

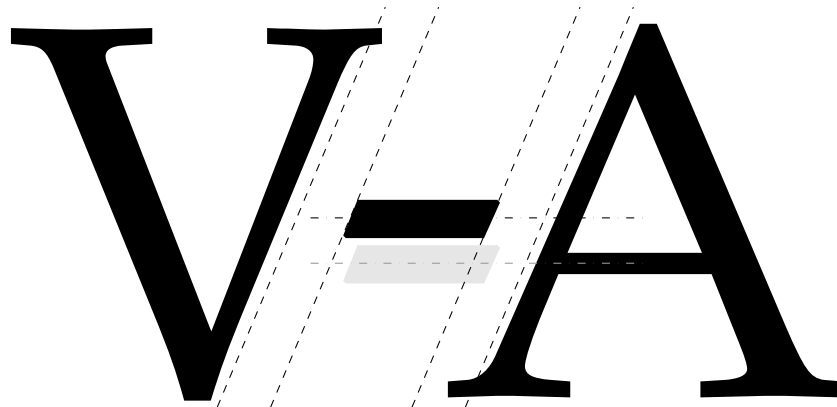
# TypoG – Typographic Fine-Tuning

Ch. L. Spiel\*

v0.5      2024/09/26

## 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 high-level front-ends for packages microtype and setspace.



This package is copyright © 2024 Ch. L. Spiel. It may be distributed and/or modified under the conditions of the [L<sup>A</sup>T<sub>E</sub>X 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](mailto:cspiel@users.sourceforge.org)

## Contents

Hoffentlich wird es nicht so schlimm, wie es schon ist!  
— KARL VALENTIN

### Quick Reference [v](#)

# 1

## Introduction [1](#)

- 1.1 Overview [1](#)
- 1.2 Prerequisites [1](#)

# 2

## Package Options [2](#)

# 3

## Macros and Environments [5](#)

- 3.1 Setup and Reconfiguration [5](#)
- 3.2 Information [6](#)
  - 3.2.1 Font Information [6](#)
  - 3.2.2 Paragraph- and Page-Breaking Trace [7](#)
- 3.3 Hyphenation [10](#)
- 3.4 Disable/Break Ligatures [14](#)
- 3.5 Manual Italic Correction [15](#)
- 3.6 Apply Extra Kerning [16](#)
  - 3.6.1 Slash [16](#)
  - 3.6.2 Hyphen [17](#)
- 3.7 Raise Selected Characters [18](#)
  - 3.7.1 Capital Hyphen [18](#)
  - 3.7.2 Capital Dash [19](#)
  - 3.7.3 Number Dash (Figure Dash) [19](#)
  - 3.7.4 Multiplication Sign [20](#)
  - 3.7.5 Guillemets [20](#)
  - 3.7.6 Inverted Marks [22](#)
- 3.8 Vertically Adjust Label Items [24](#)
- 3.9 Align Last Line [27](#)

*Table of Contents continued on next page.*

3.10	Fill Last Line	29
3.10.1	Problem Definition	29
3.10.2	Manual Changes	30
3.10.3	Multi-Purpose Environments	31
3.10.4	Specialized Environments	32
3.10.5	Consistent Spacing of Last Line	33
3.11	Spacing	33
3.11.1	Looser/Tighter	33
3.11.2	Wide Space	35
3.11.3	Narrow Space	36
3.12	Microtype Front-End	36
3.12.1	Tracking	37
3.12.2	Font Expansion	38
3.12.3	Character Protrusion	39
3.13	Sloppy Paragraphs	40
3.14	Vertically Partially-Tied Paragraphs	41
3.15	Breakable Displayed Equations	44
3.16	Setspace Front-End	46
3.17	Smooth Ragged	49

## 4

Limitations and Known Problems	53
--------------------------------	----

## 5

Other Packages for Fine L <sup>A</sup> T <sub>E</sub> X Typography	54
--	----

## A

typog-grep	55
------------	----

## B

ε-T <sub>E</sub> X: Breaking Paragraphs into Lines	62
--	----

*Table of Contents continued on next page.*

## C

**Package Code 64**

C.1	Setup and Reconfiguration	69	C.10	Fill Last Line	87
C.2	Information	70	C.11	Spacing	89
C.3	Hyphenation	72	C.12	Microtype Front-End	90
C.4	Disable/Break Ligatures	73	C.13	Sloppy Paragraphs	94
C.5	Manual Italic Correction	73	C.14	Vert. Tie Paragraphs	96
C.6	Apply Extra Kerning	74	C.15	Breakable Disp. Eqs.	100
C.7	Raise Selected Characters	76	C.16	Setspace Front-End	101
C.8	Vert. Adjust Label Items	80	C.17	Smooth Ragged	104
C.9	Align Last Line	87			

**Change History 111****References 112****Index 115****List of Tables**

1	Hyphens and automatic hyphenation	12
2	Suggested raise amounts for <code>\figuredash</code>	20
3	Suggested raise amounts for guillemets	21
4	Label item adjustment suggestions	28
5	Spacing changes made by <code>loosespacing</code>	34
6	Spacing changes made by <code>tightspacing</code>	34
7	<code>\fontdimen&lt;number&gt;</code> parameters	35
8	Comparison of some space sizes	37
9	Shrink values of <code>setfontshrink</code>	38
10	Stretch values of <code>setfontstretch</code>	38
11	Shrink and stretch values of <code>setfontexpand</code>	39
12	Parameter adjustments of <code>\slightlyloppy</code>	40
13	Partial paragraph line counts	44
14	Env. <code>breakabledisplay</code> and <code>\interdisplaylinepenalty</code>	45

## Quick Reference

We reduce the line spacing in the multi-column parts to 120% with `\setbaselineskippercentage`. List items get tied with `\vtietop`.

This is an alphabetically sorted list of all user macros and environments defined by package `typog` along with the page numbers of their respective descriptions. A list of all package options can be found on page 2 and 4. The [Index](#) on pages 115 to 119 may provide some more detailed insights.

If a line-break occurs between the command and its arguments or between any of the arguments we indicate the break with triangle at the end of the initial line and at the beginning of the following line.

<b>A</b>		
<code>\allowhyphenation</code>	(Re-)enable automatic hyphenation.	10
<b>B</b>		
<code>breakabledisplay</code>	[ <i>&lt;level&gt;</i> ] Adjust penalty associated with <code>\allowdisplaybreaks</code> .	44
<code>\breakpoint*</code>	Insert an empty discretionary.	13
<code>\breakpoint</code>	Insert an empty discretionary and re-enable automatic hyphenation.	13
<b>C</b>		
<code>\capitaldash*</code>	Typeset a vertically adjusted <code>\textendash</code> . Alias for <code>\capitalendash*</code> .	19
<code>\capitaldash</code>	Typeset a vertically adjusted <code>\textendash</code> that is hyphenatable. Alias for <code>\capitalendash</code> .	19
<code>\capitalemdash*</code>	Typeset a vertically adjusted <code>\textemdash</code> .	19
<code>\capitalemdash</code>	Typeset a vertically adjusted <code>\textemdash</code> that is hyphenatable.	19
<code>\capitalendash*</code>	Typeset a vertically adjusted <code>\textendash</code> .	19
<code>\capitalendash</code>	Typeset a vertically adjusted <code>\textendash</code> that is hyphenatable.	19
<code>\capitalhyphen*</code>	Typeset a vertically adjusted hyphen character.	18
<code>\capitalhyphen</code>	Typeset a vertically adjusted hyphen character that is hyphenatable.	18
<code>\capitalinvertedexclamationmark</code>	<{ <i>&lt;number&gt;</i> } Typeset an inverted (>upside-down<) exclamation mark that is level with the baseline.	22
<code>\capitalinvertedquestionmark</code>	<{ <i>&lt;number&gt;</i> } Typeset an inverted (>upside-down<) question mark that is level with the baseline.	22
<code>\capitaltimes</code>	Typeset a vertically adjusted <code>\texttimes</code> .	20
<code>covernextindentpar</code>	[ <i>&lt;dim&gt;</i> ] Extend the last line of a paragraph.	32
<b>D</b>		
<code>\Doubleguillemetleft</code>	Typeset left double guillemets vertically adjusted for uppercase.	21
<code>\Doubleguillemetright</code>	Typeset right double guillemets vertically adjusted for uppercase.	21
<code>\doubleguillemetleft</code>	Typeset left double guillemets vertically adjusted for lowercase.	21
<code>\doubleguillemetright</code>	Typeset right double guillemets vertically adjusted for lowercase.	21

## F

`\figuredash*`

Typeset a `\textendash` vertically adjusted for figures (numerals). 19

`\figuredash`

Typeset a hyphenatable `\textendash` vertically adjusted for figures (numerals). 19

`\fontsizeinfo{⟨cs-name⟩}`

Store the current em-height and `\base-lineskip` in a pair of macros. 6

## H

`hyphenmin[⟨left-min⟩][⟨right-min⟩]`

Set the values of `\lefthyphenmin` and `\righthyphenmin`. 13

## I

`\itcorr*{⟨strength⟩}`

Apply italic correction in the for of a `\kern` scaled by `textitalicscorrection`. 15

`\itcorr{⟨strength⟩}`

Apply italic correction in the for of a `\kern` scaled by `\fontdim1` or `textitalicscorrection`. 15

## K

`\kernedhyphen*[⟨raise⟩]▷`

◁{⟨left-kern⟩}{⟨right-kern⟩}  
Typeset an unbreakable hyphen and apply kerning to its left and right. 17

`\kernedhyphen[⟨raise⟩]▷`

◁{⟨left-kern⟩}{⟨right-kern⟩}  
Typeset a breakable hyphen and apply kerning to its left and right. 17

`\kernedslash*`

Typeset an unbreakable forward slash and apply kerning to its left and right. 16

`\kernedslash`

Typeset an breakable forward slash and apply kerning to its left and right. 16

## L

`lastlinecenteredpar`

Center the last lines of a paragraph. 27

`lastlinefitpar`

Match spacing of last line and next-to-last line. 33

`lastlineflushrightpar`

Align the last line of a paragraph flush-right. Alias for `lastlinerraggedleftpar`. 27

`lastlinerraggedleftpar`

Align the last line of paragraph flush-right. 27

`\leftkernedhyphen*[⟨raise⟩][⟨left-kern⟩]`

Typeset a hyphen and apply kerning to its left-hand side. 17

`\leftkernedhyphen[⟨raise⟩][⟨left-kern⟩]`

Typeset a hyphen, apply kerning to its left-hand side, and insert a breakpoint after it. 17

`loosespacing[⟨level⟩]`

Increase the width of the space character. 34

`\lowercaseadjustlabelitems{⟨levels⟩}`

Activate the lowercase height-adjustment values inside `itemize` environments. 24

## N

`\narrowspace*`

Typeset a narrow space whose width depends on `\fontdimen7`. 36

`\narrowspace`

Typeset a narrow space whose width depends on `\fontdimen7` or `\fontdimen2`. 36

`\noadjustlabelitems{⟨levels⟩}`

Deactivate height-adjustment of label items. 24

`nocharprotrusion`

Deactivate character protrusion. 39

`nofontexpand`

Deactivate font expansion. Alias for `nofontexpansion`. 39

`nofontexpansion`

Deactivate font expansion. 39

`\nolig*[⟨kerning⟩]`

Break a ligature. 14

<code>\nolig</code> [ <i>&lt;kerling&gt;</i> ]	Break a ligature and introduce a hyphenation opportunity.	14	<code>setfontshrink</code> [ <i>&lt;level&gt;</i> ]	Set font shrink limits.	38
<b>O</b>			<code>setfontstretch</code> [ <i>&lt;level&gt;</i> ]	Set font stretch limits.	38
<code>openlastlinepar</code> [ <i>&lt;dim&gt;</i> ]	Open a paragraph's last line if it is almost full or completely filled.	32	<code>setfonttracking</code> { <i>&lt;delta&gt;</i> }	Override the default tracking for all fonts.	37
<b>P</b>			<code>\setleadingpercentage</code> { <i>&lt;percentage&gt;</i> }	Set <code>\baselineskip</code> as percentage via the leading.	47
<code>prolongpar</code>	Increase the <code>\looseness</code> of a paragraph.	32	<code>\setleading</code> { <i>&lt;leading&gt;</i> }	Set <code>\baselineskip</code> via the leading.	47
<b>R</b>			<code>shortenpar</code>	Decrease the <code>\looseness</code> of a paragraph.	32
<code>\resetbaselineskip</code>	Reset the <code>\baselineskip</code> to its original value.	46	<code>\Singleguillemetleft</code>	Typeset left single guillemets vertically adjusted for uppercase.	21
<code>\rightkernedhyphen*</code> [ <i>&lt;raise&gt;</i> ]{ <i>&lt;right-kern&gt;</i> }	Typeset a hyphen and apply kerning to its right-hand side.	17	<code>\Singleguillemetright</code>	Typeset right single guillemets vertically adjusted for uppercase.	21
<code>\rightkernedhyphen</code> [ <i>&lt;raise&gt;</i> ]{ <i>&lt;right-kern&gt;</i> }	Typeset a hyphen, apply kerning to its right-hand side, and insert a breakpoint after it.	17	<code>\singleguillemetleft</code>	Typeset left single guillemets vertically adjusted for lowercase.	21
<b>S</b>			<code>\singleguillemetright</code>	Typeset right single guillemets vertically adjusted for lowercase.	21
<code>\setbaselineskip-</code> <code>percentage</code> { <i>&lt;percentage&gt;</i> }	Set <code>\baselineskip</code> as percentage relative to font design size.	47	<code>slightlyloppypar</code> [ <i>&lt;sloppiness&gt;</i> ]	Format a paragraph with given sloppiness.	40
<code>\setbaselineskip</code> { <i>&lt;baselineskip&gt;</i> }	Set <code>\baselineskip</code> using an absolute length.	46	<code>\slightlyloppy</code> [ <i>&lt;sloppiness&gt;</i> ]	Format with given sloppiness.	40
<code>setfontexpand</code> [ <i>&lt;level&gt;</i> ]	Simultaneously set font stretch and shrink limits.	38			
<hr/>					
<code>smoothraggedrightpar</code> [ <i>&lt;option&gt;</i> ...]	Format a paragraph with one of the three smooth-ragged-right generators.	50			
<code>smoothraggedrightshapequintuplet</code> [ <i>&lt;option&gt;</i> ...]{ <i>&lt;width1&gt;</i> }...{ <i>&lt;width5&gt;</i> }	Prescribe five line lengths for formatting paragraphs.	49			
<code>smoothraggedrightshapeseptuplet</code> [ <i>&lt;option&gt;</i> ...]{ <i>&lt;width1&gt;</i> }...{ <i>&lt;width7&gt;</i> }	Prescribe seven line lengths for formatting paragraphs.	49			

<code>smoothraggedrightshapetriplet</code> $[\langle option \rangle \dots] \{ \langle width1 \rangle \} \{ \langle width2 \rangle \} \{ \langle width3 \rangle \}$	Prescribe three line lengths for formatting paragraphs.	49
<code>smoothraggedright</code> $[\langle option \rangle \dots]$	Format with one of the three smooth-ragged-right generators.	51
<hr/>		
<code>\splicevtietop</code> $\{ \langle lines \rangle \}$	Inside of a <code>list</code> -like environment fuse the first lines.	41
<b>T</b>		
<code>tightspacing</code> $[\langle level \rangle]$	Decrease the width of the space character.	34
<code>\typogadjuststairs</code> $[\langle factor \rangle] \triangleright \triangleleft \{ \langle step \rangle \} \{ \langle count \rangle \} \{ \langle sample \rangle \}$	Generate $\triangleright$ stairs $\triangleleft$ of vertically shifted label items.	25
<code>\typogfontsize</code>	Default font's quad size.	47
<code>\typoggetnth</code> $\{ \langle dest \rangle \} \{ \langle key \rangle \} \{ \langle index \rangle \}$	Retrieve single item of a typog compound configuration value.	6
<code>\typogget</code> $\{ \langle key \rangle \}$	Retrieve a typog configuration value.	5
<code>typoginspectpar</code> $[\langle option \rangle] \{ \langle id \rangle \}$	Turn on tracing of paragraphs and pages for a paragraph.	7
<code>typoginspect</code> $[\langle option \rangle] \{ \langle id \rangle \}$	Turn on tracing of paragraphs and pages.	7
<code>\typoglowercaseadjust-check</code> $[\langle factor \rangle] \{ \langle sample \rangle \}$	Typeset all four label items adjusted for lowercase with an indicator line.	26
<code>typogsetup</code> $\{ \langle keys \rangle \}$	Configure package typog.	5
<code>\typoguppercaseadjust-check</code> $[\langle factor \rangle] \{ \langle sample \rangle \}$	Typeset all four label items adjusted for uppercase with an indicator line.	26
<b>U</b>		
<code>\uppercaseadjustlabelitems</code> $\{ \langle levels \rangle \}$	Activate the uppercase height-adjustment values inside <code>itemize</code> environments.	24
<b>V</b>		
<code>vtiebotdisptoppar</code> $[\langle num-before-lines \rangle] \triangleright \triangleleft [\langle num-after-lines \rangle]$	Fuse a display with its preceding and following lines.	42
<code>vtiebotdisp</code> $[\langle num-lines \rangle]$	Fuse a display with its preceding lines.	42
<code>vtiebotpar</code> $[\langle num-lines \rangle]$	Fuse the last lines of a paragraph.	42
<code>\vtiebot</code> $[\langle num-lines \rangle]$	Fuse last lines.	42
<code>vtietoppar</code> $[\langle num-lines \rangle]$	Fuse the first lines of a paragraph.	41
<code>vtietop</code> $[\langle num-lines \rangle]$	Fuse first lines.	41
<code>\vtietop</code> $[\langle num-lines \rangle]$	Fuse first lines.	41
<b>W</b>		
<code>\widespace*</code>	Typeset a wide space whose width depends on <code>\fontdimen7</code> .	35
<code>\widespace</code>	Typeset a wide space whose width depends on <code>\fontdimen7</code> or <code>\fontdimen2</code> .	35



# 1 Introduction

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

L<sup>A</sup>T<sub>E</sub>X is the beginning 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.

Section 3.2 tends to the wish for more information in the typesetting process whether during the draft phase or in the final printed manuscript.

Section 3.3 expands the hyphenation facilities of L<sup>A</sup>T<sub>E</sub>X.

Sections 3.4 to 3.7 deal with vertically positioning glyphs in a more pleasant way. Also in the realm of vertical alignments is Sec. 3.8 that explains how to height-adjust the labels in `itemize` lists to perfection whether the items are followed by uppercase or by lowercase letters.

Sections 3.9 and 3.10 discuss dearly missed macros for better control of the last line of a paragraph.

Section 3.11 covers the manipulation of the length of a paragraph.

Section 3.12 expounds on the microtype front-end: font tracking (3.12.1), font expansion (3.12.2), and character protrusion (3.12.3).

In Sec. 3.13 we address some shortcomings of spacing control with a replacement for the macro `\sloppy` and the related environment `sloppypar`.

Section 3.14 presents several 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 Sec. 3.15.

Section 3.16 introduces the `setspace` front-end.

In the last part, Sec. 3.17, we introduce a novel way of generating ragged paragraphs, which still is experimental.

## 1.2 Prerequisites

Package `typog` requires  $\epsilon$ -T<sub>E</sub>X; it relies on the L<sup>A</sup>T<sub>E</sub>X3 interface. Parts of it are based on package `microtype`. However, if the respective functionality is not used, `typog` can be used without `microtype`. The same holds true for the `setspace` front-end.

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

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 Sec. 3.12.

\usepackage[...]{setspace} % Only required for macros in Sec. 3.16.

\usepackage[⟨OPTION⟩...]{typog}
```

The package `⟨OPTIONS⟩` also serve as configuration `⟨key⟩`s (unless noted otherwise). 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 some package-specific debug information to the *log* file. Opposite: `nodebug`. The default is not to record debug information.

These two options neither can be used with `typogsetup` nor with `\typogget`.

`ligaturekern=⟨dim⟩`

Set `⟨dim⟩` of the kern that is inserted to split a ligature in macro `\nolig`. See Sec. 3.4. Default value:  $\frac{33}{1000}$  em.

`lowercaselabelitemadjustments={⟨dim-1⟩, ..., ⟨dim-4⟩}`

Vertical shifts `⟨dim-N⟩` to apply to `\labelitem⟨N⟩`, where `⟨N⟩`=1, 2, 3, or 4 is the nesting level of the `itemize` list. Empty list elements are ignored. The special value `*` instructs `typog` to preserve `⟨dim-N⟩` at that position. The adjustments apply to the lowercase setting (`\lowercaseadjustlabelitems`). See Sec. 3.8 (in particular subsection `»Setup«` and Tab. 4 on p. 28) and also configuration option `uppercaseadjustlabelitem`.

All four lengths default to 0 pt.

### Important

Configuring `lowercaselabelitemadjustments` (or `uppercaseadjustlabelitemadjustments`) does *not* activate the correction mechanism. Use one of the macros `\lowercaseadjustlabelitems` or `\uppercaseadjustlabelitems` for that purpose. ■

`lowerslash=⟨dim⟩`

Lower the slash typeset by `\kernedslash`. Positive lengths `⟨dim⟩` lower the glyph, negative ones raise it. This is the opposite `»direction«` of `\raisebox`. See Sec. 3.6.1. Default value: 0 pt.

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 all the (default) configuration values with `\typogget`.

SINCE V0.4

SINCE V0.5

`mathitalicscorrection=<dim>`

Italics correction in math mode. See Sec. 3.5 and also the complementary configuration option `textitalicscorrection`. Default value: 0.4mu.<sup>1</sup>

`raise*=<dim>`

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

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

This option neither can be used with `typogsetup` nor with `\typogget`, however, the specific options influenced by it can.

`raisecapitaldash=<dim>`

Set the length that the `\textendash` is raised in `\capitaldash`. See Sec. 3.7.2. Default value: 0 pt.

`raisecapitalhyphen=<dim>`

Set the length that the hyphen character `⁂` is raised in `\capitalhyphen`. See Sec. 3.7.1. Default value: 0 pt.

`raisecapitaltimes=<dim>`

Set the length that the multiplication symbol `×` is raised in `\capital-times`. See Sec. 3.7.4. Default value: 0 pt.

`raisecapitalguillemets=<dim>`

Set the length that single and double guillemets are raised in the uppercase versions of the guillemet macros. See Sec. 3.7.5. Default value: 0 pt.

`raiseguillemets=<dim>`

Set the length that single and double guillemets are raised in the lowercase versions of the guillemet macros. See Sec. 3.7.5. Default value: 0 pt.

`raisefiguredash=<dim>`

Set the length that the `\textendash` is raised in `\figuredash`. See Sec. 3.7.3. Default value: 0 pt.

`raiseinvertedmarks={<dim-1>, <dim-2>, <dim-3>}`

SINCE V0.5

Set the lengths by which the macros `\capitalinvertedexclamationmark` and `\capitalinvertedquestionmark` raise their associated inverted exclamation marks and inverted question marks. Each dimension corresponds to the respective optional indices of the macros. See Sec. 3.7.6.

A `<dim-N>` of 0 pt means to »auto level« the mark; if a (quasi-)zero manual correction is desired use e. g. 1 sp. Empty list elements are ignored. The special value `⁂` instructs typog to preserve `<dim-N>` at that position. Default values: 0 pt, 0 pt, 0 pt.

<sup>1</sup> Note that 1 mu is 1/18 em of the mathematical font's em.

`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. 3.12.2.

New `⟨limit-#⟩` values replace old ones. If one or more limits of the triple should remain unchanged pass a  $\text{⌘}$  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. 3.6.1. Default value:  $\frac{50}{1000}$  em.

`textitalicscorrection=⟨dim⟩`

Italics correction fallback-value; used if `\fontdimen1` is zero. See Sec. 3.5 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. 3.12.1.

The argument `⟨outer-spacing⟩` gets passed to microtype's `\SetTracking` option `outer spacing` [22, 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.

`uppercaselabelitemadjustments={⟨dim-1⟩, ..., ⟨dim-4⟩}`

SINCE V0.4

Vertical shifts `⟨dim-N⟩` to apply to `\labelitem⟨N⟩`, where `⟨N⟩`=1, 2, 3, or 4 is the nesting level of the `itemize` list. Empty list elements are ignored. The special value  $\text{⌘}$  instructs typog to preserve `⟨dim-N⟩` at that position. The adjustments apply to the uppercase setting (`\uppercaseadjustlabelitems`). See Sec. 3.8 (in particular subsection `›Setup‹` and Tab. 4 on p. 28) and also configuration option `lowercaseadjustlabelitem`.

All four lengths default to 0 pt.

### Important

Configuring `uppercaselabelitemadjustments` (or `lowercase-labelitemadjustments`) does *not* activate the correction mechanism. Use one of the macros `\uppercaseadjustlabelitems` or `\lowercaseadjustlabelitems` for that purpose. ■

## 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`.

### 3.1 Setup and Reconfiguration

`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.

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

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.

#### Note

Use `\PassOptionsToPackage{\langle keys \rangle}{typog}` to pass  $\langle keys \rangle$  to `typog` before loading it and `\typogsetup{\langle keys \rangle}` after `\usepackage{typog}`. ■

#### 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 configuration values of package `typog`. This can be done in a safe way without resorting to code that is bracketed by `\makeatletter` and `\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. Compare with the solution in Sec. 3.8 offered by `typog` since v0.4. ■

`\typoggetnth`  
SINCE V0.4

If a configuration item is associated with a list as `lowercaselabelitemadjustments`, `shrinklimits`, `stretchlimits`, `trackingttspacing`, and `uppercaselabelitemadjustments` are, it may be convenient to fetch a particular list element of it.

```

\typoggetnth{<csname>}{<key>}{<index>}
\typoggetnth{<dimen-register>}{<key>}{<index>}

```

Retrieve the configuration value – which is a comma-separated list – that is associated with `<key>` and store the item having position `<index>` in `<dimen-register>` or the parameter-less, global macro `<csname>`. The destination `<dimen-register>` may be predefined like, e. g., `\dimen0` or user-defined. Dimensions can also be stored in a macro by using the `<csname>` form of `\typoggetnth` but not *vice versa*.

Index into the list either from left-to-right with positive indices starting at 1 up to the length of the list, or from right-to-left with negative indices starting at -1 down to the negative length.

#### Note

Macro `\typoggetnth` *only* works with `<key>`s that are associated with a list of values. ■

## 3.2 Information

Never forget: The visual output counts; it must always be checked, [...].  
— UDO WERMUTH [29]

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

We define some functions for introspection of the typesetting process.

### 3.2.1 Font Information

`\fontsizeinfo` Capture the font size<sup>2</sup> and line spacing<sup>3</sup> 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

<sup>2</sup> We use `\fontdimen6`, the em-height as the font size.

<sup>3</sup> The line spacing simply is `\baselineskip`.

units (i. e., pt), the starred version `\cs-name*` omits the units. The separating slash is `\kernedslash`, which is introduced in Sec. 3.6.1.

#### Note

The `\baselineskip` can contain a rubber (stretch/shrink) component, however, `\fontsizeinfo` will not display these parts. ■

#### Use Cases

Colophon. ¶ Font test pages. ■

### 3.2.2 Paragraph- and Page-Breaking Trace

`typoginspect` (*env.*)  
`typoginspectpar` (*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.

```
\begin{typoginspect}[\langle option \rangle]{\langle id \rangle} ... \end{typoginspect}
\begin{typoginspectpar}[\langle option \rangle]{\langle id \rangle}
...
\end{typoginspectpar}
```

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

#### Option

##### **tracingboxes**[=*size*]

Specify the maximum box breadth and box depth reported in the log. If `\langle size \rangle` 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. L<sup>A</sup>T<sub>E</sub>X reliably logs the requested trace information, but the write operations for trace data and `\immediate\write` which is used to print the end-tag are not synchronized. The workaround in such a situation is to enclose more text in the `typoginspect` environment (respecting the nesting of other environments of course). ■

**L<sup>A</sup>T<sub>E</sub>X 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  $\langle id \rangle$  is the user-supplied, unique<sup>4</sup> identifier of the group,  $\langle jobname \rangle$  is the value of `\jobname`,  $\langle line-number \rangle$  records the `\inputlineno` of the `\begin` of the group, and  $\langle page-number \rangle$  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 <regex>`) 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 A.

### Tips

- It may be necessary to run whatever L<sup>A</sup>T<sub>E</sub>X 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<sup>A</sup>T<sub>E</sub>X.

As both **pdf<sub>l</sub>atex** and **lua<sub>l</sub>atex** 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}`.

<sup>4</sup> It has turned out advantageous to use unique  $\langle id \rangle$ s. However,  $\langle id \rangle$ s are *not required* to be distinct.



- As the overhead of `\typoginspect` is relatively low, hairy parts of a document can permanently be furnished with them, for example, the Index.
- Any labeled part can treat their ids to `\id`. Think of `\captions` or any theorem-like environment and their associated, unique `\labels`. ■

**Investigating the badness of a paragraph.** It is generally unnecessary to determine the *exact* classification of a paragraph's badness [15, p. 97n], though the curious user can switch on logging of T<sub>E</sub>X's line-break information with `\tracingparagraphs=1`<sup>5</sup> 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 [15, 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`  $\neq$  0:
  - (a) The shortfall `s` and
  - (b) glue `a` or `g`.<sup>6</sup>
5. The best<sup>7</sup> way to get here, i. e., `@@17` is via `->` breakpoint `@@` 16. ■

### 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 thousands 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`. ■

<sup>5</sup> Reference 28 provides an exceptionally detailed discussion of the output of `\tracingparagraphs`.

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

<sup>7</sup> »Best« means the minimum-demerits path in the graph of the feasible breakpoints, which has been constructed for the paragraph.

**Investigating page-breaks.** Use `\tracingpages=1` or the `typoginspect` environment to switch on tracing of T<sub>E</sub>X's page-break information [15, p. 112n].<sup>8</sup>

The first time vertical material enters a new page, T<sub>E</sub>X logs

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

where `<text-height>` is the total height T<sub>E</sub>X 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<sub>E</sub>X 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. ■

### 3.3 Hyphenation

T<sub>E</sub>X's and thus L<sup>A</sup>T<sub>E</sub>X'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`

T<sub>E</sub>X inhibits breaks of the component words by default. The following macro rectifies the problem.

```
\allowhyphenation
```

Macro `\allowhyphenation` re-enables automatic hyphenation after T<sub>E</sub>X has turned it off, for example, in the innocuous case of a hyphenated compound.

The admittedly simple rules when T<sub>E</sub>X 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<`

<sup>8</sup> See also the discussion of the T<sub>E</sub>X output routines by SOLOMON [24].

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. 12.
- 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 T<sub>E</sub>X into hyphenating the first word of a paragraph is to put `\hskip0pt` in front of it.
- Enable automatic hyphenation in the rare but weird cases when T<sub>E</sub>X does not hyphenate a word that is hyphenatable despite the result is an overfull box.<sup>9</sup> ■

Whenever using `\-`, the short-hand form of `\discretionary{-}{ }{ }`, authors writing in a foreign language should reconsider whether it really beats `\hyphenation` or `\babelhyphenation`<sup>10</sup> 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}}}
```

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.

T<sub>E</sub>X never hyphenates the initial word in a paragraph and `\allowhyphenation` cannot help in this case. Start the paragraph with `\hskip 0pt` to enable hyphenation even for the first word.

<sup>9</sup> The author of typog has no idea, why this happens, but he has been successful in fixing the problems with code like `(\allowhyphenation Superpositionsprinzip)` or `(\allowhyphenation Superauswahlregel)`. Elucidation by a T<sub>E</sub>X-savant is highly welcome.

<sup>10</sup> `\babelhyphenation` is the multi-lingual extension of T<sub>E</sub>X's `\hyphenation` and it is defined in package babel [5].

TABLE 1:  $\text{\TeX}$  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‹ → ›hy-phen-ated‹ and ›compound‹ →  
 ›com-pound‹.

$\text{\LaTeX}$ -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> [35]
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

**Tip — Typewriter Fonts**

Sometimes it is desired to get a hyphenatable typewriter font. L<sup>A</sup>T<sub>E</sub>X 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 [15, 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<sup>A</sup>T<sub>E</sub>X macro `\allowbreak` is not only that the former has a starred form, but the penalty associated with `\breakpoint` is the current<sup>11</sup> `\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 ■
```

`hyphenmin (env.)`  
**SINCE V0.3**

Set the values of `\lefthyphenmin` and `\righthyphenmin` confined to an environment.

```
\begin{hyphenmin} [⟨left-hyphen-minimum⟩] {⟨hyphen-minimum⟩}
...
\end{hyphenmin}
```

Without optional argument `hyphenmin` sets both `\lefthyphenmin` and `\righthyphenmin` to `⟨hyphen-minimum⟩`. When called with an optional argument it sets `\lefthyphenmin` to `⟨left-hyphen-minimum⟩` and `\righthyphenmin` to `⟨hyphen-minimum⟩`.<sup>12</sup>

**Use Case**

If the hyphen minimums were *increased* e.g. in the preamble: Reduce the hyphen minimum in the index or other multi-column environments with narrow lines to regain hyphenation possibilities. ¶ Use a large `⟨hyphen-minimum⟩` to disable hyphenation. ■

<sup>11</sup> At this point in the document `\exhyphenpenalty=50` holds.

<sup>12</sup> The current values for `\lefthyphenmin` and `\righthyphenmin` in this document are 2 and 3, respectively.

### 3.4 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<sup>A</sup>T<sub>E</sub>X will bail out with an error if the extra argument is not passed to `\nolig` in these situations.

Alternatively use `\texorpdfstring` [21, 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 distracting here, for it occurs at the boundary of two syllables). ■

### 3.5 Manual Italic Correction

`\itcorr`      The italic correction offered by  $\TeX$  or  $\LaTeX$  sometimes needs a helping hand.  
`\itcorr*`

```
\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`<sup>13</sup> 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 side<sup>14</sup> 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.**  $\TeX$  records the slant angle  $\alpha$  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  $\alpha$  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  $\alpha$  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  $\alpha$  exchange the reference points.

Figure 1 shows how `<strength>` and  $\alpha$  are related. Moreover, it demonstrates how intricate italics correction is.

<sup>13</sup> Separate adjustments may be desirable if the math font's italics have markedly different slants.

<sup>14</sup> 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 `\,^`.

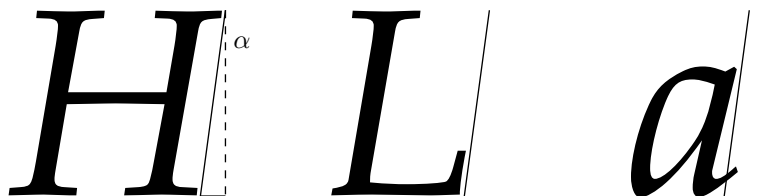


FIGURE 1: Some letters of an italics font. We use the capital `H` to measure the angle  $\alpha$  between the plumb-line (drawn dashed) and a tangent to the rightmost parts of the glyph. The length of the plumb-line is proportional to `<strength>` and the short, thick part of the baseline symbolizes the resulting italics correction. ¶ The middle example, the capital `L`, shares  $\alpha$  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<sub>E</sub>X allows to assign an italic correction to each individual character of a font. Not only features the lowercase `a` a larger  $\alpha$  – 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`.

### 3.6 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.

#### 3.6.1 Slash

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

```
\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`.

The kerned slash can typeset lowered (or raised), where the offset with respect to the baseline is configured with `lowerslash`.

#### Tip

Define specialized macros for slashes surrounded by lowercase letters or small caps if they are needed often or require a specific value for `slashkern`.

```
\newcommand*\lowercasekernedslash{
  {\begin{typogsetup}{lowerslash=.1em,
    slashkern=.02em}
    \kernedslash
  }
}
```

Generic solutions could build upon `\fontdimen5` and `\fontdimen6` (see Tab. 7 on p. 35) or measure the height and depth of the slash-glyph (abstracted

OPTION `lowerslash`  
INTRODUCED IN V0.5



in `\typog@forwardslash`) and compare it to e.g. the height of selected lowercase characters. ■

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`). ¶ Use option `lowerslash` to adjust the slash to surrounding lowercase letters. ¶ Fix a too high raising slash of a font (e.g. Cochineal) with option `lowerslash`. ■

### 3.6.2 Hyphen

`\kernedhyphen`      Macros `\kernedhyphen*` and `\kernedhyphen` expand to a hyphen (·) with  
`\kernedhyphen*`      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` [35]) 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. 3.7. 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⟩`. ■

`\leftkernedhyphen`  
`\leftkernedhyphen*`  
`\rightkernedhyphen`  
`\rightkernedhyphen*`

### 3.7 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 typeset these characters at a user definable, adjusted height above the baseline. Users can base their own definitions of raised characters on their associated dimensions.<sup>15</sup>

#### 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.

#### 3.7.1 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. ■

<sup>15</sup> Also compare with Ex. 12 in Ref. 34 for an attempt to automate vertical alignment.

### 3.7.2 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. ■

### 3.7.3 Number Dash (Figure Dash)

`\figuredash`  
`\figuredash*`

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. 3.7.1 and 3.7.2.

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

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 [10]`).

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

TABLE 2: Suggested values for raising `\figuredash`, which actually is an en-dash, between lining numerals of some selected fonts in multiples of 1 em.

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

\* Free Bembo.

#### 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*`. ■

#### 3.7.4 Multiplication Sign – Times <sup>x</sup>

`\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. ■

#### 3.7.5 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 T1<sup>16</sup> font encoding not on package babel.

<sup>16</sup> Font encoding T1 can be forced via `\usepackage[T1]{fontenc}` in the document preamble.

`\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](#).

`\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 .025 em to .075 em 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 in multiples of 1 em.

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

#### 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. ■

### 3.7.6 Inverted Exclamation Mark and Inverted Question Mark

The Spanish (Castilian, Asturian, etc.) language requires exclamatory sentences and questions to be followed by exclamation marks and question marks, respectively and recommends that they are preceded by inverted (or ›upside-down‹) versions of these punctuation characters.

Usually the horizontally-mirrored versions' designs follow that of the lowercase letters. In particular the inverted marks feature descenders (see left-hand side of Fig. 2). For all-uppercase or all-smallcaps phrases, e. g. in headlines the descending parts of these marks disturb the tight block structure and it may be preferable to have inverted marks that align with the baseline.

`\capitalinvertedexclamationmark`    The macros `\capitalinvertedexclamationmark` and `\capital-`  
`\capitalinvertedquestionmark`        `invertedquestionmark` simultaneously provide bottom-aligned in-  
BOTH SINCE V0.5                        verted marks for up to three different sets of exclamation marks or  
question marks, e. g., for uppercase, med-caps, and small-caps text<sup>17</sup> or just for  
some stylistic alternatives offered by particularly well equipped fonts.

```
\capitalinvertedexclamationmark{<number>}
\capitalinvertedquestionmark{<number>}
```

<sup>17</sup> The small exclamation and question marks even made it into Unicode.

Typeset inverted exclamation marks and inverted question mark. Select the appropriate mark and the associated raise-amount with  $\langle number \rangle=1, 2$ , or  $3$ . We sketch the result of the automated correction process in Figure 2.

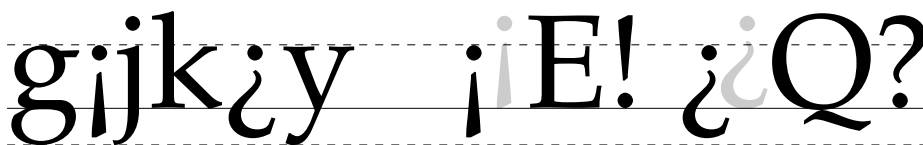


FIGURE 2: The left-hand sample shows the inverted exclamation mark and question marks wedged between some lowercase letters. The baseline is drawn solid and the ex-height as well as the descender depth are indicated with dashed lines. ¶ On the right hand side we display the same marks in conjunction with capital letters. The gray glyphs demonstrate the result of shifting them up such that they exactly touch the baseline, which mimics the auto-raise mode described in the text. ¶

The sample font once again is URW Palladio .

Inverted exclamation and inverted question marks by the same  $\langle number \rangle$  are raised by the same amount which is configurable with option `raiseinverted-marks`, where  $\langle number \rangle$  corresponds to the dimension at position  $\langle number \rangle$ . If the dimension is equal to 0 pt the marks are auto-raised as shown in Fig. 2. This may amount to different lengths for the inverted exclamation mark and inverted question mark even at the same  $\langle number \rangle$ . A non-zero dimension raises the marks by exactly that length. To get a (quasi-)zero length use e. g. 1 sp.

#### Tip

The letters  $\text{V}$ ,  $\text{W}$ , and  $\text{Y}$  following an inverted question mark may need some manual kerning as well as  $\text{J}$  following an inverted exclamation mark. Macro `\itcorr` (p. 15) offers a solution that is almost portable across font changes and a  $\langle strength \rangle$  of  $-1$  is a good starting point. ■

The user-side macros `\capitalinvertedexclamationmark{\langle number \rangle}` and `\capitalinvertedquestionmark{\langle number \rangle}` call the parameter-less, internal macros

```
\typog@inverted@exclamationmark@<roman-numeral>
\typog@inverted@questionmark@<roman-numeral>
```

where  $\langle roman-numeral \rangle=i, ii$ , and  $iii$  correspond to  $\langle number \rangle=1, 2$ , and  $3$ . Their defaults are `\textexclamdown` and `\textquestiondown` for all three numbers, respectively. Users can redefine the internal macros to hook up special fonts or characters.

### 3.8 Vertically Adjust Label Items of Environment `itemize`

Perfection of planned layout  
is achieved only by institutions  
on the point of collapse.  
— CYRIL NORTHCOTE PARKINSON

The symbols that L<sup>A</sup>T<sub>E</sub>X uses to distinguish the items of `itemize` lists do not always align well in the vertical direction with the following text. Sometimes the label is too low, especially if followed by an uppercase (initial) letter. In rare occasions the label is placed too far above the baseline. If any label has been taken from a math-font vertical alignment with the text font is almost purely accidental.<sup>18</sup>

`\uppercaseadjustlabelitems`      Package typog lets the user vertically align the `itemize` labels for subsequent uppercase or lowercase letters, where the designations `>uppercase<` and `>lowercase<` are just names for two four-tuples of lengths (technically: `dimens`) to shift the labels up or down.

`\lowercaseadjustlabelitems`

`\noadjustlabelitems`

ALL THREE SINCE V0.4

```
\uppercaseadjustlabelitems{<levels-to-adjust>}
\lowercaseadjustlabelitems{<levels-to-adjust>}
\noadjustlabelitems{<levels-to-adjust>}
```

Apply uppercase adjustment, lowercase adjustment or no adjustment to the labels in `itemize` environments at the `<levels-to-adjust>`. The adjustment values themselves, this is the vertical shifts are configured with options `uppercase-labelitemadjustments` and `lowercaselabelitemadjustments`. They are doubly font dependent: on the one hand the font where the label itself comes from and on the other hand the font of the copy.

The argument `<levels-to-adjust>` is a – possibly empty – comma separated list of the levels the respective adjustments are to be applied to. The levels themselves are given as *decimal* numbers, this is, 1, 2, 3, 4 or the special value `*` which stands for all four levels. An empty argument list also has a special meaning. Used within any `itemize` environment it automatically applies the adjustment to exactly this level.

#### Example

With the flexible syntax the following settings are possible.

- ▷ Correct all `itemize` labels for uppercase letters.

```
\uppercaseadjustlabelitems{*}
```

- ▷ Adjust nesting levels 1, 2, and 3 for uppercase letters and level 4 for lowercase.

```
\lowercaseadjustlabelitems{4}
\uppercaseadjustlabelitems{2,3,1}
```

- ▷ Within an `itemize` environment just turn off any correction for this level whatever it may be.

<sup>18</sup> The exception being mathematics typeset as text via package `mathastext` [7].



```

\begin{itemize}
\noadjustlabelitems{}
\item ...
\end{itemize}

```

- ▷ Override `\labelitemi` with a right-pointing triangle and adjust its vertical position inside of a `typogsetup` environment.

```

\begin{typogsetup}
  {uppercaselabelitemadjustments={.1em}}
  \renewcommand*\labelitemi{{\small$\rhd$}}
  \begin{itemize}
  \uppercaseadjustlabelitems{}
  \item ...
  \end{itemize}
\end{typogsetup}

```

The observant reader will have noticed that the itemized list in this emphasized section uses the code of the last example. ■

**Setup.** To assist the user in finding the desired adjustments of the labels of `typog` provides macros that help setting up `lowercaseadjustments` and `uppercaseadjustments`. Their intended uses are in the draft phase of a document or in non-printed sections of the text.

The macros assume a ›correct‹ height that they derive from the measured height of a sample text scaled by a user-defined factor, which defaults to  $\frac{1}{2}$ .<sup>19</sup> The then correct height gets indicated by a thin horizontal line parallel to the baseline. Thus, at sufficiently high magnifications it is possible to judge whether a label gets typeset too high or too low with respect to this reference line.

#### Note

The macros use the actual height of a given sample text. So, a lowercase sample should not contain any letters with ascenders.

Swashes whether upper- or lowercase always need special attention. ■

`\typogadjuststairs`  
SINCE V0.4

To get a quick overview how the four `itemize` labels align vertically `\typogadjuststairs` draws them at user-defined steps, typically  $\frac{1}{4}$  pt,  $\frac{1}{3}$  pt, or  $\frac{1}{2}$  pt. It ignores any existing adjustments and in that way can be utilized as a first configuration step or, for a small *⟨step-size⟩* and a high *⟨number-of-steps⟩*, for an easy refinement.

```

\typogadjuststairs[⟨scale-factor⟩=.5]
  {⟨step-size⟩}{⟨number-of-steps⟩}
  {⟨sample⟩}

```

Generate stairs of *⟨number-of-steps⟩* vertically shifted label items; use the next odd number, if *⟨number-of-steps⟩* is even. Draw a reference hairline at *⟨scale-factor⟩*

<sup>19</sup> The default factor of .5 hearkens back to STRIZVER's suggestion that ›[b]ullets should be centered on either the cap height or x-height of the neighboring text, ...‹ [25, p. 220].

times the height of  $\langle sample \rangle$ , where  $\langle scale-factor \rangle$  defaults to .5. The stairs start at a vertical shift of

$$-\frac{\langle number-of-steps \rangle - 1}{2} \times \langle step-size \rangle$$

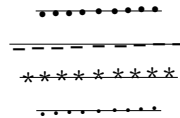
and repeat up

$$\frac{\langle number-of-steps \rangle - 1}{2} \times \langle step-size \rangle.$$

The central step – which is always surrounded by a bit more space – shows the neutral alignment, this is 0 pt. `\typogadjuststairs` never prints the contents of  $\langle sample \rangle$ .

#### Example

Play ball!



This is the result of `\typogadjuststairs{.25pt}{9}{ABC}` with the document's definitions of `\labelitem{N}`. The fifth label item in each line is the uncorrected one. ■

`\typoguppercaseadjustcheck` For a quick and easy check how the four label items vertically align *as configured* use `\typoguppercaseadjustcheck` and `\typoglowercaseadjustcheck`. Experienced users with a keen eye for type can apply these macros even in the initial setup. An accurate determination of uppercase-labelitemadjustments and lowercase-labelitemadjustments is preferably done at a high magnification (400% to 600% on a 100 dpi screen) with a representative sample of initial letters.

**BOTH SINCE V0.4**

```
\typoguppercaseadjustcheck[ $\langle scale-factor \rangle$ =.5]{ $\langle sample \rangle$ }
\typoglowercaseadjustcheck[ $\langle scale-factor \rangle$ =.5]{ $\langle sample \rangle$ }
```

Typeset all four label items adjusted for uppercase or for lowercase with an indicator line at  $\langle scale-factor \rangle$  times the  $\langle sample \rangle$ 's actual height. The default  $\langle scale-factor \rangle$  is .5. Both macros refer to the currently configured values for the uppercase or lowercase adjustments but they are independent of any settings done with `\uppercaseadjustlabelitems`, `\lowercaseadjustlabelitems`, or `\noadjustlabelitems`. Again,  $\langle sample \rangle$  does not get printed.

#### Example

Uppercase check with `\typoguppercaseadjustcheck{ABCXYZ}`:

ABGH•—•QWYZ, 0123•—•4567

and similarly for lowercase: `\typoglowercaseadjustcheck{acexyz}`:

ace•—•mno, bdf•—•gji, 0123•—•4567,

where we have bracketed the macro calls with selected uppercase and lowercase letters, or suitable figures. ■

In Table 4 on p. 28 we collected some suggestions for adjustment values in the *default* case when the label items are not redefined by the user and expand like

```
\labelitemi  ⋮ \labelitemfont \textbullet,
\labelitemii ⋮ \labelitemfont \bfseries \textendash,
\labelitemiii ⋮ \labelitemfont \textasteriskcentered, and
\labelitemiv ⋮ \labelitemfont \textperiodcentered.
```

They display as  $\cdot$ ,  $\text{---}$ ,  $\ast$ , and  $\text{---}$ , respectively.

### 3.9 Align Last Line of a Paragraph

The usual algorithms of L<sup>A</sup>T<sub>E</sub>X typeset the last line of a paragraph flush with the left margin unless center, raggedleft or Centering, FlushRight (package ragged2e [23]) are in effect. For an instructive discussion consult Ch. 17, »Paragraph End«, of Ref. 11. The following environments 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.)

```
\begin{lastlinerraggedleftpar}
...
\end{lastlinerraggedleftpar}
lastlineflushrightpar (alias)
```

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

Center the last lines of the paragraphs enclosed by this environment.<sup>20</sup>

`lastlinecenteredpar`  
(env.)

```
\begin{lastlinecenteredpar}
...
\end{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. ■

<sup>20</sup> Also compare the approach taken in Ref. 31.

TABLE 4: Some suggested values for the vertical adjustments of label items. The table assumes that the default definitions (of class article) for `\labelitem<N>` are in effect. The `itemize`-list levels `i`, `ii`, `iii` and `iv` are referred to with  $N = 1, 2, 3, 4$ . All lengths are given as printer points (pt) and refer to a document font size of 10 pt.

Font Name	Uppercase Adjustments				Lowercase Adjustments			
	1	2	3	4	1	2	3	4
ADF Accanthis	.5	1.5	-1.5	1.125	-.75	.25	-2.75	-.125
ADF Venturis	.0	1.0	.75	1.0	-.75	.0	-.25	.0
CM Roman	1.0	.75	1.0	1.0	-.25	-.5	-.25	-.25
Domitian	.25	1.0	.75	1.0	-1.0	.0	-.325	-.25
Cochineal	1.0	1.0	1.0	1.0	-.125	-.125	-.125	-.125
EB Garamond	.0	1.25	-.875	1.325	-1.5	.0	-2.0	0.125
etbb*	.25	1.0	1.125	.5	-1.0	-.0	-.0	-.5
Extended Charter†	-.25	1.0	1.75	.125	-1.125	.0	.875	-.875
Gentium	.0	.75	.0	.0	-.5	-.25	-.75	-1.0
GFS Bodoni	-.25	.625	1.0	1.125	-1.25	-.625	-.5	-.25
GFS Didot	-1.5	.75	1.0	1.25	-2.75	-.25	-.25	.25
IBM Plex Serif	.5	.5	-1.325	.5	-.25	-.25	-2.25	-.25
KP Serif‡	.0	1.0	1.25	.75	-1.0	.0	.0	-.5
Libertinus Serif	1.0	.75	1.125	.75	.0	-.325	.0	-.25
ML Modern	1.25	.75	1.0	.125	.0	-.5	-.25	-.125
Source Serif Pro	.125	.75	-1.0	.125	-.75	.0	-2.0	-.75
Spectral	.625	.625	-1.5	.625	-.25	-.25	-2.5	-.25
STIX	1.0	1.0	.75	1.0	.0	.0	.0	.0
URW Palladio§	.25	1.125	1.0	1.0	-1.0	-.125	-.125	-.125
Utopia Regular	.0	1.0	.75	1.0	-.75	.0	-.25	.0

\* EDWARD TUFTE's Bembo in package ETbb. Note the two initial capital letters in the filename.

† Found in package XCharter. Again note the two initial capitals in the filename.

‡ In package kpfnts.

§ Contained in package mathpazo.

|| Utopia is available through package fourier or package mathdesign. In the latter case pass option `adobe-utopia` to the package.

### 3.10 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.

#### 3.10.1 Problem Definition

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

1. `\parindent > 0` [31, 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-left<sup>21</sup> setting, e. g., `amsmath` [2] and option `fleqn`.<sup>22</sup>

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` [31, 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.<sup>23</sup>



<sup>21</sup> The common practice of centering displayed equations does not call for the manipulations of a paragraph's last line discussed here.

<sup>22</sup> For displayed equations and `amsmath` the relevant parameter is `\mathindent`.

<sup>23</sup> Package `parskip` defines `\parskip` as 6 pt plus 2 pt for a base size of 10 pt.

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`<sup>24</sup> for the gap [8].

### 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.

$\text{\TeX}$  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 for example Sec. 3.13) to achieve a low badness such that the desired paragraph end is deemed feasible by  $\text{\TeX}$ .
- 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. ■

#### 3.10.2 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.

###### (b) `\mbox`

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

<sup>24</sup> For example,  $\text{\LaTeX}$ 's class article uses a `\parindent` of 25 pt.

This itemize list demonstrates vertically adjusted label items (Sec. 3.8).

(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 [33]. In turn the next-to-last lines may become unsightly. Counteract this degradation e. g. with recipes 2a to 2c.

Tying and `\mboxing` 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. 3.11.1.

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. 3.12.1. Increase or decrease the tracking in steps of  $\frac{1}{1000}$  em until the last line looks good.

## (c) Vary font expansion.

Enclose the paragraph in a `setfontexpand` group. See Sec. 3.12.2.

## 3. A combination of any of the above items.

## 4. Some curveballs.

(a) If the paragraph already suffers from one of the problems that  $\text{\TeX}$  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. 3.12.2.

**3.10.3 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 line<sup>25</sup> or a club line<sup>26</sup> [15, p. 104 and 19]. (See also Sec. 3.14

`prolongpar` (*env.*)

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

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

for special functions to avoid clubs or widows.) ›Accidentally‹, they also change the length of the last line of the paragraph.

```
\begin{shortenpar} ... \end{shortenpar}
```

Environment `shortenpar` decreases the `\looseness` of the paragraph.<sup>27</sup> It performs well if the last line of the paragraph is short or the whole paragraph is loose.

```
\begin{prolongpar} ... \end{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.

#### 3.10.4 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.

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

```
\begin{covernextindentpar}[\langle dim \rangle]
...
\end{covernextindentpar}
```

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

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

```
\begin{openlastlinepar}[\langle dim \rangle]
...
\end{openlastlinepar}
```

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  $2\text{\parindent}$  (if  $\text{\parindent} \neq 0$ ) or  $2\text{em}$  (if  $\text{\parindent} = 0$ ). 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.

<sup>27</sup> Command `\looseness` is a  $\text{\TeX}$  primitive [15, p. 103n]. A thorough discussion of the interaction of `\linepenalty` and `\looseness` can be found in Ref. 30.

`covernextindentpar`  
(env.)

`openlastlinepar` (env.)



### 3.10.5 Consistent Spacing of Last Line

Since  $\varepsilon$ -TeX the paragraph-breaking algorithm can be instructed to match the spacing of the last line of a paragraph to the next-to-last line with the built-in `\lastlinefit` [6, Sec. 3.8].<sup>28</sup> Package `typog` wraps this control in an environment to limit the scope as its effects are not always beneficial to the appearance of the body.

`\lastlinefitpar (env.)`  
**SINCE V0.5**

Typeset a single paragraph with `\lastlinefit` set to a given value.

```
\begin{lastlinefitpar}[\langle value \rangle]
...
\end{lastlinefitpar}
```

The parameter range of  $\langle value \rangle$  is zero to thousand. It controls the fraction of thousandths of the next-to-last-line's spacing to be used in the last line. This means that  $\langle value \rangle = 0$  requests no transfer (as if `\lastlinefit` is not used at all) and  $\langle value \rangle = 1000$  initiates exact transfer of the spacing. The default  $\langle value \rangle$  is 1000.

Note that different  $\langle value \rangle$ s can lead to different breakpoints in the affected paragraph.  $\varepsilon$ -TeX still considers the *whole* paragraph when calculating the optimal line breaks.

#### Use Cases

Adapt the spacing of the last line of a paragraph to a very loose or very tight next-to-last line or to a whole paragraph that is very loose or very tight. ■

## 3.11 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<sup>A</sup>T<sub>E</sub>X. No extra packages are required. Compare to the microtype-based functionality of Sec. 3.12.

### 3.11.1 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 [18, 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

<sup>28</sup> The actual algorithm is somewhat involved. So, for ease of reference, we have included the relevant part of the  $\varepsilon$ -TeX-manual as Appendix B on p. 62.

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.

```
\begin{loosespacing}[\langle level \rangle] ... \end{loosespacing}
```

Environment `loosespacing` increases the width of a space by the percentages given in the Tab. 5.

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

TABLE 5: 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.

```
\begin{tightspacing}[\langle level \rangle] ... \end{tightspacing}
```

Environment `tightspacing` decreases the width of a space by the percentages given in Tab. 6.

$\langle level \rangle$	Adjustment %	Note
0	n/a	neutral
1	-1.25	default
2	-2.5	
3	-5	
$\geq 4$	-10	

TABLE 6: Adjustments made by environment `tightspacing` to `\spaceskip`. The mapping of  $\langle level \rangle$  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  $\langle level \rangle$  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. ■

### 3.11.2 Wide Space

The `\widespace` macro and its companion `\narrowsspace` derive their appearances from several of the current font's `\fontdimen⟨number⟩`s. T<sub>E</sub>X addresses the latter by integers, which is totally non-memnonic. Therefore, we play softball by first presenting Tab. 7 that associates the `\fontdimen⟨number⟩`s with their meanings and also reports on their current values (for this document).<sup>29</sup>

#	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	$x^{\circ}$ height	47.5
6	<code>\quad</code> height	100
7	Extra space width	3.9

TABLE 7: All T<sub>E</sub>X font parameters normalized to the font's quad-size. The first column # states the index of the `\fontdimen⟨number⟩` parameter: `⟨number⟩`. Column 2 presents short descriptions of `\fontdimen⟨number⟩`. As examples, the values for the current font are shown in column 3.

`\widespace`  
`\widespace*`

STARRED FORM SINCE V0.2

Typeset a wide, sentence-ending space as if in `\nonfrenchspacing` mode. Consult Table 8 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.

The sentence that ends with  
! uses `\widespace` after the  
period.

<sup>29</sup> The association is given in Appendix F (p. 433) of Ref. 15. For a concise and understandable explanation of the T<sub>E</sub>X `\fontdimen` parameters consult Ref. 9.

**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`.<sup>30</sup> ■

**3.11.3 Narrow Space**`\narrowospace`

Typeset a narrow space. Consult Table 8 for a comparison of the various sizes.

`\narrowospace*`

SINCE v0.2

```
\narrowospace
\narrowospace*
```

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

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

if `\fontdimen7` is different from zero or otherwise

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

The starred version, `\narrowospace*`, unconditionally uses the `\fontdimen7 = 0` code-path. Refer to Table 7 for the meanings of the various `\fontdimen` parameters.

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`s.<sup>31</sup> ■

**3.12 Microtype Front-End**

The functionalities are just front-ends of selected macros in package `microtype` – welcome syntactic sugar.

**Important**

All macros and environments introduced in this section require that package `microtype` [22] has been loaded, preferably *before* package `typog`

```
\usepackage[⟨microtype-options⟩...]{microtype}
\usepackage[⟨typog-options⟩...]{typog}
```

in the document preamble. ■

<sup>30</sup> See also »Investigating the badness of a paragraph« on Page 9.

<sup>31</sup> Footnote 30 again applies.

TABLE 8: Exemplary comparison of standard `\space` versus `\narrow-space` and `\widespace`. All values are relative to the size of the current font’s quad-size and shown as a percentage of it. `\narrow-space` and `\widespace` use the package’s defaults. ¶ The upper values in the »Width« column for `\narrow-space` and `\widespace` refer to the `\fontdimen7 ≠ 0` case and the lower ones to the `\fontdimen7 = 0` code-path.

Macro	Width	Stretch	Shrink
<code>\narrow-space</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

### 3.12.1 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.

```
\begin{setfonttracking}{⟨delta⟩}
...
\end{setfonttracking}
```

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` [22] for a detailed explanation.

For font combinations involving monospaced fonts (T<sub>E</sub>X 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. ■

3.12.2    Font Expansion

setfontshrink (env.)      Adjust the limits of either only stretchability or only shrinkability and zero the other component, i. e., shrinkability and stretchability, respectively.  
setfontstretch (env.)

```
\begin{setfontshrink}[\langle level \rangle] ... \end{setfontshrink}
\begin{setfontstretch}[\langle level \rangle] ... \end{setfontstretch}
```

A  $\langle level \rangle$  of zero is a no-op. Tables 9 and 10 summarize the values for stretch and shrink in these environments.

$\langle level \rangle$	stretch	shrink	Note
0	n/a	n/a	no operation
1	0	5	default
2	0	10	
3	0	20	

TABLE 9:    Preconfigured values for shrink inside of environment setfontshrink as  $\frac{1}{1000}$  em. Note that all stretch values are zero, so the fonts only can shrink.

$\langle level \rangle$	stretch	shrink	
0	n/a	n/a	no operation
1	5	0	default
2	10	0	
3	20	0	

TABLE 10:    Preconfigured values for stretch inside of environment setfontstretch as  $\frac{1}{1000}$  em. 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.

```
\begin{setfontexpand}[\langle level \rangle] ... \end{setfontexpand}
```

Table 11 gives an overview of the values associated with  $\langle level \rangle$ .  
The six shrink and stretch limits of setfontexpand can be configured with package options shrinklimits and stretchlimits.

$\langle level \rangle$	stretch	shrink	Note
0	n/a	n/a	no operation
1	5	5	default
2	10	10	
3	20	20	

TABLE 11: Preconfigured values for shrink and stretch inside of environment `setfontexpand` as  $\frac{1}{1000}$  em. Note that both shrink and stretch values are nonzero, so the fonts can shrink or expand.

- 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 [22].
- Moreover, a text may not »respond« neither to `setfontshrink`, `setfontstretch`, nor `setfontexpand` because  $\text{\TeX}$  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.

```
\begin{nofontexpansion} ... \end{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`. ■

3.12.3 Character Protrusion

`nocharprotrusion`  
(*env.*)

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

```
\begin{nocharprotrusion} ... \end{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. ■

### 3.13 Sloppy Paragraphs

Experienced L<sup>A</sup>T<sub>E</sub>X users know that `\sloppy` is more of a problem by itself and not really a viable solution of the »overfull box« syndrome.

We define the macro `\slightlyloppy` and the associated environment, `slightlyloppypar`, with a user-selectable  $\langle sloppiness \rangle$  parameter. The constructions recover the known settings `\fussy` ( $\langle sloppiness \rangle = 0$ ) and `\sloppy` ( $\langle sloppiness \rangle \geq 8$ ), and introduce seven intermediate  $\langle sloppiness \rangle$  levels.<sup>32</sup> The default  $\langle sloppiness \rangle$  is 1.

`\slightlyloppy`  
`slightlyloppypar`  
 (env.)

```
\slightlyloppy[\langle sloppiness \rangle]
\begin{slightlyloppypar}[\langle sloppiness \rangle]
...
\end{slightlyloppypar}
```

Table 12 summarizes the adjustments that `\slightlyloppy` makes depending on the  $\langle sloppiness \rangle$  level.

TABLE 12: Adjustments made by `\slightlyloppy` to various T<sub>E</sub>X parameters at different levels of  $\langle sloppiness \rangle$ .

$\langle sloppiness \rangle$	<code>\tolerance</code>	<code>\hfuzz</code> <code>\vfuzz</code> pt	<code>\emergencystretch</code> $G$ em	Note
0	200	.1	0	T <sub>E</sub> X: <code>\fussy</code> default
1	330 <sup>†</sup>	.15	.375 <sup>‡</sup>	
2	530 <sup>†</sup>	.2	.75 <sup>‡</sup>	
3	870 <sup>†</sup>	.25	1.125 <sup>‡</sup>	
4	1410 <sup>†</sup>	.3	1.5 <sup>‡</sup>	
5	2310 <sup>†</sup>	.35	1.875 <sup>‡</sup>	
6	3760 <sup>†</sup>	.4	2.25 <sup>‡</sup>	
7	6130 <sup>†</sup>	.45	2.625 <sup>‡</sup>	T <sub>E</sub> X: <code>\sloppy</code>
$\geq 8$	9999	.5	3	

<sup>†</sup> All intermediate levels set `\pretolerance = \tolerance/2`.

<sup>‡</sup> 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.

Environment `slightlyloppypar[\langle sloppiness \rangle]` mimics L<sup>A</sup>T<sub>E</sub>X's `sloppy-`

<sup>32</sup> Also compare the findings for `\emergencystretch` in Ref. 29.



par, while offering the flexibility of `\slightly sloppy`.

#### 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`. ■

### 3.14 Vertically Partially-Tied Paragraphs

L<sup>A</sup>T<sub>E</sub>X provides several macros and environments to tie material vertically – most prominently `samepage` and `minipage`.<sup>33</sup> 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<sub>E</sub>X in the usual way.

The macros and environments described in this section locally set  $\epsilon$ -T<sub>E</sub>X 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⟩]
\begin{vtietoppar}[⟨number-of-lines⟩] ... \end{vtietoppar}
```

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<sup>A</sup>T<sub>E</sub>X:

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

```
\end{description}
```

33 A valuable complement to these is package `needspace` [37] which takes a different approach and reliably works in *mixed* horizontal and vertical mode situations.

Alternatively with package `enumitem` [4]:

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

or shorter and with the default  $\langle number-of-lines \rangle$ , 3, using the `enumitem` style<sup>34</sup> `vtietop`:

`vtietop` (*enumitem* key)

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

`\vtiebot`

Avoid a widow line in each partial paragraph.

`vtiebotpar` (*env.*)

```
\vtiebot[\langle number-of-lines \rangle]
\begin{vtiebotpar}[\langle number-of-lines \rangle] ... \end{vtiebotpar}
```

Vertically tie the last  $\langle number-of-lines \rangle$  in a paragraph. Zero or one for  $\langle number-of-lines \rangle$  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.*)

```
\begin{vtiebotdisp}[\langle before-disp-number-of-lines \rangle]
...
\end{vtiebotdisp}
```

Vertically tie the last  $\langle before-disp-number-of-lines \rangle$  in a paragraph before a display. Zero or one for  $\langle before-disp-number-of-lines \rangle$  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.

```
\begin{vtiebotdisptoppar}[\langle before-disp-number-of-lines \rangle
                          [\langle after-disp-number-of-lines \rangle]
...
\end{vtiebotdisptoppar}
```

<sup>34</sup> The documentation of `enumitem` prosaically calls them ›keys‹ (Section 3) not ›styles‹.

Vertically tie the last  $\langle\textit{before-disp-number-of-lines}\rangle$  in the paragraph before a display and the first  $\langle\textit{after-disp-number-of-lines}\rangle$  in the paragraph after the display. Moreover, turn the paragraphs and the display into an un-breakable unit.<sup>35</sup>

Zero or one for  $\langle\textit{before-disp-number-of-lines}\rangle$  as well as  $\langle\textit{after-disp-number-of-lines}\rangle$  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:

```
\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. 3.10.3 for other methods to avoid club or widow lines.

**Partial Paragraphs And Counting Lines.** The top-of-paragraph ties, `\vtietop` and `vtietoppar` count  $\langle\textit{number-of-lines}\rangle$  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 T<sub>E</sub>X's rules, a displayed math formula always is counted as *three* lines no matter its contents. Table 13 summarizes these rules with the help of an example.

### 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 breakable with the help of `\intertext`<sup>36</sup> or `\shortintertext`<sup>37</sup> it is desirable to make the display breakable. This is achieved with `\allowdisplaybreaks` or the environment `breakabledisplay` which will be described in Sec. 3.15. ■

### Use Cases

Fix widows and orphans, e. g., those turned up by package `widows-and-orphans` [20]. ¶  
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. ■

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

36 Introduced in package `amsmath` [2].

37 Defined in package `mathtools` [12].

TABLE 13: 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	$\backslash\text{vtietop}^\dagger$ Count	$\backslash\text{vtiebot}^\ddagger$ Count
1	Text line <sub>1</sub>	1	3
2	Text line <sub>2</sub>	2	2
3	Text line <sub>3</sub>	3	1
4	} Display math		
5			
6			
7	Text line <sub>4</sub>	1	2
8	Text line <sub>5</sub>	2	1

<sup>†</sup> This is  $\epsilon\text{-TeX}$ 's counting scheme of  $\backslash\text{clubpenalties}$ ; it also holds for  $\text{vtietoppar}$ .

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

### 3.15 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  $\langle\text{level}\rangle$  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.

```
\begin{breakabledisplay}[\langle\text{level}\rangle]
...
\end{breakabledisplay}
```

Environment `breakabledisplay` simply passes on  $\langle\text{level}\rangle$  to `\allowdisplaybreaks`. Table 14 shows the default penalties that `amsmath` associated with each of the  $\langle\text{level}\rangle$ s.

#### *Tips*

- Terminating a line with `\` $\ast$  inhibits a break after this line.
- A `\displaybreak[\langle\text{level}\rangle]` can be set for *each* line of the displayed equation separately.  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  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.,

TABLE 14: Penalties `\interdisplaylinepenalty` associated with different *⟨level⟩*s of environment `breakabledisplay`. Depending on the version of package `amsmath` the actual penalties may differ.

<i>⟨level⟩</i>	<code>\interdisplay-</code> <code>linepenalty</code>	Note
0	10000	no operation
1	9999	
2	6999	
3	2999	default
4	0 <sup>†</sup>	

<sup>†</sup> This is the default of `\allowdisplaybreaks`.

```
\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. ■

### 3.16 Setspace Front-End

Package `setspace` [26] is a base hit when it comes to consistently setting the line skip for a document via the macro `\setstretch`. The interface of `\setstretch` though is unintuitive as it asks for an obscure factor. The L<sup>A</sup>T<sub>E</sub>X user however prefers to keep her eyes on the ball and set the line skip directly (e. g. 12.5 pt) or the lines' leading to a length or percentage of the font's size.<sup>38</sup> This is where the following macros go to bat.

#### Important

All macros that are introduced in this section rely on macro `\setstretch`. So package `setspace` must have been loaded with

```
\usepackage{setspace}
```

in the document preamble. ■

`\setbaselineskip`  
SINCE V0.3

Set the line skip using an absolute length – technically: a `dimen`.

```
\setbaselineskip{<baseline-skip>}
```

Set the `\baselineskip` to `<baseline-skip>`. This is what a non-initiated user expects from the assignment

```
\setlength{\baselineskip}{<baseline-skip>}
```

The `<baseline-skip>` can contain a rubber (stretch/shrink) component, however, `\setbaselineskip` will discard of it and issue a warning that only the fixed-length part will be used in the computation.

#### Example

- Let us assume we want to lighten the gray value of the copy a tad with the `\baselineskip` increased from 12 pt to 12.5 pt. To this end we say:

```
\setbaselineskip{12.5 pt}
```

- In a generic part of the document, where the actual `\baselineskip` is not known, we can refer to its current value and rescale it:

```
\setbaselineskip{\baselineskip * 12.5 / 12}
```

Care should be taken if code like the above is implicitly or explicitly repeated, because it results in a geometric series. ■

`\resetbaselineskip`  
SINCE V0.3

Reset the `\baselineskip` to its original value.

```
\resetbaselineskip
```

This macro simply expands to `\setstretch{1}`. So, we rely on `setspace`'s notion of what is a single-line `\baselineskip`.

`\setbaselineskippercentage`  
SINCE V0.3

Set the `\baselineskip` with a relative value calculated as a percentage of the current font's design size.

<sup>38</sup> To find out about the current font's size and the `\baselineskip` in printable form check out Sec. 3.2.1 on p. 6.

```
\setbaselineskippercentage{<baselineskip-percentage>}
```

Set `\baselineskip` to  $\text{\typogfontsize} \times \langle \text{baselineskip-percentage} \rangle / 100$ .

**Example**

We modify the previous example and assume a font design size of 10 pt, but now write

```
\setbaselineskippercentage{125}
```

which sets `\baselineskip` to  $10 \text{ pt} \times 125/100 = 12.5 \text{ pt}$ . ■

`\setleading`  
SINCE V0.3

Set the `\baselineskip` with an absolute length that gets *added to* `\typogfontsize`.

```
\setleading{<leading>}
```

Set the `\baselineskip` to `\typogfontsize` plus `<leading>`. Note that `<leading>` can be negative, e. g. to set solid.

**Example**

Another solution of the previous example, given a font design size of 10 pt is to write

```
\setleading{2.5pt}
```

which sets `\baselineskip` to  $10 \text{ pt} + 2.5 \text{ pt} = 12.5 \text{ pt}$ . ■

`\setleadingpercentage`  
SINCE V0.3

Set the `\baselineskip` to `\typogfontsize` *plus* a relative value calculated as a percentage of `\typogfontsize`.

```
\setleadingpercentage{<leading-percentage>}
```

Set `\baselineskip` to  $\text{\typogfontsize} \times (1 + \langle \text{leading-percentage} \rangle / 100)$ .

**Example**

We modify the previous example and again assume a font design size of 10 pt, but now write

```
\setleadingpercentage{25}
```

which sets `\baselineskip` to  $10 \text{ pt} \times (1 + 25/100) = 12.5 \text{ pt}$ . ■

`\typogfontsize`  
(dimen)  
SINCE V0.3

The macros `\setbaselineskippercentage`, `\setleading`, and `\setleadingpercentage` all depend on the font size. By changing `\typogfontsize` they can be configured for different font sizes.

The length `\typogfontsize` gets initialized at the end of the preamble to the default font's quad size:<sup>39</sup>

```
\typogfontsize=\fontdimen6\font
```

which is also called its »nominal size« or its »design size«. This assignment can be repeated at any point in the document to record a reference font's size. To set

39 For an overview of the various `\fontdimen<number>` parameters consult Tab. 7 on p. 35.

just `\typogfontsize` without changing the current font, encapsulate the font change in a group and export the new value:

```
\begingroup
\usefont{T1}{Arvo-TLF}{m}{n}\selectfont
\normalsize
\global\typogfontsize=\fontdimen6\font
\endgroup
```

An alternative to relying on the design size is using the actual size of an upper-case letter:

```
\settoheight{\typogfontsize}{CEMNORSUVWXZ}
```

With `\typogfontsize` defined this way it becomes trivial to set solid:

```
\setleading{0pt}
```

or

```
\setleadingpercentage{0}
```

### *Tip*

All macros in this section actually accept expressions of their respective argument types, though the sick rules of  $\text{\TeX}$  *<dimen>*- and *<skip>*-expressions apply.

Here are some forms that do work:

```
\setbaselineskip{12pt + 0.6667pt}
\setbaselineskip{12pt * 110 / 100}
\setbaselineskippercentage{100 + 25}
\setleading{1pt / -2.0}
\setleadingpercentage{10 - 25 / 2} ■
```



### 3.17 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 [32] we prescribe the line-widths with  $\text{T}_{\text{E}}\text{X}$ 's `\parshape` primitive and leave alone the stretchability or shrinkability of the glue.

#### Caution

None of the following environments work inside of lists. ■

<code>smoothraggedrightshapetriplet</code> ( <i>env.</i> )	We introduce three environments that set three, five, or seven different line-lengths (which $\text{T}_{\text{E}}\text{X}$ of course will repeat for paragraphs longer than three, five, or seven lines): <code>smoothraggedrightshapetriplet</code> , <code>smoothraggedrightshapequintuplet</code> , and <code>smoothraggedrightshapeseptuplet</code> ; they work for paragraph lengths up to 99, 95, and 98 lines, respectively.
<code>smoothraggedrightshapequintuplet</code> ( <i>env.</i> )	
<code>smoothraggedrightshapeseptuplet</code> ( <i>env.</i> )	

```
\begin{smoothraggedrightshapetriplet}[\langle option \rangle...]{\langle width1 \rangle}{\langle width2 \rangle}{\langle width3 \rangle}
...
\end{smoothraggedrightshapetriplet}
\begin{smoothraggedrightshapequintuplet}[\langle option \rangle...]{\langle width1 \rangle}...{\langle width5 \rangle}
...
\end{smoothraggedrightshapequintuplet}
\begin{smoothraggedrightshapeseptuplet}[\langle option \rangle...]{\langle width1 \rangle}...{\langle width7 \rangle}
...
\end{smoothraggedrightshapeseptuplet}
```

The environments take  $N = 3, 5$ , 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**= $\langle dim \rangle$

Set the left margin for the smooth ragged paragraph to  $\langle dim \rangle$ . Similar to the  $\text{T}_{\text{E}}\text{X}$  parameter `\leftskip`.

##### **parindent**= $\langle dim \rangle$

Set the first-line indent for the smooth ragged paragraph to  $\langle dim \rangle$ . Similar to the  $\text{T}_{\text{E}}\text{X}$  parameter `\parindent`.

<code>smoothraggedrightpar</code> ( <i>env.</i> )	Environment <code>smoothraggedrightpar</code> builds upon the three generators. It typesets a single paragraph with a given $\langle ragwidth \rangle$ of the ragged, right margin, where the rag width is the length-difference of the longest and the shortest lines.
--	---

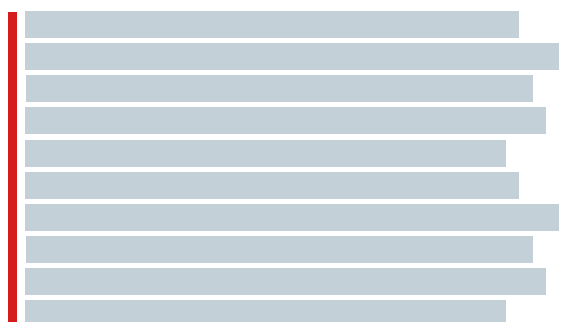
```
\begin{smoothrowrightpar}[\langle option \rangle...]
...
\end{smoothrowrightpar}
```

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

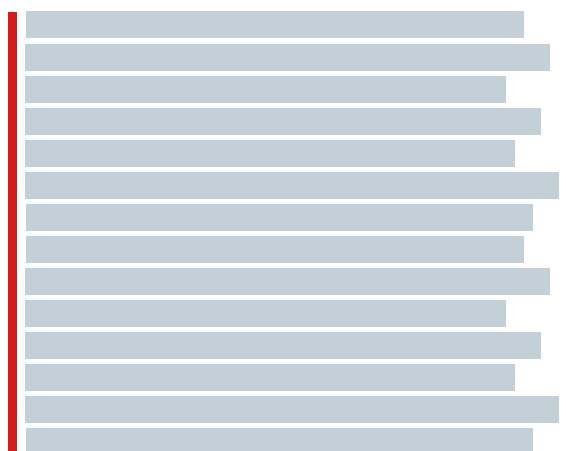
- The triplet generator repeats a *short line* – *long line* – *middle-length line* sequence. Shown below are two complete cycles.



- 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.



smoothrowright  
(env.)

Environment smoothrowright is the multi-paragraph version of smoothrow-

`raggedrightpar`. It takes the same optional arguments.

```
\begin{smoothrowright}[<option>...]
...
\end{smoothrowright}
```

### Options

#### **`\linewidth=<dim>`**

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

### Global Parameters

#### **`\smoothrowrightfuzzfactor=<factor>`**

The environment adds glue to every line-width<sup>40</sup> 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 `\smoothrowrightgenerator` (see below). The `\smoothrowrightfuzzfactor` 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).

#### **`\smoothrowrightgenerator`**

Select a generator to use. Valid generator names:

- `triplet`,
- `quintuplet`,
- `septuplet`.

The default generator is `triplet`.

#### **`\smoothrowrightleftskip=<dim>`**

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

#### **`\smoothrowrightparindent=<dim>`**

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

#### **`\smoothrowrighttragwidth=<dim>`**

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

<sup>40</sup> The shortest line only gets stretchability, the longest only receives shrinkability. All other lines are both stretchable and shrinkable.

Throughout this manual we have demonstrated how `smoothraggedright` environments work for very narrow columns namely inside of the document's margins: all maginal notes were typeset inside of `\smoothraggedright` environments (quintuplet generator, 1.5 em rag width, at footnote size in addition using environments `slightlyloppy` and `loosespacing`). Here, we utilize it with the quintuplet generator and a rag-width of only 7.0pt in a paragraph that is 355.0pt wide and averages around twelve words per line. There is much more glue to adapt to the line-ends and thus the desired rag is achieved far easier. The sloppyness is minimal, this is, `\fussy` is in effect and character protrusion into the margins is switched off. A limitation of the current implementation is that it is ineffective inside of lists. Therefore, this paragraph has not been wrapped inside of an `example`, because all examples are coded as lists.

#### *Use Cases*

Replacement for `\RaggedRight` [23]. ¶ Design alternative for fully justified paragraphs if used with a small rag-width. ■

#### *Anticipated Changes & Possible Extensions*

Translate the code to `l3galley` which is part of the `l3experimental` package [16]. Galley code is supposed to work inside of lists, too. ■

## 4 Limitations and Known Problems

Here is a list of some of the known problems of typog.

**Interference with package pdfcomment [14].** To play well with pdfcomment package typog *turns off* the default string substitution inside of PDF-bookmarks for the following macros if *both* packages are loaded:

<code>\breakpoint</code>	<code>\itcorr</code>
<code>\capitalemdash</code>	<code>\kernedhyphen</code>
<code>\capitalendash</code>	<code>\kernedslash</code>
<code>\capitalhyphen</code>	<code>\leftkernedhyphen</code>
<code>\capitaltimes</code>	<code>\nolig</code>
<code>\Doubleguillemetleft</code>	<code>\rightkernedhyphen</code>
<code>\Doubleguillemetright</code>	<code>\Singleguillemetleft</code>
<code>\doubleguillemetleft</code>	<code>\Singleguillemetright</code>
<code>\doubleguillemetright</code>	<code>\singleguillemetleft</code>
<code>\figuredash</code>	<code>\singleguillemetright</code>

Use `\texorpdfstring` – which is part of package hyperref [21] – in the relevant sectioning macros to recover string substitution on a case-by-case basis.

### *Example*

```
\section{\(CP\) \texorpdfstring{\capitalhyphen}{-}%
          Invariance}
```



## 5 Other Packages for Fine L<sup>A</sup>T<sub>E</sub>X Typography

Many other packages help with getting better output from L<sup>A</sup>T<sub>E</sub>X. Here is a list – in alphabetical order – of the ones the author considers particularly valuable.

- |           |  |
|-----------|--|
| enumitem  | Flexible and consistent definition of all basic L <sup>A</sup> T <sub>E</sub> X-list types plus in-line lists [4].   |
| geometry  | Powerful and sophisticated setup of the page layout [27]. Best accompanied by layout [17] to visualize the page geometries.  |
| hyphenat  | Hyphens that do not inhibit further auto-hyphenation of a compound word [35].  |
| microtype | Fine control of spacing, tracking, sidebearings, character protrusion into the margins, font expansion, and much more [22].<br>See Section 3.12 for a front-end to microtype offered by this package.<br>See also KHIREVICH's discussion [13]. |
| ragged2e  | Improved versions of environments <code>raggedleft</code> , <code>raggedright</code> , and <code>center</code> [23].   |
| setspace  | Consistently set the line-spacing of a document, i. e., control <code>\baselineskip</code> [26].<br>See Section 3.16 for a front-end to setspace offered by this package.  |

## A 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.

### A.1 Name

**typog-grep** - specialized grep for typog-inspect elements in L<sup>A</sup>T<sub>E</sub>X log files

### A.2 Synopsis

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

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

The first form, “discovery mode”, shows all *IDs* of

```
<typog-inspect id="ID" ...>
```

elements in *LOG-FILE*.

The second form shows the contents, *LOG-DATA*, of the elements

```
<typog-inspect id="ID" ...>
LOG-DATA
</typog-inspect>
```

whose *IDs* match *REGEXP* in *LOG-FILE*.

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

### A.3 Description

**typog-grep** is a tailored post-processor for L<sup>A</sup>T<sub>E</sub>X *log* files and the **typoginspect** environment as provided by the L<sup>A</sup>T<sub>E</sub>X package **typog**. It shares more with the venerable **sgrep** than with POSIX **grep**.

In the L<sup>A</sup>T<sub>E</sub>X source file the user brackets her text or code in a **typoginspect** environment:

```
\begin{typoginspect}{ID}
TEXT-OR-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 result of the environment shows up, packed with tracing information, as

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

where all the capital-letter sequences are meta-variables and in particular *JOB-NAME* is the expansion of `\jobname`, *LINE-NUMBER* is the L<sup>A</sup>T<sub>E</sub>X source file line number of the beginning of the `typoginspect` environment, and *PAGE-NUMBER* is the page where the output of *TEXT-OR-CODE-TO-INVESTIGATE* occurs.

**typog-grep** reveals the contents of *LOG-FILE* between `<typog-inspect id="ID" ...>` and `</typog-inspect>` excluding the XML-tags themselves. 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.

## A.4 Options

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

### **-a, --all, --any**

ID-discovery mode: Discover all `typog-inspect` elements independent of any matching patterns and print their *IDs*. The results are printed in their order of occurrence in the respective *LOG-FILES*. Pipe the output into **sort** to get alphabetically ordered *IDs*.

Augment with options `--job-name`, `--line-number`, `--log-line-number`, or `--page-number` for more information.

### **--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 section A.5 for a description of all available configuration items. Use option **--show-config** to display the default configuration.

**--debug**

Turn on debug output on *stderr*.

**-E, --encoding** *ENCODING*

Set the *ENCODING* of *LOG-FILE* for the translation to UTF-8. The default is unset.

Use this option to get rid of pesky "<HEX-DIGITS>" escapes on UTF-8 terminals. See option **--show-encodings** for the known encodings and Encode::Supported for a summary of all encodings. See also section A.5.2.

Apply **iconv** (POSIX) or **recode** (GNU) on *LOG-FILE* before this tool to avoid having to use option **--encoding**.

**-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 *latex* associated with the input file.

**-n, --[no-]line-number**

Print the line number where the `typoginspect` environment was encountered in the *L<sup>A</sup>T<sub>E</sub>X* 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.

**--show-encodings**

Show all known encodings and exit.

**-V, --version**

Show version information and exit.

**-w, --[no-]word-regexp**

Match only whole words.

**A.5 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.

### A.5.1 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

### A.5.2 Some common encodings

The following list shows some encodings that are suitable for option `--encoding`.

#### Latin-1, Western European

iso-8859-1, cp850, cp860, cp1252

#### Latin-2, Central European

iso-8859-2, cp852, cp1250

#### Latin-3, South European (Esperanto, Maltese)

iso-8859-3

#### Latin-4, North European (Baltics)

iso-8859-4

#### Cyrillics

iso-8859-5, cp855, cp866 (Ukrainian), cp1251

#### Arabic

iso-8859-6, cp864, cp1006 (Farsi), cp1256

#### Greek

iso-8859-7, cp737, cp1253

#### Hebrew

iso-8859-8, cp862, cp1255

Turkish

iso-8859-9, cp857, cp1254

Nordic

iso-8859-10, cp865, cp861 (Icelandic)

Thai

iso-8859-11, cp874

Baltic

iso-8859-13, cp775, cp1257

Celtic

iso-8859-14

Latin-9 (sometimes called Latin0)

iso-8859-15

Latin-10

iso-8859-16

## A.6 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.

## A.7 Caveats

The end tag `</typog-inspect>` sometimes gets placed too early in the output and the trace *seems* truncated. However, L<sup>A</sup>T<sub>E</sub>X reliably logs the requested the trace information, but the write operations for trace data and the code which is used to print the end tag are not synchronized.

## A.8 See also

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

## B $\epsilon$ -T<sub>E</sub>X: Breaking Paragraphs into Lines

This is an excerpt from the  $\epsilon$ -T<sub>E</sub>X manual [6], Sec. 3.8, »Breaking Paragraphs into Lines« that describes the `\lastlinefit` algorithm. Package `typog` warps `\lastlinefit` in environment `lastlinefitpar`, which was introduced in Sec. 3.10.5 on p. 33.

Traditional typesetting with lead type used to adjust (stretch or shrink) the interword spaces in the last line of a paragraph by the same amount as those in the preceding line. With T<sub>E</sub>X the last line is, however, usually typeset at its natural width due to infinitely stretchable `\parfillskip` glue.  $\epsilon$ -T<sub>E</sub>X allows interpolation between these two extremes by specifying a suitable value for `\lastlinefit`. For a value of 0 or less,  $\epsilon$ -T<sub>E</sub>X behaves as T<sub>E</sub>X, values from 1 to 1000 indicate a glue adjustment fraction  $f$  times 1000, values above 1000 are interpreted as  $f = 1$ .

The new algorithm is used only if

1. `\lastlinefit` is positive;
2. `\parfillskip` has infinite stretchability; and
3. the stretchability of `\leftskip` plus `\rightskip` is finite.<sup>41</sup>

Thus the last line of a paragraph would normally be typeset at its natural width and the stretchability of `\parfillskip` glue would be used to achieve the desired line width. The algorithm proceeds as usual, considering all possible sequences of feasible break points and accumulating demerits for the stretching or shrinking of lines as well as for visually incompatible lines. When a candidate for the last line has been reached, the following conditions are tested:

4. the previous line was not »infinitely bad« and was stretched with positive finite stretchability or was shrunk with positive shrinkability;
5. the last line has infinite stretchability entirely due to `\parfillskip` glue;
6. if the previous line was stretched or shrunk the last line has positive finite stretchability or shrinkability respectively.

If all three conditions are satisfied, a glue adjustment factor of  $f$  times that of the preceding line will be applied to the relevant stretch or shrink components of all glue nodes in the last line, and the corresponding demerits are computed. (The last line will, however, not be stretched beyond the desired line width.)

When all possible candidates for the last line of the paragraph have been examined, the one having fewest accumulated demerits is chosen. If  $\epsilon$ -T<sub>E</sub>X's modified algorithm was applied to that last line, the actual stretching or shrinking is achieved by suitably modifying the `\parfillskip` glue node.

<sup>41</sup> As usual for parameters influencing T<sub>E</sub>X's line-breaking algorithm, the values current at the end of the (partial) paragraph are used.

This paragraph benefits from being enclosed in a `covernextindentpar` environment.

All computations described so far are performed with machine-independent integer arithmetic. Note, however, that the actual stretching requires machine-dependent floating point arithmetic. Therefore, when a paragraph is interrupted by a displayed equation and the line preceding the display is subject to the adjustment just described, the display will in general be preceded by `\abovedisplayskip` and not by `\abovedisplayshortskip` glue.

Section 3.8 of the  $\epsilon$ -T<sub>E</sub>X manual closes with a description of the generalizations of `\clubpenalties`, `\displaywidowpenalties`, `\interlinepenalties`, and `\widowpenalties`. See also Sec. 3.14, »Vertically Partially-Tied Paragraphs«, p. 41 in this manual.

## C 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}
4          [2024/09/26 v0.5 TypoGraphic extensions]
5
6\RequirePackage{etoolbox}
7\RequirePackage{everyhook}
8\RequirePackage{xkeyval}
9

```

### Declarations of Lengths, Skips, etc.

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

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

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

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

\iftypog@debug Our switch for debug information.

```
12\newif\iftypog@debug
```

\typog@typeout Our information printer. Just adds a prefix so that we can tease apart the *log* later.

```
13\newcommand*{\typog@typeout}[1]{\typeout{typog: #1}}
14
```

\typog@typeout Our debug information printer.

```
15\newcommand*{\typog@debug@typeout}[1]
16          {\iftypog@debug\typog@typeout{#1}\fi}
17
```

typog@@iteration (*counter*) We want our own counter (currently for keeping track of iterations) that does not get trampled underfoot too easily.

```
18\newcounter{typog@@iteration}
19
```

\typog@trim@spaces Pull \tl\_trim\_spaces into the ›classic‹ namespace.

```
20\ExplSyntaxOn
21\let\typog@trim@spaces=\tl_trim_spaces:o
22\ExplSyntaxOff
23
```



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

```

24 \newcommand{\typog@register@pdfsubstitute}[1]{%
25   \AtBeginDocument{%
26     \ifdefined\pdfstringdefDisableCommands
27       \@ifpackageloaded{pdfcomment}
28         {\PackageWarningNoLine{typog}
29           {package pdfcomment loaded -
30             -\space
31               typog will not touch PDF in-
32               terface}}
33         {\pdfstringdefDisableCommands{#1}}%
34     \fi}}

```

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`.

`\iftypog@microtype@preloaded`

```

34 \newif\iftypog@microtype@preloaded
35

```

`\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.

```

36 \ifdefined\MT@MT
37   \typog@typeout{package microtype preloaded}%
38   \typog@microtype@preloadedtrue
39   \def\typog@require@preloaded@microtype{\relax}
40 \else
41   \typog@microtype@preloadedfalse
42   \def\typog@require@preloaded@microtype
43     {\PackageError{typog}%
44       {package microtype not (pre-)loaded}%
45       {package microtype must be loaded before pack-
46         age typog}}
47 \fi

```

`\iftypog@microtype@loaded`

```

48 \newif\iftypog@microtype@loaded
49

```

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

```

50 \AtBeginDocument{
51   \ifdefined\MT@MT
52     \typog@typeout{package microtype loaded}%
53     \typog@microtype@loadedtrue
54     \def\typog@require@microtype{\relax}

```

```

55 \else
56 \typog@microtype@loadedfalse
57 \def\typog@require@microtype
58 {\PackageError{typog}%
59 {package microtype not loaded}%
60 {require package microtype before package ty-
61   pog}}%
62 }
63

```

Our own state...

`\newmuskip\typog@config@mathitalicscorrection`

```

64 \newmuskip\typog@config@mathitalicscorrection
65

```

Actual `\labelitem<N>` corrections.

`\typog@adjust@labelitemi (dimen)`

```

66 \newdimen{\typog@adjust@labelitemi}

```

`\typog@adjust@labelitemii (dimen)`

```

67 \newdimen{\typog@adjust@labelitemii}

```

`\typog@adjust@labelitemiii (dimen)`

```

68 \newdimen{\typog@adjust@labelitemiii}

```

`\typog@adjust@labelitemiv (dimen)`

```

69 \newdimen{\typog@adjust@labelitemiv}

```

Configuration constants for `\labelitem<N>` corrections.

`\lowercase@labelitemi (dimen)`

```

70 \newdimen{\typog@adjust@lowercase@labelitemi}

```

`\lowercase@labelitemii (dimen)`

```

71 \newdimen{\typog@adjust@lowercase@labelitemii}

```

`\lowercase@labelitemiii (dimen)`

```

72 \newdimen{\typog@adjust@lowercase@labelitemiii}

```

`\lowercase@labelitemiv (dimen)`

```

73 \newdimen{\typog@adjust@lowercase@labelitemiv}

```

`\uppercase@labelitemi (dimen)`

```

74 \newdimen{\typog@adjust@uppercase@labelitemi}

```

```

ppercase@labelitemii (dimen)
75 \newdimen{\typog@adjust@uppercase@labelitemii}

ppercase@labelitemiii (dimen)
76 \newdimen{\typog@adjust@uppercase@labelitemiii}

ppercase@labelitemiv (dimen)
77 \newdimen{\typog@adjust@uppercase@labelitemiv}
78

    Other lengths...

nfig@textitalicscorrection
79 \newlength{\typog@config@textitalicscorrection}

\typog@config@ligaturekern
80 \newlength{\typog@config@ligaturekern}

\typog@config@lowerslash
81 \newlength{\typog@config@lowerslash}

og@config@raisecapitaldash
82 \newlength{\typog@config@raisecapitaldash}

fig@raisecapitalguillemets
83 \newlength{\typog@config@raisecapitalguillemets}

@config@raisecapitalhyphen
84 \newlength{\typog@config@raisecapitalhyphen}

g@config@raisecapitaltimes
85 \newlength{\typog@config@raisecapitaltimes}

pog@config@raiseguillemets
86 \newlength{\typog@config@raiseguillemets}

pog@config@raisefiguredash
87 \newlength{\typog@config@raisefiguredash}

\typog@config@slashkern
88 \newlength{\typog@config@slashkern}

\typog@config@breakpenalty
89 \newcommand*{\typog@config@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.
90 \newlength{\typog@dim@unit}
91 \setlength{\typog@dim@unit}{.001em}

```

```

g@config@trackingttspacing
    92 \newcommand*{\typog@config@trackingttspacing}{300, 90, 60}

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

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

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

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

\typog@shrink@ii
    97 \newcommand*{\typog@shrink@ii}{}

\typog@shrink@iii
    98 \newcommand*{\typog@shrink@iii}{}

\typog@default@stretch@i The default configuration for stretch values.
    99 \newcommand*{\typog@default@stretch@i}{5}

\typog@default@stretch@ii
    100 \newcommand*{\typog@default@stretch@ii}{10}

\typog@default@stretch@iii
    101 \newcommand*{\typog@default@stretch@iii}{20}

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

\typog@stretch@ii
    103 \newcommand*{\typog@stretch@ii}{}

\typog@stretch@iii
    104 \newcommand*{\typog@stretch@iii}{}

```

## C.1 Setup and Reconfiguration

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

```

105 \NewDocumentEnvironment{typogsetup}{m}
106   {\def\typog@@arg{#1}%
107    \ifx\typog@@arg\empty
108      \typog@initialize@options
109    \else
110      \setkeys{typog}{#1}%
111    \fi
112    \ignorespaces}
113   {\ignorespacesafterend}

```

`\typogget` Access the package's configuration (name-)space.

```

114 \NewDocumentCommand{\typogget}{m}
115     {\csname typog@config@#1\endcsname}
116

```

`\typoggetnth` Access the  $n^{\text{th}}$  element of a comma-separated, list-like configuration item's value.

```

117 \ExplSyntaxOn
118 \cs_generate_variant:Nn \seq_set_split:Nnn {Nne}
119 \cs_new:Npn \typog_get_nth_csname:cnn #1#2#3
120   {
121     \seq_set_split:Nne \l_tmpa_seq {,} {\cs:w typog@config@#2 \cs_end:}
122     \cs_gset:cpn {#1} {\seq_item:Nn \l_tmpa_seq {#3}}
123   }
124 \cs_new:Npn \typog_get_nth_dimen:nnn #1#2#3
125   {
126     \seq_set_split:Nne \l_tmpa_seq {,} {\cs:w typog@config@#2 \cs_end:}
127     \dim_set:Nn {#1} {\seq_item:Nn \l_tmpa_seq {#3}}
128   }
129 \NewDocumentCommand{\typoggetnth}{m m m}{
130   \token_if_dim_register:NTF {#1}
131   {
132     \typog_get_nth_dimen:nnn {#1} {#2} {#3}
133   }
134   {
135     \typog_get_nth_csname:cnn {#1} {#2} {#3}
136   }
137 }
138 \ExplSyntaxOff
139

```

## C.2 Information

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

```

140 \ExplSyntaxOn
141 \newcommand*{\typog@round@dim@to@tenths}[1]
142   {\fp_to_decimal:n {round(10 * \dim_to_fp:n{#1} / 1\p@) / 10}}
143 \ExplSyntaxOff
144
```

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

```

145 \newcommand*{\typog@formatsizeinfo}[3]
146   {#1#3\kernedslash #2#3}
147
```

`\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.

```

148 \NewDocumentCommand{\fontsizeinfo}{s m}
149   {\global\expandafter\edef\csname typog@fontsize@#2\endcsname
150     {\typog@round@dim@to@tenths{\fontdimen6\font}}}%
151   \global\expandafter\edef\csname typog@linespacing@#2\endcsname
152     {\typog@round@dim@to@tenths{\baselineskip}}}%
153   \protected\expandafter\gdef\csname #2\endcsname
154     {\@ifnextchar*\typog@formatsizeinfo
155       {\csname typog@fontsize@#2\endcsname}%
156       {\csname typog@linespacing@#2\endcsname}%
157       }% no unit
158       \ignorespaces % eat spaces after star
159       \@gobble} % consume the star itself
160   {\typog@formatsizeinfo
161     {\csname typog@fontsize@#2\endcsname}%
162     {\csname typog@linespacing@#2\endcsname}%
163     {\,pt}% decorative unit 'pt'
164   }}
165
```

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

```

166 \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.

```

167 \newcounter{typog@inspect@count}
168
```

`typoginspect` (*env.*)

```

169 \define@key[typog]{typoginspect}{tracingboxes}[\maxdimen]%
170     {\def\typog@@typoginspect@tracingboxes{#1}}
171 \NewDocumentEnvironment{typoginspect}{0}{ m}
172   {\def\typog@@typoginspect@tracingboxes{\m@ne}%
173     \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.

```

174   \edef\typog@@arg{#2}%
175   \ifx\typog@@arg\empty
176     \stepcounter{typog@inspect@count}%
177     \edef\typog@@id{\typog@default@inspect@id@prefix
178                   \arabic{typog@inspect@count}}%
179   \else
180     \edef\typog@@id{\typog@trim@spaces{\typog@@arg}}%
181   \fi
182   \typeout{<typog-inspect\space
183           id="\typog@@id"\space
184           job="\jobname"\space
185           line="\the\inputlineno"\space
186           page="\the\value{page}">}%

```

Set both badness thresholds to absurdly low values as to activate T<sub>E</sub>X's reports.

```

187   \hbadness=\m@ne
188   \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.

```

189   \tracingnone
190   \tracingpages=\@ne
191   \tracingparagraphs=\@ne
192   \showboxbreadth=\typog@@typoginspect@tracingboxes
193   \showboxdepth=\typog@@typoginspect@tracingboxes}
194   {\typeout{</typog-inspect>}}%
195   \ignorespacesafterend}

```

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

```

196 \NewDocumentEnvironment{typoginspectpar}{m}
197   {\typoginspect{#1}}
198   {\par\endtypoginspect}
199

```

### C.3 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`.

```
200 \newcommand*{\typog@allowhyphenation}
201   {\ifvmode
202     \relax
203   \else
204     \nobreak
205     \hskip\z@skip
206   \fi}
207
```

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

```
208 \unless\ifdefined\allowhyphenation
209   \let\allowhyphenation=\typog@allowhyphenation
210 \fi
211
```

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

```
212 \NewDocumentCommand{\breakpoint}{s}
213   {\discretionary{}{}{}}%
214   \IfBooleanTF{#1}%
215     {\ignorespaces}%
216     {\typog@allowhyphenation}}
217
```

PDF-substitute definition

```
218 \typog@register@pdfsubstitute{
219   \def\breakpoint#1{\if*\detokenize{#1}\ignorespaces\fi}%
220 }
221
```

`hyphenmin (env.)` No trickery here. – We use the mandatory argument for the value of `\lefthyphenmin` if the optional argument has been omitted.

```
222 \NewDocumentEnvironment{hyphenmin}{o m}
223   {\lefthyphenmin=\IfNoValueTF{#1}{#2}{#1}%
224   \righthyphenmin=#2}
225 {}
226
```



## C.4 Disable/Break Ligatures

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

```
227 \newcommand*{\typog@hyphen}{\char‘-’}
228
```

`\nolig`

```
229 \NewDocumentCommand{\nolig}{s o}
230   {\dimen0=\IfNoValueTF{#2}{\typog@config@ligaturekern}{#2\typog@dim@unit}%
231    \IfBooleanTF{#1}%
232     {\kern\dimen0\ignorespaces}%
233     {\discretionary{\typog@hyphen}{\kern\dimen0}%
234      \typog@allowhyphenation
235      \IfNoValueF{#2}{\ignorespaces}}}
236
```

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

```
237 \typog@register@pdfsubstitute{
238   \RenewExpandableDocumentCommand{\nolig}{s o m}{%
239     \ifx\typog@TYPOG#3\typog@TYPOG
240       \relax
241     \else
242       \ifx\relax#3\relax
243         \relax
244       \else
245         \PackageError{typog}
246                               {Missing third argument of \nolig}
247                               {Append empty group or \relax after macro in-
248                                vocation}
248       \fi
249     \fi}
250 }
251
```

## C.5 Manual Italic Correction

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

```
252 \newcommand*{\typog@itcorr@text@unconditional}[1]
253   {\kern#1\typog@config@textitalicscorrection}
```

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

```
254 \newcommand*{\typog@itcorr@text}[1]
255   {\def\typog@@strength{#1}%
256    \dimen0=\fontdimen1\font
257    \ifdim\dimen0=\z@
258      \typog@itcorr@text@unconditional{\typog@@strength}%
259    \else
260      \kern\typog@@strength\dimen0
261    \fi}
```

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

```
262 \newcommand*{\typog@itcorr@math}[1]
263   {\mkern#1\typog@config@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.

```
264 \NewDocumentCommand{\itcorr}{s m}
265   {\ifmmode
266     \typog@itcorr@math{#2}%
267   \else
268     \IfBooleanTF{#1}%
269       {\typog@itcorr@text{#2}}%
270       {\typog@itcorr@text@unconditional{#2}}%
271   \fi}
```

PDF-substitute definition

```
272 \typog@register@pdfsubstitute{
273   \RenewExpandableDocumentCommand{\itcorr}{s m}{}
274 }
275
```

## C.6 Apply Extra Kerning

### Slash

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

```
276 \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’`.

```
277 \NewDocumentCommand{\kernedslash}{s}
278   {\hspace*{\typog@config@slashkern}%
279   \raisebox{-\typog@config@lowerslash}{\typog@forwardslash}%
280   \IfBooleanTF{#1}%
281     {\hspace*{\typog@config@slashkern}\ignorespaces}%
282     {\typog@breakpoint
283     \typog@allowhyphenation
284     \hspace*{\typog@config@slashkern}}}
```

PDF-substitute definition

```
285 \typog@register@pdfsubstitute{
286   \def\kernedslash#1{\if* \detokenize{#1}/\ignorespaces\else/#1\fi}%
287 }
288
```

**Hyphen****\kernedhyphen**

```

289 \NewDocumentCommand{\kernedhyphen}{s O{0} m m}
290 {
291   \mspace{\muexpr(#3 mu) * 18 / 1000}%
292   \raisebox{#2\typog@dim@unit}{\m@th\mathord{-}}%
293   \mspace{\muexpr(#4 mu) * 18 / 1000}%
294   \else
295     \def\typog@@auto{*}%
296     \def\typog@@optarg{#2}%
297     \hspace*{#3\typog@dim@unit}%
298     \raisebox{\ifx\typog@@optarg\typog@@auto
299               \typog@config@raisecapitalhyphen
300               \else
301                 \typog@@optarg\typog@dim@unit
302               \fi}{\typog@hyphen}%
303     \hspace{#4\typog@dim@unit}%
304     \IfBooleanT{#1}{\nobreak}%
305   \fi}

PDF-substitute definition

306 \typog@register@pdfsubstitute{
307   \RenewExpandableDocumentCommand{\kernedhyphen}{s o m m}{-}
308 }

```

One-argument shorthands.

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

```

309 \NewDocumentCommand{\leftkernedhyphen}{s O{0} m}
310 {
311   \IfBooleanTF{#1}%
312   {\kernedhyphen*{#2}{#3}{0}\ignorespaces}%
313   {\kernedhyphen[#2]{#3}{0}}}

PDF-substitute definition

313 \typog@register@pdfsubstitute{
314   \RenewExpandableDocumentCommand{\leftkernedhyphen}{s o m}{-}
315 }
316

```

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

```

317 \NewDocumentCommand{\rightkernedhyphen}{s O{0} m}
318 {
319   \IfBooleanTF{#1}%
320   {\kernedhyphen*{#2}{0}{#3}\ignorespaces}%
321   {\kernedhyphen[#2]{0}{#3}}}

PDF-substitute definition

321 \typog@register@pdfsubstitute{
322   \RenewExpandableDocumentCommand{\rightkernedhyphen}{s o m}{-}
323 }
324

```

## C.7 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.

```
325 \newcommand*{\typog@breakpoint}
326   {\penalty\typog@config@breakpenalty}
```

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

```
327 \NewDocumentCommand{\capitalhyphen}{s}
328   {\raisebox{\typog@config@raisecapitalhyphen}{\typog@hyphen}%
329     \IfBooleanTF{#1}%
330       {\ignorespaces}%
331       {\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.

```
332 \typog@register@pdfsubstitute{
333   \def\capitalhyphen#1{%
334     \if*\detokenize{#1}%
335       -\ignorespaces
336     \else
337       -#1%
338     \fi}
339 }
340
```

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

```
341 \NewDocumentCommand{\capitalendash}{s}
342   {\raisebox{\typog@config@raisecapitaldash}{\textendash}%
343     \IfBooleanTF{#1}%
344       {\ignorespaces}%
345       {\typog@breakpoint\typog@allowhyphenation}}
346 \let\capitaldash=\capitalendash
```

PDF-substitute definition

```
347 \typog@register@pdfsubstitute{
348   \def\capitalendash#1{%
349     \if*\detokenize{#1}%
350       \textendash\ignorespaces
351     \else
352       \textendash#1%
353     \fi}
354 \let\capitaldash=\capitalendash
355 }
356
```

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

```
357 \NewDocumentCommand{\capitalemdash}{s}
358   {\raisebox{\typog@config@raisecapitaldash}{\textemdash}%
359     \IfBooleanTF{#1}%
360       {\ignorespaces}%
361       {\typog@breakpoint\typog@allowhyphenation}}}
```

PDF-substitute definition

```
362 \typog@register@pdfsubstitute{
363   \def\capitalemdash#1{%
364     \if*\detokenize{#1}%
365       \textemdash\ignorespaces
366     \else
367       \textemdash#1%
368     \fi}
369 }
370
```

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

```
371 \NewDocumentCommand{\figuredash}{s}
372   {\raisebox{\typog@config@raisefiguredash}{\textendash}%
373     \IfBooleanTF{#1}%
374       {\ignorespaces}%
375       {\typog@breakpoint\typog@allowhyphenation}}}
```

PDF-substitute definition

```
376 \typog@register@pdfsubstitute{\let\figuredash=\capitaldash}
377
```

`\capitaltimes`

```
378 \NewDocumentCommand{\capitaltimes}{}
379   {\ifmmode
380     \mathbin{\raisebox{\typog@config@raisecapitaltimes}{\math@th\times}}}%
381   \else
382     \raisebox{\typog@config@raisecapitaltimes}{\textttimes}%
383   \fi}
```

PDF-substitute definition

```
384 \typog@register@pdfsubstitute{
385   \RenewExpandableDocumentCommand{\capitaltimes}{}{\textttimes}
386 }
387
```

`\singleguillemetleft`

```
388 \NewDocumentCommand{\singleguillemetleft}{}
389   {\typog@allowhyphenation
390     \raisebox{\typog@config@raiseguillemets}{\guilsinglleft}}
```

PDF-substitute definition

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

`\singleguillemetright`

```
392 \NewDocumentCommand{\singleguillemetright}{}
393   {\raisebox{\typog@config@raiseguilletts}{\guilsinglright}%
394     \typog@allowhyphenation}
    PDF-substitute definition
395 \typog@register@pdfsubstitute{\let\singleguillemetright\guilsinglright}
```

`\doubleguillemetleft`

```
396 \NewDocumentCommand{\doubleguillemetleft}{}
397   {\typog@allowhyphenation
398     \raisebox{\typog@config@raiseguilletts}{\guillemotleft}}
    PDF-substitute definition
399 \typog@register@pdfsubstitute{\let\doubleguillemetleft\guillemotleft}
```

`\doubleguillemetright`

```
400 \NewDocumentCommand{\doubleguillemetright}{}
401   {\raisebox{\typog@config@raiseguilletts}{\guillemotright}%
402     \typog@allowhyphenation}
    PDF-substitute definition
403 \typog@register@pdfsubstitute{\let\doubleguillemetright\guillemotright}
```

`\Singleguillemetleft`

```
404 \NewDocumentCommand{\Singleguillemetleft}{}
405   {\typog@allowhyphenation
406     \raisebox{\typog@config@raisecapitalguilletts}{\guilsinglleft}}
    PDF-substitute definition
407 \typog@register@pdfsubstitute{\let\Singleguillemetleft\guilsinglleft}
```

`\Singleguillemetright`

```
408 \NewDocumentCommand{\Singleguillemetright}{}
409   {\raisebox{\typog@config@raisecapitalguilletts}{\guilsinglright}%
410     \typog@allowhyphenation}
    PDF-substitute definition
411 \typog@register@pdfsubstitute{\let\Singleguillemetright\guilsinglright}
```

`\Doubleguillemetleft`

```
412 \NewDocumentCommand{\Doubleguillemetleft}{}
413   {\typog@allowhyphenation
414     \raisebox{\typog@config@raisecapitalguilletts}{\guillemotleft}}
    PDF-substitute definition
415 \typog@register@pdfsubstitute{\let\Doubleguillemetleft\guillemotleft}
```

`\Doubleguillemetright`

```
416 \NewDocumentCommand{\Doubleguillemetright}{}
417   {\raisebox{\typog@config@raisecapitalguilletts}{\guillemotright}%
418     \typog@allowhyphenation}
```

## PDF-substitute definition

```
419 \typog@register@pdfsubstitute{\let\Doubleguillemetright\guillemotright}
420
```

We need three lengths for three (pairs of) inverted exclamation marks and inverted question marks.

```
onfig@raiseinvertedmarks@i
```

```
421 \newlength{\typog@config@raiseinvertedmarks@i}
```

```
nfig@raiseinvertedmarks@ii
```

```
422 \newlength{\typog@config@raiseinvertedmarks@ii}
```

```
fig@raiseinvertedmarks@iii
```

```
423 \newlength{\typog@config@raiseinvertedmarks@iii}
```

And the three (pairs of) inverted exclamation marks and inverted question marks themselves. Configurable.

```
inverted@exclamationmark@i
```

```
424 \newcommand*{\typog@inverted@exclamationmark@i}{\textexclamdown}
```

```
og@inverted@questionmark@i
```

```
425 \newcommand*{\typog@inverted@questionmark@i}{\textquestiondown}
```

```
nverted@exclamationmark@ii
```

```
426 \newcommand*{\typog@inverted@exclamationmark@ii}{\textexclamdown}
```

```
g@inverted@questionmark@ii
```

```
427 \newcommand*{\typog@inverted@questionmark@ii}{\textquestiondown}
```

```
verted@exclamationmark@iii
```

```
428 \newcommand*{\typog@inverted@exclamationmark@iii}{\textexclamdown}
```

```
@inverted@questionmark@iii
```

```
429 \newcommand*{\typog@inverted@questionmark@iii}{\textquestiondown}
```

```
\typog@raise@inverted@mark
```

The generic ›raise‹-macro handles all interesting cases. The first argument selects the mark type and the second the associated raise-amount.

If the raise-amount is exactly zero we shift-up the horizontal box with the mark to zero its depth.

```
430 \newcommand*{\typog@raise@inverted@mark}[2]
431 {\letcs{\@tempa}{\typog@inverted@#1@romannumeral#2}%
432 \letcs{\@tempb}{\typog@config@raiseinvertedmarks@romannumeral#2}%
433 \ifdim\@tempb=\z@
434 \setbox0=\hbox{\@tempa}%
435 \dimen0=\dp0
436 \else
437 \dimen0=\@tempb
438 \fi
439 \raisebox{\dimen0}{\@tempa}}
```

The user-side macros are almost trivial now.

capitalinvertedexclamationmark

```

440 \NewDocumentCommand{\capitalinvertedexclamationmark}{m}
441   {\typog@raise@inverted@mark{exclamationmark}{#1}%
442    \typog@allowhyphenation
443    \ignorespaces}

PDF-substitute definition

444 \typog@register@pdfsubstitute{
445   \def\capitalinvertedexclamationmark#1{%
446     \csname typog@inverted@exclamationmark@\romannumeral#1\endcsname
447   }
448 }
```

capitalinvertedquestionmark

```

449 \NewDocumentCommand{\capitalinvertedquestionmark}{m}
450   {\typog@raise@inverted@mark{questionmark}{#1}%
451    \typog@allowhyphenation
452    \ignorespaces}

PDF-substitute definition

453 \typog@register@pdfsubstitute{
454   \def\capitalinvertedquestionmark#1{%
455     \csname typog@inverted@questionmark@\romannumeral#1\endcsname
456   }
457 }
```

## C.8 Vertically Adjusted Label Items

uppercase@adjust@labelitem Handle all possible requests for uppercase label item correction. Patch itemize environments.

```

458 \newcommand*{\@typog@uppercase@adjust@labelitem}[1]
459   {\@typog@maybe@patch@itemize
460    \ifstrequal{#1}{*}
461      {\setlength{\typog@adjust@labelitemi}
462       {\typog@adjust@uppercase@labelitemi}
463       \setlength{\typog@adjust@labelitemii}
464       {\typog@adjust@uppercase@labelitemii}
465       \setlength{\typog@adjust@labelitemiii}
466       {\typog@adjust@uppercase@labelitemiii}
467       \setlength{\typog@adjust@labelitemiv}
468       {\typog@adjust@uppercase@labelitemiv}}
469     {\ifcase #1% 0
470      \relax % outside of any itemize environment
471      \or % 1
472       \setlength{\typog@adjust@labelitemi}
473       {\typog@adjust@uppercase@labelitemi}
474      \or % 2
475       \setlength{\typog@adjust@labelitemii}
476       {\typog@adjust@uppercase@labelitemii}
477      \or % 3
```



```

478         \setlength{\typog@adjust@labelitemiii}
479             {\typog@adjust@uppercase@labelitemiii}
480     \or % 4
481         \setlength{\typog@adjust@labelitemiv}
482             {\typog@adjust@uppercase@labelitemiv}
483     \else
484         \PackageError{typog}
485             {Itemize level out of range}
486             {Valid levels are 1, 2, 3, 4, and *}
487     \fi}}
488

```

`\lowercase@adjust@labelitem` Handle all possible requests for lowercase label item correction. Patch `itemize` environments.

```

489 \newcommand*{\@typog@lowercase@adjust@labelitem}[1]
490 { \@typog@maybe@patch@itemize
491   \ifstrequal{#1}{*}
492     {\setlength{\typog@adjust@labelitemi}
493       {\typog@adjust@lowercase@labelitemi}
494       \setlength{\typog@adjust@labelitemii}
495         {\typog@adjust@lowercase@labelitemii}
496       \setlength{\typog@adjust@labelitemiii}
497         {\typog@adjust@lowercase@labelitemiii}
498       \setlength{\typog@adjust@labelitemiv}
499         {\typog@adjust@lowercase@labelitemiv}}
500   {\ifcase #1% 0
501     \relax % outside of any itemize environment
502   \or % 1
503     \setlength{\typog@adjust@labelitemi}
504       {\typog@adjust@lowercase@labelitemi}
505   \or % 2
506     \setlength{\typog@adjust@labelitemii}
507       {\typog@adjust@lowercase@labelitemii}
508   \or % 3
509     \setlength{\typog@adjust@labelitemiii}
510       {\typog@adjust@lowercase@labelitemiii}
511   \or % 4
512     \setlength{\typog@adjust@labelitemiv}
513       {\typog@adjust@lowercase@labelitemiv}
514   \else
515     \PackageError{typog}
516       {Itemize level out of range}
517       {Valid levels are 1, 2, 3, 4, and *}
518   \fi}}
519

```

`\@typog@noadjust@labelitem` Neutralize all label item corrections. This function *does not* request patching any `itemize` environment!

```

520 \newcommand*{\@typog@noadjust@labelitem}[1]
521 { \ifstrequal{#1}{*}
522   {\setlength{\typog@adjust@labelitemi}{\z@}

```

```

523         \setlength{\typog@adjust@labelitemii}{\z@}
524         \setlength{\typog@adjust@labelitemiii}{\z@}
525         \setlength{\typog@adjust@labelitemiv}{\z@}
526     {\ifcase #1% 0
527         \relax % outside of any itemize environment
528         \or % 1
529         \setlength{\typog@adjust@labelitemi}{\z@}
530         \or % 2
531         \setlength{\typog@adjust@labelitemii}{\z@}
532         \or % 3
533         \setlength{\typog@adjust@labelitemiii}{\z@}
534         \or % 4
535         \setlength{\typog@adjust@labelitemiv}{\z@}
536     \else
537         \PackageError{typog}
538             {Itemize level out of range}
539             {Valid levels are 1, 2, 3, 4, and *}
540     \fi}}
541

```

`\uppercaseadjustlabelitems` User macro that handles lists and the treats the empty list specially. We wrap the code into `\AfterPreamble` because it may be called in the document's preamble where we don't know whether package `enumitem` already has been loaded and we can patch its variant of `itemize`.

```

542 \NewDocumentCommand{\uppercaseadjustlabelitems}{m}
543 {
544     \AfterPreamble{
545         \ifblank{#1}
546             {\@typog@uppercase@adjust@labelitem{\@itemdepth}}
547             {\forcsvlist{\@typog@uppercase@adjust@labelitem}{#1}}%
548         \ignorespaces}
549

```

`\lowercaseadjustlabelitems` User macro that handles lists and the treats the empty list specially.

```

549 \NewDocumentCommand{\lowercaseadjustlabelitems}{m}
550 {
551     \AfterPreamble{
552         \ifblank{#1}
553             {\@typog@lowercase@adjust@labelitem{\@itemdepth}}
554             {\forcsvlist{\@typog@lowercase@adjust@labelitem}{#1}}%
555         \ignorespaces}
556

```

`\noadjustlabelitems` User macro that handles lists and the treats the empty list specially.

```

556 \NewDocumentCommand{\noadjustlabelitems}{m}
557 {
558     \ifblank{#1}
559         {\@typog@noadjust@labelitem{\@itemdepth}}
560         {\forcsvlist{\@typog@noadjust@labelitem}{#1}}%
561     \ignorespaces}
562

```

Now we get to the dirty part. All the above definitions do not get called until we hack the existing `itemize`-environments, either the one of plain `LATEX` or the one modified by package `enumitem`.

Here comes the result of `latexdef -c article -s itemize`, which was used to derive the patch code:

```
% \def\itemize{%
%   \ifnum \@itemdepth > \thr@@
%     \@toodeep
%   \else
%     \advance\@itemdepth\@ne
%     \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
%     \expandafter
%     \list
%     \csname\@itemitem\endcsname
%     {\def\makelabel##1{\hss\llap{##1}}}%
%   \fi}
```

`\@typog@itemize@patch` This is the additional code we inject into plain L<sup>A</sup>T<sub>E</sub>X's or package `enumitem`'s `\itemize`.

```
562 \newcommand*{\@typog@itemize@patch}
```

Save the original definition of `\@itemitem` for chain-calling it later on.

```
563 {\letcs{\@typog@old@itemitem}{\@itemitem}}
```

Sneak in our own macro's name.

```
564 \edef\@itemitem{@typog@labelitem\romannumeral\the\@itemdepth}
```

Redefine under the original macro's name so that our code gets called and the old code (`\@typog@old@itemitem`) is expanded.

```
565 \expandafter\def\csname\@itemitem\endcsname
```

```
566   {\raisebox{\csname typog@adjust@labelitem\romannumeral\the\@itemdepth\endcsname
```

```
567     {\@typog@old@itemitem}}}
```

```
568
```

If package `enumitem` has been loaded, we use the *same* patch. Here comes the result of `latexdef -c article -p enumitem -s enit@itemize@i` that explains, why no change is required:

```
% \def\enit@itemize@i#1#2#3#4{%
%   \ifnum #1 > #3 \relax
%     \enit@toodeep
%   \else
%     \enit@prelist\@ne{#1}{#2}%
%     \edef\@itemitem{label#2\romannumeral#1}%
%     \expandafter
%     \enit@list
%     \csname\@itemitem\endcsname
%     {\let\enit@calc\z@
%     \def\makelabel##1{\enit@align{\enit@format{##1}}}%
%     \enit@preset{#2}{#1}{#4}%
%     \enit@calcleft
%     \enit@before
```

```
%          \enit@negwidth}%
%          \enit@keyfirst
%          \fi}
```

\@typog@patch@itemize Unconditionally apply the patches that are just *single* macro calls to disturb the original macros as little as possible. If we detect enumitem to be present we modify its definition of `itemize` otherwise we wrestle L<sup>A</sup>T<sub>E</sub>X's macro.

```
569 \newcommand*{\@typog@patch@itemize}
570   {\ifdefined\enit@itemize@i
571     \patchcmd{\enit@itemize@i}
572               {\expandafter}
573               {\@typog@itemize@patch\expandafter}
574               {\typog@debug@typeout{patching enumitem \string\enit@itemize@i\space
    ceded}}
575               {\PackageError{typog}
576                           {Patching enumitem macro \string\enit@itemize@i\space
577                           {}}
578   }
579   \else
580     \patchcmd{\itemize}
581               {\expandafter}
582               {\@typog@itemize@patch\expandafter}
583               {\typog@debug@typeout{patching \string\itemize\space suc-
    ceded}}
584               {\PackageError{typog}
585                           {Patching plain LaTeX macro \string\itemize\space fai
586                           {}}
587   \fi}
```

\@typog@maybe@patch@itemize Apply the patches only once.

```
588 \newbool{\@typog@itemize@has@been@patched}
589 \newcommand*{\@typog@maybe@patch@itemize}
590   {\ifbool{\@typog@itemize@has@been@patched}
591     {\relax}
592     {\@typog@patch@itemize
593     \booltrue{\@typog@itemize@has@been@patched}}}
594
```

Here come our convenience macros to simplify an accurate setup of the label adjustments.

\typog@hairline@width Line width of the horizontal reference lines in our convenience macros.

```
595 \newcommand*{\typog@hairline@width}{.125pt}
```

\typogadjuststairsfor The arguments are: #1: *<scale-factor>*, #2: *<step-size>*, #3: *<number-of-steps>*, #4: *<sample>*, and #5: `\labelitem<N>`.

Generate an ascending stairs of argument #5.

```
596 \newcommand*{\typogadjuststairsfor}[5]
```

Store (half of) the space between two samples in `\dimen0`.

```
597   {\dimen0=1pt%
```

Load the *<number-of-steps>* and ensure that it is odd.

```
598 \count0=#3\relax
599 \unless\ifodd\count0
600   \advance\count0 by 1%
601 \fi
```

Set the iteration counter.

```
602 \setcounter{typog@@iteration}{1}%
```

Put the *<sample>* into a box so that we can measure it with `\ht`.

```
603 \setbox0=\hbox{#4}%
```

Box 1 is the accumulator for the raised samples.

```
604 \setbox1=\hbox{}%
```

Build the stairs.

```
605 \loop
606   \ifnum\thetypog@@iteration=\numexpr\count0 / 2\relax
607     \dimen1=2\dimen0
608   \else
609     \dimen1=\dimen0
610   \fi
611   \dimen2=\dimexpr#2 * (\thetypog@@iteration - \count0 / 2)\re-
  lax
612   \setbox1=\hbox{\unhbox1\raisebox{\dimen2}{\kern\dimen1 #5\kern\dimen1}}%
613   \addtocounter{typog@@iteration}{1}%
614   \unless\ifnum\thetypog@@iteration>\count0
615 \repeat
```

Merge the stairs with a hairline at #1 times the height of *<sample>*. Answer just a single box.

```
616 \mbox{\rlap{\raisebox{\fpeval{#1}\ht0}{\rule{\wd1}{\typog@hairline@width}}}\bo
617
```

`\typogadjuststairs` The arguments are: #1: *<scale-factor>*, #2: *<step-size>*, #3: *<number-of-steps>*, and #4: *<sample>*.

```
618 \NewDocumentCommand{\typogadjuststairs}{0{.5} m m m}
619 {\begingroup
620   \unless\ifdim #2>\z@
621     \PackageError{typog}
622       {\string\typogadjuststairs\space non-positive step-
  size}
623       {step-size must be a positive dimension}%
624   \fi
625   \ifnum #3<1
626     \PackageError{typog}
627       {\string\typogadjuststairs\space too few number-
  of-steps}
628       {number-of-steps must at least be 1}%
629   \fi
630   \ifblank{#4}
```

```

631         {\PackageError{typog}
632             {sample must not be empty}
633             {supply either some uppercase or some low-
        ercase letters}}
634         {}%
635     \def\arraystretch{1}%
636     \begin{tabular}{@{}c@{}}
637         \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemi} \\\
638         \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemii} \\\
639         \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemiii} \\\
640         \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemiv}
641     \end{tabular}
642 \endgroup}
643

```

`ercase@adjusted@labelitems` Return all four labelitems in a horizontal box after they have been adjusted with the uppercase-constants set.

```

644 \newcommand*{\typog@uppercase@adjusted@labelitems}
645 { \hbox{\raisebox{\typog@adjust@uppercase@labelitemi}{\labelitemi}%
646         \raisebox{\typog@adjust@uppercase@labelitemii}{\labelitemii}%
647         \raisebox{\typog@adjust@uppercase@labelitemiii}{\labelitemiii}%
648         \raisebox{\typog@adjust@uppercase@labelitemiv}{\labelitemiv}}}

```

`\typoguppercaseadjustcheck` We stuff the user's sample text into a box only to measure its height. We typeset all four labels and draw a hairline at half the height of the sample right through it.

```

649 \NewDocumentCommand{\typoguppercaseadjustcheck}{0{.5} m}
650 { \setbox0=\hbox{#2}%
651   \setbox1=\typog@uppercase@adjusted@labelitems
652   \mbox{\rlap{\raisebox{\fpeval{#1}\ht0}
653             {\rule{\wd1}{\typog@hairline@width}}}%
654         \box1}}
655

```

`ercase@adjusted@labelitems` Return all four labelitems in a horizontal box after they have been adjusted with the lowercase-constants set.

```

656 \newcommand*{\typog@lowercase@adjusted@labelitems}
657 { \hbox{\raisebox{\typog@adjust@lowercase@labelitemi}{\labelitemi}%
658         \raisebox{\typog@adjust@lowercase@labelitemii}{\labelitemii}%
659         \raisebox{\typog@adjust@lowercase@labelitemiii}{\labelitemiii}%
660         \raisebox{\typog@adjust@lowercase@labelitemiv}{\labelitemiv}}}

```

`\typoglowercaseadjustcheck` Same code as `\typoguppercaseadjustcheck` for lowercase.

```

661 \NewDocumentCommand{\typoglowercaseadjustcheck}{0{.5} m}
662 { \setbox0=\hbox{#2}%
663   \setbox1=\typog@lowercase@adjusted@labelitems
664   \mbox{\rlap{\raisebox{\fpeval{#1}\ht0}
665             {\rule{\wd1}{\typog@hairline@width}}}%
666         \box1}}
667

```

## C.9 Align Last Line of a Paragraph

The code of environment `lastlinerraggedleftpar` has been inspired by macro `\lastlinerraggedleft` [36, Sec. 2].

`lastlinerraggedleftpar (env.)`

```
668 \NewDocumentEnvironment{lastlinerraggedleftpar}{}
669   {\lastlinefit=0%
670    \setlength{\leftskip}{\z@ \@plus 1fil}%
671    \setlength{\rightskip}{-\leftskip}%
672    \setlength{\parfillskip}{\leftskip}}
673   {\par}
```

`lastlineflushrightpar (env.)` Define `lastlineflushrightpar` as an alias of `lastlinerraggedleftpar`.

```
674 \let\lastlineflushrightpar=\lastlinerraggedleftpar
675 \let\endlastlineflushrightpar=\endlastlinerraggedleftpar
676
```

`lastlinecenteredpar (env.)` The code of environment `lastlinecenteredpar` has been inspired by *Tex By Topic* [11, Sec. 18.3.1].

```
677 \NewDocumentEnvironment{lastlinecenteredpar}{}
678   {\lastlinefit=0%
679    \setlength{\leftskip}{\z@ \@plus .5fil}%
680    \setlength{\rightskip}{-\leftskip}%
681    \setlength{\parfillskip}{\z@ \@plus 1fil}}
682   {\par}
683
```

## C.10 Fill Last Line of a Paragraph

`shortenpar (env.)`

```
684 \NewDocumentEnvironment{shortenpar}{}
685   {\advance\looseness by -1
686    \ifnum\tracingparagraphs>0
687      \typeout{@ looseness \the\looseness}%
688    \fi}
689   {\par}
690
```

`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.

```
691 \NewDocumentEnvironment{prolongpar}{}
692   {\finalhyphendemerits=100000001
693    \advance\looseness by 1
694    \ifnum\tracingparagraphs>0
695      \typeout{@ looseness \the\looseness}%
696    \fi}
697   {\par}
698
```

`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`.

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

`ndindentpar@nonzero@parindent`

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

`covernextindentpar (env.)`

```
701 \NewDocumentEnvironment{covernextindentpar}{o}
702 {\IfNoValueTF{#1}
703   {\ifdim\parindent=\z@
704     \dimen0=\dimexpr\linewidth - \typog@covernextindentpar@zero@parindent
705     \else
706     \dimen0=\dimexpr\linewidth - \typog@covernextindentpar@nonzero@parindent
707     \fi}
708   {\dimen0=\dimexpr\linewidth - (#1)}%
709   \parfillskip=\dimen0 \@minus \dimen0
710   \relax}
711 {\par}
712
```

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

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

`tlnepar@nonzero@parindent`

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

`openlastlinepar (env.)` Compare with the suggestion in Ref. 31.

```
715 \NewDocumentEnvironment{openlastlinepar}{o}
716 {\IfNoValueTF{#1}
717   {\ifdim\parindent=\z@
718     \skip0=\typog@openlastlinepar@zero@parindent
719     \@plus 1fil
720     \@minus \typog@openlastlinepar@zero@parindent
721   \else
722     \skip0=\typog@openlastlinepar@nonzero@parindent
723     \@plus 1fil
724     \@minus \typog@openlastlinepar@nonzero@parindent
725   \fi}
726   {\dimen0=\dimexpr#1\relax
727     \skip0=\dimen0 \@plus 1fil \@minus \dimen0}
728   \parfillskip=\skip0}
729 {\par}
730
```

`lastlinefitpar (env.)` Set value of `\lastlinefit` for a paragraph.

```
731 \NewDocumentEnvironment{lastlinefitpar}{0{1000}}
732 {\lastlinefit=#1\relax}
733 {\par}
734
```



## C.11 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.

```
735 \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.

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

`\widespace`

```
737 \NewDocumentCommand{\widespace}{s}
738 {\IfBooleanTF{#1}%
739   {\dimen0=\widespacescale\fontdimen2\font}%
740   {\ifdim\fontdimen7\font=\z@
741     \dimen0=\widespacescale\fontdimen2\font
742     \else
743       \dimen0=\dimexpr\fontdimen2\font +
744         \widespacestrength\fontdimen7\font
745     \fi}%
746   \hskip \glueexpr\dimen0
747     \@plus \widespacescale\fontdimen3\font
748     \@minus \widespacescale\fontdimen4\font
749   \ignorespaces}
750
```

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

```
751 \newcommand*{\narrowospacestrength}{.5}
```

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

```
752 \newcommand*{\narrowospace}{.9375}
```

`\narrowospace`

```
753 \NewDocumentCommand{\narrowospace}{s}
754 {\IfBooleanTF{#1}%
755   {\dimen0=\narrowospace\fontdimen2\font}%
756   {\ifdim\fontdimen7\font=\z@
757     \dimen0=\narrowospace\fontdimen2\font
758     \else
759       \dimen0=\dimexpr\fontdimen2\font -
760         \narrowospacestrength\fontdimen7\font
761     \fi}%
762   \hskip \glueexpr\dimen0
763     \@plus \narrowospace\fontdimen3\font
```

```

764      \@minus \narrowspacescale\fontdimen4\font
765      \ignorespaces}
766

```

See also: TeX by Topic [11, ch. 20, p. 185–190].

`loosespacing` (*env.*)

```

767 \NewDocumentEnvironment{loosespacing}{0{1}}
768 {\dimen2=\fontdimen2\font
769   \ifcase #1
770     \spaceskip=\z@
771     \or % 1      +5%
772     \spaceskip=1.05\dimen2 \@plus .5\dimen2 \@minus .1\dimen2
773     \or % 2      +10%
774     \spaceskip=1.1\dimen2 \@plus .5\dimen2 \@minus .1\dimen2
775     \or % 3      +20%
776     \spaceskip=1.2\dimen2 \@plus .6\dimen2 \@minus .2\dimen2
777     \else % >= 4  +30%
778     \spaceskip=1.3\dimen2 \@plus .8\dimen2 \@minus .3\dimen2
779   \fi
780   \ignorespaces}
781 {\ignorespacesafterend}
782

```

`tightspacing` (*env.*)

```

783 \NewDocumentEnvironment{tightspacing}{0{1}}
784 {\dimen2=\fontdimen2\font
785   \ifcase #1
786     \spaceskip=\z@
787     \or % 1      -1.25%
788     \spaceskip=.9875\dimen2 \@plus .0125\dimen2 \@minus .5\dimen2
789     \or % 2      -2.5%
790     \spaceskip=.975\dimen2 \@plus .025\dimen2 \@minus .5\dimen2
791     \or % 3      -5%
792     \spaceskip=.95\dimen2 \@plus .05\dimen2 \@minus .5\dimen2
793     \else % >= 4  -10%
794     \spaceskip=.9\dimen2 \@plus .1\dimen2 \@minus .5\dimen2
795   \fi
796   \ignorespaces}
797 {\ignorespacesafterend}
798

```

## C.12 Microtype Front-End

### Tracking

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

```

799 \NewDocumentEnvironment{setfonttracking}{m}
800 {\edef\MT@letterspace@{#1}%
801   \lsstyle
802   \ignorespaces}
803 {\ignorespacesafterend}

```

804

**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.

```

805 \newcommand*{\typog@setup@font@expansion}
806 {\SetExpansion
807   [context = typog@shrink1,
808    shrink = \typog@shrink@i,
809    stretch = 0]%
810   {encoding = {*}}}%
811 {}
812 \SetExpansion
813   [context = typog@shrink2,
814    shrink = \typog@shrink@ii,
815    stretch = 0]%
816   {encoding = {*}}}%
817 {}
818 \SetExpansion
819   [context = typog@shrink3,
820    shrink = \typog@shrink@iii,
821    stretch = 0]%
822   {encoding = {*}}}%
823 {}
824
825 \SetExpansion
826   [context = typog@stretch1,
827    shrink = 0,
828    stretch = \typog@stretch@i]%
829   {encoding = {*}}}%
830 {}
831 \SetExpansion
832   [context = typog@stretch2,
833    shrink = 0,
834    stretch = \typog@stretch@ii]%
835   {encoding = {*}}}%
836 {}
837 \SetExpansion
838   [context = typog@stretch3,
839    shrink = 0,
840    stretch = \typog@stretch@iii]%
841   {encoding = {*}}}%
842 {}
843
844 \SetExpansion
845   [context = typog@expand1,
846    shrink = \typog@shrink@i,
847    stretch = \typog@stretch@i]%
848   {encoding = {*}}}%

```

```

849     {}
850   \SetExpansion
851     [context = typog@expand2,
852       shrink = \typog@shrink@ii,
853       stretch = \typog@stretch@ii]%
854     {encoding = {*}}%
855     {}
856   \SetExpansion
857     [context = typog@expand3,
858       shrink = \typog@shrink@iii,
859       stretch = \typog@stretch@iii]%
860     {encoding = {*}}%
861     {}

```

`\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.

```

862 \newcommand*{\typog@test@microtype@expansion@feature}
863 {
864   \ifMT@expansion
865     \typog@typeout{microtype preloaded -- font expansion features avail-
866       able}%
867     \def\typog@require@microtype@expansion{\relax}
868     \typog@setup@font@expansion
869   \else
870     \PackageWarning{typog}{microtype preloaded,\space
871       but font expansion is disabled}%
872     \def\typog@require@microtype@expansion
873       {\PackageError{typog}
874         {microtype font expansion disabled}
875         {pass option 'expansion' to package microtype}}
876   \fi
877 }

```

`\typog@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.

```

875 \iftypog@microtype@preloaded
876   \typog@test@microtype@expansion@feature
877 \else
878   \def\typog@require@microtype@expansion
879     {\PackageError{typog}%
880       {package microtype not (pre-)loaded, %
881        which is required for typog's font expansion}%
882       {require package microtype before package typog}}
883 \fi
884

```

`setfontshrink` (*env.*)

```

885 \NewDocumentEnvironment{setfontshrink}{0{1}}
886 {
887   \typog@require@microtype@expansion
888 }
889 }

```

```

887 \ifcase#1% 0
888   \relax
889 \or % 1
890   \microtypecontext{expansion=typog@shrink1}%
891 \or % 2
892   \microtypecontext{expansion=typog@shrink2}%
893 \else % >= 3
894   \microtypecontext{expansion=typog@shrink3}%
895 \fi
896 \ignorespaces}
897 {\ignorespacesafterend}
898

```

`setfontstretch (env.)`

```

899 \NewDocumentEnvironment{setfontstretch}{0{1}}
900 {\typog@require@microtype@expansion
901   \ifcase#1% 0
902     \relax
903   \or % 1
904     \microtypecontext{expansion=typog@stretch1}%
905   \or % 2
906     \microtypecontext{expansion=typog@stretch2}%
907   \else % >= 3
908     \microtypecontext{expansion=typog@stretch3}%
909   \fi
910   \ignorespaces}
911 {\ignorespacesafterend}
912

```

`setfontexpand (env.)`

```

913 \NewDocumentEnvironment{setfontexpand}{0{1}}
914 {\typog@require@microtype@expansion
915   \ifcase#1% 0
916     \relax
917   \or % 1
918     \microtypecontext{expansion=typog@expand1}%
919   \or % 2
920     \microtypecontext{expansion=typog@expand2}%
921   \else % >= 3
922     \microtypecontext{expansion=typog@expand3}%
923   \fi
924   \ignorespaces}
925 {\ignorespacesafterend}
926

```

`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.

```

927 \NewDocumentEnvironment{nofontexpansion}{}
928 {\ifdefined\microtypesetup
929   \microtypesetup{expansion=false}%

```

```

930 \fi
931 \ignorespaces}
932 {\ignorespacesafterend}

```

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

```

933 \let\nofontexpand=\nofontexpansion
934 \let\endnofontexpand=\endnofontexpansion
935

```

### Character Protrusion

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

```

936 \NewDocumentEnvironment{nocharprotrusion}{}
937 {\ifdefined\microtypesetup
938   \microtypesetup{protrusion=false}%
939   \fi
940   \ignorespaces}
941 {\ignorespacesafterend}
942

```

## C.13 Sloppy Paragraphs

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

```

943 \newcommand*{\typog@scaled@emergencystretch}[1]
944 {\emergencystretch=\ifdim\linewidth=\z@
945   #1%
946   \else
947     \dimexpr (#1) * \linewidth / \textwidth
948   \fi}
949

```

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

```

950 \NewDocumentCommand{\slightlyloppy}{0{1}}
951 {\ifcase #1% 0
952   % \tolerance=200
953   % \emergencystretch=\z@
954   % \hfuzz=.1\p@
955   % \vfuzz=\hfuzz
956   \fussy
957 \or % 1
958   \pretolerance=165%
959   \tolerance=330%
960   \typog@scaled@emergencystretch{.375em}%
961   \hfuzz=.15\p@
962   \vfuzz=\hfuzz
963 \or % 2
964   \pretolerance=265%

```

```

965     \tolerance=530%
966     \typog@scaled@emergencystretch{.75em}%
967     \hfuzz=.15\p@
968     \vfuzz=\hfuzz
969   \or % 3
970     \pretolerance=435%
971     \tolerance=870%
972     \typog@scaled@emergencystretch{1.125em}%
973     \hfuzz=.2\p@
974     \vfuzz=\hfuzz
975   \or % 4
976     \pretolerance=705%
977     \tolerance=1410%
978     \typog@scaled@emergencystretch{1.5em}%
979     \hfuzz=.3\p@
980     \vfuzz=\hfuzz
981   \or % 5
982     \pretolerance=1155%
983     \tolerance=2310%
984     \typog@scaled@emergencystretch{1.875em}%
985     \hfuzz=.35\p@
986     \vfuzz=\hfuzz
987   \or % 6
988     \pretolerance=1880%
989     \tolerance=3760%
990     \typog@scaled@emergencystretch{2.25em}%
991     \hfuzz=.4\p@
992     \vfuzz=\hfuzz
993   \or % 7
994     \pretolerance=3065%
995     \tolerance=6130%
996     \typog@scaled@emergencystretch{2.625em}%
997     \hfuzz=.45\p@
998     \vfuzz=\hfuzz
999   \else % >= 8
1000     % \tolerance=9999
1001     % \emergencystretch=3em
1002     % \hfuzz=.5\p@
1003     % \vfuzz=\hfuzz
1004     \sloppy
1005   \fi
1006   \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(.1, .5, 9);
```

`slightlyloppypar (env.)`

```
1007 \NewDocumentEnvironment{slightlyloppypar}{0{1}}
1008   {\par\slightlyloppy[#1]\ignorespaces}
1009   {\par}
1010
```

## C.14 Vertically Partially-Tied Paragraphs

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

```
1011 \ExplSyntaxOn
1012 \newcommand*{\typog@geometric@mean}[2]
1013   {\fp_to_int:n {sqrt((#1) * (#2))}}
1014 \ExplSyntaxOff
1015
```

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

```
1016 \newcounter{typog@mean@penalty}
1017
```

`\vtietop`

```
1018 \NewDocumentCommand{\vtietop}{0{3}}
1019   {\setcounter{typog@mean@penalty}
1020     {\typog@geometric@mean{\@M}{\clubpenalty}}%
1021     \typog@debug@typeout{vtietop: penalties \the\@M--\the\value{typog@mean@penalty}
1022       -\the\clubpenalty}%
1022     \unless\ifnum\clubpenalty<\@M
1023       \PackageWarning{typog}{vtietop: clubpenalty=\the\clubpenalty\space>= 10000}%
1024     \fi
1025     \ifcase#1% 0
1026       \relax
1027     \or % 1
1028       \relax
1029     \or % 2
1030       \clubpenalties 3
1031       \@M
1032       \value{typog@mean@penalty}
1033       \clubpenalty
```



```

1034 \or % 3
1035   \clubpenalties 4
1036     \@M \@M
1037       \value{typog@mean@penalty}
1038     \clubpenalty
1039 \or % 4
1040   \clubpenalties 5
1041     \@M \@M \@M
1042       \value{typog@mean@penalty}
1043     \clubpenalty
1044 \or % 5
1045   \clubpenalties 6
1046     \@M \@M \@M \@M
1047       \value{typog@mean@penalty}
1048     \clubpenalty
1049 \or % 6
1050   \clubpenalties 7
1051     \@M \@M \@M \@M \@M
1052       \value{typog@mean@penalty}
1053     \clubpenalty
1054 \or % 7
1055   \clubpenalties 8
1056     \@M \@M \@M \@M \@M
1057       \value{typog@mean@penalty}
1058     \clubpenalty
1059 \or % 8
1060   \clubpenalties 9
1061     \@M \@M \@M \@M \@M
1062       \value{typog@mean@penalty}
1063     \clubpenalty
1064 \else % >= 9
1065   \clubpenalties 10
1066     \@M \@M \@M \@M \@M \@M \@M \@M
1067       \value{typog@mean@penalty}
1068     \clubpenalty
1069 \fi}
1070

```

vtietoppar (*env.*)

```

1071 \NewDocumentEnvironment{vtietoppar}{0{3}}
1072 {\vtietop[#1]}
1073 {\par
1074   \ignorespacesafterend}
1075

```

\splicevtietop

```

1076 \NewDocumentCommand{\splicevtietop}{0{3}}
1077 {\let\typog@old@item=\@item
1078   \def\@item[##1]{\typog@old@item[##1]\vtietop[#1]}%
1079   \ignorespaces}
1080

```

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).

```

1081 \ifdefined\SetEnumitemKey
1082   \SetEnumitemKey{vtietop}{first=\splicevtietop}
1083 \fi
1084
\vtiebot
1085 \NewDocumentCommand{\vtiebot}{0{3}}
1086   {\setcounter{typog@mean@penalty}
1087     {\typog@geometric@mean{\@M}{\widowpenalty}}%
1088     \typog@debug@typeout{vtiebot: penalties \the\@M--\the\value{typog@mean@penalty}
1089       -\the\widowpenalty}%
1089     \unless\ifnum\widowpenalty<\@M
1090       \PackageWarning{typog}{vtiebot: widowpenalty=\the\widowpenalty\space>= 10000
1091       \fi
1092     \ifcase#1% 0
1093       \relax
1094     \or % 1
1095       \relax
1096     \or % 2
1097       \widowpenalties 3
1098         \@M
1099         \value{typog@mean@penalty}
1100         \widowpenalty
1101     \or % 3
1102       \widowpenalties 4
1103         \@M \@M
1104         \value{typog@mean@penalty}
1105         \widowpenalty
1106     \or % 4
1107       \widowpenalties 5
1108         \@M \@M \@M
1109         \value{typog@mean@penalty}
1110         \widowpenalty
1111     \or % 5
1112       \widowpenalties 6
1113         \@M \@M \@M \@M
1114         \value{typog@mean@penalty}
1115         \widowpenalty
1116     \or % 6
1117       \widowpenalties 7
1118         \@M \@M \@M \@M \@M
1119         \value{typog@mean@penalty}
1120         \widowpenalty
1121     \or % 7
1122       \widowpenalties 8
1123         \@M \@M \@M \@M \@M \@M
1124         \value{typog@mean@penalty}
1125         \widowpenalty
1126     \or % 8
1127       \widowpenalties 9

```

```

1128      \@M \@M \@M \@M \@M \@M \@M
1129      \value{typog@mean@penalty}
1130      \widowpenalty
1131 \else % >= 9
1132   \widowpenalties 10
1133   \@M \@M \@M \@M \@M \@M \@M \@M
1134   \value{typog@mean@penalty}
1135   \widowpenalty
1136 \fi}
1137

```

vtiebotpar (env.)

```

1138 \NewDocumentEnvironment{vtiebotpar}{0{3}}
1139 {\vtiebot[#1]}
1140 {\par
1141  \ignorespacesafterend}
1142

```

\typog@vtiebotdisp

```

1143 \NewDocumentCommand{\typog@vtiebotdisp}{m}
1144 {\setcounter{typog@mean@penalty}
1145   {\typog@geometric@mean{\@M}{\displaywidowpenalty}}}%
1146 \typog@debug@typeout{vtiebotdisp: penalties \the\@M--\the\value{typog@mean@pen
-\the\displaywidowpenalty}}%
1147 \unless\ifnum\displaywidowpenalty<\@M
1148   \PackageWarning{typog}{vtiebotdisp: displaywidowpenalty=\the\displaywidowpen
1149   \fi
1150   \ifcase#1% 0
1151     \relax
1152   \or % 1
1153     \relax
1154   \or % 2
1155     \displaywidowpenalties 3
1156     \@M
1157     \value{typog@mean@penalty}
1158     \displaywidowpenalty
1159   \or % 3
1160     \displaywidowpenalties 4
1161     \@M \@M
1162     \value{typog@mean@penalty}
1163     \displaywidowpenalty
1164   \or % 4
1165     \displaywidowpenalties 5
1166     \@M \@M \@M
1167     \value{typog@mean@penalty}
1168     \displaywidowpenalty
1169   \or % 5
1170     \displaywidowpenalties 6
1171     \@M \@M \@M \@M
1172     \value{typog@mean@penalty}
1173     \displaywidowpenalty
1174   \or % 6

```

```

1175 \displaywidowpenalties 7
1176 \@M \@M \@M \@M \@M
1177 \value{typog@mean@penalty}
1178 \displaywidowpenalty
1179 \or % 7
1180 \displaywidowpenalties 8
1181 \@M \@M \@M \@M \@M \@M
1182 \value{typog@mean@penalty}
1183 \displaywidowpenalty
1184 \or % 8
1185 \displaywidowpenalties 9
1186 \@M \@M \@M \@M \@M \@M \@M
1187 \value{typog@mean@penalty}
1188 \displaywidowpenalty
1189 \else % >= 9
1190 \displaywidowpenalties 10
1191 \@M \@M \@M \@M \@M \@M \@M \@M
1192 \value{typog@mean@penalty}
1193 \displaywidowpenalty
1194 \fi}
1195

```

`vtiebotdisp (env.)`

```

1196 \NewDocumentEnvironment{vtiebotdisp}{0{3}}
1197 {\typog@vtiebotdisp{#1}}
1198 {\ignorespacesafterend}
1199

```

`vtiebotdisptoppar (env.)`

```

1200 \NewDocumentEnvironment{vtiebotdisptoppar}{0{3}o}
1201 {\postdisplaypenalty=\@M
1202 \predisplaypenalty=10001% in accordance with package ‘widows-
and-orphans’
1203 \edef\typog@@top@lines{\IfNoValueTF{#2}{#1}{#2}}%
1204 \edef\typog@@after@display@math{\vtietop[\typog@@top@lines]}%
1205 \PushPostHook{display}{\aftergroup\typog@@after@display@math}%
1206 \vtiebotdisp[#1]}
1207 {\par
1208 \PopPostHook{display}%
1209 \ignorespacesafterend}
1210

```

## C.15 Breakable Displayed Equations

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

```

1211 \newenvironment*{breakabledisplay}[1][3]
1212 {\allowdisplaybreaks[#1]}
1213 {\ignorespacesafterend}
1214

```

## C.16 Setspace Front-End

`\typog@iter@limit` The maximum number of iterations we perform before bailing out with an error. Can be changed by the user if convergence is slow.

```
1215 \newcommand*{\typog@setbaselineskip@iter@limit}{10}
```

`baselineskip@relative@error` The maximum relative error of the ratio we tolerate for the final baselineskip over the target baselineskip. Can also be changed by the user if necessary.

```
1216 \newcommand*{\typog@setbaselineskip@relative@error}{.001}
```

`\typog@setbaselineskip` Given the  $\langle target-baselineskip \rangle$  as argument iterate setting `\setstretch` until the error drops below our threshold.

```
1217 \ExplSyntaxOn
```

```
1218 \cs_new:Npn \typog@setbaselineskip #1
```

```
1219 {
```

Initialize our “emergency-stop” loop counter.

```
1220 \int_set:Nn \l_tmpa_int {1}
```

```
1221 \int_set:Nn \l_tmpb_int {\typog@setbaselineskip@iter@limit}
```

Note that the call to `\glueexpr` is required to consume dimensions that carry stretchability via plus or minus.

```
1222 \dim_set:Nn \l_tmpa_dim {\glueexpr #1}
```

```
1223
```

```
1224 \typog@debug@typeout{\string\setbaselineskip:\space
```

```
1225 initial\space baselineskip:\space \the\baselineskip}
```

```
1226 \typog@debug@typeout{\string\setbaselineskip:\space
```

```
1227 target\space baselineskip:\space \dim_use:N \l_tmpa_dim}
```

```
1228
```

```
1229 \dim_compare:nNnTF {\baselineskip} > {\c_zero_dim}
```

```
1230 {}
```

```
1231 {
```

```
1232 \PackageError{typog}
```

```
1233 {\string\setbaselineskip:\space
```

```
1234 baselineskip\space not\space positive}
```

```
1235 {}
```

```
1236 }
```

```
1237
```

```
1238 \dim_compare:nNnTF {\l_tmpa_dim} > {\c_zero_dim}
```

```
1239 {}
```

```
1240 {
```

```
1241 \PackageError{typog}
```

```
1242 {\string\setbaselineskip:\space target\space
```

```
1243 baselineskip\space must\space be\space
```

```
1244 positive}
```

```
1245 {}
```

```
1246 }
```

```
1247
```

```
1248 \skip_if_eq:nNnTF {\l_tmpa_dim} {\glueexpr #1}
```

```
1249 {}
```

```
1250 {
```

```

1251 \PackageWarning{typog}
1252     {\string\setbaselineskip:\space argument\space
1253      is\space a\space skip;\space
1254      will\space ignore\space glue}
1255     {}
1256 }
1257
1258 \fp_set:Nn \l_tmpa_fp {\l_tmpa_dim / \baselineskip}
1259 \fp_until_do:nNnn {abs(\l_tmpa_dim / \baselineskip - 1)} <
1260     {\typog@setbaselineskip@relative@error}
1261 {
1262     \setstretch{\fp_use:N \l_tmpa_fp}
1263     \fp_set:Nn \l_tmpa_fp
1264         {\l_tmpa_fp * \l_tmpa_dim / \baselineskip}
1265
1266     \int_incr:N \l_tmpa_int
1267     \int_compare:nNnTF {\l_tmpa_int} > {\l_tmpb_int}
1268     {
1269         \PackageError{typog}
1270             {\string\setbaselineskip:\space excessive\space
1271              number\space of\space iterations:\space
1272              \int_use:N \l_tmpa_int\space >\space
1273              \int_use:N \l_tmpb_int}
1274         {}
1275     }
1276     {}
1277 }
1278
1279 \typog@debug@typeout{\string\setbaselineskip:\space
1280   final\space \string\setstretch\space argument:\space
1281   \fp_use:N \l_tmpa_fp}
1282 \typog@debug@typeout{\string\setbaselineskip:\space
1283   final\space baselineskip:\space \the\baselineskip}
1284 }
1285

```

`\setbaselineskip` Set the `\baselineskip` to an absolute length.

#### **Implementation Note**

Viewed as a standalone macro `\setbaselineskip` does not need the decoration `\AfterPreamble`. However, all of its siblings, `\setbaselineskippercentage`, `\setleading`, and `\setleadingpercentage` then would behave differently as they are delayed to the end of the preamble, but `\setbaselineskip` immediately becomes effective. For example, the successive calls

```

\setbaselineskippercentage{140}
\setbaselineskip{12.5pt}

```

in the preamble would set the baselineskip to 140% in the document. Therefore, `\setbaselineskip` is delayed too and the order of the calls thus preserved.

```

1286 \cs_new:Npn \setbaselineskip #1

```

```

1287 {
1288   \AfterPreamble{\typog@setbaselineskip{#1}}
1289   \ignorespaces
1290 }
1291

```

`\resetbaselineskip` Set the `\baselineskip` to ›neutral‹.

```

1292 \cs_new:Npn \resetbaselineskip
1293 {
1294   \AfterPreamble{\setstretch{1}}
1295 }
1296

```

`\typogfontsize (dimen)` Define the default font-size/quad size.

```

1297 \dim_new:N \typogfontsize

```

Initialize `\typogfontsize` at the end of the preamble, which is after all fonts have been setup.

```

1298 \AfterEndPreamble{
1299   \dim_set:Nn \typogfontsize {\fontdimen6\font}
1300   \typog@debug@typeout{\string\typogfontsize =
1301     \dim_use:N \typogfontsize\space
1302     (at\space begin\space of\space document)}
1303 }
1304

```

`\setbaselineskippercentage`

```

1305 \cs_new:Npn \setbaselineskippercentage #1
1306 {
1307   \AfterPreamble{
1308     \dim_compare:nNnTF {\typogfontsize} > {\c_zero_dim}
1309     {
1310       \typog@setbaselineskip{
1311         \fp_eval:n {(#1) / 100} \typogfontsize
1312       }
1313     }
1314     \PackageError{typog}
1315       {\string\setbaselineskippercentage:\space
1316        \string\typogfontsize <= 0}
1317       {Maybe\space \string\typogfontsize\space
1318        is\space uninitialized?}
1319   }
1320 }
1321 \ignorespaces
1322 }
1323

```

`\setleading`

```

1324 \cs_new:Npn \setleading #1
1325 {
1326   \AfterPreamble{

```

```

1327 \dim_compare:nNnTF {\typogfontsize} > {\c_zero_dim}
1328 {
1329   \typog@setbaselineskip{\typogfontsize + \dimexpr #1}
1330 }
1331 {
1332   \PackageError{typog}
1333     {\string\setleading:\space
1334     \string\typogfontsize <= 0}
1335     {Maybe\space \string\typogfontsize\space
1336     is\space uninitialized?}
1337 }
1338 }
1339 \ignorespaces
1340 }
1341

```

\setleadingpercentage

```

1342 \cs_new:Npn \setleadingpercentage #1
1343 {
1344   \AfterPreamble{
1345     \dim_compare:nNnTF {\typogfontsize} > {\c_zero_dim}
1346     {
1347       \typog@setbaselineskip{
1348         \fp_eval:n {1 + (#1) / 100} \typogfontsize}
1349     }
1350     {
1351       \PackageError{typog}
1352         {\string\setleadingpercentage:\space
1353         \string\typogfontsize <= 0}
1354         {Maybe\space \string\typogfontsize\space
1355         is\space uninitialized?}
1356     }
1357   }
1358   \ignorespaces
1359 }
1360 \ExplSyntaxOff
1361

```

## C.17 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.

```

1362 \ExplSyntaxOn
1363 \cs_new_eq:NN \typog@repeat \prg_replicate:nn
1364

```

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

```

1365 \newcommand*{\typog@mod}[2]{\int_mod:nn{#1}{#2}}
1366 \ExplSyntaxOff

```



1367

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

1368 \newcommand\*{\typog@triplet@max@lines}{99}

1369

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

```

1370 \define@key[typog]{smoothraggedrightshapetriplet}{leftskip}%
1371       {\def\typog@@triplet@leftskip{#1}}
1372 \define@key[typog]{smoothraggedrightshapetriplet}{parindent}%
1373       {\def\typog@@triplet@parindent{#1}}
1374 \NewDocumentEnvironment{smoothraggedrightshapetriplet}{0}{ m m m}
1375   {\def\typog@@triplet@leftskip{z}%
1376     \def\typog@@triplet@parindent{z}%
1377     \setkeys*{typog}{smoothraggedrightshapetriplet}{#1}%
1378     \skip0=\typog@@triplet@leftskip\relax
1379     \skip1=#2\relax
1380     \skip2=#3\relax
1381     \skip3=#4\relax
1382     \typog@debug@typeout{smoothraggedrightshapetriplet: skip0=\the\skip0}%
1383     \typog@debug@typeout{smoothraggedrightshapetriplet: skip1=\the\skip1}%
1384     \typog@debug@typeout{smoothraggedrightshapetriplet: skip2=\the\skip2}%
1385     \typog@debug@typeout{smoothraggedrightshapetriplet: skip3=\the\skip3}%
1386     \unless\ifnum\typog@mod{\typog@triplet@max@lines}{3}=0
1387       \PackageError{typog}
1388         {Line number of triplet generator\space
1389          (\typog@triplet@max@lines) not divisible by 3}
1390     {}
1391   \fi
1392   \edef\typog@@triplet@linespecs{%
1393     \glueexpr \skip0 + \typog@@triplet@parindent\relax
1394     \glueexpr \skip1 - \typog@@triplet@parindent\relax
1395     \skip0 \skip2 \skip0 \skip3
1396     \typog@repeat{\numexpr\typog@triplet@max@lines / 3 - 1}
1397       {\skip0 \skip1 \skip0 \skip2 \skip0 \skip3}}
1398   \parshape=\typog@triplet@max@lines\typog@@triplet@linespecs\relax
1399   {\par}
1400
```

typog@quintuplet@max@lines Maximum number of lines a smoothraggedright paragraph can have with the quintuplet generator. The number must be divisible by 5.

1401 \newcommand\*{\typog@quintuplet@max@lines}{95}

1402

edrightshapequintuplet (*env.*) Engine for 5-line repetitions.

```

1403 \define@key[typog]{smoothraggedrightshapequintuplet}{leftskip}
1404       {\def\typog@@quintuplet@leftskip{#1}}
1405 \define@key[typog]{smoothraggedrightshapequintuplet}{parindent}
1406       {\def\typog@@quintuplet@parindent{#1}}
1407 \NewDocumentEnvironment{smoothraggedrightshapequintuplet}{0}{ m m m m m}

```

```

1408 {\def\typog@@quintuplet@leftskip{\z@}%
1409 \def\typog@@quintuplet@parindent{\z@}%
1410 \setkeys*[typog]{smoothraggedrightshapequintuplet}{#1}%
1411 \skip0=\typog@@quintuplet@leftskip\relax
1412 \skip1=#2\relax
1413 \skip2=#3\relax
1414 \skip3=#4\relax
1415 \skip4=#5\relax
1416 \skip5=#6\relax
1417 \typog@debug@typeout{smoothraggedrightshapequintuplet: skip0=\the\skip0}%
1418 \typog@debug@typeout{smoothraggedrightshapequintuplet: skip1=\the\skip1}%
1419 \typog@debug@typeout{smoothraggedrightshapequintuplet: skip2=\the\skip2}%
1420 \typog@debug@typeout{smoothraggedrightshapequintuplet: skip3=\the\skip3}%
1421 \typog@debug@typeout{smoothraggedrightshapequintuplet: skip4=\the\skip4}%
1422 \typog@debug@typeout{smoothraggedrightshapequintuplet: skip5=\the\skip5}%
1423 \unless\ifnum\typog@mod{\typog@quintuplet@max@lines}{5}=0
1424 \PackageError{typog}
1425 {Line number of quintuplet generator\space
1426 (\typog@quintuplet@max@lines) not divisible by 5}
1427 {}
1428 \fi
1429 \edef\typog@@quintuplet@linespecs{%
1430 \glueexpr \skip0 + \typog@@quintuplet@parindent\relax
1431 \glueexpr \skip1 - \typog@@quintuplet@parindent\relax
1432 \skip0 \skip2 \skip0 \skip3
1433 \skip0 \skip4 \skip0 \skip5
1434 \typog@repeat{\numexpr\typog@quintuplet@max@lines / 5 - 1}
1435 {\skip0 \skip1 \skip0 \skip2 \skip0 \skip3
1436 \skip0 \skip4 \skip0 \skip5}}
1437 \parshape=\typog@quintuplet@max@lines\typog@@quintuplet@linespecs\relax}
1438 {\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.

```

1439 \newcommand*{\typog@septuplet@max@lines}{98}
1440

```

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

```

1441 \define@key[typog]{smoothraggedrightshapeseptuplet}{leftskip}%
1442 {\def\typog@@septuplet@leftskip{#1}}
1443 \define@key[typog]{smoothraggedrightshapeseptuplet}{parindent}%
1444 {\def\typog@@septuplet@parindent{#1}}
1445 \NewDocumentEnvironment{smoothraggedrightshapeseptuplet}{0}{ m m m m m m m}
1446 {\def\typog@@septuplet@leftskip{\z@}%
1447 \def\typog@@septuplet@parindent{\z@}%
1448 \setkeys*[typog]{smoothraggedrightshapeseptuplet}{#1}%
1449 \skip0=\typog@@septuplet@leftskip\relax
1450 \skip1=#2\relax
1451 \skip2=#3\relax
1452 \skip3=#4\relax
1453 \skip4=#5\relax
1454 \skip5=#6\relax

```

```

1455 \skip6=#7\relax
1456 \skip7=#8\relax
1457 \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip0=\the\skip0}%
1458 \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip1=\the\skip1}%
1459 \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip2=\the\skip2}%
1460 \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip3=\the\skip3}%
1461 \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip4=\the\skip4}%
1462 \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip5=\the\skip5}%
1463 \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip6=\the\skip6}%
1464 \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip7=\the\skip7}%
1465 \unless\ifnum\typog@mod{\typog@septuplet@max@lines}{7}=0
1466   \PackageError{typog}
1467     {Line number of septuplet generator\space
1468     (\typog@septuplet@max@lines) not divisible by 7}
1469   {}
1470 \fi
1471 \edef\typog@@septuplet@linespecs{%
1472   \glueexpr \skip0 + \typog@@septuplet@parindent\relax
1473   \glueexpr \skip1 - \typog@@septuplet@parindent\relax
1474   \skip0 \skip2 \skip0 \skip3 \skip0 \skip4
1475   \skip0 \skip5 \skip0 \skip6 \skip0 \skip7
1476   \typog@repeat{\numexpr\typog@septuplet@max@lines / 7 - 1}
1477   {\skip0 \skip1 \skip0 \skip2 \skip0 \skip3 \skip0 \skip4
1478    \skip0 \skip5 \skip0 \skip6 \skip0 \skip7}}
1479 \parshape=\typog@septuplet@max@lines\typog@@septuplet@linespecs\relax
1480 {\par}
1481
smoothraggedrightfuzzfactor
1482 \newcommand*{\smoothraggedrightfuzzfactor}{1.0}

smoothraggedrightgenerator
1483 \newcommand*{\smoothraggedrightgenerator}{triplet}

\smoothraggedrightleftskip
1484 \newlength{\smoothraggedrightleftskip}

smoothraggedrightparindent
1485 \newlength{\smoothraggedrightparindent}

\smoothraggedrighttragwidth
1486 \newlength{\smoothraggedrighttragwidth}
1487 \setlength{\smoothraggedrighttragwidth}{2em}
1488

\typog@fuzzwidth (dimen)
1489 \newdimen{\typog@fuzzwidth}
1490

```

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

```
1491 \define@key[typog]{smoothraggedrightpar}{linewidth}%
1492         {\def\typog@@linewidth{#1}}
1493
```

Convert generator name to an integer suitable for `\ifcase`. Indicate an unknown name with -1.

```
1494 \ExplSyntaxOn
1495 \cs_new:Npn \typog@generator@index:n #1 {
1496   \str_case:nnTF {#1}
1497     {
1498       {triplet} {0}
1499       {quintuplet} {1}
1500       {septuplet} {2}
1501     }
1502     {}
1503     {-1}
1504 }
1505 \cs_generate_variant:Nn \typog@generator@index:n {e}
1506 \let\typog@generator@index=\typog@generator@index:e
1507 \ExplSyntaxOff
1508
1509 \NewDocumentEnvironment{smoothraggedrightpar}{0{}}
1510   {\edef\typog@@linewidth{\linewidth}%
1511    \setkeys[typog]{smoothraggedrightpar}{#1}%
1512    \edef\typog@@generatorchoice{\typog@generator@index{\smoothraggedrightgenerator@choice}}
1513    \let\typog@@smoothraggedrightleftskip=\smoothraggedrightleftskip
1514    \ifnum\@listdepth>0
1515      \addtolength{\typog@@smoothraggedrightleftskip}{\leftmargin}%
1516    \fi
1517    \typog@fuzzwidth=\smoothraggedrightfuzzfactor\smoothraggedrightragwidth
1518    \ifcase\typog@@generatorchoice
1519      generator=triplet produces a »short line – long line – middle length
1520      line« sequence.
1521      \typog@fuzzwidth=.25\smoothraggedrightragwidth
1522      \typog@debug@typeout{smoothraggedright: generator=triplet, typog@fuzzwidth=\the\typog@fuzzwidth}%
1523      \smoothraggedrightshapetriplet[leftskip=\typog@@smoothraggedrightleftskip,
1524                                     parindent=\glueexpr\smoothraggedrightparindent,
1525                                     indent,
1526                                     #1]%
1527      {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth
1528       + \glueexpr \ze @\plus \typog@fuzzwidth\relax}% (1)
1529      {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth}% (3)
```

Obeys to the indentation prescribed by any list environment.

```
1513 \let\typog@@smoothraggedrightleftskip=\smoothraggedrightleftskip
1514 \ifnum\@listdepth>0
1515   \addtolength{\typog@@smoothraggedrightleftskip}{\leftmargin}%
1516 \fi
```

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

```
1517 \typog@fuzzwidth=\smoothraggedrightfuzzfactor\smoothraggedrightragwidth
```

Now for the generator-specific code...

```
1518 \ifcase\typog@@generatorchoice
1519   generator=triplet produces a »short line – long line – middle length
1520   line« sequence.
1521   \typog@fuzzwidth=.25\smoothraggedrightragwidth
1522   \typog@debug@typeout{smoothraggedright: generator=triplet, typog@fuzzwidth=\the\typog@fuzzwidth}%
1523   \smoothraggedrightshapetriplet[leftskip=\typog@@smoothraggedrightleftskip,
1524                                   parindent=\glueexpr\smoothraggedrightparindent,
1525                                   indent,
1526                                   #1]%
1527   {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth
1528    + \glueexpr \ze @\plus \typog@fuzzwidth\relax}% (1)
1529   {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth}% (3)
```

```

1527      {\glueexpr (\typog@@linewidth * 2 - \smoothraggedrightrag-
width) / 2
1528      + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (2)
1529      \or
generator=quintuplet.
1530      \typog@fuzzwidth=.125\smoothraggedrightragwidth
1531      \typog@debug@typeout{smoothraggedright: generator=quintuplet, ty-
pog@fuzzwidth=\the\typog@fuzzwidth}%
1532      \smoothraggedrightshapequintuplet[leftskip=\typog@smoothraggedrightleftskip,
1533      parindent=\glueexpr\smoothraggedrightparind
indent,
1534      #1]%
1535      {\glueexpr (\typog@@linewidth * 4 - \smoothraggedrightrag-
width * 3) / 4
1536      + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (2)
1537      {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth\relax}% (5)
1538      {\glueexpr (\typog@@linewidth * 2 - \smoothraggedrightrag-
width) / 2
1539      + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (3)
1540      {\glueexpr (\typog@@linewidth * 4 - \smoothraggedrightrag-
width) / 4
1541      + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (4)
1542      {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth
1543      + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
1544      \or
generator=septuplet.
Permutation 3 - 6 - 1 - 5 - 2 - 7 - 4 looks ›random‹ enough for our purposes.
1545      \typog@fuzzwidth=.08333\smoothraggedrightragwidth
1546      \typog@debug@typeout{smoothraggedright: generator=septuplet, ty-
pog@fuzzwidth=\the\typog@fuzzwidth}%
1547      \smoothraggedrightshapeseptuplet[leftskip=\typog@smoothraggedrightleftskip,
1548      parindent=\glueexpr\smoothraggedrightparind
indent,
1549      #1]%
1550      {\glueexpr (\typog@@linewidth * 3 - \smoothraggedrightrag-
width * 2) / 3
1551      + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (3)
1552      {\glueexpr (\typog@@linewidth * 6 - \smoothraggedrightrag-
width) / 6
1553      + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (6)
1554      {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth +
1555      + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
1556      {\glueexpr (\typog@@linewidth * 3 - \smoothraggedrightrag-
width) / 3

```

```

1557         + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
      nus \typog@fuzzwidth\relax}% (5)
1558     {\glueexpr (\typog@@linewidth * 6 - \smoothraggedrightrag-
      width * 5) / 6
1559         + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
      nus \typog@fuzzwidth\relax}% (2)
1560     {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth\relax}% (7)
1561     {\glueexpr (\typog@@linewidth * 2 - \smoothraggedrightrag-
      width) / 2
1562         + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
      nus \typog@fuzzwidth\relax}% (4)
1563     \else
1564         \PackageError{typog}
1565             {smoothraggedright: unknown generator name (\smoothraggedright-
      generator)}{
1566             {valid generator names are triplet, quintuplet, and sep-
      tuplet}
1567         \fi}
1568     {\ifcase\typog@@generatorchoice
1569         \endsmoothraggedrightshapetriplet
1570     \or
1571         \endsmoothraggedrightshapequintuplet
1572     \or
1573         \endsmoothraggedrightshapeseptuplet
1574     \fi}
1575
smoothraggedright (env.)
1576 \NewDocumentEnvironment{smoothraggedright}{0{}}
1577 {\PushPostHook{par}{\hskip-\parindent\smoothraggedrightpar[#1]\relax}}
1578 {\par\PopPostHook{par}}
1579

```

## Change History

### v0.1

General: Initial version. [i](#)

### v0.2

`\narrowospace`: New macro. [89](#)

`\widespace`: Add fallback if `\fontdimen7` is zero. Extend with a starred version. [89](#)

### v0.3

`hyphenmin`: New environment. [72](#)

`\resetbaselineskip`: New macro. [103](#)

`\setbaselineskip`: New macro. [102](#)

`\setbaselineskippercentage`: New macro. [103](#)

`\setleading`: New macro. [103](#)

`\setleadingpercentage`: New macro. [104](#)

`\typogfontsize`: New dimen. [103](#)

### v0.4

`\lowercaseadjustlabelitems`: New macro. [82](#)

`\noadjustlabelitems`: New macro. [82](#)

`\typogadjuststairs`: New macro. [85](#)

`\typoggetnth`: New macro. [69](#)

`\typoglowercaseadjustcheck`: New macro. [86](#)

`\typoguppercaseadjustcheck`: New macro. [86](#)

`\uppercaseadjustlabelitems`: New macro. [82](#)

### v0.4a

`smoothraggedrightshapeseptuplet`: Add missing backslash to `\typog@@septuplet@parindent` in local macro `\typog@@septuplet@linespecs` which rendered the septuplet generator unusable. [106](#)

`\typog@register@pdfsubstitute`: Using `\pdfstringdefDisableCommands` breaks package `pdfcomment`. So we just refrain from using the macro if `pdfcomment` was loaded. [65](#)

### v0.5

General: Rewrite parser for package option `lowercaselabelitemadjustments` to accept a star as a placeholder. [68](#)

Rewrite parser for package option `uppercaselabelitemadjustments` to accept a star as a placeholder. [68](#)

`\kernedslash`: Allow for lowering (or raising) `\kernedslash` with package option `lowerslash`. [74](#)

`lastlinefitpar`: New environment. [88](#)

## References

- [1] ABRAHAM, PAUL W., HARGREAVES, KATHRYN A., and KARL BERRY. *T<sub>E</sub>X for the Impatient*. 2020, <http://tug.ctan.org/info/impatient/book.pdf>.
- [2] AMERICAN MATHEMATICAL SOCIETY and the L<sup>A</sup>T<sub>E</sub>X3 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<sub>T</sub>S TEAM. *ε-T<sub>E</sub>X*. 1998, [https://mirrors.ctan.org/systems/doc/etex/etex\\_man.pdf](https://mirrors.ctan.org/systems/doc/etex/etex_man.pdf).
- [7] BURNOL, JEAN-FRANÇOIS. *Package mathastext*. 2023, <https://ctan.org/pkg/mathastext>.
- [8] 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>.
- [9] CARLISLE, DAVID. *What do different \fontdimen<num> mean*. 2013–1–2, <https://tex.stackexchange.com/questions/88991/what-do-different-fontdimennum-mean>.
- [10] CUBITT, TOBY. *Package cleveref*. 2018, <https://ctan.org/pkg/cleveref>.
- [11] EIJKHOUT, VICTOR. *T<sub>E</sub>X By Topic, A Texnician's Reference*. 2007, <https://www.eijkhout.net/tex/tex-by-topic.html>.
- [12] HØGHOLM, MORTEN, MADSEN, LARS and the L<sup>A</sup>T<sub>E</sub>X3 PROJECT TEAM. *Package mathtools*. 2020, <https://ctan.org/pkg/mathtools>.
- [13] KHIRIVICH, SIARHEI. *Tips on Writing a Thesis in L<sup>A</sup>T<sub>E</sub>X*. 2013, <http://www.khirevich.com/latex/microtype>.
- [14] KLEBER, JOSEF. *Package pdfcomment*. 2018, <https://ctan.org/pkg/pdfcomment>.
- [15] KNUTH, DONALD ERVIN. *The T<sub>E</sub>Xbook*. Addison Wesley, Reading/MA, 1986.
- [16] THE L<sup>A</sup>T<sub>E</sub>X PROJECT. *Package l3experimental*. 2024, <https://ctan.org/pkg/l3experimental>.
- [17] MCPHERSON, KENT. *Package layout*. 2014, <https://ctan.org/pkg/layout>. The package was converted to L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> by J. L. BRAAMS and modified by H. UMEKI.



- [18] MIDDENDORP, JAN. *Shaping Text*. BIS publishers, Amsterdam, 2014.
- [19] MITTELBACH, FRANK. *Managing forlorn paragraph lines (a. k. a. widows and orphans) in L<sup>A</sup>T<sub>E</sub>X*. TUGboat, 39(3), 246–251, 2018, <https://tug.org/TUGboat/tb39-3/tb123mitt-widows.pdf>.
- [20] MITTELBACH, FRANK. *Package widows-and-orphans*. 2020, <https://ctan.org/pkg/widows-and-orphans>.
- [21] RAHTZ, SEBASTIAN, and FRANK MITTELBACH. *Package hyperref*. 2020, <https://ctan.org/pkg/hyperref>. The package is maintained by the L<sup>A</sup>T<sub>E</sub>X3 Project Team.
- [22] SCHLICHT, ROBERT. *Package microtype*. 2020, <https://ctan.org/pkg/microtype>.
- [23] SCHRÖDER, MARTIN. *Package ragged2e*. 2019, <https://ctan.org/pkg/ragged2e>.
- [24] 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>.
- [25] STRIZVER, ILENE. *Type rules!: the designer's guide to professional typography*, 4<sup>th</sup> ed. John Wiley & Sons, Hoboken/NJ, 2014.
- [26] TOBIN, GEOFFREY, and ROBIN FAIRBAIRNS. *Package setspace*. 2011, <https://ctan.org/pkg/setspace>.
- [27] UMEKI, HIDEO. *Package geometry*. 2020, <https://ctan.org/pkg/geometry>.
- [28] WERMUTH, UDO. *Tracing paragraphs*. TUGboat, 37(3), 358–373, 2016, <https://tug.org/TUGboat/tb37-3/tb117wermuth.pdf>.
- [29] WERMUTH, UDO. *The optimal value for \emergencystretch*. TUGboat, 38(1), 65–86, 2017, <https://tug.org/TUGboat/tb38-1/tb118wermuth.pdf>.
- [30] WERMUTH, UDO. *A note on \linepenalty*. TUGboat, 38(3), 400–414, 2017, <https://tug.org/TUGboat/tb38-3/tb120wermuth.pdf>.
- [31] WERMUTH, UDO. *Experiments with \parfillskip*. TUGboat, 39(3), 276–303, 2018, <https://tug.org/TUGboat/tb39-3/tb123wermuth-parfillskip.pdf>.
- [32] WERMUTH, UDO. *An attempt at ragged-right typesetting*. TUGboat, 41(1), 73–94, 2020, <https://tug.org/TUGboat/tb41-1/tb127wermuth-ragged.pdf>.
- [33] WERMUTH, UDO. Personal communication. August 2, 2022.
- [34] WERMUTH, UDO. *Vertical alignments in plain T<sub>E</sub>X*. TUGboat, 44(3), 427–440, 2023, <https://tug.org/TUGboat/tb44-3/tb138wermuth-valign.pdf>.

- [35] WILSON, PETER. *Package hyphenat*. 2004, <https://ctan.org/pkg/hyphenat>. The package is maintained by W. ROBERTSON.
- [36] WILSON, PETER. *Glisterings*. TUGboat, 28(2), 229–232, 2007, <https://tug.org/TUGboat/tb28-2/tb89glister.pdf>.
- [37] 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. We prefix all references to code lines with >ℓ<.

### A

\allowhyphenation [10](#), [ℓ208](#)  
amsmath (package) [44](#)

### B

baseline skip [46](#)  
bookmark [14](#)  
breakabledisplay (env.) [44](#), [ℓ1211](#)  
breakpenalty (option) [2](#), [14](#), [16](#), [18](#), [19](#)  
\breakpoint [13](#), [ℓ212](#)  
\breakpoint\* [13](#)

### C

\capitaldash [19](#), [ℓ346](#), [ℓ354](#), [ℓ376](#)  
\capitaldash\* [19](#)  
\capitalemdash [19](#), [ℓ357](#)  
\capitalemdash\* [19](#)  
\capitalendash [19](#), [ℓ341](#)  
\capitalendash\* [19](#)  
\capitalhyphen [18](#), [ℓ327](#)  
\capitalhyphen\* [18](#)  
\capitalinvertedexclamationmark [22](#),  
[ℓ440](#)  
\capitalinvertedquestionmark [22](#),  
[ℓ449](#)  
\capitaltimes [20](#), [ℓ378](#)  
configuration [5](#)  
covernextindentpar (env.) [32](#), [ℓ701](#)  
csquotes (package) [22](#)

### D

debug (option) [2](#)  
dimensions:  
  \typogfontsize [47](#), [ℓ1297](#), [ℓ1308](#),  
    [ℓ1311](#), [ℓ1316](#), [ℓ1317](#), [ℓ1327](#), [ℓ1329](#),  
    [ℓ1334](#), [ℓ1335](#), [ℓ1345](#), [ℓ1348](#), [ℓ1353](#),  
    [ℓ1354](#)  
\Doubleguillemetleft [21](#), [ℓ412](#)  
\doubleguillemetleft [21](#), [ℓ396](#)  
\Doubleguillemetright [21](#), [ℓ416](#)  
\doubleguillemetright [21](#), [ℓ400](#)

### E

enumitem-keys:  
  vtietop [42](#)  
environments:  
  breakabledisplay [44](#), [ℓ1211](#)  
  covernextindentpar [32](#), [ℓ701](#)  
  hyphenmin [13](#), [ℓ222](#)  
  lastlinecenteredpar [27](#), [ℓ677](#)  
  lastlinefitpar [33](#), [ℓ731](#)  
  lastlineflushrightpar [27](#),  
    [ℓ674](#)  
  lastlineraggedleftpar [27](#),  
    [ℓ668](#)  
  loosespacing [34](#), [ℓ767](#)  
  nocharprotrusion [39](#), [ℓ936](#)  
  nofontexpand [ℓ933](#)  
  nofontexpansion [39](#), [ℓ927](#)  
  openlastlinepar [32](#), [ℓ715](#)  
  prolongpar [31](#), [ℓ691](#)  
  setfontexpand [38](#), [ℓ913](#)  
  setfontshrink [38](#), [ℓ885](#)  
  setfontstretch [38](#), [ℓ899](#)  
  setfonttracking [37](#), [ℓ799](#)  
  shortenpar [31](#), [ℓ684](#)  
  slightlyloppypar [40](#), [ℓ1007](#)  
  smoothraggedright [50](#), [ℓ1576](#)  
  smoothraggedrightpar [49](#), [ℓ1491](#)  
  smoothraggedrightshape-  
    quintuplet [49](#), [ℓ1403](#)  
  smoothraggedrightshape-  
    septuplet [49](#), [ℓ1441](#)  
  smoothraggedrightshape-  
    triplet [49](#), [ℓ1370](#)  
  tightspacing [34](#), [ℓ783](#)  
  typoginspect [7](#), [ℓ169](#)  
  typoginspectpar [7](#), [ℓ196](#)  
  typogsetup [5](#), [ℓ105](#)  
  vtiebotdisp [42](#), [ℓ1196](#)  
  vtiebotdisptoppar [42](#), [ℓ1200](#)  
  vtiebotpar [42](#), [ℓ1138](#)  
  vtietoppar [41](#), [ℓ1071](#)

**F**`\figuredash` [19](#), [ℓ371](#)`\figuredash*` [19](#)

## font

encoding [20](#)expansion [31](#), [38](#)information [6](#)protrusion [39](#)size [6](#)spacing [31](#), [33](#)loose [33](#)tight [33](#)tracking [31](#), [37](#)

## typeface

ADF Baskervald [21](#)ADF Accanthis [28](#)ADF Venturis [20](#), [28](#)Alegreya [20](#)Arvo [20](#)Bitter [20](#)Clara [20](#)CM Roman [28](#)Cochineal [17](#), [28](#)Coelacanth [20](#)Crimson Pro [20](#)Crimson Text [20](#)Domitian [28](#)Droid Serif [20](#)EB Garamond [20](#), [21](#), [28](#)Erewhon [20](#)etbb [28](#)Extended Charter [28](#)fbb [20](#)Gentium [20](#), [21](#), [28](#)GFS Artemisia [20](#), [21](#)GFS Bodoni [28](#)GFS Didot [21](#), [28](#)Ibarra Real Nova [20](#)IBM Plex Serif [28](#)INRIA Serif [20](#)KP Serif [28](#)Libertine [20](#)Libertinus Serif [20](#), [21](#), [28](#)Libre Baskerville [20](#)Libre Caslon [20](#)Merriweather [20](#), [21](#)ML Modern [28](#)PT Serif [20](#)Quattrocento [20](#)Roboto Slab [20](#)Source Serif Pro [20](#), [28](#)Spectral [20](#), [28](#)STIX [20](#), [28](#)TeX Gyre Pagella [20](#)TX Fonts [20](#)URW Palladio [ii](#), [23](#), [28](#)Utopia Regular [28](#)`\fontsizeinfo` [6](#), [ℓ148](#)

## forlorn line

club [31](#), [41](#)display widow [42](#)orphan [31](#)widow [31](#), [42](#)**H**hyperref (package) [14](#)hyphenation [10](#)empty discretionary [13](#)first word [11](#)`\hskip` [11](#)re-enable automatic [10](#)hyphenmin (env.) [13](#), [ℓ222](#)**I**information [6](#)

## inverted

exclamation mark [22](#)question mark [22](#)`\itcorr` [ℓ11](#), [15](#), [23](#), [ℓ264](#)`\itcorr*` [15](#)**K**`\kernedhyphen` [17](#), [ℓ289](#), [ℓ311](#), [ℓ312](#),[ℓ319](#), [ℓ320](#)`\kernedhyphen*` [17](#)`\kernedslash` [16](#), [ℓ146](#), [ℓ277](#)`\kernedslash*` [16](#)

## kerning

extra [16](#)forward slash [16](#)hyphen [17](#)ligature [14](#)known problems [53](#)

**L**

label items [24](#)  
 lastlinecenteredpar (env.) [27](#), [ℓ677](#)  
 \lastlinefit [9](#), [33](#), [62](#)  
 lastlinefitpar (env.) [33](#), [62](#), [ℓ731](#)  
 lastlineflushrightpar (env.) [27](#),  
     [ℓ674](#)  
 lastlineraggedleftpar (env.) [27](#),  
     [ℓ668](#)  
 leading [46](#)  
 \leftkernedhyphen [17](#), [ℓ309](#), [ℓ314](#)  
 \leftkernedhyphen\* [17](#)  
 ligature [14](#)  
 ligaturekern (option) [2](#), [14](#)  
 limitations [53](#)  
 line spacing [6](#)  
 list [41](#)  
 loosespacing (env.) [34](#), [ℓ767](#)  
 lowercaselabelitem-  
     adjustments (option) [2](#), [24](#)  
 \lowercaseadjustlabelitems [24](#),  
     [ℓ549](#)  
 lowerslash (option) [2](#), [16](#)

**M**

mathitalicscorrection (option) [3](#),  
     [15](#)  
 microtype (package) [36](#)

**N**

narrow space [36](#)  
 \narrowspace [36](#), [ℓ753](#)  
 \narrowspace\* [36](#)  
 \narrowspacescale [36](#), [ℓ752](#), [ℓ755](#),  
     [ℓ757](#), [ℓ763](#), [ℓ764](#)  
 \narrowpacestrength [36](#), [ℓ751](#),  
     [ℓ760](#)  
 \noadjustlabelitems [24](#), [ℓ556](#)  
 nocharprotrusion (env.) [39](#), [ℓ936](#)  
 nodebug (option) [2](#)  
 nofontexpand (env.) [ℓ933](#)  
 nofontexpansion (env.) [39](#), [ℓ927](#)  
 \nolig [14](#), [ℓ229](#)  
 \nolig\* [14](#)

**O**

openlastlinepar (env.) [32](#), [ℓ715](#)

**P**

package option [2–4](#)  
     breakpenalty [2](#), [14](#), [16](#), [18](#), [19](#)  
     debug [2](#)  
     ligaturekern [2](#), [14](#)  
     lowercaselabelitem-  
         adjustments [2](#), [24](#)  
     lowerslash [2](#), [16](#)  
     mathitalicscorrection [3](#), [15](#)  
     nodebug [2](#)  
     raise\* [3](#)  
     raisecapitaldash [3](#), [19](#)  
     raisecapitalguillemets [3](#), [21](#)  
     raisecapitalhyphen [3](#), [17](#)  
     raisecapitaltimes [3](#), [20](#)  
     raisefiguredash [3](#), [19](#)  
     raiseguillemets [3](#), [21](#)  
     raiseinvertedmarks [3](#), [23](#)  
     shrinklimits [4](#), [38](#)  
     slashkern [4](#), [16](#)  
     stretchlimits [4](#), [38](#)  
     textitalicscorrection [4](#), [15](#)  
     trackingttspacing [4](#), [37](#)  
     uppercaselabelitem-  
         adjustments [4](#), [24](#)  
 page break [10](#), [44](#)  
 paragraph  
     align last line [27](#)  
         centered [27](#)  
         flush right [27](#)  
     badness [9](#)  
     fill last line [29](#)  
         covernextindentpar [32](#)  
         \linebreak [31](#)  
         \mbox [30](#)  
         openlastlinepar [32](#)  
         prolongpar [31](#)  
         shortenpar [31](#)  
         tie [30](#)  
     sloppy [40](#)  
     vertically tied [41](#)  
 PDF [14](#)  
 prolongpar (env.) [31](#), [ℓ691](#)

**Q**

quick reference [v–viii](#)

**R**

ragged right [49](#)  
 raise\* (option) [3](#)  
 raisecapitaldash (option) [3](#), [19](#)  
 raisecapitalguillemets (option) [3](#), [21](#)  
 raisecapitalhyphen (option) [3](#), [17](#)  
 raisecapitaltimes (option) [3](#), [20](#)  
 raised character [18](#)  
   en-dash [19](#)  
   guillemets [20](#)  
   hyphen [18](#)  
   inverted exclamation mark [22](#)  
   inverted question mark [22](#)  
   multiplication sign [20](#)  
   number dash [19](#)  
 raisefiguredash (option) [3](#), [19](#)  
 raiseguillemets (option) [3](#), [21](#)  
 raiseinvertedmarks (option) [3](#), [23](#)  
 reconfigure [5](#)  
 \resetbaselineskip [46](#), [1292](#)  
 \rightkernedhyphen [17](#), [317](#), [322](#)  
 \rightkernedhyphen\* [17](#)

**S**

\setbaselineskip [46](#), [1224](#), [1226](#),  
   [1233](#), [1242](#), [1252](#), [1270](#), [1279](#),  
   [1282](#), [1286](#)  
 \setbaselineskippercentage [46](#),  
   [1305](#)  
 setfontexpand (env.) [38](#), [913](#)  
 setfontshrink (env.) [38](#), [885](#)  
 setfontstretch (env.) [38](#), [899](#)  
 setfonttracking (env.) [37](#), [799](#)  
 \setleading [47](#), [1324](#)  
 \setleadingpercentage [47](#), [1342](#)  
 setup [5](#)  
 shortenpar (env.) [31](#), [684](#)  
 shrinklimits (option) [4](#), [38](#)  
 \Singleguillemetleft [21](#), [404](#)  
 \singleguillemetleft [21](#), [388](#)  
 \Singleguillemetright [21](#), [408](#)  
 \singleguillemetright [21](#), [392](#)  
 slashkern (option) [4](#), [16](#)  
 \slightlyloppy [40](#), [950](#), [1008](#)  
 \slightlyloppypar (env.) [40](#), [1007](#)  
 smoothraggedright (env.) [50](#), [1576](#)  
 \smoothraggedrightfuzzfactor [1482](#),  
   [1517](#)  
 \smoothraggedrightgenerator [1483](#),  
   [1512](#), [1565](#)

\smoothraggedrightleftskip [1484](#),  
   [1513](#)  
 smoothraggedrightpar (env.) [49](#),  
   [1491](#)  
 \smoothraggedrightparindent [1485](#),  
   [1522](#), [1533](#), [1548](#)  
 \smoothraggedrightragwidth [1486](#),  
   [1517](#), [1519](#), [1524](#), [1527](#), [1530](#),  
   [1535](#), [1538](#), [1540](#), [1542](#), [1545](#),  
   [1550](#), [1552](#), [1554](#), [1556](#), [1558](#),  
   [1561](#)  
 smoothraggedrightshape-  
   quintuplet (env.) [49](#), [1403](#)  
 smoothraggedrightshape-  
   septuplet (env.) [49](#), [1441](#)  
 smoothraggedrightshapetriplet  
   (env.) [49](#), [1370](#)  
 \splicevtietop [41](#), [1076](#)  
 stretchlimits (option) [4](#), [38](#)

**T**

\textexclamdown [23](#)  
 textitalicscorrection (option) [4](#),  
   [15](#)  
 \textquestiondown [23](#)  
 tightspacing (env.) [34](#), [783](#)  
 trackingttspace (option) [4](#), [37](#)  
 \typogadjuststairs [25](#), [618](#)  
 \typogadjuststairsfor [596](#)  
 \typogfontsize (dim.) [47](#), [1297](#),  
   [1308](#), [1311](#), [1316](#), [1317](#), [1327](#),  
   [1329](#), [1334](#), [1335](#), [1345](#), [1348](#),  
   [1353](#), [1354](#)  
 \typogget [5](#), [114](#)  
 \typoggetnth [6](#), [117](#)  
 typoginspect (env.) [7](#), [169](#)  
 typoginspectpar (env.) [7](#), [196](#)  
 \typoglogo [11](#)  
 \typoglowercaseadjustcheck [26](#),  
   [661](#)  
 typogsetup (env.) [5](#), [105](#)  
 \typoguppercaseadjustcheck [26](#),  
   [649](#)

**U**

uppercaselabelitem-  
   adjustments (option) [4](#), [24](#)  
 \uppercaseadjustlabelitems [24](#),  
   [542](#)

**V**

`\vtiebot` [42](#), [ℓ1085](#), [ℓ1139](#)  
`\vtiebotdisp` [ℓ1206](#)  
`vtiebotdisp (env.)` [42](#), [ℓ1196](#)  
`vtiebotdisptoppar (env.)` [42](#), [ℓ1200](#)  
`vtiebotpar (env.)` [42](#), [ℓ1138](#)  
`\vtietop` [41](#), [ℓ1018](#), [ℓ1072](#), [ℓ1078](#), [ℓ1204](#)  
`vtietop (enumitem-key)` [42](#)  
`vtietoppar (env.)` [41](#), [ℓ1071](#)

**W**

wide space [35](#)  
`\widespace` [35](#), [ℓ737](#)  
`\widespace*` [35](#)  
`\widespacescale` [35](#), [ℓ736](#), [ℓ739](#), [ℓ741](#),  
[ℓ747](#), [ℓ748](#)  
`\widespacestrength` [35](#), [ℓ735](#), [ℓ744](#)