@breakpoint\typog@allowhyphenation}}
286 \let\capitaldash=\capitalendash

```

PDF-substitute definition

```

287 \typog@register@pdfsubstitute{
288   \def\capitalendash#1{%
289     \if*\detokenize{#1}%
290       \textendash\ignorespaces
291     \else
292       \textendash#1%
293     \fi}
294   \let\capitaldash=\capitalendash
295 }
296

```

`\capitalemdash` Macro `\capitalemdash` introduces a hyphenation possibility right after the dash; its starred version does not.

```

297 \NewDocumentCommand{\capitalemdash}{s}
298   {\raisebox{\typog@raisecapitaldash}{\textemdash}%
299    \IfBooleanTF{#1}%
300      {\ignorespaces}%
301      {\typog@breakpoint\typog@allowhyphenation}}

```

PDF-substitute definition

```

302 \typog@register@pdfsubstitute{
303   \def\capitalemdash#1{%
304     \if*\detokenize{#1}%
305       \textemdash\ignorespaces
306     \else
307       \textemdash#1%
308     \fi}
309 }
310

```

`\figuredash` Macro `\figuredash` introduces a hyphenation possibility right after the dash; its starred version does not.

```
311 \NewDocumentCommand{\figuredash}{s}
312   {\raisebox{\typog@raisefiguredash}{\textendash}%
313    \IfBooleanTF{#1}%
314     {\ignorespaces}%
315     {\typog@breakpoint\typog@allowhyphenation}}
```

PDF-substitute definition

```
316 \typog@register@pdfsubstitute{\let\figuredash=\capitaldash}
317
```

`\capitaltimes`

```
318 \NewDocumentCommand{\capitaltimes}{}
319   {\ifmmode
320     \mathbin{\raisebox{\typog@raisecapitaltimes}{\math@th\times}}%
321     \else
322     \raisebox{\typog@raisecapitaltimes}{\texttimes}%
323     \fi}
```

PDF-substitute definition

```
324 \typog@register@pdfsubstitute{
325   \RenewExpandableDocumentCommand{\capitaltimes}{}{\texttimes}
326 }
327
```

`\singleguillemetleft`

```
328 \NewDocumentCommand{\singleguillemetleft}{}
329   {\typog@allowhyphenation
330    \raisebox{\typog@raiseguilletts}{\guilsinglleft}}
```

PDF-substitute definition

```
331 \typog@register@pdfsubstitute{\let\singleguillemetleft\guilsinglleft}
```

`\singleguillemetright`

```
332 \NewDocumentCommand{\singleguillemetright}{}
333   {\raisebox{\typog@raiseguilletts}{\guilsinglright}%
334    \typog@allowhyphenation}
```

PDF-substitute definition

```
335 \typog@register@pdfsubstitute{\let\singleguillemetright\guilsinglright}
```

`\doubleguillemetleft`

```
336 \NewDocumentCommand{\doubleguillemetleft}{}
337   {\typog@allowhyphenation
338    \raisebox{\typog@raiseguilletts}{\guillemotleft}}
```


PDF-substitute definition

```
339 \typog@register@pdfsubstitute{\let\doubleguillemetleft\guillemotleft}
```

```
\doubleguillemetright
```

```
340 \NewDocumentCommand{\doubleguillemetright}{}
341   {\raisebox{\typog@raiseguilletts}{\guillemotright}%
342    \typog@allowhyphenation}
```

PDF-substitute definition

```
343 \typog@register@pdfsubstitute{\let\doubleguillemetright\guillemotright}
```

```
\Singleguillemetleft
```

```
344 \NewDocumentCommand{\Singleguillemetleft}{}
345   {\typog@allowhyphenation
346    \raisebox{\typog@raisecapitalguilletts}{\guilsinglleft}}
```

PDF-substitute definition

```
347 \typog@register@pdfsubstitute{\let\Singleguillemetleft\guilsinglleft}
```

```
\Singleguillemetright
```

```
348 \NewDocumentCommand{\Singleguillemetright}{}
349   {\raisebox{\typog@raisecapitalguilletts}{\guilsinglright}%
350    \typog@allowhyphenation}
```

PDF-substitute definition

```
351 \typog@register@pdfsubstitute{\let\Singleguillemetright\guilsinglright}
```

```
\Doubleguillemetleft
```

```
352 \NewDocumentCommand{\Doubleguillemetleft}{}
353   {\typog@allowhyphenation
354    \raisebox{\typog@raisecapitalguilletts}{\guillemotleft}}
```

PDF-substitute definition

```
355 \typog@register@pdfsubstitute{\let\Doubleguillemetleft\guillemotleft}
```

```
\Doubleguillemetright
```

```
356 \NewDocumentCommand{\Doubleguillemetright}{}
357   {\raisebox{\typog@raisecapitalguilletts}{\guillemotright}%
358    \typog@allowhyphenation}
```

PDF-substitute definition

```
359 \typog@register@pdfsubstitute{\let\Doubleguillemetright\guillemotright}
```

```
360
```

A.7 Align Last Line of a Paragraph

The code of environment `lastlineraggedleftpar` has been inspired by macro `\lastlineraggedleft` [32, Sec. 2].

`lastlineraggedleftpar` (*env.*)

```
361 \NewDocumentEnvironment{lastlineraggedleftpar}{}
362   {\lastlinefit=0%
363    \setlength{\leftskip}{\z@ \@plus 1fil}%
364    \setlength{\rightskip}{-\leftskip}%
365    \setlength{\parfillskip}{\leftskip}}
366   {\par}
```

`lastlineflushrightpar` (*env.*) Define `lastlineflushrightpar` as an alias of `lastlineraggedleftpar`.

```
367 \let\lastlineflushrightpar=\lastlineraggedleftpar
368 \let\endlastlineflushrightpar=\endlastlineraggedleftpar
369
```

`lastlinecenteredpar` (*env.*) The code of environment `lastlinecenteredpar` has been inspired by *Tex By Topic* [10, Sec. 18.3.1].

```
370 \NewDocumentEnvironment{lastlinecenteredpar}{}
371   {\lastlinefit=0%
372    \setlength{\leftskip}{\z@ \@plus .5fil}%
373    \setlength{\rightskip}{-\leftskip}%
374    \setlength{\parfillskip}{\z@ \@plus 1fil}}
375   {\par}
376
```

A.8 Fill Last Line of a Paragraph

`shortenpar` (*env.*)

```
377 \NewDocumentEnvironment{shortenpar}{}
378   {\advance\looseness by -1
379    \ifnum\tracingparagraphs>0
380      \typeout{@ looseness \the\looseness}%
381      \fi}
382   {\par}
383
```

`prolongpar` (*env.*) We try to be prudent and inhibit hyphenation of the next-to-last line just in case the longer paragraph could be cheaply achieved by hyphenation – at the worst – of the last word.

```
384 \NewDocumentEnvironment{prolongpar}{}
385   {\finalhyphendemerits=100000001
386    \advance\looseness by 1
387    \ifnum\tracingparagraphs>0
388      \typeout{@ looseness \the\looseness}%
389      \fi}
390   {\par}
391
```

`xtindentpar@zero@parindent` This auxiliary macro and the following one are meant as an easy means to override the defaults of the user-visible environment `covernextindentpar`.

```
392 \newcommand*{\typog@covernextindentpar@zero@parindent}{2em}
```

`ndindentpar@nonzero@parindent`

```
393 \newcommand*{\typog@covernextindentpar@nonzero@parindent}{2\parindent}
```

`covernextindentpar (env.)`

```
394 \NewDocumentEnvironment{covernextindentpar}{o}
395   {\IfNoValueTF{#1}
396     {\ifdim\parindent=\z@
397       \dimen0=\dimexpr\linewidth - \typog@covernextindentpar@zero@parindent
398     \else
399       \dimen0=\dimexpr\linewidth - \typog@covernextindentpar@nonzero@parindent
400     \fi}
401     {\dimen0=\dimexpr\linewidth - (#1)}%
402     \parfillskip=\dimen0 \@minus \dimen0
403     \relax}
404   {\par}
405
```

`lastlinepar@zero@parindent` These auxiliary macros are meant as a means to override the defaults of the user-visible environment `openlastlinepar`.

```
406 \newcommand*{\typog@openlastlinepar@zero@parindent}{2em}
```

`ntlinepar@nonzero@parindent`

```
407 \newcommand*{\typog@openlastlinepar@nonzero@parindent}{2\parindent}
```

`openlastlinepar (env.)` Compare with the suggestion in Ref. [27](#).

```
408 \NewDocumentEnvironment{openlastlinepar}{o}
409   {\IfNoValueTF{#1}
410     {\ifdim\parindent=\z@
411       \skip0=\typog@openlastlinepar@zero@parindent
412       \@plus 1fil
413       \@minus \typog@openlastlinepar@zero@parindent
414     \else
415       \skip0=\typog@openlastlinepar@nonzero@parindent
416       \@plus 1fil
417       \@minus \typog@openlastlinepar@nonzero@parindent
418     \fi}
419     {\dimen0=\dimexpr#1\relax
420     \skip0=\dimen0 \@plus 1fil \@minus \dimen0}
421     \parfillskip=\skip0}
422   {\par}
423
```

A.9 Spacing

`\widespacestrength` Weight factor (“strength”) for `\fontdimen7`, the extra width of a sentence-ending space, we apply to construct our `\widespace` if `\fontdimen7` $\neq 0$. Can be increased to get a more pronounced effect.

```
424 \newcommand*{\widespacestrength}{1.}
```

`\widespacescale` Scale factor we apply to the glue of the normal space to setup the glue of our `\widespacescale`. Also used in the fall-back calculation for the width if `\fontdimen7` = 0.

```
425 \newcommand*{\widespacescale}{1.125}
```

`\widespace`

```
426 \NewDocumentCommand{\widespace}{s}
427   {\IfBooleanTF{#1}%
428     {\dimen0=\widespacescale\fontdimen2\font}%
429     {\ifdim\fontdimen7\font=\z@
430       \dimen0=\widespacescale\fontdimen2\font
431       \else
432         \dimen0=\dimexpr\fontdimen2\font +
433           \widespacestrength\fontdimen7\font
434       \fi}%
435     \hskip \glueexpr\dimen0
436           \@plus \widespacescale\fontdimen3\font
437           \@minus \widespacescale\fontdimen4\font
438     \ignorespaces}
439
```

`\narrowpacestrength` Weight factor (“strength”) for `\fontdimen7`, the extra width of a sentence-ending space, we apply to construct our `\narrowpace` if `\fontdimen7` $\neq 0$. Can be increased to get a more pronounced effect.

```
440 \newcommand*{\narrowpacestrength}{.5}
```

`\narrowspacescale` Scale factor we apply to the glue of the normal space to setup the glue of our `\narrowspacescale`. Also used in the fall-back calculation for the width if `\fontdimen7` = 0.

```
441 \newcommand*{\narrowspacescale}{.9375}
```

`\narrowpace`

```
442 \NewDocumentCommand{\narrowpace}{s}
443   {\IfBooleanTF{#1}%
444     {\dimen0=\narrowspacescale\fontdimen2\font}%
445     {\ifdim\fontdimen7\font=\z@
446       \dimen0=\narrowspacescale\fontdimen2\font
447       \else
448         \dimen0=\dimexpr\fontdimen2\font -
449           \narrowpacestrength\fontdimen7\font
```

```

450     \fi}%
451     \hskip \glueexpr\dimen0
452           \@plus \narrowspacescale\fontdimen3\font
453           \@minus \narrowspacescale\fontdimen4\font
454     \ignorespaces}
455

```

See also: TeX by Topic [10, ch. 20, p. 185–190].

loosespacing (*env.*)

```

456 \NewDocumentEnvironment{loosespacing}{0{1}}
457 {\dimen2=\fontdimen2\font
458   \ifcase #1
459     \spaceskip=\z@
460     \or % 1      +5%
461       \spaceskip=1.05\dimen2 \@plus .5\dimen2 \@minus .1\dimen2
462       \or % 2      +10%
463       \spaceskip=1.1\dimen2 \@plus .5\dimen2 \@minus .1\dimen2
464       \or % 3      +20%
465       \spaceskip=1.2\dimen2 \@plus .6\dimen2 \@minus .2\dimen2
466       \else % >= 4  +30%
467       \spaceskip=1.3\dimen2 \@plus .8\dimen2 \@minus .3\dimen2
468     \fi
469     \ignorespaces}
470 {\ignorespacesafterend}
471

```

tightspacing (*env.*)

```

472 \NewDocumentEnvironment{tightspacing}{0{1}}
473 {\dimen2=\fontdimen2\font
474   \ifcase #1
475     \spaceskip=\z@
476     \or % 1      -1.25%
477       \spaceskip=.9875\dimen2 \@plus .0125\dimen2 \@minus .5\dimen2
478       \or % 2      -2.5%
479       \spaceskip=.975\dimen2 \@plus .025\dimen2 \@minus .5\dimen2
480       \or % 3      -5%
481       \spaceskip=.95\dimen2 \@plus .05\dimen2 \@minus .5\dimen2
482       \else % >= 4  -10%
483       \spaceskip=.9\dimen2 \@plus .1\dimen2 \@minus .5\dimen2
484     \fi
485     \ignorespaces}
486 {\ignorespacesafterend}
487

```

A.10 Microtype Front-End

Tracking

`setfonttracking` (*env.*) To achieve the control we want, we must tinker with microtype's internals. Doh!

```

488 \NewDocumentEnvironment{setfonttracking}{m}
489   {\edef\MT@letterspace@{#1}%
490    \lsstyle
491    \ignorespaces}
492   {\ignorespacesafterend}
493 
```

Font Expansion

`typog@setup@font@expansion` Note that we cannot factor the encodings into a macro; a single encoding would qualify, though. We need to support multiple encodings and thus go with the literal solution.

```

494 \newcommand*{\typog@setup@font@expansion}
495   {\SetExpansion
496    [context = typog@shrink1,
497     shrink = \typog@shrink@i,
498     stretch = 0]%
499    {encoding = {*}}%
500    {}
501   \SetExpansion
502    [context = typog@shrink2,
503     shrink = \typog@shrink@ii,
504     stretch = 0]%
505    {encoding = {*}}%
506    {}
507   \SetExpansion
508    [context = typog@shrink3,
509     shrink = \typog@shrink@iii,
510     stretch = 0]%
511    {encoding = {*}}%
512    {}
513
514   \SetExpansion
515    [context = typog@stretch1,
516     shrink = 0,
517     stretch = \typog@stretch@i]%
518    {encoding = {*}}%
519    {}
520   \SetExpansion
521    [context = typog@stretch2,
522     shrink = 0,
523     stretch = \typog@stretch@ii]%
524    {encoding = {*}}%
525    {}
526   \SetExpansion
527    [context = typog@stretch3,
```

```

528     shrink = 0,
529     stretch = \typog@stretch@iii]%
530     {encoding = {*}}}%
531     {}
532
533   \SetExpansion
534     [context = typog@expand1,
535     shrink = \typog@shrink@i,
536     stretch = \typog@stretch@i]%
537     {encoding = {*}}}%
538     {}
539   \SetExpansion
540     [context = typog@expand2,
541     shrink = \typog@shrink@ii,
542     stretch = \typog@stretch@ii]%
543     {encoding = {*}}}%
544     {}
545   \SetExpansion
546     [context = typog@expand3,
547     shrink = \typog@shrink@iii,
548     stretch = \typog@stretch@iii]%
549     {encoding = {*}}}%
550     {}

```

`microtype@expansion@feature` We cannot even parse the `\iftypog@microtype@preloaded` part further down unless the `\ifMT@expansion` conditional exists. So we hoist this test in a macro of its own. It only gets called if package `microtype` already has been sourced.

```

551 \newcommand*{\typog@test@microtype@expansion@feature}
552   {\ifMT@expansion
553     \typog@typeout{microtype preloaded -- font expansion features avail-
554     able}%
555     \def\typog@require@microtype@expansion{\relax}
556     \typog@setup@font@expansion
557   \else
558     \PackageWarning{typog}{microtype preloaded,\space
559     but font expansion is disabled}%
560     \def\typog@require@microtype@expansion
561     {\PackageError{typog}
562     {microtype font expansion disabled}
563     {pass option 'expansion' to package microtype}}
564   \fi}

```

`require@microtype@expansion` We are all set for the initialization of the font expansion, however, we must be careful in which (load-)state package `microtype` is in. Compare the code for `\typog@require@microtype` and `\typog@require@preloaded@microtype`. Initialize our own flag and setup meaningful messages for later feature checks.

```

564 \iftypog@microtype@preloaded
565   \typog@test@microtype@expansion@feature
566 \else

```

```

567 \def\typog@require@microtype@expansion
568   {\PackageError{typog}%
569     {package microtype not (pre-)loaded, %
570     which is required for typog's font expansion}%
571     {require package microtype before package typog}}
572 \fi
573

```

setfontshrink (*env.*)

```

574 \NewDocumentEnvironment{setfontshrink}{0{1}}
575   {\typog@require@microtype@expansion
576     \ifcase#1% 0
577       \relax
578     \or % 1
579       \microtypecontext{expansion=typog@shrink1}%
580     \or % 2
581       \microtypecontext{expansion=typog@shrink2}%
582     \else % >= 3
583       \microtypecontext{expansion=typog@shrink3}%
584     \fi
585     \ignorespaces}
586   {\ignorespacesafterend}
587

```

setfontstretch (*env.*)

```

588 \NewDocumentEnvironment{setfontstretch}{0{1}}
589   {\typog@require@microtype@expansion
590     \ifcase#1% 0
591       \relax
592     \or % 1
593       \microtypecontext{expansion=typog@stretch1}%
594     \or % 2
595       \microtypecontext{expansion=typog@stretch2}%
596     \else % >= 3
597       \microtypecontext{expansion=typog@stretch3}%
598     \fi
599     \ignorespaces}
600   {\ignorespacesafterend}
601

```

setfontexpand (*env.*)

```

602 \NewDocumentEnvironment{setfontexpand}{0{1}}
603   {\typog@require@microtype@expansion
604     \ifcase#1% 0
605       \relax
606     \or % 1
607       \microtypecontext{expansion=typog@expand1}%
608     \or % 2
609       \microtypecontext{expansion=typog@expand2}%
610     \else % >= 3

```



```

611     \microtypecontext{expansion=typog@expand3}%
612     \fi
613     \ignorespaces}
614     {\ignorespacesafterend}
615

```

`nofontexpansion` (*env.*) Implementation: We proceed a different approach with respect to requiring package microtype. The semantics of the macro is to switch something off. If it is not on because the necessary package was not loaded, a no-op is ok.

```

616 \NewDocumentEnvironment{nofontexpansion}{}
617   {\ifdefined\microtypesetup
618     \microtypesetup{expansion=false}%
619     \fi
620     \ignorespaces}
621   {\ignorespacesafterend}

```

`nofontexpand` (*env.*) Define `nofontexpand` as an alias of `nofontexpansion`.

```

622 \let\nofontexpand=\nofontexpansion
623 \let\endnofontexpand=\endnofontexpansion
624

```

Character Protrusion

`nocharprotrusion` (*env.*) See Implementation comment of `nofontexpansion`.

```

625 \NewDocumentEnvironment{nocharprotrusion}{}
626   {\ifdefined\microtypesetup
627     \microtypesetup{protrusion=false}%
628     \fi
629     \ignorespaces}
630   {\ignorespacesafterend}
631

```

A.11 Sloppy Paragraphs

`og@scaled@emergencystretch` Compute the correct scale factor for the emergency stretch even if we do not have a valid `\linewidth`.

```

632 \newcommand*{\typog@scaled@emergencystretch}[1]
633   {\emergencystretch=\ifdim\linewidth=\z@
634     #1%
635     \else
636       \dimexpr (#1) * \linewidth / \textwidth
637     \fi}
638

```

`\slightlyloppy` Macro `\slightlyloppy` takes an optional *<sloppiness>* index ranging from 0 to 8, where 0 means the same as `\fussy` and 8 or more works like `\sloppy`. The default *<sloppiness>* is 1.

```

639 \NewDocumentCommand{\slightlyloppy}{0{1}}
640 {
641   \ifcase #1% 0
642     % \tolerance=200
643     % \emergencystretch=\z@
644     % \hfuzz=.1\p@
645     % \vfuzz=\hfuzz
646     \fussy
647   \or % 1
648     \pretolerance=165%
649     \tolerance=330%
650     \typog@scaled@emergencystretch{.375em}%
651     \hfuzz=.15\p@
652     \vfuzz=\hfuzz
653   \or % 2
654     \pretolerance=265%
655     \tolerance=530%
656     \typog@scaled@emergencystretch{.75em}%
657     \hfuzz=.15\p@
658     \vfuzz=\hfuzz
659   \or % 3
660     \pretolerance=435%
661     \tolerance=870%
662     \typog@scaled@emergencystretch{1.125em}%
663     \hfuzz=.2\p@
664     \vfuzz=\hfuzz
665   \or % 4
666     \pretolerance=705%
667     \tolerance=1410%
668     \typog@scaled@emergencystretch{1.5em}%
669     \hfuzz=.3\p@
670     \vfuzz=\hfuzz
671   \or % 5
672     \pretolerance=1155%
673     \tolerance=2310%
674     \typog@scaled@emergencystretch{1.875em}%
675     \hfuzz=.35\p@
676     \vfuzz=\hfuzz
677   \or % 6
678     \pretolerance=1880%
679     \tolerance=3760%
680     \typog@scaled@emergencystretch{2.25em}%
681     \hfuzz=.4\p@
682     \vfuzz=\hfuzz
683   \or % 7
684     \pretolerance=3065%
685     \tolerance=6130%
686     \typog@scaled@emergencystretch{2.625em}%
687     \hfuzz=.45\p@
688     \vfuzz=\hfuzz
689   \else % >= 8
690     % \tolerance=9999
691     % \emergencystretch=3em

```

```

691      % \hfuzz=.5\p@
692      % \vfuzz=\hfuzz
693      \sloppy
694      \fi
695      \ignorespaces}

```

Implementation Note

- The `\tolerance` values are calculated as the geometric mean of the extreme values 200 and 9999. This means the factor

$$f = \left(\frac{9999}{200} \right)^{1/8} \approx 1.63$$

defines additional tolerances which we generously round values in the actual implementation.

- The `\emergencystretch` is scaled linearly with $\langle sloppiness \rangle$ and the ratio of the actual `\linewidth` to the (maximum) `\textwidth`.
- The `\hfuzz` values are interpolated linearly with $\langle sloppiness \rangle$ between .1pt and .5pt.

Maxima code to calculate the intermediate values.

```

Initialize. load("list_functions")$
\tolerance: logspace(log10(200), log10(9999), 9), numer;
\emergencystretch: linspace(0, 3, 9), numer;
\hfuzz: linspace(0.1, 0.5, 9);

```

`slightlyloppypar (env.)`

```

696 \NewDocumentEnvironment{slightlyloppypar}{0{1}}
697   {\par\sloppypar[#1]\ignorespaces}
698   {\par}
699

```

A.12 Vertically Partially-Tied Paragraphs

`\typog@geometric@mean` This is just the usual geometric mean of two values x and y : \sqrt{xy} .

```

700 \ExplSyntaxOn
701 \newcommand*{\typog@geometric@mean}[2]
702   {\fp_to_int:n {\sqrt{(#1) * (#2)}}}
703 \ExplSyntaxOff
704

```

`typog@mean@penalty` Reserve a private counter for the geometric-mean penalties.

```

705 \newcounter{typog@mean@penalty}
706

```

`\vtietop`

```

707 \NewDocumentCommand{\vtietop}{0{3}}
708   {\setcounter{typog@mean@penalty}
709     {\typog@geometric@mean{\@M}{\clubpenalty}}%
710   \typog@typeout{vtietop: penalties \the\@M--\the\value{typog@mean@penalty}-
    -\the\clubpenalty}%

```

```

711 \unless\ifnum\clubpenalty<\@M
712 \PackageWarning{typog}{vtietop: clubpenalty=\the\clubpenalty\space>= 10000}%
713 \fi
714 \ifcase#1% 0
715 \relax
716 \or % 1
717 \relax
718 \or % 2
719 \clubpenalties 3
720 \@M
721 \value{typog@mean@penalty}
722 \clubpenalty
723 \or % 3
724 \clubpenalties 4
725 \@M \@M
726 \value{typog@mean@penalty}
727 \clubpenalty
728 \or % 4
729 \clubpenalties 5
730 \@M \@M \@M
731 \value{typog@mean@penalty}
732 \clubpenalty
733 \or % 5
734 \clubpenalties 6
735 \@M \@M \@M \@M
736 \value{typog@mean@penalty}
737 \clubpenalty
738 \or % 6
739 \clubpenalties 7
740 \@M \@M \@M \@M \@M
741 \value{typog@mean@penalty}
742 \clubpenalty
743 \or % 7
744 \clubpenalties 8
745 \@M \@M \@M \@M \@M
746 \value{typog@mean@penalty}
747 \clubpenalty
748 \or % 8
749 \clubpenalties 9
750 \@M \@M \@M \@M \@M
751 \value{typog@mean@penalty}
752 \clubpenalty
753 \else % >= 9
754 \clubpenalties 10
755 \@M \@M \@M \@M \@M \@M \@M \@M
756 \value{typog@mean@penalty}
757 \clubpenalty
758 \fi}
759

```

vtietoppar(*env.*)

```

760 \NewDocumentEnvironment{vtietoppar}{0{3}}

```

```

761 {\vtietop[#1]}
762 {\par
763   \ignorespacesafterend}
764

```

\splicevtietop

```

765 \NewDocumentCommand{\splicevtietop}{0{3}}
766 {\let\typog@old@item=\@item
767   \def\@item[##1]{\typog@old@item[##1]\vtietop[#1]}%
768   \ignorespaces}
769

```

We define an extra style for the users of enumitem. Its only drawback is that it hard-codes the default number of tied lines (3).

```

770 \ifdefined\SetEnumitemKey
771   \SetEnumitemKey{vtietop}{first=\splicevtietop}
772 \fi
773

```

\vtiebot

```

774 \NewDocumentCommand{\vtiebot}{0{3}}
775 {\setcounter{typog@mean@penalty}
776   {\typog@geometric@mean{\@M}{\widowpenalty}}%
777   \typog@typeout{vtiebot: penalties \the\@M--\the\value{typog@mean@penalty}-
778 -\the\widowpenalty}%
779   \unless\ifnum\widowpenalty<\@M
780     \PackageWarning{typog}{vtiebot: widowpenalty=\the\widowpenalty\space>= 10000}
781   \fi
782   \ifcase#1% 0
783     \relax
784   \or % 1
785     \relax
786   \or % 2
787     \widowpenalties 3
788     \@M
789     \value{typog@mean@penalty}
790     \widowpenalty
791   \or % 3
792     \widowpenalties 4
793     \@M \@M
794     \value{typog@mean@penalty}
795     \widowpenalty
796   \or % 4
797     \widowpenalties 5
798     \@M \@M \@M
799     \value{typog@mean@penalty}
800     \widowpenalty
801   \or % 5
802     \widowpenalties 6
803     \@M \@M \@M \@M
804     \value{typog@mean@penalty}
805     \widowpenalty

```

```

805 \or % 6
806   \widowpenalties 7
807     \@M \@M \@M \@M \@M
808     \value{typog@mean@penalty}
809     \widowpenalty
810 \or % 7
811   \widowpenalties 8
812     \@M \@M \@M \@M \@M \@M
813     \value{typog@mean@penalty}
814     \widowpenalty
815 \or % 8
816   \widowpenalties 9
817     \@M \@M \@M \@M \@M \@M \@M
818     \value{typog@mean@penalty}
819     \widowpenalty
820 \else % >= 9
821   \widowpenalties 10
822     \@M \@M \@M \@M \@M \@M \@M \@M
823     \value{typog@mean@penalty}
824     \widowpenalty
825 \fi}
826

```

vtiebotpar (env.)

```

827 \NewDocumentEnvironment{vtiebotpar}{0{3}}
828   {\vtiebot[#1]}
829   {\par
830     \ignorespacesafterend}
831

```

\typog@vtiebotdisp

```

832 \NewDocumentCommand{\typog@vtiebotdisp}{m}
833   {\setcounter{typog@mean@penalty}
834     {\typog@geometric@mean{\@M}{\displaywidowpenalty}}%
835     \typog@typeout{vtiebotdisp: penalties \the\@M--\the\value{typog@mean@penalty}-
836       -\the\displaywidowpenalty}%
837     \unless\ifnum\displaywidowpenalty<\@M
838       \PackageWarning{typog}{vtiebotdisp: displaywidowpenalty=\the\displaywidowpen
839     \fi
840     \ifcase#1% 0
841       \relax
842     \or % 1
843       \relax
844     \or % 2
845       \displaywidowpenalties 3
846       \@M
847       \value{typog@mean@penalty}
848       \displaywidowpenalty
849     \or % 3
850       \displaywidowpenalties 4
851       \@M \@M
852       \value{typog@mean@penalty}

```

```

852         \displaywidowpenalty
853     \or % 4
854         \displaywidowpenalties 5
855         \@M \@M \@M
856         \value{typog@mean@penalty}
857         \displaywidowpenalty
858     \or % 5
859         \displaywidowpenalties 6
860         \@M \@M \@M \@M
861         \value{typog@mean@penalty}
862         \displaywidowpenalty
863     \or % 6
864         \displaywidowpenalties 7
865         \@M \@M \@M \@M \@M
866         \value{typog@mean@penalty}
867         \displaywidowpenalty
868     \or % 7
869         \displaywidowpenalties 8
870         \@M \@M \@M \@M \@M \@M
871         \value{typog@mean@penalty}
872         \displaywidowpenalty
873     \or % 8
874         \displaywidowpenalties 9
875         \@M \@M \@M \@M \@M \@M \@M
876         \value{typog@mean@penalty}
877         \displaywidowpenalty
878     \else % >= 9
879         \displaywidowpenalties 10
880         \@M \@M \@M \@M \@M \@M \@M \@M
881         \value{typog@mean@penalty}
882         \displaywidowpenalty
883     \fi}
884

```

vtiebotdisp (env.)

```

885 \NewDocumentEnvironment{vtiebotdisp}{0{3}}
886 {\typog@vtiebotdisp{#1}}
887 {\ignorespacesafterend}
888

```

vtiebotdisptoppar (env.)

```

889 \NewDocumentEnvironment{vtiebotdisptoppar}{0{3}o}
890 {\postdisplaypenalty=\@M
891 \predisplaypenalty=10001% in accordance with package ‘widows-
and-orphans’
892 \edef\typog@@top@lines{\IfNoValueTF{#2}{#1}{#2}}%
893 \edef\typog@@after@display@math{\vtietop[\typog@@top@lines]}%
894 \PushPostHook{display}{\aftergroup\typog@@after@display@math}%
895 \vtiebotdisp[#1]}
896 {\par
897 \PopPostHook{display}%
898 \ignorespacesafterend}

```

899

A.13 Breakable Disp. Eqs.

`breakabledisplay (env.)` We use a different default, 3, than `\allowdisplaybreaks` which utilizes 4 as its default.

```
900 \newenvironment*{breakabledisplay}[1][3]
901   {\allowdisplaybreaks[#1]}
902   {\ignorespacesafterend}
903
```

A.14 Smooth Ragged

`\typog@repeat` As we shall have to repeat the line specifications for our paragraphs so often we introduce the two argument macro `\typog@repeat` that takes a *repeat-count* and a *body* that is repeated.

```
904 \ExplSyntaxOn
905 \cs_new_eq:NN \typog@repeat \prg_replicate:nn
906
```

`\typog@mod` For error checking we shall need the modulo operation on integers, i. e., the remainder of an integral division.

```
907 \newcommand*{\typog@mod}[2]{\int_mod:nn{#1}{#2}}
908 \ExplSyntaxOff
909
```

`\typog@triplet@max@lines` Maximum number of lines a `smoothraggedright` paragraph can have with the triplet generator. The number must be divisible by 3.

```
910 \newcommand*{\typog@triplet@max@lines}{99}
911
```

`aggedrightshapetriplet (env.)` Engine for 3-line repetitions.

```
912 \define@key[typog]{smoothraggedrightshapetriplet}{leftskip}%
913   {\def\typog@@triplet@leftskip{#1}}
914 \define@key[typog]{smoothraggedrightshapetriplet}{parindent}%
915   {\def\typog@@triplet@parindent{#1}}
916 \NewDocumentEnvironment{smoothraggedrightshapetriplet}{0}{m m m}
917   {\def\typog@@triplet@leftskip{\z@}%
918    \def\typog@@triplet@parindent{\z@}%
919    \setkeys*{typog}{smoothraggedrightshapetriplet}{#1}%
920    \skip0=\typog@@triplet@leftskip\relax
921    \skip1=#2\relax
922    \skip2=#3\relax
923    \skip3=#4\relax
924    \typog@typeout{smoothraggedrightshapetriplet: skip0=\the\skip0}%
925    \typog@typeout{smoothraggedrightshapetriplet: skip1=\the\skip1}%
926    \typog@typeout{smoothraggedrightshapetriplet: skip2=\the\skip2}%
927    \typog@typeout{smoothraggedrightshapetriplet: skip3=\the\skip3}%
928    \unless\ifnum\typog@mod{\typog@triplet@max@lines}{3}=0
929      \PackageError{typog}
```



```

930             {Line number of triplet generator %
931               (\typog@triplet@max@lines) not divisible by 3}
932             {}
933     \fi
934     \edef\typog@@triplet@linespecs{%
935       \glueexpr \skip0 + \typog@@triplet@parindent\relax
936       \glueexpr \skip1 - \typog@@triplet@parindent\relax
937       \skip0 \skip2 \skip0 \skip3
938       \typog@repeat{\numexpr\typog@triplet@max@lines / 3 - 1}
939       {\skip0 \skip1 \skip0 \skip2 \skip0 \skip3}}
940     \parshape=\typog@triplet@max@lines\typog@@triplet@linespecs\relax
941     {\par}
942
typog@quintuplet@max@lines Maximum number of lines a smoothraggedright paragraph can have with the
quintuplet generator. The number must be divisible by 5.
943 \newcommand*{\typog@quintuplet@max@lines}{95}
944
edrightshapequintuplet (env.) Engine for 5-line repetitions.
945 \define@key[typog]{smoothraggedrightshapequintuplet}{leftskip}
946     {\def\typog@@quintuplet@leftskip{#1}}
947 \define@key[typog]{smoothraggedrightshapequintuplet}{parindent}
948     {\def\typog@@quintuplet@parindent{#1}}
949 \NewDocumentEnvironment{smoothraggedrightshapequintuplet}{0{} m m m m m}
950     {\def\typog@@quintuplet@leftskip{\z@}%
951     \def\typog@@quintuplet@parindent{\z@}%
952     \setkeys*{typog}{smoothraggedrightshapequintuplet}{#1}%
953     \skip0=\typog@@quintuplet@leftskip
954     \skip1=#2\relax
955     \skip2=#3\relax
956     \skip3=#4\relax
957     \skip4=#5\relax
958     \skip5=#6\relax
959     \typog@typeout{smoothraggedrightshapequintuplet: skip0=\the\skip0}%
960     \typog@typeout{smoothraggedrightshapequintuplet: skip1=\the\skip1}%
961     \typog@typeout{smoothraggedrightshapequintuplet: skip2=\the\skip2}%
962     \typog@typeout{smoothraggedrightshapequintuplet: skip3=\the\skip3}%
963     \typog@typeout{smoothraggedrightshapequintuplet: skip4=\the\skip4}%
964     \typog@typeout{smoothraggedrightshapequintuplet: skip5=\the\skip5}%
965     \unless\ifnum\typog@mod{\typog@quintuplet@max@lines}{5}=0
966     \PackageError{typog}
967         {Line number of quintuplet generator %
968           (\typog@quintuplet@max@lines) not divisible by 5}
969     {}
970     \fi
971     \edef\typog@@quintuplet@linespecs{%
972       \glueexpr \skip0 + \typog@@quintuplet@parindent\relax
973       \glueexpr \skip1 - \typog@@quintuplet@parindent\relax
974       \skip0 \skip2 \skip0 \skip3 \skip0 \skip4 \skip0 \skip5
975       \typog@repeat{\numexpr\typog@quintuplet@max@lines / 5 - 1}
976       {\skip0 \skip1 \skip0 \skip2 \skip0 \skip3 \skip0 \skip4 \s

```

```

977 \parshape=\typog@quintuplet@max@lines\typog@@quintuplet@linespecs\relax}
978 {\par}

\typog@septuplet@max@lines Maximum number of lines a smoothraggedright paragraph can have with the
septuplet generator. The number must be divisible by 7.

979 \newcommand*{\typog@septuplet@max@lines}{98}
980

gedrightshapeseptuplet (env.) Engine for 7-line repetitions.

981 \define@key[typog]{smoothraggedrightshapeseptuplet}{leftskip}%
982 {\def\typog@septuplet@leftskip{#1}}
983 \define@key[typog]{smoothraggedrightshapeseptuplet}{parindent}%
984 {\def\typog@septuplet@parindent{#1}}
985 \NewDocumentEnvironment{smoothraggedrightshapeseptuplet}{0}{ m m m m m m m}
986 {\def\typog@@septuplet@leftskip{\z@}%
987 \def\typog@@septuplet@parindent{\z@}%
988 \setkeys*{typog}{smoothraggedrightshapeseptuplet}{#1}%
989 \skip0=\typog@septuplet@leftskip
990 \skip1=#2\relax
991 \skip2=#3\relax
992 \skip3=#4\relax
993 \skip4=#5\relax
994 \skip5=#6\relax
995 \skip6=#7\relax
996 \skip7=#8\relax
997 \typog@typeout{smoothraggedrightshapeseptuplet: skip0=\the\skip0}%
998 \typog@typeout{smoothraggedrightshapeseptuplet: skip1=\the\skip1}%
999 \typog@typeout{smoothraggedrightshapeseptuplet: skip2=\the\skip2}%
1000 \typog@typeout{smoothraggedrightshapeseptuplet: skip3=\the\skip3}%
1001 \typog@typeout{smoothraggedrightshapeseptuplet: skip4=\the\skip4}%
1002 \typog@typeout{smoothraggedrightshapeseptuplet: skip5=\the\skip5}%
1003 \typog@typeout{smoothraggedrightshapeseptuplet: skip6=\the\skip6}%
1004 \typog@typeout{smoothraggedrightshapeseptuplet: skip7=\the\skip7}%
1005 \unless\ifnum\typog@mod{\typog@septuplet@max@lines}{7}=0
1006 \PackageError{typog}
1007 {Line number of septuplet generator %
1008 (\typog@septuplet@max@lines) not divisible by 7}
1009 {}
1010 \fi
1011 \edef\typog@@septuplet@linespecs{%
1012 \glueexpr \skip0 + \typog@septuplet@parindent\relax
1013 \glueexpr \skip1 - \typog@septuplet@parindent\relax
1014 \skip0 \skip2 \skip0 \skip3 \skip0 \skip4 \skip0 \skip5 \
1015 \typog@repeat{\numexpr\typog@septuplet@max@lines / 7 - 1}
1016 {\skip0 \skip1 \skip0 \skip2 \skip0 \skip3 \skip0 \skip4 \s
1017 \parshape=\typog@septuplet@max@lines\typog@@septuplet@linespecs\relax}
1018 {\par}
1019

smoothraggedrightfuzzfactor

1020 \newcommand*{\smoothraggedrightfuzzfactor}{1.0}

```

smoothraggedrightgenerator

```
1021 \newcommand*{\smoothraggedrightgenerator}{triplet}
```

\smoothraggedrightleftskip

```
1022 \newlength{\smoothraggedrightleftskip}
```

smoothraggedrightparindent

```
1023 \newlength{\smoothraggedrightparindent}
```

\smoothraggedrighttragwidth

```
1024 \newlength{\smoothraggedrighttragwidth}
1025 \setlength{\smoothraggedrighttragwidth}{2em}
1026
```

smoothraggedrightpar (*env.*) The longest line will be `\linewidth` wide unless overridden by optional argument `linewidth`.

```
1027 \newdimen{\typog@fuzzwidth}
1028
1029 \define@key[typog]{smoothraggedrightpar}{linewidth}%
1030         {\def\typog@@linewidth{#1}}
1031
1032 \NewDocumentEnvironment{smoothraggedrightpar}{0{}}
1033     {\edef\typog@@linewidth{\linewidth}%
1034      \setkeys[typog]{smoothraggedrightpar}{#1}%

```

Convert generator name to an integer suitable for `\ifcase`.

```
1035     \edef\typog@@generatorchoice{%
1036         \ifnum\pdf@strcmp{\smoothraggedrightgenerator}{triplet}=\z@
1037             0%
1038         \else
1039             \ifnum\pdf@strcmp{\smoothraggedrightgenerator}{quintuplet}=\z@
1040                 1%
1041             \else
1042                 \ifnum\pdf@strcmp{\smoothraggedrightgenerator}{septuplet}=\z@
1043                     2%
1044                 \else
1045                     \PackageError{typog}
1046                         {smoothraggedright: unknown generator name}
1047                         {valid generator names are triplet, quin-
1048                             tuplet, and septuplet}%
1048                     \fi
1049                 \fi
1049             \fi}%
1050
```

Obey to the indentation prescribed by any list environment.

```
1051     \let\typog@@smoothraggedrightleftskip=\smoothraggedrightleftskip
1052     \ifnum\@listdepth>0
1053         \addtolength{\typog@@smoothraggedrightleftskip}{\leftmargin}%
1054     \fi

```

Scale the fuzz-width by the user's factor. Later we shall rescale again specifically for each generator.

```
1055     \typog@fuzzwidth=\smoothraggedrightfuzzfactor\smoothraggedrighttragwidth

```

Now for the generator-specific code...

```

1056 \ifcase\typog@@generatorchoice

      generator=triplet produces a »short line – long line – middle length
line« sequence.

1057 \typog@fuzzwidth=.25\smoothraggedrightragwidth
1058 \typog@typeout{smoothraggedright: generator=triplet, typog@fuzzwidth=\the\typog@fuzzwidth}%
1059 \smoothraggedrightshapetriplet[leftskip=\typog@@smoothraggedrightleftskip,
1060                                parindent=\glueexpr\smoothraggedrightparindent-
indent,
1061                                #1]%
1062 {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth
1063  + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
1064 {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth}% (3)
1065 {\glueexpr (\typog@@linewidth * 2 - \smoothraggedrightrag-
width) / 2
1066  + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (2)
1067 \or

      generator=quintuplet.

1068 \typog@fuzzwidth=.125\smoothraggedrightragwidth
1069 \typog@typeout{smoothraggedright: generator=quintuplet, typog@fuzzwidth=\the\typog@fuzzwidth}%
1070 \smoothraggedrightshapequintuplet[leftskip=\typog@@smoothraggedrightleftskip,
1071                                   parindent=\glueexpr\smoothraggedrightparindent-
indent,
1072                                   #1]%
1073 {\glueexpr (\typog@@linewidth * 4 - \smoothraggedrightrag-
width * 3) / 4
1074  + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (2)
1075 {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth\relax}% (5)
1076 {\glueexpr (\typog@@linewidth * 2 - \smoothraggedrightrag-
width) / 2
1077  + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (3)
1078 {\glueexpr (\typog@@linewidth * 4 - \smoothraggedrightrag-
width) / 4
1079  + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (4)
1080 {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth
1081  + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
1082 \or

      generator=septuplet.
      Permutation 3 – 6 – 1 – 5 – 2 – 7 – 4 looks »random« enough for our purposes.

1083 \typog@fuzzwidth=.08333\smoothraggedrightragwidth
1084 \typog@typeout{smoothraggedright: generator=septuplet, typog@fuzzwidth=\the\typog@fuzzwidth}%
1085 \smoothraggedrightshapeseptuplet[leftskip=\typog@@smoothraggedrightleftskip,
1086                                   parindent=\glueexpr\smoothraggedrightparindent-
indent,

```

```

1087                                     #1]%
1088     {\glueexpr (\typog@@linewidth * 3 - \smoothraggedrightrag-
width * 2) / 3
1089         + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (3)
1090     {\glueexpr (\typog@@linewidth * 6 - \smoothraggedrightrag-
width) / 6
1091         + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (6)
1092     {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth +
1093         + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
1094     {\glueexpr (\typog@@linewidth * 3 - \smoothraggedrightrag-
width) / 3
1095         + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (5)
1096     {\glueexpr (\typog@@linewidth * 6 - \smoothraggedrightrag-
width * 5) / 6
1097         + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (2)
1098     {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth\relax}% (7)
1099     {\glueexpr (\typog@@linewidth * 2 - \smoothraggedrightrag-
width) / 2
1100         + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (4)
1101     \fi}
1102     {\ifcase\typog@@generatorchoice
1103         \endsmoothraggedrightshapetriplet
1104     \or
1105         \endsmoothraggedrightshapequintuplet
1106     \or
1107         \endsmoothraggedrightshapeseptuplet
1108     \fi}
1109
smoothraggedright (env.)

1110 \NewDocumentEnvironment{smoothraggedright}{0{}}
1111 {\PushPostHook{par}{\hskip-\parindent\smoothraggedrightpar[#1]\relax}}
1112 {\par\PopPostHook{par}}
1113

```

B typog-grep

The companion program **typog-grep** for analyzing the output of **typoginspect** and **typoginspectpar** has its own manual page. We reproduce it here for completeness of the documentation.

NAME

typog-grep - grep for typog-inspect elements in L^AT_EX log files

SYNOPSIS

typog-grep -a|--all|--any [OPTION...] LOG-FILE...

typog-grep [OPTION...] REGEXP LOG-FILE...

The first form shows all `<typog-inspect id="ID" ...>` elements in *LOG-FILE*.

The second form shows the contents of `<typog-inspect id="ID" ...>` elements whose *IDs* match *REGEXP* in *LOG-FILE*.

If no *LOG-FILE* is given read from *stdin*. The filename - is synonymous to *stdin*.

DESCRIPTION

typog-grep is a tailored post-processor for L^AT_EX log files and the **typoginspect** environment as provided by package **typog**. It shares more with the venerable **sgrep** than with POSIX **grep**.

The L^AT_EX user brackets her text in

```
\begin{typoginspect}{ID}
  Text and code to investigate
\end{typoginspect}
```

where *ID* is used to identify one or more bracketed snippets. *ID* does not have to be unique. The *REGEXP* mechanism makes it easy to select groups of related *IDs* if they are named accordingly.

In *LOG-FILE* the environment shows up, packed with tracing information, as

```
<typog-inspect id="ID" job="JOB-NAME" line="LINE-NUMBER" page="PAGE-NUMBER">
  Trace Data
</typog-inspect>
```

all the capital-letter sequences are meta-variables and in particular *JOB-NAME* is the expansion of `\jobname`, *LINE-NUMBER* is the L^AT_EX source file line number

of the beginning of the `typoginspect` environment, and *PAGE-NUMBER* is the page where the output of `Text` and `code` to investigate occurs.

typog-grep reveals the contents of *LOG-FILE* between `<typog-inspect id="ID" ...>` and `</typog-inspect>` excluding the XML-tags. Access the *JOB-NAME*, *LINE-NUMBER*, and *PAGE-NUMBER* with the commandline options **--job-name**, **--line-number**, and **--page-number**, respectively. Use **--id** to show the name of the IDs that matched *REGEXP*.

`typoginspect` environments can be nested. **typog-grep** respects the nesting, i.e., if the *ID* of the nested environment does not match *REGEXP* it will not be included in the program's output.

OPTIONS

The list of options is sorted by the names of the long options.

- a, --all, --any**
ID-discovery mode: Show all `typog-inspect` elements independent of any matching patterns.
- color, colour WHEN**
Colorize specific log contents for the matching ids. The argument *WHEN* determines when to apply color: *always*, *never*, or *auto*. The setting *auto* checks whether standard output has been redirected. This is the default.
- C, --config KEY=VALUE[:KEY=VALUE[:...]]**
Set one or more configuration *KEY* to *VALUE* pairs. See Sec. CONFIGURATION below for a description of all available configuration items. Use option **--show-config** to display the default configuration.
- debug**
Turn on debug output on *stderr*.
- h, --help**
Display brief help then exit.
- i, --[no-]id**
Print the actual id name that matched *REGEXP*. Control the appearance of the matching id with configuration item *id-heading*.
- y, --[no-]ignore-case**
Match ids while ignoring case distinctions in patterns and data.
- j, --[no-]job-name**
Print the `\jobname` that `tex` associated with the input file.
- n, --[no-]line-number**
Print the line number where the `typoginspect` environment was encountered in the *L^AT_EX* source file.
- N, --[no-]log-line-number**
Print the line number of the *log*-file where the current line was encountered.
- p, --[no-]page-number**
Print page number where the contents of the `typoginspect` environment starts in the typeset document.

- P, --[no-]pager**
Redirect output from *stdout* to the configured pager.
- show-config**
Show the default configuration and exit.
- V, --version**
Show version information and exit.
- w, --[no-]word-regexp**
Match only whole words.

CONFIGURATION

- id-format=FORMAT**
Control the *FORMAT* for printing matching ids in inline-mode, where *FORMAT* is passed to Perl's `printf`. Default: `%s`.
- id-heading=0|1**
Choose between printing the matching ids with option **--id**: Inline (0) or heading before the matching data (1). Default: 0.
- id-heading-format=FORMAT**
Control the *FORMAT* for printing matching ids in heading-mode, where *FORMAT* is passed to Perl's `printf`. Default: `--> %s <--`.
- id-indent=INDENT**
Indentation of nested typog-inspect tags. Only used in “discovery” mode (first form), i.e., if **--all** is active. Default: 8.
- id-max-length=MAXIMUM-LENGTH**
Set the maximum length of a matching id for printing. If a matching id exceeds this length it will be truncated and the last three characters (short of *MAXIMUM-LENGTH*) will be replaced by dots. Default: 40.
- line-number-format=FORMAT**
Control the *FORMAT* for printing TeX source line numbers, where *FORMAT* is passed to Perl's `printf`. Default: `%5d`.
- log-line-number-format=FORMAT**
Control the *FORMAT* for printing log line numbers, where *FORMAT* is passed to Perl's `printf`. Default: `%6d`.
- page-number-format=FORMAT**
Control the *FORMAT* for printing page numbers, where *FORMAT* is passed to Perl's `printf`. Default: `[%3d]`.
- pager=PAGER**
Name of pager application to pipe output into if run with option **--pager**. Default: `less`.
- pager-flags=FLAGS**
Pass *FLAGS* to *PAGER*. Default: `--quit-if-one-screen`.
- Color Configuration**
For the syntax of the color specifications consult the manual page of `Term::ANSIColor`(pm).

`file-header-color`
 Color of the filename header.
`fill-state-color`
 Color of the messages that report “Underfull hbox” or “Overfull hbox”.
`first-vbox-color`
 Color of the first vbox on a page.
`font-spec-color`
 Color of font specifications.
`horizontal-break-candidate-color`
 Color of lines with horizontal-breakpoint candidates @.
`horizontal-breakpoint-color`
 Color of lines with horizontal breakpoints @@.
`id-color`
 Color of matching ids when printed inline.
`id-heading-color`
 Color of matching ids when printed in heading form.
`line-break-pass-color`
 Color of the lines showing which pass (e.g., @firstpass) of the line-breaking algorithm is active.
`line-number-color`
 Color of TeX-source-file line numbers.
`log-line-number-color`
 Color of log-file line numbers.
`math-color`
 Color used for math expressions including their font specs.
`page-number-color`
 Color of page numbers of the final output.
`tightness-color`
 Color of lines with Tight/Loose hbox reports.
`vertical-breakpoint-color`
 Color of possible vertical breakpoints.

Brief summary of colors and attributes

Foreground Color

black, red, green, yellow, blue, magenta, cyan, white,
 Prefix with `bright_` for high-intensity or bold foreground.

Foreground Grey

`grey0`, ..., `grey23`

Background Color

`on_black`, `on_red`, `on_green`, `on_yellow`, `on_blue`, `on_magenta`, `on_cyan`,
`on_white`
 Replace `on_` with `on_bright_` for high-intensity or bold background.

Background Grey

`on_grey0`, ..., `on_grey23`

Text Attribute

`bold, dark, italic, underline, reverse`

EXIT STATUS

The exit status is 0 if at least one *ID* matched *REGEXP*, 1 if no *ID* matched *REGEXP*, and 2 if an error occurred.

SEE ALSO

`grep(1)`, `printf(3)`, `Term::ANSIColor(pm)`

Change History

v0.1		
	General: Initial version	i
v0.2		
	\narrowospace: New macro.	57
	\widespace: Add fallback if \fontdimen7 is zero. Extend with a starred version.	57

References

- [1] ABRAHAM, PAUL W., HARGREAVES, KATHRYN A., and KARL BERRY. *T_EX for the Impatient*. 2020, <http://tug.ctan.org/info/impatient/book.pdf>.
- [2] AMERICAN MATHEMATICAL SOCIETY and the L^AT_EX3 PROJECT TEAM. *Package amsmath*. 2020, <https://ctan.org/pkg/amsmath>.
- [3] ARSENEAU, DONALD. *Package cite*. 2015, <https://ctan.org/pkg/cite>.
- [4] BEZOS, JAVIER. *Package enumitem*. 2019, <https://ctan.org/pkg/enumitem>.
- [5] BEZOS, JAVIER. *Package babel*. 2021, <https://ctan.org/pkg/babel>. The original author of package babel was J. L. BRAAMS.
- [6] BREITENLOHNER, PETER and the N_TS TEAM. *ε-T_EX*. 1998, https://mirrors.ctan.org/systems/doc/etex/etex_man.pdf.
- [7] CARLISLE, DAVID. *Russian Paragraph Shapes*. Baskerville, 6(1), 13–15, 1996, <http://uk-tug-archive.tug.org/wp-installed-content/uploads/2008/12/61.pdf>.
- [8] CARLISLE, DAVID. *What do different \fontdimen<num> mean*. 2013–1–2, <https://tex.stackexchange.com/questions/88991/what-do-different-fontdimennum-mean>.
- [9] CUBITT, TOBY. *Package cleveref*. 2018, <https://ctan.org/pkg/cleveref>.
- [10] EIJKHOUT, VICTOR. *T_EX By Topic, A Texnician's Reference*. 2007, <https://www.eijkhout.net/tex/tex-by-topic.html>.
- [11] HØGHOLM, MORTEN, MADSEN, LARS and the L^AT_EX3 PROJECT TEAM. *Package mathtools*. 2020, <https://ctan.org/pkg/mathtools>.
- [12] KHIREVICH, SIARHEI. *Tips on Writing a Thesis in L^AT_EX*. 2013, <http://www.khirevich.com/latex/microtype>.
- [13] KNUTH, DONALD ERVIN. *The T_EXbook*. Addison Wesley, Reading/MA, 1986.
- [14] MCPHERSON, KENT. *Package layout*. 2014, <https://ctan.org/pkg/layout>. The package was converted to L^AT_EX 2_ε by J. L. BRAAMS and modified by H. UMEKI.
- [15] MIDDENDORP, JAN. *Shaping Text*. BIS publishers, Amsterdam, 2014.
- [16] MITTELBACH, FRANK. *Managing forlorn paragraph lines (a. k. a. widows and orphans) in L^AT_EX*. TUGboat, 39(3), 246–251, 2018, <https://tug.org/TUGboat/tb39-3/tb123mitt-widows.pdf>.
- [17] MITTELBACH, FRANK. *Package widows-and-orphans*. 2020, <https://ctan.org/pkg/widows-and-orphans>.
- [18] RAHTZ, SEBASTIAN, and FRANK MITTELBACH. *Package hyperref*. 2020, <https://ctan.org/pkg/hyperref>. The package is maintained by the L^AT_EX3 Project Team.
- [19] SCHLICHT, ROBERT. *Package microtype*. 2020, <https://ctan.org/pkg/microtype>.
- [20] SCHRÖDER, MARTIN. *Package ragged2e*. 2019, <https://ctan.org/pkg/ragged2e>.

- [21] SOLOMON, DAVID. *Output Routines: Examples and Techniques. Part I: Introduction and Examples*. TUGboat, 11(1), 69–85, 1990, <http://www.tug.org/TUGboat/Articles/tb11-1/tb27salomon.pdf>.
- [22] TOBIN, GEOFFREY, and ROBIN FAIRBAIRNS. *Package setspace*. 2011, <https://ctan.org/pkg/setspace>.
- [23] UMEKI, HIDEO. *Package geometry*. 2020, <https://ctan.org/pkg/geometry>.
- [24] WERMUTH, UDO. *Tracing paragraphs*. TUGboat, 37(3), 358–373, 2016, <https://tug.org/TUGboat/tb37-3/tb117wermuth.pdf>.
- [25] WERMUTH, UDO. *The optimal value for \emergencystretch*. TUGboat, 38(1), 65–86, 2017, <https://tug.org/TUGboat/tb38-1/tb118wermuth.pdf>.
- [26] WERMUTH, UDO. *A note on \linepenalty*. TUGboat, 38(3), 400–414, 2017, <https://tug.org/TUGboat/tb38-3/tb120wermuth.pdf>.
- [27] WERMUTH, UDO. *Experiments with \parfillskip*. TUGboat, 39(3), 276–303, 2018, <https://tug.org/TUGboat/tb39-3/tb123wermuth-parfillskip.pdf>.
- [28] WERMUTH, UDO. *An attempt at ragged-right typesetting*. TUGboat, 41(1), 73–94, 2020, <https://tug.org/TUGboat/tb41-1/tb127wermuth-ragged.pdf>.
- [29] WERMUTH, UDO. Personal communication. August 2, 2022.
- [30] WERMUTH, UDO. *Vertical alignments in plain T_EX*. TUGboat, 44(3), 427–440, 2023, <https://tug.org/TUGboat/tb44-3/tb138wermuth-valign.pdf>.
- [31] WILSON, PETER. *Package hyphenat*. 2004, <https://ctan.org/pkg/hyphenat>. The package is maintained by W. ROBERTSON.
- [32] WILSON, PETER. *Glisterings*. TUGboat, 28(2), 229–232, 2007, <https://tug.org/TUGboat/tb28-2/tb89glister.pdf>.
- [33] WILSON, PETER. *Package needspace*. 2010, <https://ctan.org/pkg/needspace>. The package is maintained by W. ROBERTSON.

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

In the Index page ranges are stuck together with `\figure-dash*`.

A

`\allowhyphenation`, [9](#), [155](#)
`amsmath` (package), 36

B

`bookmark`, 12
`breakabledisplay` (env.), [36](#), [900](#)
`\breakpoint`, [11](#), [159](#)
`\breakpoint*`, [11](#)

C

`\capitaldash`, [16](#), [286](#), [294](#), [316](#)
`\capitaldash*`, [16](#)
`\capitalemdash`, [16](#), [297](#)
`\capitalemdash*`, [16](#)
`\capitalendash`, [16](#), [281](#)
`\capitalendash*`, [16](#)
`\capitalhyphen`, [16](#), [267](#)
`\capitalhyphen*`, [16](#)
`\capitaltimes`, [18](#), [318](#)
`configuration`, 4
`covernextindentpar` (env.), [24](#), [394](#)
`\cs`, [905](#)
`csquotes` (package), 20

D

`debug` (option), 2
`\Doubleguillemetleft`, [19](#), [352](#)
`\doubleguillemetleft`, [18](#), [336](#)
`\Doubleguillemetright`, [19](#), [356](#)
`\doubleguillemetright`, [18](#), [340](#)

E

`enumitem` (package)
 style `vtietop`, 33
`environments`:
 `breakabledisplay`, [36](#), [900](#)
 `covernextindentpar`, [24](#), [394](#)
 `lastlinecenteredpar`, [20](#), [370](#)
 `lastlineflushrightpar`, [20](#), [367](#)
 `lastlineraggedleftpar`, [20](#), [361](#)
 `loosespacing`, [25](#), [456](#)
 `nocharprotrusion`, [31](#), [625](#)
 `nofontexpand`, [622](#)
 `nofontexpansion`, [30](#), [616](#)

`openlastlinepar`, [24](#), [408](#)
`prolongpar`, [24](#), [384](#)
`setfontexpand`, [30](#), [602](#)
`setfontshrink`, [29](#), [574](#)
`setfontstretch`, [29](#), [588](#)
`setfonttracking`, [28](#), [488](#)
`shortenpar`, [24](#), [377](#)
`slightlysloppypar`, [31](#), [696](#)
`smoothraggedright`, [39](#), [1110](#)
`smoothraggedrightpar`, [38](#), [1027](#)
`smoothraggedrightshape-`
 `quintuplet`, [38](#), [945](#)
`smoothraggedrightshape-`
 `septuplet`, [38](#), [981](#)
`smoothraggedrightshape-`
 `triplet`, [38](#), [912](#)
`tightspacing`, [25](#), [472](#)
`typoginspect`, [5](#), [121](#)
`typoginspectpar`, [5](#), [143](#)
`typogsetup`, [4](#), [82](#)
`vtiebotdisp`, [34](#), [885](#)
`vtiebotdisptoppar`, [34](#), [889](#)
`vtiebotpar`, [34](#), [827](#)
`vtietoppar`, [33](#), [760](#)

F

`\figuredash`, [17](#), [311](#)
`\figuredash*`, [17](#)
`font`, 17
 ADF Baskervald, 19
 ADF Venturis, 17
 Alegreya, 17
 Arvo, 17
 Bitter, 17
 Clara, 17
 Coelacanth, 17
 Crimson Pro, 17
 Crimson Text, 17
 Droid Serif, 17
 EB Garamond, 17, 19
 encoding, 18
 Erewhon, 17
 expansion, 23, 29
 fbb, 17
 Gentium, 17, 19

GFS Artemisia, 17, 19
 GFS Didot, 19
 Ibarra Real Nova, 17
 information, 5
 INRIA Serif, 17
 Libertine, 17
 Libertinus, 17, 19
 Libre Baskerville, 17
 Libre Caslon, 17
 Merriweather, 17, 19
 protrusion, 31
 PT Serif, 17
 Roboto Slab, 17
 size, 5
 Source Serif Pro, 17
 spacing, 23, 25
 loose, 25
 tight, 25
 Spectral, 17
 STIX, 17
 T_EX Gyre Pagella, 17
 tracking, 23, 28
 TX Fonts, 17
 URW Palladio L, iii
`\fontsizeinfo`, 5, [101](#)
 forlorn line
 club, 24, 33
 display widow, 34
 orphan, 24
 widow, 24, 34

H

hyperref (package), 12
 hyphenation, 9
 empty discretionary, 11
 re-enable automatic, 9

I

information, 5
`\int`, [907](#)
`\itcorr`, 9, [13](#), [206](#)
`\itcorr*`, [13](#)

K

`\kernedhyphen`, [15](#), [229](#), [251](#), [252](#),
 [259](#), [260](#)
`\kernedhyphen*`, [15](#)
`\kernedslash`, [14](#), [99](#), [219](#)
`\kernedslash*`, [14](#)
 kerning
 extra, 14
 forward slash, 14

hyphen, 15
 ligature, 12

L

`lastlinecenteredpar` (env.), [20](#), [370](#)
`lastlineflushrightpar` (env.),
 [20](#), [367](#)
`lastlinerraggedleftpar` (env.),
 [20](#), [361](#)
`\leftkernedhyphen`, [15](#), [249](#), [254](#)
`\leftkernedhyphen*`, [15](#)
 ligature, 12
 ligaturekern (option), 2
 line spacing, 5
 list, 33
`loosespacing` (env.), [25](#), [456](#)

M

`mathitalicscorrection` (op-
 tion), 2
 microtype (package), 28

N

narrow space, 27
`\narrow space`, [27](#), [442](#)
`\narrow space*`, [27](#)
`\narrow spacescale`, [441](#), [444](#),
 [446](#), [452](#), [453](#)
`\narrow space strength`, [440](#), [449](#)
`nocharprotrusion` (env.), [31](#), [625](#)
 nodebug (option), 2
`nofontexpand` (env.), [622](#)
`nofontexpansion` (env.), [30](#), [616](#)
`\nolig`, [12](#), [171](#)
`\nolig*`, [12](#)

O

`openlastlinepar` (env.), [24](#), [408](#)

P

package option
 debug, 2
 ligaturekern, 2
 mathitalicscorrection, 2
 nodebug, 2
 penalty, 2
 raise*, 2
 raisecapitaldash, 2
 raisecapitalguillemets, 3
 raisecapitalhyphen,
 2

raisecapitaltimes, 2
 raisefiguredash, 3
 raiseguillemets, 3
 shrinklimits, 3
 slashkern, 3
 stretchlimits, 3
 textitalicscorrection, 3
 trackingttspacing, 3
 package options, 2–3
 page break, 8, 36
 paragraph
 align last line, 20
 centered, 20
 flush right, 20
 badness, 7
 fill last line
 openlastlinepar, 24
 prolongpar, 24
 shortenpar, 24
 fill last line, 21
 covernextindentpar, 24
 \linebreak, 23
 \mbox, 23
 tie, 22
 sloppy, 31
 vertically tied, 33
 PDF, 12
 penalty (option), 2
 \prg, 905
 prolongpar (env.), 24, 384
R
 ragged right, 38
 raise* (option), 2
 raisecapitaldash (option), 2
 raisecapitalguillemets (option), 3
 raisecapitalhyphen (option), 2
 raisecapitaltimes (option), 2
 raised character, 15
 en-dash, 16
 guillemets, 18
 hyphen, 16
 multiplication sign, 18
 number dash, 17
 raisefiguredash (option), 3
 raiseguillemets (option), 3
 reconfigure, 4
 \rightkernedhyphen, 15, 257, 262
 \rightkernedhyphen*, 15

S

setfontexpand (env.), 30, 602
 setfontshrink (env.), 29, 574
 setfontstretch (env.), 29, 588
 setfonttracking (env.), 28, 488
 setup, 4
 shortenpar (env.), 24, 377
 shrinklimits (option), 3
 \Singleguillemetleft, 19, 344
 \singleguillemetleft, 18, 328
 \Singleguillemetright, 19, 348
 \singleguillemetright, 18, 332
 slashkern (option), 3
 \slightlyloppy, 31, 639, 697
 slightlyloppypar (env.), 31, 696
 smoothraggedright (env.), 39, 1110
 \smoothraggedrightfuzzfactor, 1020, 1055
 \smoothraggedrightgenerator, 1021, 1036, 1039, 1042
 \smoothraggedrightleftskip, 1022, 1051
 smoothraggedrightpar (env.), 38, 1027
 \smoothraggedrightparindent, 1023, 1060, 1071, 1086
 \smoothraggedrightragwidth, 1024, 1055, 1057, 1062, 1065, 1068, 1073, 1076, 1078, 1080, 1083, 1088, 1090, 1092, 1094, 1096, 1099
 smoothraggedrightshape-quintuplet (env.), 38, 945
 smoothraggedrightshape-septuplet (env.), 38, 981
 smoothraggedrightshape-triplet (env.), 38, 912
 \splicevtietop, 33, 765
 stretchlimits (option), 3

T

textitalicscorrection (option), 3
 tightspacing (env.), 25, 472
 trackingttspacing (option), 3
 \typogget, 4, 91
 typoginspect (env.), 5, 121
 typoginspectpar (env.), 5, 143
 \typoglogo, 9
 typogsetup (env.), 4, 82

V

`\vtiebot`, [34](#), [774](#), [828](#)
`\vtiebotdisp`, [895](#)
`vtiebotdisp (env.)`, [34](#), [885](#)
`vtiebotdisptoppar (env.)`, [34](#), [889](#)
`vtiebotpar (env.)`, [34](#), [827](#)
`\vtietop`, [33](#), [707](#), [761](#), [767](#), [893](#)
`vtietoppar (env.)`, [33](#), [760](#)

W

wide space, [26](#)
`\widespace`, [26](#), [426](#)
`\widespace*`, [26](#)
`\widespacescale`, [425](#), [428](#), [430](#),
[436](#), [437](#)
`\widespacestrength`, [424](#), [433](#)