

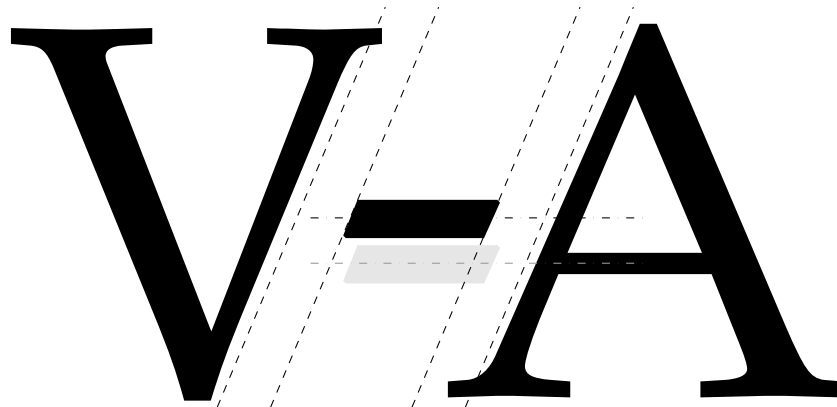
TypoG – Typographic Fine-Tuning

Ch. L. Spiel*

v0.5 2025/04/15

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^AT_EX Project Public License](#) (LPPL), either version 1.3c of this license or – at your option – any later version. This work has the LPPL maintenance status »author-maintained«.

* cspiel@users.sourceforge.org

Contents

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 [6](#)

- 3.1 Setup and Reconfiguration [6](#)
- 3.2 Information [7](#)
 - 3.2.1 Font Information [7](#)
 - 3.2.2 Paragraph- and Page-Breaking Trace [8](#)
- 3.3 Hyphenation [11](#)
- 3.4 Disable/Break Ligatures [15](#)
- 3.5 Manual Italic Correction [16](#)
- 3.6 Apply Extra Kerning or Spacing [17](#)
 - 3.6.1 Slash [17](#)
 - 3.6.2 Hyphen [18](#)
 - 3.6.3 En-Dash and Em-Dash [19](#)
- 3.7 Raise Selected Characters [20](#)
 - 3.7.1 Capital Hyphen [21](#)
 - 3.7.2 Capital Dash [21](#)
 - 3.7.3 Number Dash (Figure Dash) [22](#)
 - 3.7.4 Multiplication Sign [23](#)
 - 3.7.5 Guillemets [23](#)
 - 3.7.6 Inverted Marks [25](#)
- 3.8 Vertically Adjust Label Items [27](#)
- 3.9 Align Last Line [30](#)

Table of Contents continued on next page.

3.10	Fill Last Line	32
3.10.1	Problem Definition	32
3.10.2	Manual Changes	34
3.10.3	Multi-Purpose Environments	35
3.10.4	Specialized Environments	35
3.10.5	Consistent Spacing of Last Line	36
3.11	Spacing	36
3.11.1	Looser/Tighter	36
3.11.2	Wide Space	38
3.11.3	Narrow Space	39
3.12	Microtype Front-End	39
3.12.1	Tracking	40
3.12.2	Font Expansion	41
3.12.3	Character Protrusion	42
3.13	Sloppy Paragraphs	42
3.14	Vertically Partially-Tied Paragraphs	44
3.15	Breakable Displayed Equations	47
3.16	Setspace Front-End	48
3.17	Smooth Ragged	51

4 Limitations and Known Problems 55

5 Other Packages for Fine L^AT_EX Typography 56

A typog-grep 57

B e-_TE_X: Breaking Paragraphs into Lines 64

Table of Contents continued on next page.

C

Package Code 66

C.1	Setup and Reconfiguration	71	C.10	Fill Last Line	93
C.2	Information	71	C.11	Spacing	95
C.3	Hyphenation	73	C.12	Microtype Front-End	96
C.4	Disable/Break Ligatures	74	C.13	Sloppy Paragraphs	100
C.5	Manual Italic Correction	75	C.14	Vert. Tie Paragraphs	102
C.6	Apply Extra Kerning	76	C.15	Breakable Disp. Eqs.	106
C.7	Raise Selected Characters	79	C.16	Setspace Front-End	107
C.8	Vert. Adjust Label Items	85	C.17	Smooth Ragged	110
C.9	Align Last Line	92			

Change History 117**References 119****Index 122****List of Tables**

1	Hyphens and automatic hyphenation	13
2	Suggested raise amounts for <code>\figuredash</code>	23
3	Suggested raise amounts for guillemets	24
4	Label item adjustment suggestions	31
5	Spacing changes made by <code>loosespacing</code>	37
6	Spacing changes made by <code>tightspacing</code>	37
7	<code>\fontdimen<number></code> parameters	38
8	Comparison of some space sizes	40
9	Shrink values of <code>setfontshrink</code>	41
10	Stretch values of <code>setfontstretch</code>	41
11	Shrink and stretch values of <code>setfontexpand</code>	41
12	Parameter adjustments of <code>\slightlyloppy</code>	43
13	Partial paragraph line counts	46
14	Env. <code>breakabledisplay</code> and <code>\interdisplaylinepenalty</code>	48

Quick Reference

We reduce the line spacing in the multi-column parts to 120% with `\setbaselineskippercentage`. List `\item`s 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 pages 2 to 5. The [Index](#) on pages 122 to 126 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.

A		<code>\breakpoint*</code>	Insert an empty discretionary.	14	
<code>\Adjustedlabelitemi</code>	Typeset an uppercase-adjusted <code>\labelitemi</code> .	28	<code>\breakpoint</code>	Insert an empty discretionary and re-enable automatic hyphenation.	14
<code>\adjustedlabelitemi</code>	Typeset a lowercase-adjusted <code>\labelitemi</code> .	28	C		
<code>\Adjustedlabelitemii</code>	Typeset an uppercase-adjusted <code>\labelitemii</code> .	28	<code>\capitaldash*</code>	Alias for <code>\capitalendash*</code> .	21
<code>\adjustedlabelitemii</code>	Typeset a lowercase-adjusted <code>\labelitemii</code> .	28	<code>\capitaldash</code>	Alias for <code>\capitalendash</code> .	21
<code>\Adjustedlabelitemiii</code>	Typeset an uppercase-adjusted <code>\labelitemiii</code> .	28	<code>\capitalemdash*</code>	Typeset a vertically adjusted <code>\textemdash</code> .	22
<code>\adjustedlabelitemiii</code>	Typeset a lowercase-adjusted <code>\labelitemiii</code> .	28	<code>\capitalemdash</code>	Typeset a vertically adjusted <code>\textemdash</code> that is hyphenatable.	22
<code>\Adjustedlabelitemiv</code>	Typeset an uppercase-adjusted <code>\labelitemiv</code> .	28	<code>\capitalendash*</code>	Typeset a vertically adjusted <code>\textendash</code> .	21
<code>\adjustedlabelitemiv</code>	Typeset a height-adjusted <code>\labelitemiv</code> .	28	<code>\capitalendash</code>	Typeset a vertically adjusted <code>\textendash</code> that is hyphenatable.	21
<code>\allowhyphenation</code>	(Re-)enable automatic hyphenation.	11	<code>\capitalhyphen*</code>	Typeset a vertically adjusted hyphen character.	21
B		<code>\capitalhyphen</code>	Typeset a vertically adjusted hyphen character that is hyphenatable.	21	
<code>breakabledisplay</code> [<i>⟨level⟩</i>]	Adjust penalty associated with <code>\allowdisplaybreaks</code> .	47	<code>\capitalinverted-exclamationmark</code> { <i>⟨number⟩</i> }	Typeset an inverted (upside-down) exclamation mark that is level with the baseline.	26

- `\capitalinverted-questionmark`{*<number>*}
Typeset an inverted (>upside-down<) question mark that is level with the baseline. 26
- `\capitaltimes`
Typeset a vertically adjusted `\texttimes`. 23
- `covernextindentpar`[*<dim>*]
Extend the last line of a paragraph. 35
- D**
- `\Doubleguillemetleft`
Typeset left double guillemets vertically adjusted for uppercase. 24
- `\Doubleguillemetright`
Typeset right double guillemets vertically adjusted for uppercase. 24
- `\doubleguillemetleft`
Typeset left double guillemets vertically adjusted for lowercase. 24
- `\doubleguillemetright`
Typeset right double guillemets vertically adjusted for lowercase. 24
- F**
- `\figuredash*`
Typeset a `\textendash` vertically adjusted for figures (numerals). 22
- `\figuredash`
Typeset a hyphenatable `\textendash` vertically adjusted for figures (numerals). 22
- `\fontsizeinfo`{*<cs-name>*}
Store the current em-height and `\baselineskip` in a pair of macros. 7
- H**
- `hyphenmin`[*<left-min>*]{*<right-min>*}
Set the values of `\lefthyphenmin` and `\righthyphenmin`. 14
- I**
- `\itcorr`{*<strength>*}
Apply italic correction in the for of a `\kern` scaled by `\fontdim1` or `\textitalicscorrection`. 16
- `\itcorr`{*<strength>*}
Apply italic correction in the for of a `\kern` scaled by `\fontdim1` or `\textitalicscorrection`. 16
- K**
- `\kernedhyphen*`[*<raise>*]{*<left-kern>*}{*<right-kern>*}
Typeset an unbreakable hyphen and apply kerning to its left and right. 18
- `\kernedhyphen`[*<raise>*]{*<left-kern>*}{*<right-kern>*}
Typeset a breakable hyphen and apply kerning to its left and right. 18
- `\kernedslash*`
Typeset an unbreakable forward slash and apply kerning to its left and right. 17
- `\kernedslash`
Typeset a breakable forward slash and apply kerning to its left and right. 17
- L**
- `lastlinecenteredpar`
Center the last lines of a paragraph. 32
- `lastlinefitpar`
Match spacing of last line and next-to-last line. 36
- `lastlineflushrightpar`
Alias for `lastlineraggedleftpar`. 30
- `lastlineraggedleftpar`
Align the last line of paragraph flush-right. 30
- `\leftkernedhyphen*`[*<raise>*]{*<left-kern>*}
Typeset a hyphen and apply kerning to its left-hand side. 18
- `\leftkernedhyphen`[*<raise>*]{*<left-kern>*}
Typeset a hyphen, apply kerning to its left-hand side, and insert a breakpoint after it. 18
- `\leftspaceddash*`[*<raise>*]
Alias for `\leftspacedendash*`.
- `\leftspaceddash`[*<raise>*]
Alias for `\leftspacedendash`.

`\leftspacedendash*[\langle raise \rangle]`
Typeset an en-dash with some space around it. Prohibit line-breaks before and after it.

`\leftspacedendash[\langle raise \rangle]`
Typeset an en-dash with some space around it. Allow line-breaks at its left-hand side.

`\loosespacing[\langle level \rangle]`
Increase the width of the space character. 37

`\lowercaseadjustlabelitems{\langle levels \rangle}`
Activate the lowercase height-adjustment values inside `itemize` environments. 27

N

`\narrowspace*`
Typeset a narrow space whose width depends on `\fontdimen7`. 39

`\narrowspace`
Typeset a narrow space whose width depends on `\fontdimen7` or `\fontdimen2`. 39

`\noadjustlabelitems{\langle levels \rangle}`
Deactivate height-adjustment of label items. 27

`nocharprotrusion`
Deactivate character protrusion. 42

`nofontexpand`
Alias for `nofontexpansion`. 42

`nofontexpansion`
Deactivate font expansion. 42

`\nolig*[\langle kerning \rangle]`
Break a ligature. 15

`\nolig[\langle kerning \rangle]`
Break a ligature and introduce a hyphenation opportunity. 15

O

`openlastlinepar[\langle dim \rangle]`
Open a paragraph's last line if it is almost full or completely filled. 35

P

`prolongpar`
Increase the `\looseness` of a paragraph. 35

R

`\resetbaselineskip`
Reset the `\baselineskip` to its original value. 49

`\rightkernedhyphen*[\langle raise \rangle]{\langle right-kern \rangle}`
Typeset a hyphen and apply kerning to its right-hand side. 18

`\rightkernedhyphen[\langle raise \rangle]{\langle right-kern \rangle}`
Typeset a hyphen, apply kerning to its right-hand side, and insert a breakpoint after it. 18

`\rightspaceddash*[\langle raise \rangle]`
Alias for `\rightspacedendash*`.

`\rightspaceddash[\langle raise \rangle]`
Alias for `\rightspacedendash`.

`\rightspacedendash*[\langle raise \rangle]`
Typeset an en-dash with some space around it. Prohibit line-breaks before and after it.

`\rightspacedendash[\langle raise \rangle]`
Typeset an en-dash with some space around it. Allow line-breaks at its right-hand side.

S

`\setbaselineskip-
percentage{\langle percentage \rangle}`
Set `\baselineskip` as percentage relative to font design size. 49

`\setbaselineskip{\langle baselineskip \rangle}`
Set `\baselineskip` using an absolute length. 48

`setfontexpand[\langle level \rangle]`
Simultaneously set font stretch and shrink limits. 41

`setfontshrink[\langle level \rangle]`
Set font shrink limits. 41

`setfontstretch[\langle level \rangle]`
Set font stretch limits. 41

`setfonttracking{\langle delta \rangle}`
Override the default tracking for all fonts. 40

`\setleadingpercentage{\langle percentage \rangle}`
Set `\baselineskip` as percentage via the leading. 49

<code>\setleading{⟨leading⟩}</code>	Set <code>\baselineskip</code> via the leading.	49	<code>\singleguillemetleft</code>	Typeset left single guillemets vertically adjusted for lowercase.	24
<code>shortenpar</code>	Decrease the <code>\looseness</code> of a paragraph.	35	<code>\singleguillemetright</code>	Typeset right single guillemets vertically adjusted for lowercase.	24
<code>\Singleguillemetleft</code>	Typeset left single guillemets vertically adjusted for uppercase.	24	<code>slightlyloppypar[⟨sloppiness⟩]</code>	Format a paragraph with given sloppiness.	43
<code>\Singleguillemetright</code>	Typeset right single guillemets vertically adjusted for uppercase.	24	<code>\slightlyloppy[⟨sloppiness⟩]</code>	Format with given sloppiness.	43
<hr/>					
<code>smoothraggedrightpar[⟨option⟩...]</code>	Format a paragraph with one of the three smooth-ragged-right generators.	52			
<code>smoothraggedrightshapequintuplet[⟨option⟩...]{⟨width1⟩}...{⟨width5⟩}</code>	Prescribe five line lengths for formatting paragraphs.	51			
<code>smoothraggedrightshapeseptuplet[⟨option⟩...]{⟨width1⟩}...{⟨width7⟩}</code>	Prescribe seven line lengths for formatting paragraphs.	51			
<code>smoothraggedrightshapetriplet[⟨option⟩...]{⟨width1⟩}{⟨width2⟩}{⟨width3⟩}</code>	Prescribe three line lengths for formatting paragraphs.	51			
<code>smoothraggedright[⟨option⟩...]</code>	Format with one of the three smooth-ragged-right generators.	53			
<hr/>					
<code>\spacedcapitalemdash</code>	Typeset a height-adjusted em-dash with some space around it.	22	<code>\spacedendash[⟨raise⟩]</code>	Alias for <code>\rightspacedendash</code> .	19
<code>\spacedcapitalendash*</code>	Typeset a height-adjusted en-dash with some space around it. Prohibit line breaks before and after the dash.	22	<code>\spacedemdash[⟨raise⟩]</code>	Typeset an em-dash with some space around it.	20
<code>\spacedcapitalendash</code>	Typeset a height-adjusted em-dash with some space around it.	22	<code>\spacedendash*[⟨raise⟩]</code>	Typeset an en-dash with some space around it. Prohibit line breaks before and after the dash.	19
<code>\spaceddash*[⟨raise⟩]</code>	Alias for <code>\rightspacedendash*</code> .	19	<code>\spacedendash[⟨raise⟩]</code>	Typeset an em-dash with some space around it.	19
<code>\spaceddash[⟨raise⟩]</code>	Alias for <code>\rightspacedendash</code> .	19	<code>\splicevtietop{⟨lines⟩}</code>	Inside of a <code>list</code> -like environment fuse the first lines.	44
<code>\spacedendash*[⟨raise⟩]</code>	Alias for <code>\rightspacedendash*</code> .	19			

T

<code>tightspacing</code> [<i><level></i>]	
Decrease the width of the space character.	37
<code>\typogadjuststairs</code> [<i><factor></i>] ▶ ◀ [<i><step></i>] { <i><count></i> } { <i><sample></i> }	
Generate ›stairs‹ of vertically shifted label items.	29
<code>\typogfontsize</code>	
Default font's quad size.	50
<code>\typoggetnth</code> { <i><dest></i> } { <i><key></i> } { <i><index></i> }	
Retrieve single item of a typog compound configuration value.	7
<code>\typogget</code> { <i><key></i> }	
Retrieve a typog configuration value.	6
<code>typoginspectpar</code> [<i><option></i>] { <i><id></i> }	
Turn on tracing of paragraphs and pages for a paragraph.	8
<code>typoginspect</code> [<i><option></i>] { <i><id></i> }	
Turn on tracing of paragraphs and pages.	8
<code>\typoglowercaseadjust-check</code> [<i><factor></i>] { <i><sample></i> }	
Typeset all four label items adjusted for lowercase with an indicator line.	30
<code>typogsetup</code> { <i><keys></i> }	
Configure package typog.	6
<code>\typoguppercaseadjust-check</code> [<i><factor></i>] { <i><sample></i> }	
Typeset all four label items adjusted for uppercase with an indicator line.	30

U

<code>\uppercaseadjustlabelitems</code> { <i><levels></i> }	
Activate the uppercase height-adjustment values inside <code>itemize</code> environments.	27

V

<code>vtiebotdisptoppar</code> [<i><num-before-lines></i>] ▶ ◀ [<i><num-after-lines></i>]	
Fuse a display with its preceding and following lines.	45
<code>vtiebotdisp</code> [<i><num-lines></i>]	
Fuse a display with its preceding lines.	45
<code>vtiebotpar</code> [<i><num-lines></i>]	
Fuse the last lines of a paragraph.	45
<code>\vtiebot</code> [<i><num-lines></i>]	
Fuse last lines.	45
<code>vtietoppar</code> [<i><num-lines></i>]	
Fuse the first lines of a paragraph.	44
<code>vtietop</code> [<i><num-lines></i>]	
Fuse first lines.	44
<code>\vtietop</code> [<i><num-lines></i>]	
Fuse first lines.	44

W

<code>\widespace*</code>	
Typeset a wide space whose width depends on <code>\fontdimen7</code> .	38
<code>\widespace</code>	
Typeset a wide space whose width depends on <code>\fontdimen7</code> or <code>\fontdimen2</code> .	38

1 Introduction

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

L^AT_EX 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 3.1 presents how to reconfigure package typog after has been loaded and how to access its configuration values for introspection or inclusion in the user's own code.

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^AT_EX.

Sections 3.4 and 3.5 treat the breaking of ligatures and also manually applying italic correction – in a generalized way.

Section 3.6 introduces macros to kern or space some punctuation signs.

Section 3.7 deals 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 `sloppy`.

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 e-_T_EX; it relies on the L^AT_EX3 interface. Parts of it are based on package microtype. However, if the respective functionality is not used, typog can be used without microtype. The same holds true for the setspace front-end.

The package was tested with pdf_T_EX 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`.

`emdashspace=⟨glue⟩`

SINCE V0.5

Set the horizontal skip that is inserted before and after the em-dash in macros `\spacedemdash` and `\spacedcapitaldash` to `⟨glue⟩`. Default value: $\frac{56}{1000}$ em.

`endashspace=⟨glue⟩`

SINCE V0.5

Set the horizontal skip that is inserted before and after the en-dash in macros `\spacedendash` and `\spacedcapitalendash` to `⟨glue⟩`. Default value: $\frac{250}{1000}$ em plus $\frac{50}{1000}$ em minus $\frac{10}{1000}$ em.

`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⟩}`

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 `*` 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. 31) and also configuration option `uppercaseadjustlabelitem`.

All four lengths default to 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.

Important

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

SINCE V0.5

`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.

`mathitalicscorrection=<dim>`

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

`raise*=<dim>`

Set the length by which selected characters (dash, hyphen, times, and number dash) are raised. Default value: 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 `\capitalinvertedexclamation-`

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

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

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

`mark` 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 $\langle dim-N \rangle$ 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 \star instructs typog to preserve $\langle dim-N \rangle$ at that position. Default values: 0 pt, 0 pt, 0 pt.

`shrinklimits`={ $\langle limit-1 \rangle$, $\langle limit-2 \rangle$, $\langle limit-3 \rangle$ } microtype req.

`stretchlimits`={ $\langle limit-1 \rangle$, $\langle limit-2 \rangle$, $\langle limit-3 \rangle$ } 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 $\langle limit-\# \rangle$ values replace old ones. If one or more limits of the triple should remain unchanged pass a \star 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`= $\langle dim \rangle$

Set the size of the kerns before and after `\kernedslash`. See Sec. 3.6.1. Default value: $\frac{50}{1000}$ em.

`textitalicscorrection`= $\langle dim \rangle$

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`={ $\langle outer-spacing \rangle$ } 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 $\langle outer-spacing \rangle$ gets passed to microtype's `\SetTracking` option `outer spacing` [23, 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`={ $\langle dim-1 \rangle$, ..., $\langle dim-4 \rangle$ }

SINCE V0.4

Vertical shifts $\langle dim-N \rangle$ to apply to `\labelitem<N>`, where $\langle N \rangle$ =1, 2, 3, or 4 is the nesting level of the `itemize` list. Empty list elements are ignored. The special value \star instructs typog to preserve $\langle dim-N \rangle$ at that position. The adjustments apply to the uppercase setting (`\uppercaseadjustlabelitems`). See Sec. 3.8 (in particular subsection »Setup« and Tab. 4 on p. 31) and also configuration option `lowercaseadjustlabelitem`.

All four lengths default to 0 pt.

Important

Configuring `uppercase-label-itemadjustments` (or `lowercase-label-itemadjustments`) does *not* activate the correction mechanism. Use one of the macros `\uppercaseadjust-label-items` or `\lowercaseadjust-label-items` 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 [30]

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

```

\fontsizeinfo{<cs-name>}

```

The call to `\fontsizeinfo` introduces a pair of macros to access the stored values. The unstarred version `\cs-name` expands to the lengths including their

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

³ 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[=`\langle size \rangle`]

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

```
<typog-inspect_id="\langle id \rangle" _job="\langle jobname \rangle" _line="\langle line-number \rangle" _page="\langle page-number \rangle">
...
</typog-inspect>
```

where the $\langle id \rangle$ is the user-supplied, unique⁴ 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 $\langle regexp \rangle$) 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^AT_EX engine with a larger log-file line length, to prevent wrapped lines. With short lines the wannabe XML opening tags can get wrapped and thus become unrecognizable to dumb post-processors. To avoid wrapped lines prepend

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

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

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

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

to the respective command-line.

- If more trace information is needed just add `\tracing...` calls right after `\begin{typoginspect}` or `\begin{typoginspectpar}`.
- 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 $\langle id \rangle$. Think of `\captions` or any theorem-like environment and their associated, unique `\labels`. ■

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

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_EX’s line-break information with `\tracingparagraphs=1`⁵ or simply use the `typoginspect` environment and check the suffixes

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

of each line in the paragraph, where for `⟨suffix⟩` the following mapping holds [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`.⁶
5. The best⁷ 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 thousandths of the current font’s em) like, e. g.,

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

or

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

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

⁵ Reference 29 provides an exceptionally detailed discussion of the output of `\tracingparagraphs`.

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

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

Investigating page-breaks. Use `\tracingpages=1` or the `typoginspect` environment to switch on tracing of T_EX's page-break information [15, p. 112n].⁸

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

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

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

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

For every vertical breakpoint T_EX records

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

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

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

Example

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

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

3.3 Hyphenation

T_EX's and thus L^AT_EX'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_EX inhibits breaks of the component words by default. The following macro rectifies the problem.

```
\allowhyphenation
```

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

The admittedly simple rules when T_EX 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-1/2<`

⁸ See also the discussion of the T_EX output routines by SOLOMON [25].

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. 13.
- 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_EX 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_EX does not hyphenate a word that is hyphenatable despite the result is an overfull box.⁹ ■

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

⁹ 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_EX-savant is highly welcome.

¹⁰ `\babelhyphenation` is the multi-lingual extension of T_EX's `\hyphenation` and it is defined in package babel [5].

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

\LaTeX -Code	Result	Note
hyphenated-compound	hyphenated- compound	Most frequently used code; the <code>hyphen_</code> expands to <code>\discretionary{-}{-}{-}</code> rendering the parts un-breakable
hyphenated\mbox{-}% compound	hyphenated-compound	Suppress hyphenation with the <code>\mbox</code> in the compound
\mbox{hyphenated-% compound}	hyphenated-compound	Avoid line break and thus hyphenation
hyphenated\hyp compound	hy- phen- ated- com- pound	Macro <code>\hyp</code> defined in package <code>hyphenat</code> [36]
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^AT_EX suppresses any hyphenation for fonts in `\ttfamily` by un-defining their `\hyphenchars`. If these are reassigned, the usual hyphenation occurs again.

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

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

`\breakpoint`
`\breakpoint*`

The empty discretionary construct [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^AT_EX macro `\allowbreak` is not only that the former has a starred form, but the penalty associated with `\breakpoint` is the current¹¹ `\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⟩`.¹²

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. ■

¹¹ At this point in the document `\exhyphenpenalty=50` holds.

¹² 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^AT_EX will trip with »Undefined control sequence« on `\typog@missing@-` argument if the extra argument is not passed to `\leftspacedendash`, `\rightspacedendash`, or any of its aliases in these situations.

Alternatively use `\texorpdfstring` [22, 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 T_EX or L^AT_EX 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`¹³ 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 side¹⁴ of italics, i. e., compensate »shift-to-the-right effect« of italics. ¶ Positive correction at the left-hand side of italics, e.g., an opening parenthesis or square bracket followed by an italic *f* (before: 8, after: 7) or *y* (before: 4, after: 1) reaching far to the left below the baseline. ■

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

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

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

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

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

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

¹⁴ Groff has the machinery for left-italic-correction. Its font-metrics files support per glyph left-italic-correction values and users can access them conveniently via `_`.

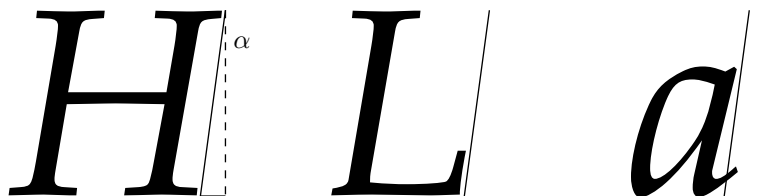


FIGURE 1: Some letters of an italics font. We use the capital `H` to measure the angle α between the plumb-line (drawn dashed) and a tangent to the rightmost parts of the glyph. The length of the plumb-line is proportional to `<strength>` and the short, thick part of the baseline symbolizes the resulting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

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

3.6 Apply Extra Kerning or Spacing

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 and dashes.

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`.

OPTION `lowerslash`
INTRODUCED IN V0.5

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
\end{typogsetup}}
```

Generic solutions could build upon `\fontdimen5` and `\fontdimen6` (see Tab. 7 on p. 38) or measure the height and depth of the slash-glyph (abstracted

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` [36]) 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 `raisecapital-hyphen`. 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*`

3.6.3 En-Dash and Em-Dash

`\leftspacedendash`
`\leftspaceddash`
`\leftspacedendash*`
`\leftspaceddash*`
`\rightspacedendash`
`\rightspaceddash`
`\spacedendash`
`\spaceddash`
`\rightspacedendash*`
`\rightspaceddash*`
`\spacedendash*`
`spaceddash*`
ALL SINCE V0.5

The macros `\leftspacedendash`, `\leftspacedendash*`, `\rightspacedendash`, and `\rightspacedendash*` all expand to an en-dash “–” that is surrounded by whitespace. They differ in the ways they handle line breaking before or after the en-dash.

A *single* en-dash is used – among many other cases – to indicate a pause (e.g. announcing an action to be continued), instead of a comma, or even instead of a period. Here, the typographic rules favor either no line break at all or a break at the right-hand side of the en-dash.

A *pair* of en-dashes is used for appositions or explanatory insertions (as exemplified in the first sentence of the previous paragraph). In this instance there are two camps: One emphasizes the insertion and thus couples the lead-in and lead-out en-dashes to it (see item 1 below). The other avoids a dash at the beginning of a line at the cost of severing it from the insertion (item 2).

<code>\leftspacedendash[⟨raise⟩]</code>	<code>\leftspaceddash (alias)</code>
<code>\leftspacedendash*[⟨raise⟩]</code>	<code>\leftspaceddash* (alias)</code>
<code>\rightspacedendash[⟨raise⟩]</code>	<code>\rightspaceddash (alias)</code>
	<code>\spacedendash (alias)</code>
	<code>\spaceddash (alias)</code>
<code>\rightspacedendash*[⟨raise⟩]</code>	<code>\spacedendash* (alias)</code>
	<code>\spaceddash* (alias)</code>

Typeset an unbreakable en-dash with any of the starred variants.

The distinction between `\leftspacedendash` and `\rightspacedendash` and the definition of the alias `\spacedendash` for `\rightspacedendash` caters two different typographic preferences.

1. One rule set prefers insertions to be preceded and followed by dashes. The elementary way to solve this is to code

before ---insertion--- after

where we used the unbreakable space “ ” as example. For this solution typog offers `\leftspacedendash` and `\rightspacedendash`:

before \leftspacedendash insertion \rightspacedendash after

2. Another camp insists to avoid dashes at the start of lines. Now the simple solution looks like this:

before--- insertion--- after

For this scenario both dashes behave like `\rightspacedendash` and the alias `\spacedendash` comes in handy:

before \spacedendash insertion \spacedendash after

For convenience all spaced-dash macros eliminate white-space to their left. (Technically they issue an `\unskip` right at the beginning.)

Configure the size of the space before and after the dash with option `en-dashspace`.

The optional argument $\langle raise \rangle$ allows to adjust the height of the en-dash just as the macros in Sec. 3.7; it is meant for one-off solutions. If an en-dash is needed that is both raised and spaced prefer macro `\spacedcapitalendash`.

Important — hyperref bookmarks

If any of `\leftspacedendash`, `\rightspacedendash`, `\spacedendash`, `\spaceddash`, or `\spacedemdash` (see below) – whether starred or unstarred – occur 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 `\{\}`.

We showcase the modifications of the plain calls for macro `\spaceddash`:

```
\spaceddash*[\langle raise \rangle]\relax   \spaceddash[\langle raise \rangle]\relax
\spaceddash*[\langle raise \rangle]\{\}   \spaceddash[\langle raise \rangle]\{\}
```

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

\LaTeX will trip with »Undefined control sequence« on `\typog@missing@`-argument if the extra argument is not passed to `\leftspacedendash`, `\rightspacedendash`, or any of its aliases in these situations.

Also note that space to the left of the macros is not gobbled in PDF-bookmarks.

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

`\spacedemdash`
SINCE V0.5

Macro `\spacedemdash` expands to an em-dash `—` that is surrounded by whitespace.

```
\spacedemdash[\langle raise \rangle]
```

Typeset an unbreakable em-dash with `\spacedemdash`. The size of the space before and after the dash is configured with option `emdashspace`.

The optional argument $\langle raise \rangle$ allows to adjust the height of the em-dash just as the macros in Sec. 3.7; it is meant for one-off solutions. If an em-dash is needed that is both raised and spaced prefer macro `\spacedcapitalemdash`.

The same *Important note* on `hyperref` bookmarks holds for `\spacedemdash` as for `\spaceddash`.

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. Readers familiar with OpenType fonts will be reminded of the case feature. Users can base their own definitions of raised characters on their associated dimensions.¹⁵

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` 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, `\capitalhyphen*` this may be too low for compounds involving capitals.

```
\capitalhyphen      \capitalhyphen*
```

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

Use Cases

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

3.7.2 Capital Dash

`\capitalendash` 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`
`\capitaldash*`

```
\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`.

¹⁵ Also compare with Ex. 12 in Ref. 35 for an attempt to automate vertical alignment.

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. ■

`\spacedcapitalemdash`
`\leftspacedcapitalendash`
`\leftspacedcapitaldash`
`\leftspacedcapitalendash*`
`\leftspacedcapitaldash*`
`\rightspacedcapitalendash`
`\rightspacedcapitaldash`
`\rightspacedcapitalendash*`
`\rightspacedcapitaldash*`

Combine raising a dash and adding some extra space around it.

`\spacedcapitalemdash`
`\leftspacedcapitalendash` `\leftspacedcapitaldash` (alias)
`\leftspacedcapitalendash*` `\leftspacedcapitaldash*` (alias)
`\rightspacedcapitalendash` `\rightspacedcapitaldash` (alias)
`\spacedcapitalendash` (alias)
`\spacedcapitaldash` (alias)
`\rightspacedcapitalendash*` `\spacedcapitalendash*` (alias)
`\spacedcapitaldash*` (alias)

ALL SINCE V0.5

These macros unite the features of `\capitalemdash`, `\capitalendash`, and `\capitalendash*`, respectively with those of `\spacedendash` as described in Sec. 3.6.3.

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

`\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]`).

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
Charis SIL, fbb*, Source Serif Pro	.05
Libre Baskerville, Crimson Pro, Erewhon, Droid Serif	.0667
GFS Artemisia, Libre Caslon, Coelacanth, Crimson Pro, Crimson Text, T _E 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 ^x

`\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 T1¹⁶ font encoding not on package babel.

¹⁶ 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
Charis SIL	.0667	.0333
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.

<code>\capitalinvertedexclamationmark</code> <code>\capitalinvertedquestionmark</code>	The macros <code>\capitalinvertedexclamationmark</code> and <code>\capitalinvertedquestionmark</code> simultaneously provide bottom-aligned inverted marks for up to three different sets of exclamation marks or question marks, e. g., for uppercase, med-caps, and small-caps text ¹⁷ or just for some stylistic alternatives offered by particularly well equipped fonts.
---	--

BOTH SINCE V0.5

17 The small exclamation and question marks even made it into Unicode.

```
\capitalinvertedexclamationmark{⟨number⟩}
\capitalinvertedquestionmark{⟨number⟩}
```

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.

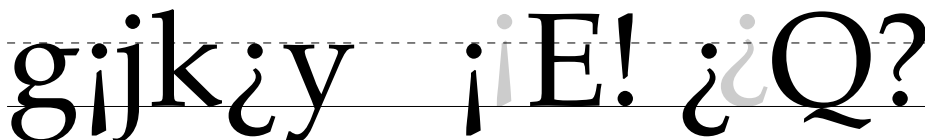


FIGURE 2: The left-hand sample shows the inverted exclamation mark and question marks wedged between some lowercase letters. The baseline is drawn solid and the ex-height as well as the descender depth are indicated with dashed lines. ¶ On the right hand side we display the same marks in conjunction 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 nonzero dimension raises the marks by exactly that length. To get a (quasi-)zero length use e. g. 1 sp.

Tip

The letters `V`, `W`, and `Y` following an inverted question mark may need some manual kerning as well as `J` following an inverted exclamation mark. Macro `\itcorr` (p. 16) 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{⟨number⟩}` and `\capitalinvertedquestionmark{⟨number⟩}` 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^AT_EX 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.¹⁸

`\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.

¹⁸ 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 $\frac{1}{2}$.¹⁹ 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. ■

Sometimes uppercase-adjusted or lowercase-adjusted label items are needed outside of `itemize` environments.

<code>\Adjustedlabelitemi</code>	<code>\adjustedlabelitemi</code>
<code>\Adjustedlabelitemii</code>	<code>\adjustedlabelitemii</code>
<code>\Adjustedlabelitemiii</code>	<code>\adjustedlabelitemiii</code>
<code>\Adjustedlabelitemiv</code>	<code>\adjustedlabelitemiv</code>

Typeset label items that are height-adjusted according to `uppercaseadjustments` or `lowercaseadjustments` either for uppercase (macros starting with a capital `A`) or lowercase (macros starting with

¹⁹ 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,‹ [26, p. 220].

`\Adjustedlabelitemi`
`\Adjustedlabelitemii`
`\Adjustedlabelitemiii`
`\Adjustedlabelitemiv`
`\adjustedlabelitemi`
`\adjustedlabelitemii`
`\adjustedlabelitemiii`
`\adjustedlabelitemiv`
ALL EIGHT SINCE V0.5

a small `a`). These macros can be used anywhere. They are *not* controlled by `\uppercaseadjustlabelitems`, `\lowercaseadjustlabelitems`, nor `\noadjustlabelitems`.

Use Cases

Free, user-defined lists. ¶ Injection of typog's height-adjustment functionality in other packages as, for example, tasks [21]. ■

`\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 $\langle\text{step-size}\rangle$ and a high $\langle\text{number-of-steps}\rangle$, for an easy refinement.

```
\typogadjuststairs[ $\langle\text{scale-factor}\rangle=.5$ ]
                  { $\langle\text{step-size}\rangle$ } { $\langle\text{number-of-steps}\rangle$ }
                  { $\langle\text{sample}\rangle$ }
```

Generate stairs of $\langle\text{number-of-steps}\rangle$ vertically shifted label items; use the next odd number, if $\langle\text{number-of-steps}\rangle$ is even. Draw a reference hairline at $\langle\text{scale-factor}\rangle$ times the height of $\langle\text{sample}\rangle$, where $\langle\text{scale-factor}\rangle$ defaults to .5. The stairs start at a vertical shift of

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

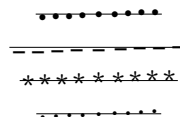
and repeat up

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

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 $\langle\text{sample}\rangle$.

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`
`\typoglowercaseadjustcheck`
BOTH SINCE V0.4

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 `\labelitemadjustments` and `\lowercaselabelitemadjustments` is preferably done at a high magnification (400% to 600% on a 100 dpi screen) with a representative sample of initial letters.

```
\typoguppercaseadjustcheck[⟨scale-factor⟩=.5]{⟨sample⟩}
\typoglowercaseadjustcheck[⟨scale-factor⟩=.5]{⟨sample⟩}
```

Typeset all four label items adjusted for uppercase or for lowercase with an indicator line at `⟨scale-factor⟩` times the `⟨sample⟩`'s actual height. The default `⟨scale-factor⟩` 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, `⟨sample⟩` does not get printed.

Example

Uppercase check with `\typoguppercaseadjustcheck{ABCXYZ}`:

ABGH•••QWYZ, 0123•••4567

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

ace•••mno, bdf•••gij, 0123•••4567,

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

In Table 4 on p. 31 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 \bullet , --- , \ast , and ~ , respectively.

3.9 Align Last Line of a Paragraph

The usual algorithms of L^AT_EX typeset the last line of a paragraph flush with the left margin unless `center`, `raggedleft` or `Centering`, `FlushRight` (package `ragged2e` [24]) 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.

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

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

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
Aleo	.75	.75	-1.5	.75	-.25	-.5	-2.75	-.325
Charis SIL	.375	1.0	-.75	.25	-.75	.0	-1.75	-.875
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 kpfonts.

§ Contained in package mathpazo.

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

`\lastlinecenteredpar`
(*env.*)

The name `\lastlineflushrightpar` is an alias for `\lastlineraggedleftpar`.

Center the last lines of the paragraphs enclosed by this environment.²⁰

```
\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. ■

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` [32, 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-left²¹ setting, e. g., `amsmath` [2] and option `fleqn`.²²

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`.



²⁰ Also compare the approach taken in Ref. 32.

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

²² For displayed equations and `amsmath` the relevant parameter is `\mathindent`.

2. `\parindent = 0` [32, 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.²³



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`²⁴ 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.

\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 \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. ■

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

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

²⁴ For example, \LaTeX 's class `article` uses a `\parindent` of 25 pt.

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 `\tie`

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`.

- (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 [\[34\]](#). In turn the next-to-last lines may become unsightly. Counteract this degradation e. g. with recipes [2a](#) to [2c](#).

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

2. Uniform paragraph change.

- (a) Vary spacing.

Modify the inter-word spacing, for example, with the macros introduced in Sec. [3.11.1](#).

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 \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 line²⁵ or a club line²⁶ [15, p. 104 and 19]. (See also Sec. 3.14 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.²⁷ 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.

`covernextindentpar` (*env.*) Environment `covernextindentpar` can be helpful for [case 01](#), i. e., a too short last line.

```
\begin{covernextindentpar} [⟨dim⟩]
...
\end{covernextindentpar}
```

The environment asks `TEX` to extend the last line of a paragraph such that it takes at least `2\parindent` (if `\parindent ≠ 0`), `2em` (if `\parindent = 0`), or `⟨dim⟩` if called with an optional argument.

`openlastlinepar` (*env.*) The next environment, `openlastlinepar`, takes care of [case 02](#), i. e., a last line in a paragraph that is almost full or completely filled.

```
\begin{openlastlinepar} [⟨dim⟩] ... \end{openlastlinepar}
```

It may resolve [case 02](#) as it attempts to prevent a completely filled line by introducing a partly unshrinkable `\parfillskip`. Without optional argument the

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.

27 Command `\looseness` is a `TEX` primitive [15, p. 103n]. A thorough discussion of the interaction of `\linepenalty` and `\looseness` can be found in Ref. 31.

threshold of unused last-line length is either 2\parindent (if $\text{\parindent} \neq 0$) or 2em (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.

3.10.5 Consistent Spacing of Last Line

Since e-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].²⁸ 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. e-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^AT_EX. 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]

²⁸ The actual algorithm is somewhat involved. So, for ease of reference, we have included the relevant part of the e-TeX-manual as Appendix B on p. 64.

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

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

`loosespacing (env.)`

`tightspacing (env.)`

Environments `loosespacing` and `tightspacing` introduce four $\langle level \rangle$ s of $\rangle looseness \langle$ or $\rangle tightness \langle$, 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	
≥ 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	
≥ 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 `\narrowospace` derive their appearances from several of the current font's `\fontdimen\langle number \rangle`s. T_EX addresses the latter by integers, which is totally non-memnonic. Therefore, we play softball by first presenting Tab. 7 that associates the `\fontdimen\langle number \rangle`s with their meanings and also reports on their current values (for this document).²⁹

#	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° height	47.5
6	<code>\quad</code> height	100
7	Extra space width	3.9

TABLE 7: All T_EX font parameters normalized to the font's quad-size. The first column $\#$ states the index of the `\fontdimen\langle number \rangle` parameter: $\langle number \rangle$. Column 2 presents short descriptions of `\fontdimen\langle number \rangle`. 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`.

²⁹ The association is given in Appendix F (p. 433) of Ref. 15. For a concise and understandable explanation of the T_EX `\fontdimen` parameters consult Ref. 9.

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

The `\widespacescale` too can be redefined by the user to achieve different effects.

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

Use Case

Useful as a sentence-ending space if, for example, the sentence ends in an abbreviation with a period or decimal number without trailing digits *and* the next sentence should be delimited in a clearer way. ¶ Open tight lines with a series of `\widespaces`.³⁰ ■

3.11.3 Narrow Space

`\narrowospace`
`\narrowospace*`
 SINCE v0.2

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

<code>\narrowospace</code>	<code>\narrowospace*</code>
----------------------------	-----------------------------

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

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

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 `\narrowospacestrength`. Both factors, `\narrowospacestrength` and `\narrowospacestrength` can be redefined by the user; their defaults are .5 and .9375, respectively.

Use Case

Tighten loose lines with a series of `\narrowospace`s.³¹ ■

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` [23] has been loaded, preferably *before* package `typog`

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

in the document preamble. ■

³⁰ See also »Investigating the badness of a paragraph« on Page 10.

³¹ 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` [23] for a detailed explanation.

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

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

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

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

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

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

Use Cases

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

3.12.2 Font Expansion

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

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

```
\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	Note
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 $\langle level \rangle$.

$\langle level \rangle$	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.

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

Notes

- Environment `setfontexpand` shares its `shrinklimits` with `setfontshrink` and its `stretchlimits` with `setfontstretch`.
- These environments do not nail down any font's expansion but only set up its available range. See Sec. 3.3, »Font Expansion«, in the microtype documentation [23].

Moreover, a text may not ›respond‹ neither to `setfontshrink`, `setfontstretch`, nor `setfontexpand` because \TeX already considers it optimal without expansion or within the previous expansion limits, e. g., those set at microtype load time as opposed to typog's load time. ■

Use Cases

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

`nofontexpansion` (env.)

`nofontexpand` (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

`\slightlyloppy`
`slightlyloppypar`
(env.)

Experienced \LaTeX 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.³² The default $\langle sloppiness \rangle$ is 1.

³² Also compare the findings for `\emergencystretch` in Ref. 30.

```

\slightlyloppy[⟨sloppiness⟩]
\begin{slightlyloppypar}[⟨sloppiness⟩]
...
\end{slightlyloppypar}

```

Table 12 summarizes the adjustments that `\slightlyloppy` makes depending on the `⟨sloppiness⟩` level.

TABLE 12: Adjustments made by `\slightlyloppy` to various \TeX parameters at different levels of `⟨sloppiness⟩`.

<code>⟨sloppiness⟩</code>	<code>\tolerance</code>	<code>\hfuzz</code> <code>\vfuzz</code> pt	<code>\emergencystretch</code> <code>G</code> em	Note
0	200	.1	0	\TeX : <code>\fussy</code> default
1	330 [†]	.15	.375 [‡]	
2	530 [†]	.2	.75 [‡]	
3	870 [†]	.25	1.125 [‡]	
4	1410 [†]	.3	1.5 [‡]	
5	2310 [†]	.35	1.875 [‡]	
6	3760 [†]	.4	2.25 [‡]	
7	6130 [†]	.45	2.625 [‡]	
≥ 8	9999	.5	3	\TeX : <code>\sloppy</code>

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

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

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

to prevent excessive stretchability in narrow lines.

Environment `slightlyloppypar[⟨sloppiness⟩]` mimics \LaTeX 's `sloppypar`, while offering the flexibility of `\slightlyloppy`.

Use Cases

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

3.14 Vertically Partially-Tied Paragraphs

L^AT_EX provides several macros and environments to tie material vertically – most prominently `samepage` and `minipage`.³³ Typog’s macros and environments constitute more sophisticated but weaker forms of these. They tie only the first or last couple of lines in a paragraph while the rest of the paragraph gets broken into pages by T_EX in the usual way.

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

Avoid a club line in each partial paragraph.

`\vtietop`

`vtietoppar (env.)`

```
\vtietop[⟨number-of-lines⟩]
\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^AT_EX:

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

Alternatively with package `enumitem` [4]:

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

or shorter and with the default *⟨number-of-lines⟩*, 3, using the `enumitem` style³⁴ `vtietop`:

³³ A valuable complement to these is package `needspace` [38] which takes a different approach and reliably works in *mixed* horizontal and vertical mode situations.

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

`vtietop` (*enumitem* key)

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

`\vtiebot`

Avoid a widow line in each partial paragraph.

`vtiebotpar` (*env.*)

```
\vtiebot[⟨number-of-lines⟩]
\begin{vtiebotpar}[⟨number-of-lines⟩] ... \end{vtiebotpar}
```

Vertically tie the last $\langle \text{number-of-lines} \rangle$ in a paragraph. Zero or one for $\langle \text{number-of-lines} \rangle$ 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}[⟨before-disp-number-of-lines⟩]
...
\end{vtiebotdisp}
```

Vertically tie the last $\langle \text{before-disp-number-of-lines} \rangle$ in a paragraph before a display. Zero or one for $\langle \text{before-disp-number-of-lines} \rangle$ 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}[⟨before-disp-number-of-lines⟩]
                        [⟨after-disp-number-of-lines⟩]
...
\end{vtiebotdisptoppar}
```

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

Zero or one for $\langle \text{before-disp-number-of-lines} \rangle$ as well as $\langle \text{after-disp-number-of-lines} \rangle$ are no-ops for the respective paragraph. Up to nine lines each can be fused.

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

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 *<number-of-lines>* from the beginning of every partial paragraph. Each displayed math in the paragraph resets the count. The bottom-paragraph ties, `\vtiebot`, `vtiebotpar`, `\vtiebotdisp`, and `vtiebotdisppar` count backward from the end of each partial paragraph. Again, each displayed math in the paragraph resets the count. According to T_EX'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.

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

[†] This is e-T_EX's counting scheme of `\clubpenalties`; it also holds for `vtietoppar`.

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

Tips

- The environments can be combined to arrive at paragraphs that simultaneously are protected against club lines and (display) widow lines.
- For very long derivations that are not interrupted and thus made breakable with the help of `\intertext`³⁶ or `\shortintertext`³⁷ 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. ■

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

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

Tips

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

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

³⁶ Introduced in package `amsmath` [2].



³⁷ Defined in package `mathtools` [12].

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

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

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

Use Cases

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

3.16 Setspace Front-End

Package `setspace` [27] 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 \LaTeX 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.³⁸ 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. 

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

`\setbaselineskip`
SINCE V0.3

```
\setbaselineskip{ $\langle baseline-skip \rangle$ }
```

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

```
\setlength{\baselineskip}{ $\langle baseline-skip \rangle$ }
```

The $\langle baseline-skip \rangle$ 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.

The copy of this document gets typeset with 10/12.5.

38 To find out about the current font’s size and the `\baselineskip` in printable form check out Sec. 3.2.1 on p. 7.

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{12.5pt}
```

- 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.

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

Set `\baselineskip` to `\typogfontsize × <baselineskip-percentage> / 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 `\typogfontsize × (1 + <leading-percentage> / 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:³⁹

```
\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 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 uppercase 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 \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} ■
```

39 For an overview of the various `\fontdimen<number>` parameters consult Tab. 7 on p. 38.

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 [33] we prescribe the line-widths with \TeX '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 \TeX 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 \TeX parameter `\leftskip`.

parindent= $\langle dim \rangle$

Set the first-line indent for the smooth ragged paragraph to $\langle dim \rangle$. Similar to the \TeX 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.
--	---

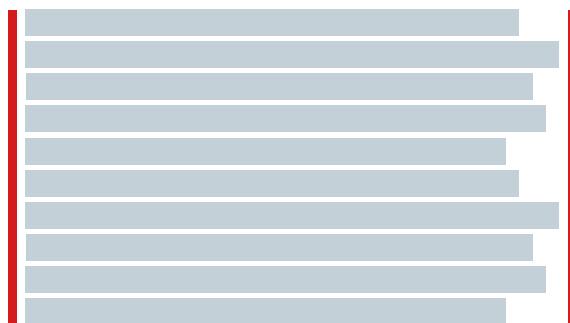
```
\begin{smoothrowrightpar}[\langle option \rangle...]
...
\end{smoothrowrightpar}
```

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

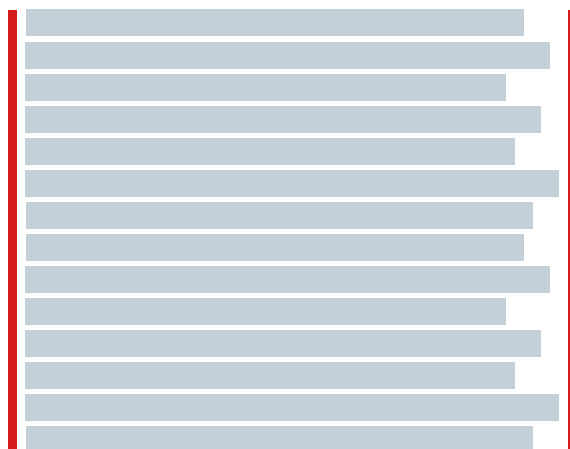
- The triplet generator repeats a *short line – long line – middle-length line* sequence. Shown below are two complete cycles.



- 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 smoothrowright

`raggedrightpar`. It takes the same optional arguments.

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

Options

\linewidth= $\langle dim \rangle$

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

Global Parameters

\smoothraggedrightfuzzfactor= $\langle factor \rangle$

The environment adds glue to every line-width⁴⁰ to achieve a more convincing »ragged appearance« and to reduce the number of overfull lines. The algorithm divides the smooth margin into 3, 5, or 7 parts depending on the chosen `\smoothraggedrightgenerator` (see below). The `\smoothraggedrightfuzzfactor` is the amount of glue of each line expressed as a multiple of the distance between the division points. The default of 1.0 means to add as much glue such that the lines just do not overlap (assuming justification is feasible).

\smoothraggedrightgenerator

Select a generator to use. Valid generator names:

- triplet,
- quintuplet,
- septuplet.

The default generator is triplet.

\smoothraggedrightleftskip= $\langle dim \rangle$

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

\smoothraggedrightparindent= $\langle dim \rangle$

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

\smoothraggedrightragwidth= $\langle dim \rangle$

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

⁴⁰ 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` [24]. ¶ 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>\Adjustedlabelitemi</code>	28	<code>\itcorr</code>	16
<code>\Adjustedlabelitemii</code>	28	<code>\kernedhyphen</code>	18
<code>\Adjustedlabelitemiii</code>	28	<code>\kernedslash</code>	17
<code>\Adjustedlabelitemiv</code>	28	<code>\leftkernedhyphen</code>	18
<code>\adjustedlabelitemi</code>	28	<code>\leftspacedendash</code>	19
<code>\adjustedlabelitemii</code>	28	<code>\nolig</code>	15
<code>\adjustedlabelitemiii</code>	28	<code>\rightkernedhyphen</code>	18
<code>\adjustedlabelitemiv</code>	28	<code>\rightspacedendash</code>	19
<code>\breakpoint</code>	14	<code>\Singleguillemetleft</code>	24
<code>\capitalemdash</code>	22	<code>\Singleguillemetright</code>	24
<code>\capitalendash</code>	21	<code>\singleguillemetleft</code>	24
<code>\capitalhyphen</code>	21	<code>\singleguillemetright</code>	24
<code>\capitaltimes</code>	23	<code>\spacedcapitalemdash</code>	22
<code>\Doubleguillemetleft</code>	24	<code>\spacedcapitalendash</code>	22
<code>\Doubleguillemetright</code>	24	<code>\spaceddash</code>	19
<code>\doubleguillemetleft</code>	24	<code>\spacedemdash</code>	20
<code>\doubleguillemetright</code>	24	<code>\spacedendash</code>	19
<code>\figuredash</code>	22		

The sentence uses `\spaced-` endashes.

Use `\texorpdfstring` – which is part of package hyperref [22] – 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^AT_EX Typography

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

- | | |
|-----------|---|
| enumitem | Flexible and consistent definition of all basic L ^A T _E X-list types plus in-line lists [4]. |
| geometry | Powerful and sophisticated setup of the page layout [28]. Best accompanied by layout [17] to visualize the page geometries. |
| hyphenat | Hyphens that do not inhibit further auto-hyphenation of a compound word [36]. |
| microtype | Fine control of spacing, tracking, sidebearings, character protrusion into the margins, font expansion, and much more [23].
See Section 3.12 for a front-end to microtype offered by this package.
See also KHIRVICH's discussion [13]. |
| ragged2e | Improved versions of environments <code>raggedleft</code> , <code>raggedright</code> , and <code>center</code> [24]. |
| setspace | Consistently set the line-spacing of a document, i. e., control <code>\baselineskip</code> [27].
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^AT_EX 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^AT_EX log files and the **typoginspect** environment as provided by the L^AT_EX package **typog**. It shares more with the venerable **sgrep** than with POSIX **grep**.

In the L^AT_EX 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^AT_EX 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^AT_EX** source file.

-N, --[no-]log-line-number

Print the line number of the *log*-file where the current line was encountered.

-p, --[no-]page-number

Print page number where the contents of the `typoginspect` environment starts in the typeset document.

-P, --[no-]pager

Redirect output from *stdout* to the configured pager.

--show-config

Show the default configuration and exit.

--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^AT_EX 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 e-TeX: Breaking Paragraphs into Lines

This is an excerpt from the e-TeX 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. 36.

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 TeX the last line is, however, usually typeset at its natural width due to infinitely stretchable `\parfillskip` glue. e-TeX allows interpolation between these two extremes by specifying a suitable value for `\lastlinefit`. For a value of 0 or less, e-TeX behaves as TeX, 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.⁴¹

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 e-TeX's modified algorithm was applied to that last line, the actual stretching or shrinking is achieved by suitably modifying the `\parfillskip` glue node.

41 As usual for parameters influencing TeX'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 e-TeX manual closes with a description of the generalizations of `\clubpenalties`, `\displaywidowpenalties`, `\interlinepenalties`, and `\widowpenalties`. See also Sec. 3.14, »Vertically Partially-Tied Paragraphs«, p. 44 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          [2025/04/15 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`.

`ftypog@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 \fi
63

```

Our own state...

`\typog@config@mathitalicscorrection`

```

64 \newmuskip\typog@config@mathitalicscorrection
65

```

Space around em-dash.

`\typog@config@emdashspace (dimen)`

```

66 \newlength{\typog@config@emdashspace}

```

Space around en-dash.

`\typog@config@endashspace (dimen)`

```

67 \newlength{\typog@config@endashspace}

```

Actual `\labelitem⟨N⟩` corrections.

`\typog@adjust@labelitemi (dimen)`

```

68 \newdimen{\typog@adjust@labelitemi}

```

`\typog@adjust@labelitemii (dimen)`

```

69 \newdimen{\typog@adjust@labelitemii}

```

`\typog@adjust@labelitemiii (dimen)`

```

70 \newdimen{\typog@adjust@labelitemiii}

```

`\typog@adjust@labelitemiv (dimen)`

```

71 \newdimen{\typog@adjust@labelitemiv}

```

Configuration constants for `\labelitem⟨N⟩` corrections.

`\typog@lowercase@labelitemi (dimen)`

```

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

```

`\typog@lowercase@labelitemii (dimen)`

```

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

```

```

lowercase@labelitemiii (dimen)
74 \newdimen{\typog@adjust@lowercase@labelitemiii}

lowercase@labelitemiv (dimen)
75 \newdimen{\typog@adjust@lowercase@labelitemiv}

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

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

percase@labelitemiii (dimen)
78 \newdimen{\typog@adjust@uppercase@labelitemiii}

ppercase@labelitemiv (dimen)
79 \newdimen{\typog@adjust@uppercase@labelitemiv}
80

    Other lengths...

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

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

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

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

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

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

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

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

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

```

```

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

\typog@config@breakpenalty
    91 \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.
    92 \newlength{\typog@dim@unit}
    93 \setlength{\typog@dim@unit}{.001em}

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

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

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

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

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

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

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

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

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

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

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

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

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

```

C.1 Setup and Reconfiguration

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

```

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

```

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

```

116 \NewDocumentCommand{\typogget}{m}
117     {\csname typog@config@#1\endcsname}
118

```

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

```

119 \ExplSyntaxOn
120 \cs_generate_variant:Nn \seq_set_split:Nnn {Nne}
121 \cs_new:Npn \typog_get_nth_csname:cnn #1#2#3
122   {
123     \seq_set_split:Nne \l_tmpa_seq {,} {\cs:w typog@config@#2 \cs_end:}
124     \cs_gset:cpn {#1} {\seq_item:Nn \l_tmpa_seq {#3}}
125   }
126 \cs_new:Npn \typog_get_nth_dimen:nnn #1#2#3
127   {
128     \seq_set_split:Nne \l_tmpa_seq {,} {\cs:w typog@config@#2 \cs_end:}
129     \dim_set:Nn {#1} {\seq_item:Nn \l_tmpa_seq {#3}}
130   }
131 \NewDocumentCommand{\typoggetnth}{m m m}{
132   \token_if_dim_register:NTF {#1}
133   {
134     \typog_get_nth_dimen:nnn {#1} {#2} {#3}
135   }
136   {
137     \typog_get_nth_csname:cnn {#1} {#2} {#3}
138   }
139 }
140 \ExplSyntaxOff
141

```

C.2 Information

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

```

142 \ExplSyntaxOn
143 \newcommand*{\typog@round@dim@to@tenths}[1]
144   {\fp_to_decimal:n {round(10 * \dim_to_fp:n{#1} / 1\p@) / 10}}
145 \ExplSyntaxOff
146

```


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

```
147 \newcommand*{\typog@formatsizeinfo}[3]
148   {\#1#3\kernedslash #2#3}
149
```

`\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.

```
150 \NewDocumentCommand{\fontsizeinfo}{s m}
151   {\global\expandafter\edef\csname typog@fontsize@#2\endcsname
152     {\typog@round@dim@to@tenths{\fontdimen6\font}}}%
153   \global\expandafter\edef\csname typog@linespacing@#2\endcsname
154     {\typog@round@dim@to@tenths{\baselineskip}}}%
155   \protected\expandafter\gdef\csname #2\endcsname
156     {\@ifnextchar*{\typog@formatsizeinfo
157                   {\csname typog@fontsize@#2\endcsname}%
158                   {\csname typog@linespacing@#2\endcsname}%
159                   }% no unit
160                   \ignorespaces % eat spaces after star
161                   \@gobble} % consume the star itself
162     {\typog@formatsizeinfo
163       {\csname typog@fontsize@#2\endcsname}%
164       {\csname typog@linespacing@#2\endcsname}%
165       {\,pt}% decorative unit 'pt'
166     }}}}
167
```

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

```
168 \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.

```
169 \newcounter{typog@inspect@count}
170
```

`typoginspect (env.)`

```
171 \define@key[typog]{typoginspect}{tracingboxes}[\maxdimen]%
172       {\def\typog@atypoginspect@tracingboxes{#1}}
173 \NewDocumentEnvironment{typoginspect}{0}{ m}
174   {\def\typog@atypoginspect@tracingboxes{\m@ne}%
175     \setkeys[typog]{typoginspect}{#1}%
176   }
```

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.

```

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

```

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

```

189 \hbadness=\m@ne
190 \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.

```

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

```

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

```

198 \NewDocumentEnvironment{typoginspectpar}{m}
199 {\typoginspect{#1}}
200 {\par\endtypoginspect}
201

```

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`.

```

202 \newcommand*{\typog@allowhyphenation}
203 {\ifvmode
204   \relax
205 \else
206   \nobreak
207   \hskip\z@skip
208 \fi}
209

```

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

```
210 \unless\ifdefined\allowhyphenation
211   \let\allowhyphenation=\typog@allowhyphenation
212 \fi
213
```

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

```
214 \NewDocumentCommand{\breakpoint}{s}
215   {\discretionary{}{}{}}%
216   \IfBooleanTF{#1}%
217     {\ignorespaces}%
218     {\typog@allowhyphenation}}
219
```

PDF-substitute definition

```
220 \typog@register@pdfsubstitute{
221   \def\breakpoint#1{\if*\detokenize{#1}\ignorespaces\fi}%
222 }
223
```

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

```
224 \NewDocumentEnvironment{hyphenmin}{o m}
225   {\lefthyphenmin=\IfNoValueTF{#1}{#2}{#1}%
226    \righthyphenmin=#2}
227   {}
228
```

C.4 Disable/Break Ligatures

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

```
229 \newcommand*{\typog@hyphen}{\char‘-}
230
```

`\nolig`

```
231 \NewDocumentCommand{\nolig}{s o}
232   {\dimen0=\IfNoValueTF{#2}{\typog@config@ligaturekern}{#2\typog@dim@unit}%
233    \IfBooleanTF{#1}%
234      {\kern\dimen0\ignorespaces}%
235      {\discretionary{\typog@hyphen}{}{\kern\dimen0}%
236       \typog@allowhyphenation
237       \IfNoValueF{#2}{\ignorespaces}}}
238
```

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

```
239 \typog@register@pdfsubstitute{
240   \RenewExpandableDocumentCommand{\nolig}{s o m}{%
241     \ifx\typog@TYPOG#3\typog@TYPOG
242       \relax
```

```

243     \else
244         \ifx\relax#3\relax
245             \relax
246         \else
247             \typog@missing@argument
248         \fi
249     \fi}
250 }
251

```

C.5 Manual Italic Correction

`\typog@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   {\ifmmode
291     \mspace{\muexpr(#3 mu) * 18 / 1000}%
292     \raisebox{#2\typog@dim@unit}{\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   {\IfBooleanTF{#1}%
311     {\kernedhyphen*[#2]{#3}{0}\ignorespaces}%
312     {\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   {\IfBooleanTF{#1}%
319     {\kernedhyphen*[#2]{0}{#3}\ignorespaces}%
320     {\kernedhyphen[#2]{0}{#3}}}
```

PDF-substitute definition

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

En-Dash and Em-Dash

`\typog@wrap@endash` Wrapper for the en-dash. The first and second arguments are used to control the line-breaking; the third argument is the actual en-dash macro.

```
325 \newcommand*{\typog@wrap@endash}[3]
326   {\unskip
327     #1\hspace{\typog@config@endashspace}%
328     #3%
329     #2\hspace{\typog@config@endashspace}}
```

`\typog@wrap@emdash` Wrapper for the em-dash. The first argument is used to control the line-breaking; the second argument is the actual em-dash macro.

```
330 \newcommand*{\typog@wrap@emdash}[2]
331   {\unskip
332     #1\hspace{\typog@config@emdashspace}%
333     #2%
334     #1\hspace{\typog@config@emdashspace}}
335
```

`\leftspacedendash` User-land macro for the left (aka opening or introducing) spaced en-dash. The unstarred variant introduces a breakpoint *before* the en-dash.

```
336 \NewDocumentCommand{\leftspacedendash}{s o}
337   {\IfBooleanTF{#1}
338     {\IfNoValueTF{#2}
339       {\typog@wrap@endash{\nobreak}{\nobreak}{\textendash}}
340       {\typog@wrap@endash{\nobreak}{\nobreak}{\raisebox{#2}{\textendash}}}}}
```

```

341     {\IfNoValueTF{#2}
342       {\typog@wrap@endash{\relax}{\nobreak}{\textendash}}
343       {\typog@wrap@endash{\nobreak}{\nobreak}{\raisebox{#2}{\textendash}}}}%
344     \ignorespaces}
345 \let\leftspaceddash=\leftspacedendash

    PDF-substitute definition
346 \typog@register@pdfsubstitute{
347   \RenewExpandableDocumentCommand{\leftspacedendash}{s o m}{%
348     \ifx\typog@TYPOG#3\typog@TYPOG
349       \textendash
350     \else
351       \ifx\relax#3\relax
352         \textendash
353       \else
354         \PackageError{typog}
355           {Missing third argument of \leftspacedendash}
356           {Append empty group or \relax after macro in-
    vocation}
357       \fi
358     \fi}
359 \let\leftspaceddash=\leftspacedendash
360 }
361
362 % \begin{macro}{\rightspacedendash}
363 % \changes{v0.5}{2025-05-19}{New macro.}
364 % User-land macro for the right (aka closing) spaced en-dash.
365 % The unstarred variant introduces a breakpoint \emph{after} the en-
    dash.
366 %
367 % \begin{macrocode}
368 \NewDocumentCommand{\rightspacedendash}{s o}
369 {\IfBooleanTF{#1}
370   {\IfNoValueTF{#2}
371     {\typog@wrap@endash{\nobreak}{\nobreak}{\textendash}}
372     {\typog@wrap@endash{\nobreak}{\nobreak}{\raisebox{#2}{\textendash}}}}
373   {\IfNoValueTF{#2}
374     {\typog@wrap@endash{\nobreak}{\relax}{\textendash}}
375     {\typog@wrap@endash{\nobreak}{\nobreak}{\raisebox{#2}{\textendash}}}}%
376   \ignorespaces}
377 \let\rightspaceddash=\rightspacedendash
378 \let\spacedendash=\rightspacedendash
379 \let\spaceddash=\rightspacedendash

    PDF-substitute definition
380 \typog@register@pdfsubstitute{
381   \RenewExpandableDocumentCommand{\rightspacedendash}{s o m}{%
382     \ifx\typog@TYPOG#3\typog@TYPOG
383       \space\textendash\space
384     \else
385       \ifx\relax#3\relax
386         \space\textendash\space

```

```

387     \else
388     \typog@missing@argument
389     \fi
390 \fi
391 \ignorespaces}
392 \let\rightspaceddash=\rightspacedendash
393 \let\spacedendash=\rightspacedendash
394 \let\spaceddash=\rightspacedendash
395 }
396

```

`\spacedemdash` User-land macro for the spaced em-dash.

```

397 \NewDocumentCommand{\spacedemdash}{o}
398 { \IfNoValueTF{#1}
399   { \typog@wrap@emdash{\nobreak}{\textemdash}}
400   { \typog@wrap@emdash{\nobreak}{\raisebox{#1}{\textemdash}}}%
401   \ignorespaces}
402

```

PDF-substitute definition

```

403 \typog@register@pdfsubstitute{
404   \RenewExpandableDocumentCommand{\spacedemdash}{o m}{%
405     \ifx\typog@TYPOG#2\typog@TYPOG
406       \textemdash
407     \else
408       \ifx\relax#2\relax
409         \textemdash
410       \else
411         \typog@missing@argument
412       \fi
413     \fi
414     \ignorespaces
415   }
416 }
417

```

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.

```

418 \newcommand*{\typog@breakpoint}
419 {\penalty\typog@config@breakpenalty}

```

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

```

420 \NewDocumentCommand{\capitalhyphen}{s}
421 { \raisebox{\typog@config@raisecapitalhyphen}{\typog@hyphen}%
422   \IfBooleanTF{#1}%
423   { \ignorespaces}%
424   { \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.

```

425 \typog@register@pdfsubstitute{
426   \def\capitalhyphen#1{%
427     \if*\detokenize{#1}%
428       -\ignorespaces
429     \else
430       -#1%
431     \fi}
432 }
433

```

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

```

434 \NewDocumentCommand{\capitalendash}{s}
435 { \raisebox{\typog@config@raisecapitaldash}{\textendash}%
436   \IfBooleanTF{#1}%
437     {\ignorespaces}%
438     {\typog@breakpoint\typog@allowhyphenation}}
439 \let\capitaldash=\capitalendash

```

PDF-substitute definition

```

440 \typog@register@pdfsubstitute{
441   \def\capitalendash#1{%
442     \if*\detokenize{#1}%
443       \textendash\ignorespaces
444     \else
445       \textendash#1%
446     \fi}
447 \let\capitaldash=\capitalendash
448 }
449

```

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

```

450 \NewDocumentCommand{\capitalemdash}{s}
451 { \raisebox{\typog@config@raisecapitaldash}{\textemdash}%
452   \IfBooleanTF{#1}%
453     {\ignorespaces}%
454     {\typog@breakpoint\typog@allowhyphenation}}

```

PDF-substitute definition

```

455 \typog@register@pdfsubstitute{
456   \def\capitalemdash#1{%
457     \if*\detokenize{#1}%
458       \textemdash\ignorespaces
459     \else
460       \textemdash#1%

```

```

461     \fi}
462 }
463

```

`\leftspacedcapitalendash` Thanks to our wrapper macro the definition of `\leftspacedcapitalendash` is easy to write.

```

464 \NewDocumentCommand{\leftspacedcapitalendash}{s}
465   {\IfBooleanTF{#1}%
466     {\typog@wrap@endash{\nobreak}{\nobreak}{\capitalendash*}}
467     {\typog@wrap@endash{\relax}{\nobreak}{\capitalendash}}}
468 \let\leftspacedcapitaldash=\leftspacedcapitalendash

    PDF-substitute definition

469 \typog@register@pdfsubstitute{
470   \def\leftspacedcapitalendash#1{%
471     \if*\detokenize{#1}%
472       \textendash\ignorespaces
473     \else
474       \textendash#1%
475     \fi}
476 \let\leftspacedcapitaldash=\leftspacedcapitalendash
477 }
478

```

`\rightspacedcapitalendash` Thanks to our wrapper macro the definition of `\rightspacedcapitalendash` is easy to write.

```

479 \NewDocumentCommand{\rightspacedcapitalendash}{s}
480   {\IfBooleanTF{#1}%
481     {\typog@wrap@endash{\nobreak}{\nobreak}{\capitalendash*}}
482     {\typog@wrap@endash{\nobreak}{\relax}{\capitalendash}}}
483 \let\rightspacedcapitaldash=\rightspacedcapitalendash
484 \let\spacedcapitalendash=\rightspacedcapitalendash
485 \let\spacedcapitaldash=\rightspacedcapitalendash

    PDF-substitute definition

486 \typog@register@pdfsubstitute{
487   \def\rightspacedcapitalendash#1{%
488     \if*\detokenize{#1}%
489       \textendash\ignorespaces
490     \else
491       \textendash#1%
492     \fi}
493 \let\rightspacedcapitaldash=\rightspacedcapitalendash
494 \let\spacedcapitalendash=\rightspacedcapitalendash
495 \let\spacedcapitaldash=\rightspacedcapitalendash
496 }
497

```

`\spacedcapitalemdash` The same ease of definition holds for `\spacedcapitalemdash`.

```

498 \NewDocumentCommand{\spacedcapitalemdash}{}
499   {\typog@wrap@emdash{\nobreak}{\nobreak}{\capitalemdash*}}

```

PDF-substitute definition

```
500 \typog@register@pdfsubstitute{\def\spacedcapitalemdash{\textemdash}}
501
```

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

```
502 \NewDocumentCommand{\figuredash}{s}
503   {\raisebox{\typog@config@raisefiguredash}{\textendash}%
504    \IfBooleanTF{#1}%
505      {\ignorespaces}%
506      {\typog@breakpoint\typog@allowhyphenation}}
```

PDF-substitute definition

```
507 \typog@register@pdfsubstitute{\let\figuredash=\capitaldash}
508
```

`\capitaltimes`

```
509 \NewDocumentCommand{\capitaltimes}{}
510   {\ifmmode
511     \mathbin{\raisebox{\typog@config@raisecapitaltimes}{\mathchar"2014}}%
512     \else
513       \raisebox{\typog@config@raisecapitaltimes}{\texttimes}%
514       \fi}
```

PDF-substitute definition

```
515 \typog@register@pdfsubstitute{
516   \RenewExpandableDocumentCommand{\capitaltimes}{}{\texttimes}
517 }
518
```

`\singleguillemetleft`

```
519 \NewDocumentCommand{\singleguillemetleft}{}
520   {\typog@allowhyphenation
521    \raisebox{\typog@config@raiseguillemets}{\guilsinglleft}}
```

PDF-substitute definition

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

`\singleguillemetright`

```
523 \NewDocumentCommand{\singleguillemetright}{}
524   {\raisebox{\typog@config@raiseguillemets}{\guilsinglright}%
525    \typog@allowhyphenation}
```

PDF-substitute definition

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

`\doubleguillemetleft`

```
527 \NewDocumentCommand{\doubleguillemetleft}{}
528   {\typog@allowhyphenation
529    \raisebox{\typog@config@raiseguillemets}{\guillemotleft}}
```

PDF-substitute definition

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

```
\doubleguillemetright
```

```
531 \NewDocumentCommand{\doubleguillemetright}{}
532 {\raisebox{\typog@config@raiseguilletts}{\guillemotright}%
533 \typog@allowhyphenation}
```

PDF-substitute definition

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

```
\Singleguillemetleft
```

```
535 \NewDocumentCommand{\Singleguillemetleft}{}
536 {\typog@allowhyphenation
537 \raisebox{\typog@config@raisecapitalguilletts}{\guilsinglleft}}
```

PDF-substitute definition

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

```
\Singleguillemetright
```

```
539 \NewDocumentCommand{\Singleguillemetright}{}
540 {\raisebox{\typog@config@raisecapitalguilletts}{\guilsinglright}%
541 \typog@allowhyphenation}
```

PDF-substitute definition

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

```
\Doubleguillemetleft
```

```
543 \NewDocumentCommand{\Doubleguillemetleft}{}
544 {\typog@allowhyphenation
545 \raisebox{\typog@config@raisecapitalguilletts}{\guillemotleft}}
```

PDF-substitute definition

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

```
\Doubleguillemetright
```

```
547 \NewDocumentCommand{\Doubleguillemetright}{}
548 {\raisebox{\typog@config@raisecapitalguilletts}{\guillemotright}%
549 \typog@allowhyphenation}
```

PDF-substitute definition

```
550 \typog@register@pdfsubstitute{\let\Doubleguillemetright\guillemotright}
551
```

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

```
onfig@raiseinvertedmarks@i
```

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

```
nfig@raiseinvertedmarks@ii
```

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

fig@raiseinvertedmarks@iii

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

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

inverted@exclamationmark@i

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

og@inverted@questionmark@i

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

nverted@exclamationmark@ii

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

g@inverted@questionmark@ii

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

verted@exclamationmark@iii

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

@inverted@questionmark@iii

```
560 \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.

```
561 \newcommand*{\typog@raise@inverted@mark}[2]
562 {\letcs{\@tempa}{\typog@inverted@#1@\romannumeral#2}%
563 \letcs{\@tempb}{\typog@config@raiseinvertedmarks@\romannumeral#2}%
564 \ifdim\@tempb=\z@
565 \setbox0=\hbox{\@tempa}%
566 \dimen0=\dp0
567 \else
568 \dimen0=\@tempb
569 \fi
570 \raisebox{\dimen0}{\@tempa}}
```

The user-side macros are almost trivial now.

talinvertedexclamationmark

```
571 \NewDocumentCommand{\capitalinvertedexclamationmark}{m}
572 {\typog@raise@inverted@mark{exclamationmark}{#1}%
573 \typog@allowhyphenation}
```

PDF-substitute definition

```
574 \typog@register@pdfsubstitute{
575 \def\capitalinvertedexclamationmark#1{%
576 \csname typog@inverted@exclamationmark@\romannumeral#1\endcsname
577 }
578 }
```

capitalinvertedquestionmark

```

579 \NewDocumentCommand{\capitalinvertedquestionmark}{m}
580   {\typog@raise@inverted@mark{questionmark}{#1}%
581     \typog@allowhyphenation}

    PDF-substitute definition

582 \typog@register@pdfsubstitute{
583   \def\capitalinvertedquestionmark#1{%
584     \csname typog@inverted@questionmark@romannumeral#1\endcsname
585   }
586 }
```

C.8 Vertically Adjusted Label Items

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

```

587 \newcommand*{\@typog@uppercase@adjust@labelitem}[1]
588   {\@typog@maybe@patch@itemize
589     \ifstrequal{#1}{*}
590       {\setlength{\typog@adjust@labelitemi}
591         {\typog@adjust@uppercase@labelitemi}
592         \setlength{\typog@adjust@labelitemii}
593           {\typog@adjust@uppercase@labelitemii}
594         \setlength{\typog@adjust@labelitemiii}
595           {\typog@adjust@uppercase@labelitemiii}
596         \setlength{\typog@adjust@labelitemiv}
597           {\typog@adjust@uppercase@labelitemiv}}
598       {\ifcase #1% 0
599         \relax % outside of any itemize environment
600         \or % 1
601           \setlength{\typog@adjust@labelitemi}
602             {\typog@adjust@uppercase@labelitemi}
603         \or % 2
604           \setlength{\typog@adjust@labelitemii}
605             {\typog@adjust@uppercase@labelitemii}
606         \or % 3
607           \setlength{\typog@adjust@labelitemiii}
608             {\typog@adjust@uppercase@labelitemiii}
609         \or % 4
610           \setlength{\typog@adjust@labelitemiv}
611             {\typog@adjust@uppercase@labelitemiv}
612         \else
613           \PackageError{typog}
614             {Itemize level out of range}
615             {Valid levels are 1, 2, 3, 4, and *}
616         \fi}}
617 }
```

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

```

618 \newcommand*{\@typog@lowercase@adjust@labelitem}[1]
```

```

619 {\@typog@maybe@patch@itemize
620   \ifstrequal{#1}{*}
621     {\setlength{\typog@adjust@labelitemi}
622       {\typog@adjust@lowercase@labelitemi}
623       \setlength{\typog@adjust@labelitemii}
624         {\typog@adjust@lowercase@labelitemii}
625       \setlength{\typog@adjust@labelitemiii}
626         {\typog@adjust@lowercase@labelitemiii}
627       \setlength{\typog@adjust@labelitemiv}
628         {\typog@adjust@lowercase@labelitemiv}}
629   {\ifcase #1% 0
630     \relax % outside of any itemize environment
631     \or % 1
632       \setlength{\typog@adjust@labelitemi}
633         {\typog@adjust@lowercase@labelitemi}
634     \or % 2
635       \setlength{\typog@adjust@labelitemii}
636         {\typog@adjust@lowercase@labelitemii}
637     \or % 3
638       \setlength{\typog@adjust@labelitemiii}
639         {\typog@adjust@lowercase@labelitemiii}
640     \or % 4
641       \setlength{\typog@adjust@labelitemiv}
642         {\typog@adjust@lowercase@labelitemiv}
643     \else
644       \PackageError{typog}
645         {Itemize level out of range}
646         {Valid levels are 1, 2, 3, 4, and *}
647     \fi}}
648

```

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

```

649 \newcommand*{\@typog@noadjust@labelitem}[1]
650   {\ifstrequal{#1}{*}
651     {\setlength{\typog@adjust@labelitemi}{\z@}
652       \setlength{\typog@adjust@labelitemii}{\z@}
653       \setlength{\typog@adjust@labelitemiii}{\z@}
654       \setlength{\typog@adjust@labelitemiv}{\z@}}
655     {\ifcase #1% 0
656       \relax % outside of any itemize environment
657       \or % 1
658         \setlength{\typog@adjust@labelitemi}{\z@}
659       \or % 2
660         \setlength{\typog@adjust@labelitemii}{\z@}
661       \or % 3
662         \setlength{\typog@adjust@labelitemiii}{\z@}
663       \or % 4
664         \setlength{\typog@adjust@labelitemiv}{\z@}
665       \else
666         \PackageError{typog}
667           {Itemize level out of range}

```

```

668                                     {Valid levels are 1, 2, 3, 4, and *}
669                                     \fi}}
670

```

`\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`.

```

671 \NewDocumentCommand{\uppercaseadjustlabelitems}{m}
672   {\AfterPreamble{%
673     \ifblank{#1}
674       {\@typog@uppercase@adjust@labelitem{\@itemdepth}}
675       {\forcsvlist{\@typog@uppercase@adjust@labelitem}{#1}}}%
676     \ignorespaces}}
677

```

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

```

678 \NewDocumentCommand{\lowercaseadjustlabelitems}{m}
679   {\AfterPreamble{%
680     \ifblank{#1}
681       {\@typog@lowercase@adjust@labelitem{\@itemdepth}}
682       {\forcsvlist{\@typog@lowercase@adjust@labelitem}{#1}}}%
683     \ignorespaces}}
684

```

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

```

685 \NewDocumentCommand{\noadjustlabelitems}{m}
686   {\ifblank{#1}
687     {\@typog@noadjust@labelitem{\@itemdepth}}
688     {\forcsvlist{\@typog@noadjust@labelitem}{#1}}}%
689     \ignorespaces}
690

```

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 L^AT_EX 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^AT_EX's or package enumitem's \itemize.

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

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

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

Sneak in our own macro's name.

```
693 \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.

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

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

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

```
697
```

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^AT_EX's macro.

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

```
699   {\ifdefined\enit@itemize@i
```

```
700     \patchcmd{\enit@itemize@i
```

```
701       {\expandafter}
```

```
702       {\@typog@itemize@patch\expandafter}
```

```
703       {\typog@debug@typeout{patching enumitem \string\enit@itemize@i\space
cedeed}}}
```

```
704       {\PackageError{typog}
```

```

705             {Patching enumitem macro \string\enit@itemize@i\space
706             {}
707     \else
708       \patchcmd{\itemize}
709         {\expandafter}
710         {\@typog@itemize@patch\expandafter}
711         {\typog@debug@typeout{patching \string\itemize\space suc-
       ceeded}}
712       {\PackageError{typog}
713         {Patching plain LaTeX macro \string\itemize\space fai
714         {}
715     \fi}
716

```

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

```

717 \newbool{@typog@itemize@has@been@patched}
718 \newcommand*{@typog@maybe@patch@itemize}
719   {\ifbool{@typog@itemize@has@been@patched}
720     {\relax}
721     {@typog@patch@itemize
722     \booltrue{@typog@itemize@has@been@patched}}}
723

```

Freestanding height-adjusted label items.

\Adjustedlabelitemi

```

724 \NewDocumentCommand{\Adjustedlabelitemi}{}
725   {\raisebox{@typog@adjust@uppercase@labelitemi}
726     {\labelitemi}}

```

\adjustedlabelitemi

```

727 \NewDocumentCommand{\adjustedlabelitemi}{}
728   {\raisebox{@typog@adjust@lowercase@labelitemi}
729     {\labelitemi}}

```

\Adjustedlabelitemii

```

730 \NewDocumentCommand{\Adjustedlabelitemii}{}
731   {\raisebox{@typog@adjust@uppercase@labelitemii}
732     {\labelitemii}}

```

\adjustedlabelitemii

```

733 \NewDocumentCommand{\adjustedlabelitemii}{}
734   {\raisebox{@typog@adjust@lowercase@labelitemii}
735     {\labelitemii}}

```

\Adjustedlabelitemiii

```

736 \NewDocumentCommand{\Adjustedlabelitemiii}{}
737   {\raisebox{@typog@adjust@uppercase@labelitemiii}
738     {\labelitemiii}}

```

`\adjustedlabelitemiii`

```
739 \NewDocumentCommand{\adjustedlabelitemiii}{}
740   {\raisebox{\typog@adjust@lowercase@labelitemiii}
741     {\labelitemiii}}
```

`\Adjustedlabelitemiv`

```
742 \NewDocumentCommand{\Adjustedlabelitemiv}{}
743   {\raisebox{\typog@adjust@uppercase@labelitemiv}
744     {\labelitemiv}}
```

`\adjustedlabelitemiv`

```
745 \NewDocumentCommand{\adjustedlabelitemiv}{}
746   {\raisebox{\typog@adjust@lowercase@labelitemiv}
747     {\labelitemiv}}
748
```

And now for their PDFsubstitutes.

```
749 \typog@register@pdfsubstitute{
750   \def\Adjustedlabelitemi{\labelitemi}
751   \def\adjustedlabelitemi{\labelitemi}
752   \def\Adjustedlabelitemii{\labelitemii}
753   \def\adjustedlabelitemii{\labelitemii}
754   \def\Adjustedlabelitemiii{\labelitemiii}
755   \def\adjustedlabelitemiii{\labelitemiii}
756   \def\Adjustedlabelitemiv{\labelitemiv}
757   \def\adjustedlabelitemiv{\labelitemiv}
758 }
759
```

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.

```
760 \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.

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

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

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

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

```
763   \count0=#3\relax
764   \unless\ifodd\count0
765     \advance\count0 by 1%
766   \fi
```

Set the iteration counter.

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

Put the $\langle sample \rangle$ into a box so that we can measure it with $\backslash ht$.

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

Box 1 is the accumulator for the raised samples.

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

Build the stairs.

```
770 \loop
771   \ifnum\thetypog@@iteration=\numexpr\count0 / 2\relax
772     \dimen1=2\dimen0
773   \else
774     \dimen1=\dimen0
775   \fi
776   \dimen2=\dimexpr#2 * (\thetypog@@iteration - \count0 / 2)\relax
777   \setbox1=\hbox{\unhbox1\raisebox{\dimen2}{\kern\dimen1 #5\kern\dimen1}}%
778   \addtocounter{typog@@iteration}{1}%
779   \unless\ifnum\thetypog@@iteration>\count0
780 \repeat
```

Merge the stairs with a hairline at #1 times the height of $\langle sample \rangle$. Answer just a single box.

```
781 \mbox{\rlap{\raisebox{\fpeval{\#1}\ht0}{\rule{\wd1}{\typog@hairline@width}}}\bo
782
```

$\backslash typogadjuststairs$ The arguments are: #1: $\langle scale-factor \rangle$, #2: $\langle step-size \rangle$, #3: $\langle number-of-steps \rangle$, and #4: $\langle sample \rangle$.

```
783 \NewDocumentCommand{\typogadjuststairs}{0{.5} m m m}
784 {\begingroup
785   \unless\ifdim #2>\z@
786     \PackageError{typog}
787       {\string\typogadjuststairs\space non-positive step-
788         size}
789       {step-size must be a positive dimension}%
789   \fi
790   \ifnum #3<1
791     \PackageError{typog}
792       {\string\typogadjuststairs\space too few number-
793         of-steps}
794       {number-of-steps must at least be 1}%
794   \fi
795   \ifblank{\#4}
796     {\PackageError{typog}
797       {sample must not be empty}
798       {supply either some uppercase or some low-
799         ercase letters}}
800   {\def\arraystretch{1}%
801     \begin{tabular}{@{}c@{}}
802       \typogadjuststairsfor{\#1}{\#2}{\#3}{\#4}{\labelitemi} \\
803       \typogadjuststairsfor{\#1}{\#2}{\#3}{\#4}{\labelitemii} \\
```

```

804     \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemiii}  \\\
805     \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemiv}
806   \end{tabular}
807   \endgroup
808

```

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

```

809 \newcommand*{\typoguppercaseadjusted@labelitems}
810   {\hbox{\raisebox{\typogadjust@uppercase@labelitemi}{\labelitemi}%
811           \raisebox{\typogadjust@uppercase@labelitemii}{\labelitemii}%
812           \raisebox{\typogadjust@uppercase@labelitemiii}{\labelitemiii}%
813           \raisebox{\typogadjust@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.

```

814 \NewDocumentCommand{\typoguppercaseadjustcheck}{0{.5} m}
815   {\setbox0=\hbox{#2}%
816    \setbox1=\typoguppercaseadjusted@labelitems
817    \mbox{\rlap{\raisebox{\fpeval{#1}\ht0}
818               {\rule{\wd1}{\typog@hairline@width}}}%
819          \box1}}
820

```

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

```

821 \newcommand*{\typoglowercaseadjusted@labelitems}
822   {\hbox{\raisebox{\typogadjust@lowercase@labelitemi}{\labelitemi}%
823           \raisebox{\typogadjust@lowercase@labelitemii}{\labelitemii}%
824           \raisebox{\typogadjust@lowercase@labelitemiii}{\labelitemiii}%
825           \raisebox{\typogadjust@lowercase@labelitemiv}{\labelitemiv}}}

```

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

```

826 \NewDocumentCommand{\typoglowercaseadjustcheck}{0{.5} m}
827   {\setbox0=\hbox{#2}%
828    \setbox1=\typoglowercaseadjusted@labelitems
829    \mbox{\rlap{\raisebox{\fpeval{#1}\ht0}
830               {\rule{\wd1}{\typog@hairline@width}}}%
831          \box1}}
832

```

C.9 Align Last Line of a Paragraph

The code of environment `lastlinerraggedleftpar` has been inspired by macro `\lastlinerraggedleft` [37, Sec. 2].

`lastlinerraggedleftpar` (*env.*)

```

833 \NewDocumentEnvironment{lastlinerraggedleftpar}{}
834   {\lastlinefit=0%

```

```

835 \setlength{\leftskip}{\z@ \@plus 1fil}%
836 \setlength{\rightskip}{-\leftskip}%
837 \setlength{\parfillskip}{\leftskip}}
838 {\par}

```

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

```

839 \let\lastlineflushrightpar=\lastlineraggedleftpar
840 \let\endlastlineflushrightpar=\endlastlineraggedleftpar
841

```

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

```

842 \NewDocumentEnvironment{lastlinecenteredpar}{}
843 {\lastlinefit=0%
844 \setlength{\leftskip}{\z@ \@plus .5fil}%
845 \setlength{\rightskip}{-\leftskip}%
846 \setlength{\parfillskip}{\z@ \@plus 1fil}}
847 {\par}
848

```

C.10 Fill Last Line of a Paragraph

`shortenpar` (*env.*)

```

849 \NewDocumentEnvironment{shortenpar}{}
850 {\advance\looseness by -1
851 \ifnum\tracingparagraphs>0
852 \typeout{@ looseness \the\looseness}%
853 \fi}
854 {\par}
855

```

`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.

```

856 \NewDocumentEnvironment{prolongpar}{}
857 {\finalhyphendemerits=100000001
858 \advance\looseness by 1
859 \ifnum\tracingparagraphs>0
860 \typeout{@ looseness \the\looseness}%
861 \fi}
862 {\par}
863

```

`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`.

```

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

```

`ndentpar@nonzero@parindent`

```

865 \newcommand*{\typog@covernextindentpar@nonzero@parindent}{2\parindent}

```

covernextindentpar (*env.*)

```

866 \NewDocumentEnvironment{covernextindentpar}{o}
867   {\IfNoValueTF{#1}
868     {\ifdim\parindent=\z@
869       \dimen0=\dimexpr\linewidth - \typog@covernextindentpar@zero@parindent
870     \else
871       \dimen0=\dimexpr\linewidth - \typog@covernextindentpar@nonzero@parindent
872     \fi}
873     {\dimen0=\dimexpr\linewidth - (#1)}%
874     \parfillskip=\dimen0 \@minus \dimen0
875     \relax}
876   {\par}
877

```

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

```

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

```

tlpar@nonzero@parindent

```

879 \newcommand*{\typog@openlastlinepar@nonzero@parindent}{2\parindent}

```

openlastlinepar (*env.*) Compare with the suggestion in Ref. [32](#).

```

880 \NewDocumentEnvironment{openlastlinepar}{o}
881   {\IfNoValueTF{#1}
882     {\ifdim\parindent=\z@
883       \skip0=\typog@openlastlinepar@zero@parindent
884       \@plus 1fil
885       \@minus \typog@openlastlinepar@zero@parindent
886     \else
887       \skip0=\typog@openlastlinepar@nonzero@parindent
888       \@plus 1fil
889       \@minus \typog@openlastlinepar@nonzero@parindent
890     \fi}
891     {\dimen0=\dimexpr#1\relax
892     \skip0=\dimen0 \@plus 1fil \@minus \dimen0}
893     \parfillskip=\skip0}
894   {\par}
895

```

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

```

896 \NewDocumentEnvironment{lastlinefitpar}{0{1000}}
897   {\lastlinefit=#1\relax}
898   {\par}
899

```

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.

```
900 \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.

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

`\widespace`

```
902 \NewDocumentCommand{\widespace}{s}
903   {\IfBooleanTF{#1}%
904     {\dimen0=\widespacescale\fontdimen2\font}%
905     {\ifdim\fontdimen7\font=\z@
906       \dimen0=\widespacescale\fontdimen2\font
907       \else
908         \dimen0=\dimexpr\fontdimen2\font +
909           \widespacestrength\fontdimen7\font
910       \fi}%
911   \hskip \glueexpr\dimen0
912         \@plus \widespacescale\fontdimen3\font
913         \@minus \widespacescale\fontdimen4\font
914   \ignorespaces}
915
```

`\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.

```
916 \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.

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

`\narrowospace`

```
918 \NewDocumentCommand{\narrowospace}{s}
919   {\IfBooleanTF{#1}%
920     {\dimen0=\narrowospace\fontdimen2\font}%
921     {\ifdim\fontdimen7\font=\z@
922       \dimen0=\narrowospace\fontdimen2\font
923       \else
924         \dimen0=\dimexpr\fontdimen2\font -
925           \narrowospacestrength\fontdimen7\font
926       \fi}%
927   \hskip \glueexpr\dimen0
928         \@plus \narrowospace\fontdimen3\font
```



```

929      \@minus \narrowspacescale\fontdimen4\font
930      \ignorespaces}
931

```

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

`loosespacing` (*env.*)

```

932 \NewDocumentEnvironment{loosespacing}{0{1}}
933   {\dimen2=\fontdimen2\font
934    \ifcase #1
935      \spaceskip=\z@
936      \or % 1      +5%
937      \spaceskip=1.05\dimen2 \@plus .5\dimen2 \@minus .1\dimen2
938      \or % 2      +10%
939      \spaceskip=1.1\dimen2 \@plus .5\dimen2 \@minus .1\dimen2
940      \or % 3      +20%
941      \spaceskip=1.2\dimen2 \@plus .6\dimen2 \@minus .2\dimen2
942      \else % >= 4  +30%
943      \spaceskip=1.3\dimen2 \@plus .8\dimen2 \@minus .3\dimen2
944      \fi
945      \ignorespaces}
946   {\ignorespacesafterend}
947

```

`tightspacing` (*env.*)

```

948 \NewDocumentEnvironment{tightspacing}{0{1}}
949   {\dimen2=\fontdimen2\font
950    \ifcase #1
951      \spaceskip=\z@
952      \or % 1      -1.25%
953      \spaceskip=.9875\dimen2 \@plus .0125\dimen2 \@minus .5\dimen2
954      \or % 2      -2.5%
955      \spaceskip=.975\dimen2 \@plus .025\dimen2 \@minus .5\dimen2
956      \or % 3      -5%
957      \spaceskip=.95\dimen2 \@plus .05\dimen2 \@minus .5\dimen2
958      \else % >= 4  -10%
959      \spaceskip=.9\dimen2 \@plus .1\dimen2 \@minus .5\dimen2
960      \fi
961      \ignorespaces}
962   {\ignorespacesafterend}
963

```

C.12 Microtype Front-End

Tracking

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

```

964 \NewDocumentEnvironment{setfonttracking}{m}
965   {\edef\MT@letterspace@{#1}%
966    \lsstyle
967    \ignorespaces}
968   {\ignorespacesafterend}

```

969

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.

```

970 \newcommand*{\typog@setup@font@expansion}
971   {\SetExpansion
972     [context = typog@shrink1,
973     shrink = \typog@shrink@i,
974     stretch = 0]%
975     {encoding = {*}}%
976     {}
977   \SetExpansion
978     [context = typog@shrink2,
979     shrink = \typog@shrink@ii,
980     stretch = 0]%
981     {encoding = {*}}%
982     {}
983   \SetExpansion
984     [context = typog@shrink3,
985     shrink = \typog@shrink@iii,
986     stretch = 0]%
987     {encoding = {*}}%
988     {}
989
990   \SetExpansion
991     [context = typog@stretch1,
992     shrink = 0,
993     stretch = \typog@stretch@i]%
994     {encoding = {*}}%
995     {}
996   \SetExpansion
997     [context = typog@stretch2,
998     shrink = 0,
999     stretch = \typog@stretch@ii]%
1000     {encoding = {*}}%
1001     {}
1002   \SetExpansion
1003     [context = typog@stretch3,
1004     shrink = 0,
1005     stretch = \typog@stretch@iii]%
1006     {encoding = {*}}%
1007     {}
1008
1009   \SetExpansion
1010     [context = typog@expand1,
1011     shrink = \typog@shrink@i,
1012     stretch = \typog@stretch@i]%
1013     {encoding = {*}}%

```

```

1014     {}
1015   \SetExpansion
1016     [context = typog@expand2,
1017       shrink = \typog@shrink@ii,
1018       stretch = \typog@stretch@ii]%
1019     {encoding = {*}}%
1020   {}
1021   \SetExpansion
1022     [context = typog@expand3,
1023       shrink = \typog@shrink@iii,
1024       stretch = \typog@stretch@iii]%
1025     {encoding = {*}}%
1026   {}

```

`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.

```

1027 \newcommand*{\typog@test@microtype@expansion@feature}
1028   {\ifMT@expansion
1029     \typog@typeout{microtype preloaded -- font expansion features avail-
1030       able}%
1031     \def\typog@require@microtype@expansion{\relax}
1032     \typog@setup@font@expansion
1033   \else
1034     \PackageWarning{typog}{microtype preloaded,\space
1035       but font expansion is disabled}%
1036     \def\typog@require@microtype@expansion
1037       {\PackageError{typog}
1038         {microtype font expansion disabled}
1039         {pass option 'expansion' to package microtype}}
1040   \fi}

```

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

```

1040 \iftypog@microtype@preloaded
1041   \typog@test@microtype@expansion@feature
1042 \else
1043   \def\typog@require@microtype@expansion
1044     {\PackageError{typog}%
1045       {package microtype not (pre-)loaded, %
1046        which is required for typog's font expansion}%
1047       {require package microtype before package typog}}
1048 \fi
1049

```

`setfontshrink` (*env.*)

```

1050 \NewDocumentEnvironment{setfontshrink}{0{1}}
1051   {\typog@require@microtype@expansion

```

```

1052 \ifcase#1% 0
1053   \relax
1054 \or % 1
1055   \microtypecontext{expansion=typog@shrink1}%
1056 \or % 2
1057   \microtypecontext{expansion=typog@shrink2}%
1058 \else % >= 3
1059   \microtypecontext{expansion=typog@shrink3}%
1060 \fi
1061 \ignorespaces}
1062 {\ignorespacesafterend}
1063

```

setfontstretch (*env.*)

```

1064 \NewDocumentEnvironment{setfontstretch}{0{1}}
1065 {\typog@require@microtype@expansion
1066   \ifcase#1% 0
1067     \relax
1068   \or % 1
1069     \microtypecontext{expansion=typog@stretch1}%
1070   \or % 2
1071     \microtypecontext{expansion=typog@stretch2}%
1072   \else % >= 3
1073     \microtypecontext{expansion=typog@stretch3}%
1074   \fi
1075   \ignorespaces}
1076 {\ignorespacesafterend}
1077

```

setfontexpand (*env.*)

```

1078 \NewDocumentEnvironment{setfontexpand}{0{1}}
1079 {\typog@require@microtype@expansion
1080   \ifcase#1% 0
1081     \relax
1082   \or % 1
1083     \microtypecontext{expansion=typog@expand1}%
1084   \or % 2
1085     \microtypecontext{expansion=typog@expand2}%
1086   \else % >= 3
1087     \microtypecontext{expansion=typog@expand3}%
1088   \fi
1089   \ignorespaces}
1090 {\ignorespacesafterend}
1091

```

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.

```

1092 \NewDocumentEnvironment{nofontexpansion}{}
1093 {\ifdefined\microtypesetup
1094   \microtypesetup{expansion=false}%

```

```

1095 \fi
1096 \ignorespaces}
1097 {\ignorespacesafterend}

```

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

```

1098 \let\nofontexpand=\nofontexpansion
1099 \let\endnofontexpand=\endnofontexpansion
1100

```

Character Protrusion

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

```

1101 \NewDocumentEnvironment{nocharprotrusion}{}
1102 {\ifdefined\microtypesetup
1103 \microtypesetup{protrusion=false}%
1104 \fi
1105 \ignorespaces}
1106 {\ignorespacesafterend}
1107

```

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`.

```

1108 \newcommand*{\typog@scaled@emergencystretch}[1]
1109 {\emergencystretch=\ifdim\linewidth=\z@
1110 #1%
1111 \else
1112 \dimexpr (#1) * \linewidth / \textwidth
1113 \fi}
1114

```

`\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.

```

1115 \NewDocumentCommand{\slightlyloppy}{0{1}}
1116 {\ifcase #1% 0
1117 % \tolerance=200
1118 % \emergencystretch=\z@
1119 % \hfuzz=.1\p@
1120 % \vfuzz=\hfuzz
1121 \fussy
1122 \or % 1
1123 \pretolerance=165%
1124 \tolerance=330%
1125 \typog@scaled@emergencystretch{.375em}%
1126 \hfuzz=.15\p@
1127 \vfuzz=\hfuzz
1128 \or % 2
1129 \pretolerance=265%

```

```

1130     \tolerance=530%
1131     \typog@scaled@emergencystretch{.75em}%
1132     \hfuzz=.15\p@
1133     \vfuzz=\hfuzz
1134   \or % 3
1135     \pretolerance=435%
1136     \tolerance=870%
1137     \typog@scaled@emergencystretch{1.125em}%
1138     \hfuzz=.2\p@
1139     \vfuzz=\hfuzz
1140   \or % 4
1141     \pretolerance=705%
1142     \tolerance=1410%
1143     \typog@scaled@emergencystretch{1.5em}%
1144     \hfuzz=.3\p@
1145     \vfuzz=\hfuzz
1146   \or % 5
1147     \pretolerance=1155%
1148     \tolerance=2310%
1149     \typog@scaled@emergencystretch{1.875em}%
1150     \hfuzz=.35\p@
1151     \vfuzz=\hfuzz
1152   \or % 6
1153     \pretolerance=1880%
1154     \tolerance=3760%
1155     \typog@scaled@emergencystretch{2.25em}%
1156     \hfuzz=.4\p@
1157     \vfuzz=\hfuzz
1158   \or % 7
1159     \pretolerance=3065%
1160     \tolerance=6130%
1161     \typog@scaled@emergencystretch{2.625em}%
1162     \hfuzz=.45\p@
1163     \vfuzz=\hfuzz
1164   \else % >= 8
1165     % \tolerance=9999
1166     % \emergencystretch=3em
1167     % \hfuzz=.5\p@
1168     % \vfuzz=\hfuzz
1169   \sloppy
1170   \fi
1171   \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.)`

```
1172 \NewDocumentEnvironment{slightlyloppypar}{0{1}}
1173   {\par\slightlyloppy[#1]\ignorespaces}
1174   {\par}
1175
```

C.14 Vertically Partially-Tied Paragraphs

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

```
1176 \ExplSyntaxOn
1177 \newcommand*{\typog@geometric@mean}[2]
1178     {\fp_to_int:n {sqrt((#1) * (#2))}}
1179 \ExplSyntaxOff
1180
```

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

```
1181 \newcounter{typog@mean@penalty}
1182
```

`\vtietop`

```
1183 \NewDocumentCommand{\vtietop}{0{3}}
1184   {\setcounter{typog@mean@penalty}
1185     {\typog@geometric@mean{\@M}{\clubpenalty}}%
1186     \typog@debug@typeout{vtietop: penalties \the\@M--\the\value{typog@mean@penalty}
1187       -\the\clubpenalty}%
1187     \unless\ifnum\clubpenalty<\@M
1188       \PackageWarning{typog}{vtietop: clubpenalty=\the\clubpenalty\space>= 10000}%
1189     \fi
1190     \ifcase#1% 0
1191       \relax
1192     \or % 1
1193       \relax
1194     \or % 2
1195       \clubpenalties 3
1196       \@M
1197       \value{typog@mean@penalty}
1198       \clubpenalty
```

```

1199 \or % 3
1200   \clubpenalties 4
1201     \@M \@M
1202       \value{typog@mean@penalty}
1203       \clubpenalty
1204 \or % 4
1205   \clubpenalties 5
1206     \@M \@M \@M
1207       \value{typog@mean@penalty}
1208       \clubpenalty
1209 \or % 5
1210   \clubpenalties 6
1211     \@M \@M \@M \@M
1212       \value{typog@mean@penalty}
1213       \clubpenalty
1214 \or % 6
1215   \clubpenalties 7
1216     \@M \@M \@M \@M \@M
1217       \value{typog@mean@penalty}
1218       \clubpenalty
1219 \or % 7
1220   \clubpenalties 8
1221     \@M \@M \@M \@M \@M
1222       \value{typog@mean@penalty}
1223       \clubpenalty
1224 \or % 8
1225   \clubpenalties 9
1226     \@M \@M \@M \@M \@M
1227       \value{typog@mean@penalty}
1228       \clubpenalty
1229 \else % >= 9
1230   \clubpenalties 10
1231     \@M \@M \@M \@M \@M \@M \@M \@M
1232       \value{typog@mean@penalty}
1233       \clubpenalty
1234 \fi}
1235

```

vtietoppar (*env.*)

```

1236 \NewDocumentEnvironment{vtietoppar}{0{3}}
1237   {\vtietop[#1]}
1238   {\par
1239     \ignorespacesafterend}
1240

```

\splicevtietop

```

1241 \NewDocumentCommand{\splicevtietop}{0{3}}
1242   {\let\typog@old@item=\@item
1243     \def\@item[##1]{\typog@old@item[##1]\vtietop[#1]}%
1244     \ignorespaces}
1245

```


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).

```

1246 \ifdefined\SetEnumitemKey
1247   \SetEnumitemKey{vtietop}{first=\splicevtietop}
1248 \fi
1249
\vtiebot
1250 \NewDocumentCommand{\vtiebot}{0{3}}
1251   {\setcounter{typog@mean@penalty}
1252     {\typog@geometric@mean{\@M}{\widowpenalty}}%
1253     \typog@debug@typeout{vtiebot: penalties \the\@M--\the\value{typog@mean@penalty}
1254       -\the\widowpenalty}%
1255     \unless\ifnum\widowpenalty<\@M
1256       \PackageWarning{typog}{vtiebot: widowpenalty=\the\widowpenalty\space>= 10000}
1257     \fi
1258     \ifcase#1% 0
1259       \relax
1260     \or % 1
1261       \relax
1262     \or % 2
1263       \widowpenalties 3
1264       \@M
1265       \value{typog@mean@penalty}
1266       \widowpenalty
1267     \or % 3
1268       \widowpenalties 4
1269       \@M \@M
1270       \value{typog@mean@penalty}
1271       \widowpenalty
1272     \or % 4
1273       \widowpenalties 5
1274       \@M \@M \@M
1275       \value{typog@mean@penalty}
1276       \widowpenalty
1277     \or % 5
1278       \widowpenalties 6
1279       \@M \@M \@M \@M
1280       \value{typog@mean@penalty}
1281       \widowpenalty
1282     \or % 6
1283       \widowpenalties 7
1284       \@M \@M \@M \@M \@M
1285       \value{typog@mean@penalty}
1286       \widowpenalty
1287     \or % 7
1288       \widowpenalties 8
1289       \@M \@M \@M \@M \@M \@M
1290       \value{typog@mean@penalty}
1291       \widowpenalty
1292     \or % 8
1293       \widowpenalties 9

```

```

1293      \@M \@M \@M \@M \@M \@M \@M
1294      \value{typog@mean@penalty}
1295      \widowpenalty
1296  \else % >= 9
1297      \widowpenalties 10
1298      \@M \@M \@M \@M \@M \@M \@M \@M
1299      \value{typog@mean@penalty}
1300      \widowpenalty
1301  \fi}
1302

```

vtiebotpar (env.)

```

1303 \NewDocumentEnvironment{vtiebotpar}{0{3}}
1304 {\vtiebot[#1]}
1305 {\par
1306  \ignorespacesafterend}
1307

```

\typog@vtiebotdisp

```

1308 \NewDocumentCommand{\typog@vtiebotdisp}{m}
1309 {\setcounter{typog@mean@penalty}
1310   {\typog@geometric@mean{\@M}{\displaywidowpenalty}}}%
1311  \typog@debug@typeout{vtiebotdisp: penalties \the\@M--\the\value{typog@mean@pen
- \the\displaywidowpenalty}}%
1312  \unless\ifnum\displaywidowpenalty<\@M
1313    \PackageWarning{typog}{vtiebotdisp: displaywidowpenalty=\the\displaywidowpen
1314    \fi
1315    \ifcase#1% 0
1316      \relax
1317    \or % 1
1318      \relax
1319    \or % 2
1320      \displaywidowpenalties 3
1321      \@M
1322      \value{typog@mean@penalty}
1323      \displaywidowpenalty
1324    \or % 3
1325      \displaywidowpenalties 4
1326      \@M \@M
1327      \value{typog@mean@penalty}
1328      \displaywidowpenalty
1329    \or % 4
1330      \displaywidowpenalties 5
1331      \@M \@M \@M
1332      \value{typog@mean@penalty}
1333      \displaywidowpenalty
1334    \or % 5
1335      \displaywidowpenalties 6
1336      \@M \@M \@M \@M
1337      \value{typog@mean@penalty}
1338      \displaywidowpenalty
1339    \or % 6

```

```

1340 \displaywidowpenalties 7
1341 \@M \@M \@M \@M \@M
1342 \value{typog@mean@penalty}
1343 \displaywidowpenalty
1344 \or % 7
1345 \displaywidowpenalties 8
1346 \@M \@M \@M \@M \@M \@M
1347 \value{typog@mean@penalty}
1348 \displaywidowpenalty
1349 \or % 8
1350 \displaywidowpenalties 9
1351 \@M \@M \@M \@M \@M \@M \@M
1352 \value{typog@mean@penalty}
1353 \displaywidowpenalty
1354 \else % >= 9
1355 \displaywidowpenalties 10
1356 \@M \@M \@M \@M \@M \@M \@M \@M
1357 \value{typog@mean@penalty}
1358 \displaywidowpenalty
1359 \fi}
1360

```

vtiebotdisp (*env.*)

```

1361 \NewDocumentEnvironment{vtiebotdisp}{0{3}}
1362 {\typog@vtiebotdisp{#1}}
1363 {\ignorespacesafterend}
1364

```

vtiebotdisptoppar (*env.*)

```

1365 \NewDocumentEnvironment{vtiebotdisptoppar}{0{3}o}
1366 {\postdisplaypenalty=\@M
1367 \predisplaypenalty=10001% in accordance with package ‘widows-
and-orphans’
1368 \edef\typog@@top@lines{\IfNoValueTF{#2}{#1}{#2}}%
1369 \edef\typog@@after@display@math{\vtietop[\typog@@top@lines]}%
1370 \PushPostHook{display}{\aftergroup\typog@@after@display@math}%
1371 \vtiebotdisp[#1]}
1372 {\par
1373 \PopPostHook{display}%
1374 \ignorespacesafterend}
1375

```

C.15 Breakable Displayed Equations

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

```

1376 \newenvironment*{breakabledisplay}[1][3]
1377 {\allowdisplaybreaks[#1]}
1378 {\ignorespacesafterend}
1379

```

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.

```
1380 \newcommand*{\typog@setbaselineskip@iter@limit}{10}
```

`aselineskip@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.

```
1381 \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.

```
1382 \ExplSyntaxOn
```

```
1383 \cs_new:Npn \typog@setbaselineskip #1
```

```
1384 {
```

Initialize our “emergency-stop” loop counter.

```
1385 \int_set:Nn \l_tmpa_int {1}
```

```
1386 \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.

```
1387 \dim_set:Nn \l_tmpa_dim {\glueexpr #1}
```

```
1388
```

```
1389 \typog@debug@typeout{\string\setbaselineskip:\space  
initial\space baselineskip:\space \the\baselineskip}
```

```
1390 \typog@debug@typeout{\string\setbaselineskip:\space
```

```
1391 target\space baselineskip:\space \dim_use:N \l_tmpa_dim}
```

```
1392
```

```
1393 \dim_compare:nNnTF {\baselineskip} > {\c_zero_dim}
```

```
1394 {}
```

```
1395 {
```

```
1396 \PackageError{typog}
```

```
1397 {\string\setbaselineskip:\space
```

```
1398 baselineskip\space not\space positive}
```

```
1399 {}
```

```
1400 }
```

```
1401
```

```
1402 \dim_compare:nNnTF {\l_tmpa_dim} > {\c_zero_dim}
```

```
1403 {}
```

```
1404 {
```

```
1405 \PackageError{typog}
```

```
1406 {\string\setbaselineskip:\space target\space  
baselineskip\space must\space be\space
```

```
1407 positive}
```

```
1408 {}
```

```
1409 }
```

```
1410
```

```
1411 \skip_if_eq:nnTF {\l_tmpa_dim} {\glueexpr #1}
```

```
1412 {}
```

```
1413 {
```

```

1416 \PackageWarning{typog}
1417     {\string\setbaselineskip:\space argument\space
1418      is\space a\space skip;\space
1419      will\space ignore\space glue}
1420 {}
1421 }
1422
1423 \fp_set:Nn \l_tmpa_fp {\l_tmpa_dim / \baselineskip}
1424 \fp_until_do:nNnn {abs(\l_tmpa_dim / \baselineskip - 1)} <
1425     {\typog@setbaselineskip@relative@error}
1426 {
1427     \setstretch{\fp_use:N \l_tmpa_fp}
1428     \fp_set:Nn \l_tmpa_fp
1429         {\l_tmpa_fp * \l_tmpa_dim / \baselineskip}
1430
1431     \int_incr:N \l_tmpa_int
1432     \int_compare:nNnTF {\l_tmpa_int} > {\l_tmpb_int}
1433     {
1434         \PackageError{typog}
1435             {\string\setbaselineskip:\space excessive\space
1436              number\space of\space iterations:\space
1437              \int_use:N \l_tmpa_int\space >\space
1438              \int_use:N \l_tmpb_int}
1439         {}
1440     }
1441     {}
1442 }
1443
1444 \typog@debug@typeout{\string\setbaselineskip:\space
1445   final\space \string\setstretch\space argument:\space
1446   \fp_use:N \l_tmpa_fp}
1447 \typog@debug@typeout{\string\setbaselineskip:\space
1448   final\space baselineskip:\space \the\baselineskip}
1449 }
1450

```

`\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.

```

1451 \cs_new:Npn \setbaselineskip #1

```

```

1452 {
1453   \AfterPreamble{\typog@setbaselineskip{#1}}
1454   \ignorespaces
1455 }
1456

```

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

```

1457 \cs_new:Npn \resetbaselineskip
1458 {
1459   \AfterPreamble{\setstretch{1}}
1460 }
1461

```

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

```

1462 \dim_new:N \typogfontsize

```

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

```

1463 \AfterEndPreamble{
1464   \dim_set:Nn \typogfontsize {\fontdimen6\font}
1465   \typog@debug@typeout{\string\typogfontsize =
1466     \dim_use:N \typogfontsize\space
1467     (at\space begin\space of\space document)}
1468 }
1469

```

`\setbaselineskippercentage`

```

1470 \cs_new:Npn \setbaselineskippercentage #1
1471 {
1472   \AfterPreamble{
1473     \dim_compare:nNnTF {\typogfontsize} > {\c_zero_dim}
1474     {
1475       \typog@setbaselineskip{
1476         \fp_eval:n {(#1) / 100} \typogfontsize}
1477     }
1478     {
1479       \PackageError{typog}
1480         {\string\setbaselineskippercentage:\space
1481          \string\typogfontsize <= 0}
1482         {Maybe\space \string\typogfontsize\space
1483          is\space uninitialized?}
1484     }
1485   }
1486   \ignorespaces
1487 }
1488

```

`\setleading`

```

1489 \cs_new:Npn \setleading #1
1490 {
1491   \AfterPreamble{

```

```

1492 \dim_compare:nNnTF {\typogfontsize} > {\c_zero_dim}
1493 {
1494   \typog@setbaselineskip{\typogfontsize + \dimexpr #1}
1495 }
1496 {
1497   \PackageError{typog}
1498     {\string\setleading:\space
1499     \string\typogfontsize <= 0}
1500     {Maybe\space \string\typogfontsize\space
1501     is\space uninitialized?}
1502 }
1503 }
1504 \ignorespaces
1505 }
1506

```

\setleadingpercentage

```

1507 \cs_new:Npn \setleadingpercentage #1
1508 {
1509   \AfterPreamble{
1510     \dim_compare:nNnTF {\typogfontsize} > {\c_zero_dim}
1511     {
1512       \typog@setbaselineskip{
1513         \fp_eval:n {1 + (#1) / 100} \typogfontsize}
1514     }
1515     {
1516       \PackageError{typog}
1517         {\string\setleadingpercentage:\space
1518         \string\typogfontsize <= 0}
1519         {Maybe\space \string\typogfontsize\space
1520         is\space uninitialized?}
1521     }
1522   }
1523   \ignorespaces
1524 }
1525 \ExplSyntaxOff
1526

```

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.

```

1527 \ExplSyntaxOn
1528 \cs_new_eq:NN \typog@repeat \prg_replicate:nn
1529

```

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

```

1530 \newcommand*{\typog@mod}[2]{\int_mod:nn{#1}{#2}}
1531 \ExplSyntaxOff

```

1532

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

1533 \newcommand*{\typog@triplet@max@lines}{99}

1534

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

```

1535 \define@key[typog]{smoothraggedrightshapetriplet}{leftskip}%
1536         {\def\typog@@triplet@leftskip{#1}}
1537 \define@key[typog]{smoothraggedrightshapetriplet}{parindent}%
1538         {\def\typog@@triplet@parindent{#1}}
1539 \NewDocumentEnvironment{smoothraggedrightshapetriplet}{0}{ m m m}
1540     {\def\typog@@triplet@leftskip{\z@}%
1541      \def\typog@@triplet@parindent{\z@}%
1542      \setkeys*{typog}{smoothraggedrightshapetriplet}{#1}%
1543      \skip0=\typog@@triplet@leftskip\relax
1544      \skip1=#2\relax
1545      \skip2=#3\relax
1546      \skip3=#4\relax
1547      \typog@debug@typeout{smoothraggedrightshapetriplet: skip0=\the\skip0}%
1548      \typog@debug@typeout{smoothraggedrightshapetriplet: skip1=\the\skip1}%
1549      \typog@debug@typeout{smoothraggedrightshapetriplet: skip2=\the\skip2}%
1550      \typog@debug@typeout{smoothraggedrightshapetriplet: skip3=\the\skip3}%
1551      \unless\ifnum\typog@mod{\typog@triplet@max@lines}{3}=0
1552          \PackageError{typog}
1553              {Line number of triplet generator\space
1554               (\typog@triplet@max@lines) not divisible by 3}
1555              {}
1556      \fi
1557      \edef\typog@@triplet@linespecs{%
1558          \glueexpr \skip0 + \typog@@triplet@parindent\relax
1559          \glueexpr \skip1 - \typog@@triplet@parindent\relax
1560          \skip0 \skip2 \skip0 \skip3
1561          \typog@repeat{\numexpr\typog@triplet@max@lines / 3 - 1}
1562              {\skip0 \skip1 \skip0 \skip2 \skip0 \skip3}}
1563      \parshape=\typog@triplet@max@lines\typog@@triplet@linespecs\relax
1564      {\par}
1565

```

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

1566 \newcommand*{\typog@quintuplet@max@lines}{95}

1567

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

```

1568 \define@key[typog]{smoothraggedrightshapequintuplet}{leftskip}
1569         {\def\typog@@quintuplet@leftskip{#1}}
1570 \define@key[typog]{smoothraggedrightshapequintuplet}{parindent}
1571         {\def\typog@@quintuplet@parindent{#1}}
1572 \NewDocumentEnvironment{smoothraggedrightshapequintuplet}{0}{ m m m m m}

```



```

1573 {\def\typog@@quintuplet@leftskip{\z@}%
1574 \def\typog@@quintuplet@parindent{\z@}%
1575 \setkeys*[typog]{smoothraggedrightshapequintuplet}{#1}%
1576 \skip0=\typog@@quintuplet@leftskip\relax
1577 \skip1=#2\relax
1578 \skip2=#3\relax
1579 \skip3=#4\relax
1580 \skip4=#5\relax
1581 \skip5=#6\relax
1582 \typog@debug@typeout{smoothraggedrightshapequintuplet: skip0=\the\skip0}%
1583 \typog@debug@typeout{smoothraggedrightshapequintuplet: skip1=\the\skip1}%
1584 \typog@debug@typeout{smoothraggedrightshapequintuplet: skip2=\the\skip2}%
1585 \typog@debug@typeout{smoothraggedrightshapequintuplet: skip3=\the\skip3}%
1586 \typog@debug@typeout{smoothraggedrightshapequintuplet: skip4=\the\skip4}%
1587 \typog@debug@typeout{smoothraggedrightshapequintuplet: skip5=\the\skip5}%
1588 \unless\ifnum\typog@mod{\typog@quintuplet@max@lines}{5}=0
1589 \PackageError{typog}
1590 {Line number of quintuplet generator\space
1591 (\typog@quintuplet@max@lines) not divisible by 5}
1592 {}
1593 \fi
1594 \edef\typog@@quintuplet@linespecs{%
1595 \glueexpr \skip0 + \typog@@quintuplet@parindent\relax
1596 \glueexpr \skip1 - \typog@@quintuplet@parindent\relax
1597 \skip0 \skip2 \skip0 \skip3
1598 \skip0 \skip4 \skip0 \skip5
1599 \typog@repeat{\numexpr\typog@quintuplet@max@lines / 5 - 1}
1600 {\skip0 \skip1 \skip0 \skip2 \skip0 \skip3
1601 \skip0 \skip4 \skip0 \skip5}}
1602 \parshape=\typog@quintuplet@max@lines\typog@@quintuplet@linespecs\relax}
1603 {\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.

```

1604 \newcommand*{\typog@septuplet@max@lines}{98}
1605

```

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

```

1606 \define@key[typog]{smoothraggedrightshapeseptuplet}{leftskip}%
1607 {\def\typog@septuplet@leftskip{#1}}
1608 \define@key[typog]{smoothraggedrightshapeseptuplet}{parindent}%
1609 {\def\typog@septuplet@parindent{#1}}
1610 \NewDocumentEnvironment{smoothraggedrightshapeseptuplet}{0}{ m m m m m m m}
1611 {\def\typog@@septuplet@leftskip{\z@}%
1612 \def\typog@@septuplet@parindent{\z@}%
1613 \setkeys*[typog]{smoothraggedrightshapeseptuplet}{#1}%
1614 \skip0=\typog@@septuplet@leftskip\relax
1615 \skip1=#2\relax
1616 \skip2=#3\relax
1617 \skip3=#4\relax
1618 \skip4=#5\relax
1619 \skip5=#6\relax

```

```

1620 \skip6=#7\relax
1621 \skip7=#8\relax
1622 \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip0=\the\skip0}%
1623 \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip1=\the\skip1}%
1624 \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip2=\the\skip2}%
1625 \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip3=\the\skip3}%
1626 \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip4=\the\skip4}%
1627 \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip5=\the\skip5}%
1628 \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip6=\the\skip6}%
1629 \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip7=\the\skip7}%
1630 \unless\ifnum\typog@mod{\typog@septuplet@max@lines}{7}=0
1631   \PackageError{typog}
1632     {Line number of septuplet generator\space
1633     (\typog@septuplet@max@lines) not divisible by 7}
1634   {}
1635 \fi
1636 \edef\typog@@septuplet@linespecs{%
1637   \glueexpr \skip0 + \typog@@septuplet@parindent\relax
1638   \glueexpr \skip1 - \typog@@septuplet@parindent\relax
1639   \skip0 \skip2 \skip0 \skip3 \skip0 \skip4
1640   \skip0 \skip5 \skip0 \skip6 \skip0 \skip7
1641   \typog@repeat{\numexpr\typog@septuplet@max@lines / 7 - 1}
1642     {\skip0 \skip1 \skip0 \skip2 \skip0 \skip3 \skip0 \skip4
1643     \skip0 \skip5 \skip0 \skip6 \skip0 \skip7}}
1644   \parshape=\typog@septuplet@max@lines\typog@@septuplet@linespecs\relax}
1645 {\par}
1646
smoothraggedrightfuzzfactor
1647 \newcommand*{\smoothraggedrightfuzzfactor}{1.0}

smoothraggedrightgenerator
1648 \newcommand*{\smoothraggedrightgenerator}{triplet}

\smoothraggedrightleftskip
1649 \newlength{\smoothraggedrightleftskip}

smoothraggedrightparindent
1650 \newlength{\smoothraggedrightparindent}

\smoothraggedrighttragwidth
1651 \newlength{\smoothraggedrighttragwidth}
1652 \setlength{\smoothraggedrighttragwidth}{2em}
1653

\typog@fuzzwidth (dimen)
1654 \newdimen{\typog@fuzzwidth}
1655

```

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

```
1656 \define@key[typog]{smoothraggedrightpar}{linewidth}%
1657         {\def\typog@linewidth{#1}}
1658
```

Convert generator name to an integer suitable for `\ifcase`. Indicate an unknown name with -1.

```
1659 \ExplSyntaxOn
1660 \cs_new:Npn \typog@generator@index:n #1 {
1661   \str_case:nnTF {#1}
1662     {
1663       {triplet} {0}
1664       {quintuplet} {1}
1665       {septuplet} {2}
1666     }
1667     {}
1668     {-1}
1669 }
1670 \cs_generate_variant:Nn \typog@generator@index:n {e}
1671 \let\typog@generator@index=\typog@generator@index:e
1672 \ExplSyntaxOff
1673
1674 \NewDocumentEnvironment{smoothraggedrightpar}{0{}}
1675   {\edef\typog@linewidth{\linewidth}%
1676    \setkeys[typog]{smoothraggedrightpar}{#1}%
1677    \edef\typog@@generatorchoice{\typog@generator@index{\smoothraggedrightgenerator}}
```

Obeys to the indentation prescribed by any list environment.

```
1678   \let\typog@@smoothraggedrightleftskip=\smoothraggedrightleftskip
1679   \ifnum\@listdepth>0
1680     \addtolength{\typog@@smoothraggedrightleftskip}{\leftmargin}%
1681   \fi
```

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

```
1682   \typog@fuzzwidth=\smoothraggedrightfuzzfactor\smoothraggedrightragwidth
```

Now for the generator-specific code...

```
1683   \ifcase\typog@@generatorchoice
```

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

```
1684     \typog@fuzzwidth=.25\smoothraggedrightragwidth
1685     \typog@debug@typeout{smoothraggedright: generator=triplet, ty-
1686       pog@fuzzwidth=\the\typog@fuzzwidth}%
1686     \smoothraggedrightshapetriplet[leftskip=\typog@@smoothraggedrightleftskip,
1687       parindent=\glueexpr\smoothraggedrightparinden
1688       #1]%
1689     {\glueexpr \typog@linewidth - \smoothraggedrightragwidth
1690       + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
1691     {\glueexpr \typog@linewidth \@minus \typog@fuzzwidth}% (3)
```

```

1692      {\glueexpr (\typog@@linewidth * 2 - \smoothraggedrightrag-
width) / 2
1693      + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (2)
1694      \or
      generator=quintuplet.
1695      \typog@fuzzwidth=.125\smoothraggedrightragwidth
1696      \typog@debug@typeout{smoothraggedright: generator=quintuplet, ty-
pog@fuzzwidth=\the\typog@fuzzwidth}%
1697      \smoothraggedrightshapequintuplet[leftskip=\typog@smoothraggedrightleftskip,
1698      parindent=\glueexpr\smoothraggedrightparin-
indent,
1699      #1]%
1700      {\glueexpr (\typog@@linewidth * 4 - \smoothraggedrightrag-
width * 3) / 4
1701      + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (2)
1702      {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth\relax}% (5)
1703      {\glueexpr (\typog@@linewidth * 2 - \smoothraggedrightrag-
width) / 2
1704      + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (3)
1705      {\glueexpr (\typog@@linewidth * 4 - \smoothraggedrightrag-
width) / 4
1706      + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (4)
1707      {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth
1708      + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
1709      \or
      generator=septuplet.
      Permutation 3 - 6 - 1 - 5 - 2 - 7 - 4 looks ›random‹ enough for our purposes.
1710      \typog@fuzzwidth=.08333\smoothraggedrightragwidth
1711      \typog@debug@typeout{smoothraggedright: generator=septuplet, ty-
pog@fuzzwidth=\the\typog@fuzzwidth}%
1712      \smoothraggedrightshapeseptuplet[leftskip=\typog@smoothraggedrightleftskip,
1713      parindent=\glueexpr\smoothraggedrightparind-
indent,
1714      #1]%
1715      {\glueexpr (\typog@@linewidth * 3 - \smoothraggedrightrag-
width * 2) / 3
1716      + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (3)
1717      {\glueexpr (\typog@@linewidth * 6 - \smoothraggedrightrag-
width) / 6
1718      + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (6)
1719      {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth +
1720      + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
1721      {\glueexpr (\typog@@linewidth * 3 - \smoothraggedrightrag-
width) / 3

```

```

1722          + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
      nus \typog@fuzzwidth\relax}% (5)
1723      {\glueexpr (\typog@@linewidth * 6 - \smoothraggedright-
      width * 5) / 6
1724          + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
      nus \typog@fuzzwidth\relax}% (2)
1725      {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth\relax}% (7)
1726      {\glueexpr (\typog@@linewidth * 2 - \smoothraggedright-
      width) / 2
1727          + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
      nus \typog@fuzzwidth\relax}% (4)
1728      \else
1729      \PackageError{typog}
1730      {smoothraggedright: unknown generator name (\smoothraggedright-
      generator)}{
1731      {valid generator names are triplet, quintuplet, and sep-
      tuplet}
1732      \fi}
1733      {\ifcase\typog@@generatorchoice
1734      \endsmoothraggedrightshapetriplet
1735      \or
1736      \endsmoothraggedrightshapequintuplet
1737      \or
1738      \endsmoothraggedrightshapeseptuplet
1739      \fi}
1740
smoothraggedright (env.)
1741 \NewDocumentEnvironment{smoothraggedright}{0{}}
1742 {\PushPostHook{par}{\hskip-\parindent\smoothraggedrightpar[#1]\relax}}
1743 {\par\PopPostHook{par}}
1744

```

Change History

v0.1

General: Initial version. [i](#)

v0.2

`\narrowospace`: New macro. [95](#)

`\widespace`: Add fallback if `\fontdimen7` is zero. Extend with a starred version. [95](#)

v0.3

`hyphenmin`: New environment. [74](#)

`\resetbaselineskip`: New macro. [109](#)

`\setbaselineskip`: New macro. [108](#)

`\setbaselineskippercentage`: New macro. [109](#)

`\setleading`: New macro. [109](#)

`\setleadingpercentage`: New macro. [110](#)

`\typogfontsize`: New dimen. [109](#)

v0.4

`\lowercaseadjustlabelitems`: New macro. [87](#)

`\noadjustlabelitems`: New macro. [87](#)

`\typogadjuststairs`: New macro. [91](#)

`\typoggetnth`: New macro. [71](#)

`\typoglowercaseadjustcheck`: New macro. [92](#)

`\typoguppercaseadjustcheck`: New macro. [92](#)

`\uppercaseadjustlabelitems`: New macro. [87](#)

v0.4a

`smoothraggedrightshapeseptuplet`: Add missing backslash to `\typog@septuplet@parindent` in local macro `\typog@septuplet@linespecs` which rendered the septuplet generator unusable. [112](#)

`\typog@register@pdfsubstitute`: Using `\pdfstringdefDisableCommands` breaks package `pdfcomment`. So we just refrain from using the macro if `pdfcomment` was loaded. [67](#)

v0.5

General: Rewrite parser for package option `lowercaselabelitemadjustments` to accept a star as a placeholder. [70](#)

Rewrite parser for package option `uppercaselabelitemadjustments` to accept a star as a placeholder. [70](#)

`\Adjustedlabelitemi`: New macro. [89](#)

`\adjustedlabelitemi`: New macro. [89](#)

`\Adjustedlabelitemii`: New macro. [89](#)

`\adjustedlabelitemii`: New macro. [89](#)

`\Adjustedlabelitemiii`: New macro. [89](#)

`\adjustedlabelitemiii`: New macro. [90](#)

`\Adjustedlabelitemiv`: New macro. [90](#)

`\adjustedlabelitemiv`: New macro. [90](#)

`\capitalinvertedexclamationmark`: New macro. [84](#)

`\capitalinvertedquestionmark`: New macro. [85](#)

`\kernedslash`: Allow for lowering (or raising) `\kernedslash` with package option `lowerslash`. [76](#)
`lastlinefitpar`: New environment. [94](#)
`\leftspacedcapitalendash`: New macro. [81](#)
`\leftspacedendash`: New macro. [77](#)
`\rightspacedcapitalendash`: New macro. [81](#)
`\spacedcapitalemdash`: New macro. [81](#)
`\spacedemdash`: New macro. [79](#)

References

- [1] ABRAHAM, PAUL W., HARGREAVES, KATHRYN A., and KARL BERRY. *T_EX for the Impatient*. 2020, <https://tug.ctan.org/info/impatient/book.pdf>.
- [2] AMERICAN MATHEMATICAL SOCIETY and the L^AT_EX3 PROJECT TEAM. *Package amsmath*. 2020, <https://ctan.org/pkg/amsmath>.
- [3] ARSENEAU, DONALD. *Package cite*. 2015, <https://ctan.org/pkg/cite>.
- [4] BEZOS, JAVIER. *Package enumitem*. 2019, <https://ctan.org/pkg/enumitem>.
- [5] BEZOS, JAVIER. *Package babel*. 2021, <https://ctan.org/pkg/babel>.
The original author of package babel was J. L. BRAAMS.
- [6] BREITENLOHNER, PETER and the N_TS TEAM. *e-T_EX*. 1998, 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_EX By Topic, A Texnician's Reference*. 2007, <https://www.eijkhout.net/tex/tex-by-topic.html>.
- [12] HØGHOLM, MORTEN, MADSEN, LARS and the L^AT_EX3 PROJECT TEAM. *Package mathtools*. 2020, <https://ctan.org/pkg/mathtools>.
- [13] KHIREVICH, SIARHEI. *Tips on Writing a Thesis in L^AT_EX*. 2013, <https://www.khirevich.com/latex/microtype>.
- [14] KLEBER, JOSEF. *Package pdfcomment*. 2018, <https://ctan.org/pkg/pdfcomment>.
- [15] KNUTH, DONALD ERVIN. *The T_EXbook*. Addison Wesley, Reading/MA, 1986.
- [16] THE L^AT_EX 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^AT_EX 2_ε 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^AT_EX*. 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] NIEDERBERGER, CLEMENS. *Package tasks*. 2022, <https://github.com/cgnieder/tasks>.
- [22] RAHTZ, SEBASTIAN, and FRANK MITTELBACH. *Package hyperref*. 2020, <https://ctan.org/pkg/hyperref>. The package is maintained by the L^AT_EX3 Project Team.
- [23] SCHLICHT, ROBERT. *Package microtype*. 2020, <https://ctan.org/pkg/microtype>.
- [24] SCHRÖDER, MARTIN. *Package ragged2e*. 2019, <https://ctan.org/pkg/ragged2e>.
- [25] SOLOMON, DAVID. *Output Routines: Examples and Techniques. Part I: Introduction and Examples*. TUGboat, 11(1), 69–85, 1990, <https://www.tug.org/TUGboat/Articles/tb11-1/tb27salomon.pdf>.
- [26] STRIZVER, ILENE. *Type rules!: the designer's guide to professional typography*, 4th ed. John Wiley & Sons, Hoboken/NJ, 2014.
- [27] TOBIN, GEOFFREY, and ROBIN FAIRBAIRNS. *Package setspace*. 2011, <https://ctan.org/pkg/setspace>.
- [28] UMEKI, HIDEO. *Package geometry*. 2020, <https://ctan.org/pkg/geometry>.
- [29] WERMUTH, UDO. *Tracing paragraphs*. TUGboat, 37(3), 358–373, 2016, <https://tug.org/TUGboat/tb37-3/tb117wermuth.pdf>.
- [30] WERMUTH, UDO. *The optimal value for \emergencystretch*. TUGboat, 38(1), 65–86, 2017, <https://tug.org/TUGboat/tb38-1/tb118wermuth.pdf>.
- [31] WERMUTH, UDO. *A note on \linepenalty*. TUGboat, 38(3), 400–414, 2017, <https://tug.org/TUGboat/tb38-3/tb120wermuth.pdf>.
- [32] WERMUTH, UDO. *Experiments with \parfillskip*. TUGboat, 39(3), 276–303, 2018, <https://tug.org/TUGboat/tb39-3/tb123wermuth-parfillskip.pdf>.
- [33] WERMUTH, UDO. *An attempt at ragged-right typesetting*. TUGboat, 41(1), 73–94, 2020, <https://tug.org/TUGboat/tb41-1/tb127wermuth-ragged.pdf>.
- [34] WERMUTH, UDO. Personal communication. August 2, 2022.

- [35] WERMUTH, UDO. *Vertical alignments in plain T_EX*. TUGboat, 44(3), 427–440, 2023, <https://tug.org/TUGboat/tb44-3/tb138werimuth-valign.pdf>.
- [36] WILSON, PETER. *Package hyphenat*. 2004, <https://ctan.org/pkg/hyphenat>. The package is maintained by W. ROBERTSON.
- [37] WILSON, PETER. *Glisterings*. TUGboat, 28(2), 229–232, 2007, <https://tug.org/TUGboat/tb28-2/tb89glister.pdf>.
- [38] WILSON, PETER. *Package needspace*. 2010, <https://ctan.org/pkg/needspace>. The package is maintained by W. ROBERTSON.

Index

A

`\Adjustedlabelitemi` 28, [ℓ724](#), [ℓ750](#)
`\adjustedlabelitemi` 28, [ℓ727](#), [ℓ751](#)
`\Adjustedlabelitemii` 28, [ℓ730](#),
[ℓ752](#)
`\adjustedlabelitemii` 28, [ℓ733](#),
[ℓ753](#)
`\Adjustedlabelitemiii` 28, [ℓ736](#),
[ℓ754](#)
`\adjustedlabelitemiii` 28, [ℓ739](#),
[ℓ755](#)
`\Adjustedlabelitemiv` 28, [ℓ742](#),
[ℓ756](#)
`\adjustedlabelitemiv` 28, [ℓ745](#),
[ℓ757](#)
`\allowhyphenation` 11, [ℓ210](#)
`amsmath` (package) 47

B

`baseline skip` 48
`bookmark` 15, 20
`breakabledisplay` (env.) 47, [ℓ1376](#)
`breakpenalty` (option) 2, 15, 17, 21, 22
`\breakpoint` 14, [ℓ214](#)
`\breakpoint*` 14

C

`\capitaldash` 21, [ℓ439](#), [ℓ447](#), [ℓ507](#)
`\capitaldash*` 21
`\capitalemdash` 22, [ℓ450](#), [ℓ499](#)
`\capitalemdash*` 22
`\capitalendash` 21, [ℓ434](#), [ℓ466](#), [ℓ467](#),
[ℓ481](#), [ℓ482](#)
`\capitalendash*` 21
`\capitalhyphen` 21, [ℓ420](#)
`\capitalhyphen*` 21
`\capitalinvertedexclamationmark` 25,
[ℓ571](#)
`\capitalinvertedquestionmark` 25,
[ℓ579](#)
`\capitaltimes` 23, [ℓ509](#)
`\changes` [ℓ363](#)
`configuration` 6
`covernextindentpar` (env.) 35, [ℓ866](#)
`csquotes` (package) 25

D

`debug` (option) 2
`dimensions:`
`\typogfontsize` 50, [ℓ1462](#), [ℓ1473](#),
[ℓ1476](#), [ℓ1481](#), [ℓ1482](#), [ℓ1492](#), [ℓ1494](#),
[ℓ1499](#), [ℓ1500](#), [ℓ1510](#), [ℓ1513](#), [ℓ1518](#),
[ℓ1519](#)
`\Doubleguillemetleft` 24, [ℓ543](#)
`\doubleguillemetleft` 24, [ℓ527](#)
`\Doubleguillemetright` 24, [ℓ547](#)
`\doubleguillemetright` 24, [ℓ531](#)

E

`emdashspace` (option) 2
`\emph` [ℓ365](#)
`endashspace` (option) 2
`enumitem-keys:`
`vtietop` 45
`environments:`
`breakabledisplay` 47, [ℓ1376](#)
`covernextindentpar` 35, [ℓ866](#)
`hyphenmin` 14, [ℓ224](#)
`lastlinecenteredpar` 32, [ℓ842](#)
`lastlinefitpar` 36, [ℓ896](#)
`lastlineflushrightpar` 30,
[ℓ839](#)
`lastlineraggedleftpar` 30,
[ℓ833](#)
`loosespacing` 37, [ℓ932](#)
`nocharprotrusion` 42, [ℓ1101](#)
`nofontexpand` 42, [ℓ1098](#)
`nofontexpansion` 42, [ℓ1092](#)
`openlastlinepar` 35, [ℓ880](#)
`prolongpar` 35, [ℓ856](#)
`setfontexpand` 41, [ℓ1078](#)
`setfontshrink` 41, [ℓ1050](#)
`setfontstretch` 41, [ℓ1064](#)
`setfonttracking` 40, [ℓ964](#)
`shortenpar` 35, [ℓ849](#)
`slightlyloppypar` 42, [ℓ1172](#)
`smoothraggedright` 52, [ℓ1741](#)
`smoothraggedrightpar` 51, [ℓ1656](#)
`smoothraggedrightshape-`
`quintuplet` 51, [ℓ1568](#)
`smoothraggedrightshape-`
`septuplet` 51, [ℓ1606](#)
`smoothraggedrightshape-`
`triplet` 51, [ℓ1535](#)
`tightspacing` 37, [ℓ948](#)

typoginspect 8, [ℓ171](#)
 typoginspectpar 8, [ℓ198](#)
 typogsetup 6, [ℓ107](#)
 vtiebotdisp 45, [ℓ1361](#)
 vtiebotdisptoppar 45, [ℓ1365](#)
 vtiebotpar 45, [ℓ1303](#)
 vtietoppar 44, [ℓ1236](#)
 extra spacing 17

F

`\figuredash` 22, [ℓ502](#)
`\figuredash*` 22
 font
 encoding 23
 expansion 34, 41
 information 7
 protrusion 42
 size 7
 spacing 34, 36
 loose 36
 tight 36
 tracking 34, 40
 typeface
 ADF Baskervald 24
 ADF Accanthis 31
 ADF Venturis 23, 31
 Alegreya 23
 Aleo 31
 Arvo 23
 Bitter 23
 Charis SIL 23, 24, 31
 Clara 23
 CM Roman 31
 Cochineal 18, 31
 Coelacanth 23
 Crimson Pro 23
 Crimson Text 23
 Domitian 31
 Droid Serif 23
 EB Garamond 23, 24, 31
 Erewhon 23
 etbb 31
 Extended Charter 31
 fbb 23
 Gentium 23, 24, 31
 GFS Artemisia 23, 24
 GFS Bodoni 31
 GFS Didot 24, 31
 Ibarra Real Nova 23
 IBM Plex Serif 31
 INRIA Serif 23
 KP Serif 31

Libertine 23
 Libertinus Serif 23, 24, 31
 Libre Baskerville 23
 Libre Caslon 23
 Merriweather 23, 24
 ML Modern 31
 PT Serif 23
 Quattrocento 23
 Roboto Slab 23
 Source Serif Pro 23, 31
 Spectral 23, 31
 STIX 23, 31
 T_EX Gyre Pagella 23
 TX Fonts 23
 URW Palladio ii, 26, 31
 Utopia Regular 31
`\fontsizeinfo` 7, [ℓ150](#)
 forlorn line
 club 35, 44
 display widow 45
 orphan 35
 widow 35, 45

H

hyperref (package) 15, 20
 hyphenation 11
 empty discretionary 14
 first word 12
 `\hskip` 12
 re-enable automatic 11
 hyphenmin (env.) 14, [ℓ224](#)

I

information 7
 inverted
 exclamation mark 25
 question mark 25
`\itcorr` [ℓ11](#), [16](#), [26](#), [ℓ264](#)
`\itcorr*` 16

K

`\kernedhyphen` 18, [ℓ289](#), [ℓ311](#), [ℓ312](#),
 [ℓ319](#), [ℓ320](#)
`\kernedhyphen*` 18
`\kernedslash` 17, [ℓ148](#), [ℓ277](#)
`\kernedslash*` 17
 kerning
 extra 17
 forward slash 17
 hyphen 18
 ligature 15
 known problems 55

L

label items 27
 lastlinecenteredpar (env.) 32,
 ℓ842
 \lastlinefit 10, 36, 64
 lastlinefitpar (env.) 36, 64, ℓ896
 lastlineflushrightpar (env.) 30,
 ℓ839
 lastlineraggedleftpar (env.) 30,
 ℓ833
 leading 48
 \leftkernedhyphen 18, ℓ309, ℓ314
 \leftkernedhyphen* 18
 \leftspacedcapitaldash 22, ℓ468,
 ℓ476
 \leftspacedcapitaldash* 22
 \leftspacedcapitalendash 22,
 ℓ464
 \leftspacedcapitalendash* 22
 \leftspacedddash 19, ℓ345, ℓ359
 \leftspacedddash* 19
 \leftspacedendash 19, ℓ336
 \leftspacedendash* 19
 ligature 15
 ligaturekern (option) 2, 15
 limitations 55
 line spacing 7
 list 44
 loosespacing (env.) 37, ℓ932
 lowercaselabelitem-
 adjustments (option) 2, 27
 \lowercaseadjustlabelitems 27,
 ℓ678
 lowerslash (option) 3, 17

M

mathitalicscorrection (option) 3,
 16
 microtype (package) 39

N

narrow space 39
 \narrow space 39, ℓ918
 \narrow space* 39
 \narrow spacescale 39, ℓ917, ℓ920,
 ℓ922, ℓ928, ℓ929
 \narrow spacestrength 39, ℓ916,
 ℓ925
 \noadjustlabelitems 27, ℓ685
 nocharprotrusion (env.) 42, ℓ1101
 nodebug (option) 2
 nofontexpand (env.) 42, ℓ1098

nofontexpansion (env.) 42, ℓ1092
 \nolig 15, ℓ231
 \nolig* 15

O

openlastlinepar (env.) 35, ℓ880

P

package option 2–5
 breakpenalty 2, 15, 17, 21, 22
 debug 2
 emdashspace 2
 endashspace 2
 ligaturekern 2, 15
 lowercaselabelitem-
 adjustments 2, 27
 lowerslash 3, 17
 mathitalicscorrection 3, 16
 nodebug 2
 raise* 3
 raisecapitaldash 3, 21
 raisecapitalguillemets 3, 24
 raisecapitalhyphen 3, 18
 raisecapitaltimes 3, 23
 raisefiguredash 3, 22
 raiseguillemets 3, 24
 raiseinvertedmarks 3, 26
 shrinklimits 4, 41
 slashkern 4, 17
 stretchlimits 4, 41
 textitalicscorrection 4, 16
 trackingttspacing 4, 40
 uppercaselabelitem-
 adjustments 4, 27
 page break 11, 47
 paragraph
 align last line 30
 centered 32
 flush right 30
 badness 10
 fill last line 32
 covernextindentpar 35
 \linebreak 34
 \mbox 34
 openlastlinepar 35
 prolongpar 35
 shortenpar 35
 tie 34
 sloppy 42
 vertically tied 44
 PDF 15, 20
 prolongpar (env.) 35, ℓ856

Q

quick reference [v-ix](#)

R

ragged right [51](#)

raise* (option) [3](#)

raisecapitaldash (option) [3](#), [21](#)

raisecapitalguillemets (option) [3](#), [24](#)

raisecapitalhyphen (option) [3](#), [18](#)

raisecapitaltimes (option) [3](#), [23](#)

raised character [20](#)

en-dash [21](#)

guillemets [23](#)

hyphen [21](#)

inverted exclamation mark [25](#)

inverted question mark [25](#)

multiplication sign [23](#)

number dash [22](#)

raisefiguredash (option) [3](#), [22](#)

raiseguillemets (option) [3](#), [24](#)

raiseinvertedmarks (option) [3](#), [26](#)

reconfigure [6](#)

\resetbaselineskip [49](#), [1457](#)

\rightkernedhyphen [18](#), [317](#), [322](#)

\rightkernedhyphen* [18](#)

\rightspacedcapitaldash [22](#), [483](#), [493](#)

\rightspacedcapitaldash* [22](#)

\rightspacedcapitalendash [22](#), [479](#)

\rightspacedcapitalendash* [22](#)

\rightspacedddash [19](#), [377](#), [392](#)

\rightspacedddash* [19](#)

\rightspacedendash [19](#), [362](#), [368](#), [377](#), [378](#), [379](#), [381](#), [392](#), [393](#), [394](#)

\rightspacedendash* [19](#)

S

\setbaselineskip [48](#), [1389](#), [1391](#), [1398](#), [1407](#), [1417](#), [1435](#), [1444](#), [1447](#), [1451](#)

\setbaselineskippercentage [49](#), [1470](#)

setfontexpand (env.) [41](#), [1078](#)

setfontshrink (env.) [41](#), [1050](#)

setfontstretch (env.) [41](#), [1064](#)

setfonttracking (env.) [40](#), [964](#)

\setleading [49](#), [1489](#)

\setleadingpercentage [49](#), [1507](#)

setup [6](#)

shortenpar (env.) [35](#), [849](#)

shrinklimits (option) [4](#), [41](#)

\Singleguillemetleft [24](#), [535](#)

\singleguillemetleft [24](#), [519](#)

\Singleguillemetright [24](#), [539](#)

\singleguillemetright [24](#), [523](#)

slashkern (option) [4](#), [17](#)

\slightlyloppy [42](#), [1115](#), [1173](#)

slightlyloppypar (env.) [42](#), [1172](#)

smoothraggedright (env.) [52](#), [1741](#)

\smoothraggedrightfuzzfactor [1647](#), [1682](#)

\smoothraggedrightgenerator [1648](#), [1677](#), [1730](#)

\smoothraggedrightleftskip [1649](#), [1678](#)

smoothraggedrightpar (env.) [51](#), [1656](#)

\smoothraggedrightparindent [1650](#), [1687](#), [1698](#), [1713](#)

\smoothraggedrighttragwidth [1651](#), [1682](#), [1684](#), [1689](#), [1692](#), [1695](#), [1700](#), [1703](#), [1705](#), [1707](#), [1710](#), [1715](#), [1717](#), [1719](#), [1721](#), [1723](#), [1726](#)

smoothraggedrightshape-

quintuplet (env.) [51](#), [1568](#)

smoothraggedrightshape-

septuplet (env.) [51](#), [1606](#)

smoothraggedrightshapetriplet (env.) [51](#), [1535](#)

spaced character

en-dash [19](#)

\spacedcapitaldash [485](#), [495](#)

\spacedcapitalemdash [22](#), [498](#)

\spacedcapitalendash [484](#), [494](#)

\spacedddash [19](#), [379](#), [394](#)

\spacedddash* [19](#)

\spacedemdash [20](#), [397](#)

\spacedendash [19](#), [378](#), [393](#)

\spacedendash* [19](#)

\splicevtietop [44](#), [1241](#)

stretchlimits (option) [4](#), [41](#)

T

\texttexclamdown [26](#)

textitalicscorrection (option) [4](#), [16](#)

\textquestiondown [26](#)

tightspacing (env.) [37](#), [948](#)

trackingttspace (option) [4](#), [40](#)

\typogadjuststairs [29](#), [783](#)

`\typogadjuststairsfor` [ℓ761](#)
`\typogfontsize (dim.)` [50](#), [ℓ1462](#),
[ℓ1473](#), [ℓ1476](#), [ℓ1481](#), [ℓ1482](#), [ℓ1492](#),
[ℓ1494](#), [ℓ1499](#), [ℓ1500](#), [ℓ1510](#), [ℓ1513](#),
[ℓ1518](#), [ℓ1519](#)
`\typogget` [6](#), [ℓ116](#)
`\typoggetnth` [7](#), [ℓ119](#)
`typoginspect (env.)` [8](#), [ℓ171](#)
`typoginspectpar (env.)` [8](#), [ℓ198](#)
`\typoglogo` [ℓ11](#)
`\typoglowercaseadjustcheck` [29](#),
[ℓ826](#)
`typogsetup (env.)` [6](#), [ℓ107](#)
`\typoguppercaseadjustcheck` [29](#),
[ℓ814](#)

U

`\unskip` [ℓ326](#), [ℓ331](#)
`uppercaseadjustlabelitem-`
adjustments (option) [4](#), [27](#)

`\uppercaseadjustlabelitems` [27](#),
[ℓ671](#)

V

`\vtiebot` [45](#), [ℓ1250](#), [ℓ1304](#)
`\vtiebotdisp` [ℓ1371](#)
`vtiebotdisp (env.)` [45](#), [ℓ1361](#)
`vtiebotdisptoppar (env.)` [45](#), [ℓ1365](#)
`vtiebotpar (env.)` [45](#), [ℓ1303](#)
`\vtietop` [44](#), [ℓ1183](#), [ℓ1237](#), [ℓ1243](#), [ℓ1369](#)
`vtietop (enumitem-key)` [45](#)
`vtietoppar (env.)` [44](#), [ℓ1236](#)

W

wide space [38](#)
`\widespace` [38](#), [ℓ902](#)
`\widespace*` [38](#)
`\widespacescale` [38](#), [ℓ901](#), [ℓ904](#),
[ℓ906](#), [ℓ912](#), [ℓ913](#)
`\widespacestrenght` [38](#), [ℓ900](#), [ℓ909](#)