

TypoG

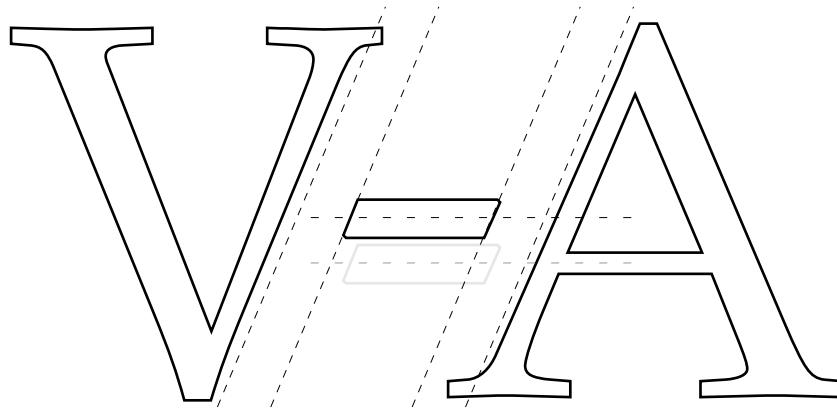
A L^AT_EX Package for Micro-Typographic Enhancements

Ch. L. Spiel*

v0.5 2025/10/27

Abstract

The `typog` package provides macros and environments to improve micro-typographic quality. It enables fine-tuning of hyphenation, spacing, font adjustments, and alignment. Features include control over T_EX's paragraph justification, customizable spacing, font characteristic adjustments, and specialized environments for precise paragraph layout and alignment.



This package is copyright © 2024, 2025 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 [16](#)
- 3.5 Manual Italic Correction [17](#)
- 3.6 Apply Extra Kerning or Spacing [18](#)
 - 3.6.1 Slash [18](#)
 - 3.6.2 Hyphen [19](#)
 - 3.6.3 En-Dash and Em-Dash [20](#)
- 3.7 Raise Selected Characters [24](#)
 - 3.7.1 Capital Hyphen [24](#)
 - 3.7.2 Capital En-Dash and Capital Em-Dash [25](#)
 - 3.7.3 Number Dash (Figure Dash) [26](#)
 - 3.7.4 Multiplication Sign [27](#)
 - 3.7.5 Guillemets [27](#)
 - 3.7.6 Inverted Marks [29](#)
- 3.8 Vertically Adjust Label Items [30](#)
- 3.9 Align Last Line [34](#)

Table of Contents continued on next page.

3.10	Fill Last Line	36
3.10.1	Problem Description	36
3.10.2	Manual Changes	37
3.10.3	Multi-Purpose Environments	38
3.10.4	Specialized Environments	39
3.10.5	Consistent Spacing of Last Line	40
3.11	Spacing	40
3.11.1	Looser/Tighter	40
3.11.2	Wide Space	42
3.11.3	Narrow Space	43
3.12	Microtype Front-End	44
3.12.1	Tracking	44
3.12.2	Font Expansion	45
3.12.3	Character Protrusion	46
3.13	Sloppy Paragraphs	46
3.14	Vertically Partially-Tied Paragraphs	48
3.15	Breakable Displayed Equations	51
3.16	Setspace Front End	52
3.17	Smooth Ragged	55

4

Limitations and Known Problems	59
--------------------------------	----

5

Other Packages for Fine L ^A T _E X Typography	60
--	----

A

typog-grep	61
------------	----

B

e- \TeX : Breaking Paragraphs into Lines	68
--	----

Table of Contents continued on next page.

C

Package Code 70

C.1	Setup and Reconfiguration	75	C.10	Fill Last Line	99
C.2	Information	76	C.11	Spacing	101
C.3	Hyphenation	78	C.12	Microtype Front-End	103
C.4	Disable/Break Ligatures	78	C.13	Sloppy Paragraphs	106
C.5	Manual Italic Correction	79	C.14	Vert. Tie Paragraphs	108
C.6	Apply Extra Kerning	80	C.15	Breakable Disp. Eqs.	113
C.7	Raise Selected Characters	85	C.16	Setspace Front End	113
C.8	Vert. Adjust Label Items	91	C.17	Smooth Ragged	117
C.9	Align Last Line	99			

Change History 123**References 125****Index 128****List of Tables**

1	Hyphens and automatic hyphenation	13
2	Comparison of <code>\rightspacedendash-comma</code> combinations	22
3	Suggested raise amounts for <code>\figuredash</code>	26
4	Suggested raise amounts for guillemets	28
5	Label item adjustment suggestions	35
6	Spacing changes made by <code>loosespacing</code>	41
7	Spacing changes made by <code>tightspacing</code>	41
8	<code>\fontdimen<number></code> parameters	42
9	Comparison of some space sizes	43
10	Shrink values of <code>setfontshrink</code>	45
11	Stretch values of <code>setfontstretch</code>	45
12	Shrink and stretch values of <code>setfontexpand</code>	45
13	Parameter adjustments of <code>\slightlyloppy</code>	47
14	Partial paragraph line counts	50
15	Env. <code>breakabledisplay</code> and <code>\interdisplaylinepenalty</code>	52

Quick Reference

This is an alphabetically sorted list of all user macros and environments defined by package `typog` along with the page numbers of their descriptions. A list of all [package options](#) can be found on pages 2 to 5. The [Index](#) on pages 128 to 132 may provide some more detailed insights.

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

In the multi-column parts of the Quick Reference, we reduce the line spacing from 125% to 120% with `\setbaseline-skippercentage`, and the inter-word spacing by 1.25% with `tightspacing`. List \items get tied with `\vtietop`.

A

<code>\Adjustedlabelitemi</code>	Typeset an uppercase-adjusted <code>\labelitemi</code> .	32
<code>\adjustedlabelitemi</code>	Typeset a lowercase-adjusted <code>\labelitemi</code> .	32
<code>\Adjustedlabelitemii</code>	Typeset an uppercase-adjusted <code>\labelitemii</code> .	32
<code>\adjustedlabelitemii</code>	Typeset a lowercase-adjusted <code>\labelitemii</code> .	32
<code>\Adjustedlabelitemiii</code>	Typeset an uppercase-adjusted <code>\labelitemiii</code> .	32
<code>\adjustedlabelitemiii</code>	Typeset a lowercase-adjusted <code>\labelitemiii</code> .	32
<code>\Adjustedlabelitemiv</code>	Typeset an uppercase-adjusted <code>\labelitemiv</code> .	32
<code>\adjustedlabelitemiv</code>	Typeset a height-adjusted <code>\labelitemiv</code> .	32
<code>\allowhyphenation</code>	(Re-)enable automatic hyphenation.	11

B

<code>breakabledisplay</code> [<i>level</i>]	Adjust the penalty associated with <code>\allowdisplaybreaks</code> .	51
--	---	----

<code>\breakpoint*</code>	Insert an empty discretionary.	14
---------------------------	--------------------------------	----

<code>\breakpoint</code>	Insert an empty discretionary and re-enable automatic hyphenation.	14
--------------------------	--	----

C

<code>\capitaldash*</code>	Alias for <code>\capitalendash*</code> .	25
----------------------------	--	----

<code>\capitaldash</code>	Alias for <code>\capitalendash</code> .	25
---------------------------	---	----

<code>\capitalemdash*</code>	Typeset a vertically adjusted <code>\textemdash</code> .	25
------------------------------	--	----

<code>\capitalemdash</code>	Typeset a vertically adjusted <code>\textemdash</code> that is breakable.	25
-----------------------------	---	----

<code>\capitalendash*</code>	Typeset a vertically adjusted <code>\textendash</code> .	25
------------------------------	--	----

<code>\capitalendash</code>	Typeset a vertically adjusted <code>\textendash</code> that is breakable.	25
-----------------------------	---	----

<code>\capitalhyphen*</code>	Typeset a vertically adjusted hyphen character.	24
------------------------------	---	----

<code>\capitalhyphen</code>	Typeset a vertically adjusted hyphen character that is breakable.	24
-----------------------------	---	----

<code>\capitalinverted-exclamationmark</code> { <i>number</i> }	Typeset an inverted ('upside-down') exclamation mark that is level with the baseline.	29
---	---	----

- `\capitalinverted-questionmark`{*<number>*}
Typeset an inverted ('upside-down') question mark that is level with the baseline. 29
- `\capitaltimes`
Typeset a vertically adjusted `\texttimes`. 27
- `covernextindentpar` [*<dim>*]
Extend the last line of a paragraph. 39
- D**
- `\Doubleguillemetleft`
Typeset left double guillemets vertically adjusted for uppercase. 27
- `\Doubleguillemetright`
Typeset right double guillemets vertically adjusted for uppercase. 27
- `\doubleguillemetleft`
Typeset left double guillemets vertically adjusted for lowercase. 27
- `\doubleguillemetright`
Typeset right double guillemets vertically adjusted for lowercase. 27
- F**
- `\figuredash*`
Typeset a `\textendash` vertically adjusted for figures (numerals). 26
- `\figuredash`
Typeset a breakable `\textendash` vertically adjusted for figures (numerals). 26
- `\fontsizeinfo`{*<csname>*}
Store the current em-height and `\baselineskip` in a pair of macros. 8
- H**
- `hyphenmin` [*<left-min>*] [*<right-min>*]
Set the values of `\lefthyphenmin` and `\righthyphenmin`. 15
- I**
- `\itcorr*`{*<strength>*}
Apply italic correction in the form of a `\kern` scaled by `\textit` correction. 17
- `\itcorr`{*<strength>*}
Apply italic correction in the form of a `\kern` scaled by `\fontdim1` or `\textit` correction. 17
- K**
- `\kernedhyphen*` [*<raise>*]▷
◁{*<left-kern>*} {*<right-kern>*}
Typeset an unbreakable hyphen and apply kerning to its left and right. 19
- `\kernedhyphen` [*<raise>*]▷
◁{*<left-kern>*} {*<right-kern>*}
Typeset a breakable hyphen and apply kerning to its left and right. 19
- `\kernedslash*`
Typeset an unbreakable forward slash and apply kerning to both its left and right sides. 18
- `\kernedslash`
Typeset a breakable forward slash and apply kerning to both its left and right sides. 18
- L**
- `lastlinecenteredpar`
Center the last lines of a paragraph. 34
- `lastlinefitpar`
Match the spacing of last line and next-to-last line. 40
- `lastlineflushrightpar`
Alias for `lastlinerraggedleftpar`. 34
- `lastlinerraggedleftpar`
Align the last line of paragraph flush right. 34
- `\leftkernedhyphen*` [*<raise>*] {*<left-kern>*}
Typeset a hyphen and apply kerning to its left-hand side. 19
- `\leftkernedhyphen` [*<raise>*] {*<left-kern>*}
Typeset a hyphen, apply kerning to its left-hand side, and insert a breakpoint after it. 19
- `\leftspaceddash*` [*<raise>*]
Alias for `\leftspacedendash*`.
- `\leftspaceddash` [*<raise>*]
Alias for `\leftspacedemdash`.

- `\leftspacedemdash*` [*raise*]
Typeset an em-dash with some space around it. Prohibit line breaks before and after it.
- `\leftspacedemdash` [*raise*]
Typeset an em-dash with some space around it. Allow line breaks at its left-hand side.
- `\leftspacedendash*` [*raise*]
Typeset an en-dash with some space around it. Prohibit line breaks before and after it.
- `\leftspacedendash` [*raise*]
Typeset an en-dash with some space around it. Allow line breaks at its left-hand side.
- `\loosespacing` [*level*]
Increase the width of the space character. 41
- `\lowercaseadjustlabelitems` {*levels*}
Activate the lowercase height-adjustment values inside `itemize` environments. 30
- N**
- `\narrowspace*`
Typeset a narrow space whose width depends on `\fontdimen7`. 43
- `\narrowspace`
Typeset a narrow space whose width depends on `\fontdimen7` or `\fontdimen2`. 43
- `\noadjustlabelitems` {*levels*}
Deactivate height-adjustment of label items. 30
- `nocharprotrusion`
Deactivate character protrusion. 46
- `nofontexpand`
Alias for `nofontexpansion`. 46
- `nofontexpansion`
Deactivate font expansion. 46
- `\nolig*` [*kerning*]
Break a ligature. 16
- `\nolig` [*kerning*]
Break a ligature and introduce a hyphenation opportunity. 16
- O**
- `openlastlinepar` [*dim*]
Open a paragraph's last line if it is almost full or completely filled. 39
- P**
- `prolongpar`
Increase the `\looseness` of a paragraph. 39
- R**
- `\resetbaselineskip`
Reset `\baselineskip` to its original value. 53
- `\rightkernedhyphen*` [*raise*] {*right-kern*}
Typeset a hyphen and apply kerning to its right-hand side. 19
- `\rightkernedhyphen` [*raise*] {*right-kern*}
Typeset a hyphen, apply kerning to its right-hand side, and insert a breakpoint after it. 19
- `\rightspaceddash*` [*raise*]
Alias for `\rightspacedendash*`.
- `\rightspaceddash` [*raise*]
Alias for `\rightspacedendash`.
- `\rightspacedemdash*` [*raise*]
Typeset an em-dash with some space around it. Prohibit line breaks before and after it.
- `\rightspacedemdash` [*raise*]
Typeset an em-dash with some space around it. Allow line breaks at its right-hand side.
- `\rightspacedendash*` [*raise*]
Typeset an en-dash with some space around it. Prohibit line breaks before and after it.
- `\rightspacedendash` [*raise*]
Typeset an en-dash with some space around it. Allow line breaks at its right-hand side.
- S**
- `\setbaselineskip-
percentage` {*percentage*}
Set `\baselineskip` as a percentage relative to the point size of the font. 53
- `\setbaselineskip` {*baselineskip*}
Set `\baselineskip` using an absolute length. 52

- `setfontexpand`[*<level>*]
Simultaneously set font stretch and shrink limits. 45
- `setfontshrink`[*<level>*]
Set font shrink limits. 45
- `setfontstretch`[*<level>*]
Set font stretch limits. 45
- `setfonttracking`{*<delta>*}
Override the default tracking for all fonts. 44
- `\setleadingpercentage`{*<percentage>*}
Set `\baselineskip` as a percentage using the leading. 53
- `\setleading`{*<leading>*}
Set `\baselineskip` using the leading. 53
- `shortenpar`
Decrease the `\looseness` of a paragraph. 39
- `\Singleguillemetleft`
Typeset left single guillemets vertically adjusted for uppercase. 27
- `\Singleguillemetright`
Typeset right single guillemets vertically adjusted for uppercase. 27
- `\singleguillemetleft`
Typeset left single guillemets vertically adjusted for lowercase. 27
- `\singleguillemetright`
Typeset right single guillemets vertically adjusted for lowercase. 27
- `slightlyloppypar`[*<sloppiness>*]
Format a paragraph with given sloppiness. 47
- `\slightlyloppy`[*<sloppiness>*]
Format using the given sloppiness. 47
- `smoothraggedright`[*<option>*...]
Format with one of the three smooth-ragged-right generators. 57
- `\smoothraggedrightfuzzfactor`{*<factor>*}
Relative amount of glue in smooth-ragged-right typeset lines. 57
- `\smoothraggedrightgenerator`{*<generator>*}
Name of the basis environment (*'generator'*) used in the environments `smoothraggedright` and `smoothraggedrightpar`. 57
- `\smoothraggedrightleftskip`
Value for `leftskip` that `smoothraggedright` and `smoothraggedrightpar` pass to the smooth-ragged-right generator. 57
- `smoothraggedrightpar`[*<option>*...]
Format a paragraph using one of the three smooth-ragged-right generators. 55
- `\smoothraggedrightparindent`
Value for `parindent` that `smoothraggedright` and `smoothraggedrightpar` pass to the smooth-ragged-right generator. 57
- `\smoothraggedrightparwidth`
Value for the width of the smooth-ragged-right margin in environments `smoothraggedright` and `smoothraggedrightpar`. 58
-
- `smoothraggedrightshapequintuplet`[*<option>*...]{*<width1>*}...{*<width5>*}
Prescribe five line lengths for formatting paragraphs. 55
- `smoothraggedrightshapeseptuplet`[*<option>*...]{*<width1>*}...{*<width7>*}
Prescribe seven line lengths for formatting paragraphs. 55
- `smoothraggedrightshapetriplet`[*<option>*...]{*<width1>*}{*<width2>*}{*<width3>*}
Prescribe three line lengths for formatting paragraphs. 55
-

- `\spacedcapitalemdash*`
Typeset a height-adjusted em-dash with some space around it. Prohibit line breaks before and after the dash. 25
- `\spacedcapitalemdash`
Typeset a height-adjusted em-dash with some space around it. 25
- `\spacedcapitallengdash*`
Typeset a height-adjusted en-dash with some space around it. Prohibit line breaks before and after the dash. 25
- `\spacedcapitallengdash`
Typeset a height-adjusted em-dash with some space around it. 25
- `\spacedddash*[\langle raise \rangle]`
Alias for `\rightspacedenddash*`. 21
- `\spacedddash[\langle raise \rangle]`
Alias for `\rightspacedenddash`. 21
- `\spacedemdash*[\langle raise \rangle]`
Alias for `\rightspacedemdash*`. 23
- `\spacedemdash[\langle raise \rangle]`
Alias for `\rightspacedemdash`. 23
- `\spacedenddash*[\langle raise \rangle]`
Alias for `\rightspacedenddash*`. 21
- `\spacedenddash[\langle raise \rangle]`
Alias for `\rightspacedenddash`. 21
- `\splicevtietop{\langle lines \rangle}`
Inside of a `list`-like environment, fuse the first lines. 48
- T**
- `tightspacing[\langle level \rangle]`
Decrease the width of the space character. 41
- `\typogadjuststairs[\langle factor \rangle]\>
<{\langle step \rangle}{\langle count \rangle}{\langle sample \rangle}`
Generate ‘stairs’ of vertically shifted label items. 32
- `\typogfontsize`
The default font’s quad size. 54
- `\typoggetnth{\langle dest \rangle}{\langle key \rangle}{\langle index \rangle}`
Retrieve single item of a typog compound configuration value. 7
- `\typogget{\langle key \rangle}`
Retrieve a typog configuration value. 6
- `typoginspectpar[\langle option \rangle]{\langle id \rangle}`
Turn on tracing of paragraphs and pages for a paragraph. 8
- `typoginspect[\langle option \rangle]{\langle id \rangle}`
Turn on tracing of paragraphs and pages. 8
- `\typoglowercaseadjust-check[\langle factor \rangle]{\langle sample \rangle}`
Typeset all four label items adjusted for lowercase with an indicator line. 33
- `typogsetup{\langle keys \rangle}`
Configure package typog. 6
- `\typoguppercaseadjust-check[\langle factor \rangle]{\langle sample \rangle}`
Typeset all four label items adjusted for uppercase with an indicator line. 33
- U**
- `\uppercaseadjustlabelitems{\langle levels \rangle}`
Activate the uppercase height-adjustment values inside `itemize` environments. 30
- V**
- `vtiebotdisptoppar[\langle num-before-lines \rangle]\>
<[\langle num-after-lines \rangle]`
Fuse a display with its preceding and following lines. 49
- `vtiebotdisp[\langle num-lines \rangle]`
Fuse a display with its preceding lines. 49
- `vtiebotpar[\langle num-lines \rangle]`
Fuse the final lines of a paragraph. 49
- `\vtiebot[\langle num-lines \rangle]`
Fuse final lines. 49
- `vtietoppar[\langle num-lines \rangle]`
Fuse the first lines of a paragraph. 48
- `vtietop[\langle num-lines \rangle]`
Fuse first lines. 48
- `\vtietop[\langle num-lines \rangle]`
Fuse first lines. 48
- W**
- `\widespace*`
Typeset a wide space whose width depends on `\fontdimen7`. 42
- `\widespace`
Typeset a wide space whose width depends on `\fontdimen7` or `\fontdimen2`. 42

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. However, it’s a long ball game.

1.1 Overview

The primary focus of `typog` are micro-typographic improvements, where it tries to hit it out of the ballpark in terms of typographic polish.

Section 3.1 presents how to reconfigure package `typog` after it has been loaded and how to access its configuration values for introspection or inclusion in the user’s own code. Section 3.2 addresses the need 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 the manual application of 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 visually pleasing manner. Section 3.8 also deals with vertical alignment and 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 much-needed macros for better control of the last line of a paragraph. Section 3.11 covers the manipulation of paragraph spacing. Section 3.12 details 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 specialized 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 is still experimental.

1.2 Prerequisites

Package `typog` requires e- \TeX ; it relies on the L^AT_EX3 interface. Parts of it are based on packages `microtype` [25] and `setspace` [29]. However, if the functionality is not used, `typog` can be used without `microtype`. The same holds true for the `setspace` front end.

The package was tested with **pdfTeX** 3.141592653-2.6-1.40.24 from the TeX Live distribution of 2022, as shipped by [Debian](#).

Throughout the whole document we indicate actual uses of the package’s macros and environments in the margin. All these notes are examples themselves, as they are typeset using `slightlyloppy`, `loose-spacing`, and `smoothragged-rightpar`. ¶ The title page has already demonstrated the effect of `lastlinecentered-par` in justified paragraphs for the abstract and the copyright notice.

2 Package Options

The `typog` package is loaded as usual via `\usepackage`. It can be configured during loading with package $\langle\text{OPTIONS}\rangle$ or later on with `typogsetup`, as described in Sec. 3.1. Since `typog` provides front ends to packages `microtype` and `setspace`, we mention them in the synopsis; they are *not* automatically loaded by `typog`.

```
\usepackage[...]{microtype} % Only required for macros and
                             % environments in Sec. 3.12.

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

\usepackage[\langle\text{OPTION}\rangle...]{typog}
```

The package $\langle\text{OPTIONS}\rangle$ also serve as configuration $\langle\text{key}\rangle$ 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 required”.

`breakpenalty`= $\langle\text{penalty}\rangle$

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 cannot be used with `typogsetup` or `\typogget`.

`emdashspace`= $\langle\text{glue}\rangle$

SINCE V0.5

Set the horizontal skip that is inserted before and after the em-dash in macros `\spacedemdash` and `\spacedcapitalemdash` to $\langle\text{glue}\rangle$. Default value: $56/1000$ em.

`endashspace`= $\langle\text{glue}\rangle$

SINCE V0.5

Set the horizontal skip that is inserted before and after the en-dash in macros `\spacedendash` and `\spacedcapitalendash` to $\langle\text{glue}\rangle$. Default value: $200/1000$ em plus $100/1000$ em minus $67/1000$ em.

`ligaturekern`= $\langle\text{dim}\rangle$

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

`lowercaselabelitemadjustments`={ $\langle\text{dim-1}\rangle$, ..., $\langle\text{dim-4}\rangle$ }

SINCE V0.4

Vertical shifts $\langle\text{dim-N}\rangle$ to apply to `\labelitem` $\langle N\rangle$, 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 \ast instructs `typog` to preserve $\langle\text{dim-N}\rangle$ at that position. The adjustments apply to the lowercase setting (`\lowercaseadjustlabelitems`). See Sec. 3.8 (in particular subsection ‘Setup’ and Tab. 5 on p. 35) and also configuration option `uppercaseadjustlabelitem`.

All four lengths default to 0 pt.

This subsection is typeset 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 $\langle dim \rangle$ lower the glyph, negative ones raise it. This is the opposite ‘direction’ of `\raisebox`. See Sec. 3.6.1. Default value: 0 pt.

RENAMED IN V0.5

`mathitaliccorrection=<dim>`

Italic correction in math mode. See Sec. 3.5 and also the complementary configuration option `textitaliccorrection`. Default value: 0.4μ .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.

`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.

`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.

`raisefiguredash=<dim>`

Set the length that the `\textendash` is raised in `\figuredash`. See Sec. 3.7.3. 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.

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

SINCE V0.5

Set the lengths by which the macros `\capitalinvertedexclamation-`

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

¹ Note that 1μ is $\frac{1}{48}$ em of the mathematical font's em.

`mark` and `\capitalinvertedquestionmark` raise their associated inverted exclamation marks and inverted question marks. Each dimension corresponds to the 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, for example, 1 sp. Empty list elements are ignored. The special value ⌘ 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 required

`stretchlimits`={ $\langle limit-1 \rangle$, $\langle limit-2 \rangle$, $\langle limit-3 \rangle$ } microtype required

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

`textitaliccorrection`= $\langle dim \rangle$

RENAMED IN V0.5

Italic correction fallback value; used if `\fontdimen1` is zero. See Sec. 3.5 on manual italic correction and also the complementary configuration option `mathitaliccorrection`. Default value: $\frac{20}{1000}$ em.

`trackingttspacing`={ $\langle outer-spacing \rangle$ } microtype required

Set the outer spacing of all typewriter fonts when used in the `settracking` environment as described in Sec. 3.12.1.

The argument $\langle outer-spacing \rangle$ gets passed to microtype’s `\SetTracking` option `outer spacing` [25, Sec. 5.3]. If the argument 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` $\langle N \rangle$, 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 ⌘ 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. 5 on p. 35) and also configuration option `lowercaseadjustlabelitem`.

All four lengths default to 0 pt.

Important

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

3 Macros and Environments

Easy things should be easy, and
hard things should be possible.
— LARRY WALL

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

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

3.1 Setup and Reconfiguration

`typogsetup` (*env.*)

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

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

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

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

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

Tip

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 could be attached to font-changing macros like `\rm`, `\sf`, or `\tt`. ■

`\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` using 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 of values as `lowercase-labelitemadjustments`, `shrinklimits`, `stretchlimits`, `trackingtt-spacing`, and `uppercase-labelitemadjustments` are, it may be convenient to fetch a particular list element of it.

```
\typoggetnth{<cname>}{<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 `<cname>`. 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 `<cname>` 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.

Important

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 [32]

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

We define some functions for inspecting 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 `<cname>`. Both dimensions are measured in points (pt) and the results are rounded to tenths.

- 2 We use `\fontdimen6`, the em-height as the font size.
- 3 The line spacing simply is `\baselineskip`.


```
\fontsizeinfo{<cname>}
```

The call to `\fontsizeinfo` defines a pair of macros to access the stored values. The unstarred version `\cname` expands to the lengths including their units (i. e., pt), the starred version `\cname*` 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 help users identify typographic issues quantitatively without getting distracted by unrelated information in the trace or the *log*-file.

```
\begin{typoginspect}[<option>]{<id>} ... \end{typoginspect}
\begin{typoginspectpar}[<option>]{<id>}
...
\end{typoginspectpar}
```

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

Option

tracingboxes[=*size*]

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

Caution

The end-of-trace marker sometimes gets placed too early and the trace seems truncated. \LaTeX reliably logs the requested 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="⟨id⟩" _job="⟨jobname⟩" _line="⟨line-number⟩" _page="⟨page-number⟩">
  ...
</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 extract the tags. EMACS users will find the function (`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.

The complete manual page of `typog-grep` is reproduced 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 `pdflatex` and `lualatex` 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 command-line.

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

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

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

Investigating the badness of a paragraph. It is generally unnecessary to determine the *exact* classification of a paragraph’s badness [17, 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 [17, p. 99]:

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

Example

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

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

Note

When package `microtype`’s font expansion 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/@/@/10\ls (+7) ...`

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

⁵ Reference 31 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., *pdf_{tex}.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 [17, 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, e. g. in a hyphenated compound.

The admittedly simple rules for when T_EX auto-hyphenates and when it does not give rise to so many different, yet interesting cases that we devote Tab. 1 to

⁸ See also the discussion of the T_EX output routines by SOLOMON [27].

them. The seemingly special cases shown there are not that uncommon, e. g., consider ‘spin-½’ which is coded as `\mbox{spin-\textfrac{1}{2}}`. A line break between the text and the fraction would garble the term.

Use Cases

- All examples from the bottom of Tab. 1 on p. 13.
- Establish additional hyphenation opportunities if these have been canceled by bricolage that is necessary for parenthesized German compounds like, e. g., “(Linear-)Impulsvektor”, “(Links-)Konjugation”, or “(Unter-)Gruppe”. All of these can be fixed by blunting the hyphen with `\mbox` and inserting a breakpoint with `\hskip` (or, as horizontal glue carries a penalty of zero, with `\breakpoint`, which is associated with `\breakpenalty`), like

```
(Linear\mbox{-})\hskip0pt Impulsvektor
```

where “Linear” won’t get hyphenated. Macro `\allowhyphenation` rectifies this situation:

```
(Linear\allowhyphenation\mbox{-})\hskip0pt Impulsvektor
```

Typographically the last code example raises the same concerns as does `\hyp`, this is, whether the hyphenation opportunities (here: *Li-ne-ar*), if chosen, distract from the construction of the compound that already contains a hyphen.

- Mend 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}}}
```

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` [6].

A `\narrowsspace` compensates the visual gap between the colon and the italic ‘L’.

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

LaTeX-Code	Result	Note
hyphenated-compound	hyphenated-compound	Most frequently used code; the hyphen <code>_</code> expands to <code>\discretionary{-}{-}{-}</code> rendering the parts unbreakable
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> [38]
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 hyphen unbreakable
hyphenated% \allowhyphenation-% \allowhyphenation compound	hy- phen- ated- com- pound	Macro <code>\allowhyphenation</code> of package <code>typog</code> ; hyphenate both parts, similar to <code>\hyp</code> shown above

and we want to have breakable “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.

\TeX never hyphenates the initial word in a paragraph even with `\allowhyphenation`. Start the paragraph with `\hskip 0pt` to enable hyphenation even for the first word.

Tip — Typewriter Fonts

Sometimes it is desired to get a breakable typewriter font. \LaTeX suppresses any hyphenation for fonts in `\ttfamily` by un-defining their `\hyphenchars`. If these are reassigned, the usual line breaking and 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 [17, p. 95], `\discretionary{}{}{}{}`, is helpful. 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 \LaTeX 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.

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

```
\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. ■

¹² The current values for `\lefthyphenmin` and `\righthyphenmin` in this document are 2 and 3.

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.

Alternatively use `\texorpdfstring` [24, 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 too 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 manual adjustment.

`\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 `textitaliccorrection` instead of `\fontdim1`. The starred version always uses `textitaliccorrection`. In math mode macro `\itcorr` uses the value set with `mathitaliccorrection`¹³ in both the starred and the unstarred form.

Typical slant angles of serif italic fonts range from 8° to 18° and thus values for `textitaliccorrection` 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 italic, i. e., compensate “shift-to-the-right effect” of italic. ¶ Positive correction at the left-hand side of italic, 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 1pt high?* Thus, `\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 italic collide with other – usually non-italic – 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 italic correction is.

¹³ Separate adjustments may be desirable if the math font’s italic 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 `\,`.

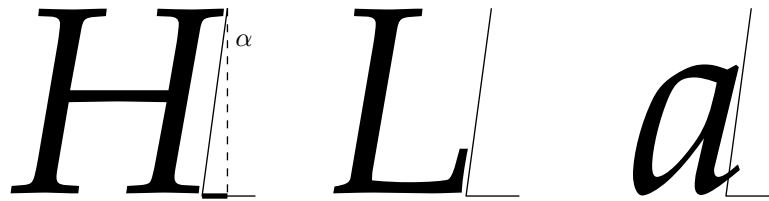


FIGURE 1: Some letters of an italic font. We use the capital \mathcal{H} to measure the angle α between the plumb-line (drawn dashed) and a tangent to the rightmost parts of the glyph. The length of the plumb-line is proportional to $\langle strength \rangle$ and the short, thick part of the baseline symbolizes the resulting italic correction. ¶ The middle example, the capital \mathcal{L} , shares α with \mathcal{H} but obviously needs a far smaller $\langle strength \rangle$ or even no correction at all. ¶ The \mathcal{a} at the right-hand side is an example of why \TeX allows to assign an italic correction to each individual character of a font. Not only features the lowercase \mathcal{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 using `lastlinecenteredpar`.

3.6 Apply Extra Kerning or Spacing

Package `typog` provides 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*
```

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), with the offset from the baseline configured using `lowerslash`.

Tip

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

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

Generic solutions could build upon `\fontdimen5` and `\fontdimen6` (see Tab. 8 on p. 42) 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. ■

OPTION `lowerslash`
INTRODUCED IN V0.5

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: $\langle year_1 \rangle / \langle year_2 \rangle$. ¶ The macro has proven helpful in many cases where the right hand side of the slash starts with a capital as, for example, $\langle city \rangle / \langle state-code \rangle$ (US-specific) or $\langle anything \rangle / \langle noun \rangle$ (any language that capitalizes $\langle noun \rangle$). ¶ Use option `lower slash` to adjust the slash to surrounding lowercase letters. ¶ Correct a too high raising slash of a font (e.g. Cochineal) with option `lower slash`. ■

3.6.2 Hyphen

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

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

Typeset an unbreakable hyphen with `\kernedhyphen*` or a breakable hyphen (like `\hyp` of package `hyphenat` [38]) with `\kernedhyphen` and apply some kerning to left and to the right of it. The values $\langle left-kerning \rangle$ and $\langle right-kerning \rangle$ are multiplied with one thousandth of the current font's em to get the size of the kern.

The optional argument $\langle raise \rangle$, 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 `*7` for $\langle raise \rangle$ applies the current value of `raisecapital-hyphen`. The default for $\langle raise \rangle$ 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 opposite 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 $\langle math \rangle - \langle noun \rangle$ in languages where nouns are capitalized. ¶ Composites where one or both sides of the hyphen are typeset in different fonts, like, $\langle small-caps \rangle - \langle roman \rangle$. ■

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

3.6.3 En-Dash and Em-Dash

The macros introduced in this subsection allow for fine-tuning of the whitespace before and after en-dashes and em-dashes. Each macro set is designed for a specific use. The spaced en-dash macros are intended to be used where an en-dash is surrounded by normal spaces:

$$\langle WORD \rangle_-_ \langle WORD \rangle.$$

Typographic improvement may come from reducing the width of the space around the en-dash with relative to the normal space.¹⁵ On the other hand, the em-dash macros assume that there is no space between the $\langle WORD \rangle$ s and the em-dash:

$$\langle WORD \rangle-\langle WORD \rangle.$$

The spaced em-dash macros add some (hairline) space between the em-dash and the $\langle WORD \rangle$ s, which helps prevent the em-dash from visually colliding with adjacent words.¹⁶

Line Breaking Around Dashes. Package `typog` offers spaced en- and em-dashes with breakpoints either on the left-hand side or the right-hand side (or none for any of the starred versions) and this is reflected in the macro names by `left` and `right`. All spaced dashes apply equal amounts of space on both sides of the dash; ‘left’ and ‘right’ refer to the breakpoints.

A single 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. The typographic rules favor either no line break at all or a break at the right-hand side [4, 6.82] of the dash, which also is \TeX ’s default.

Dashes are often used for appositions or explanatory phrases, as in the first sentence above. In this instance there exist two camps: one emphasizes the insertion and thus couples the lead-in and lead-out dashes to it (see item 1 below) [13, p. 173]. The other avoids a dash at the beginning of a line at the cost of severing it from the insertion (item 2). We demonstrate the alternatives with en-dashes.

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

$$\text{before_--}\sim\text{insertion}\sim\text{--_after}$$

For this solution `typog` offers `\leftspacedendash` and `\rightspacedendash`:

$$\text{before}\backslash\text{leftspacedendash} \\ \text{insertion}\backslash\text{rightspacedendash}\text{ after}$$

¹⁵ The current font’s space width is $233/1000$ em plus $116/1000$ em minus $78/1000$ em compared to typogs `en-dashspace` width of $200/1000$ em plus $100/1000$ em minus $67/1000$ em.

¹⁶ Some fonts offer dashes with no side bearings to facilitate drawing horizontal lines by abutting these dashes.

The insertion, “among . . .”, is separated by spaced en-dashes.

2. Another school insists to avoid dashes at the start of lines. Now the straightforward 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
```

The four 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.

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

Typeset an unbreakable en-dash with any of the starred variants. The unstarred `\left...-` macros have breakpoints at their left-hand sides and the `\right...-` macros breakpoints at their right-hand sides.

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

The optional argument `⟨raise⟩` allows adjusting the height of the en-dash above the baseline 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`. They apply to all macros that generate spaced en- and em-dashes.

<code>\spaceddash*[⟨raise⟩]\relax</code>	<code>\spaceddash[⟨raise⟩]\relax</code>
<code>\spaceddash*[⟨raise⟩]{}</code>	<code>\spaceddash[⟨raise⟩]{}</code>

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

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

L^AT_EX will trip with “Undefined control sequence” on `\typog@missing@-` argument if the extra argument is not passed.

Alternatively use `\texorpdfstring` [24, Sec. 4.1.2, p. 22]. ■

`\swapedashskip`
SINCE V0.5

If a `\rightspacedendash` is immediately followed by another punctuation mark as, e. g., a comma, the space at the right-hand side of the en-dash is ill-placed. It either should be removed by following the macro with an `\unskip` or `\swapedashskip`. Furthermore, `\rightspacedendash` should be made unbreakable.

```
\swapedashskip{<arg>}
```

Use `\swapedashskip{<arg>}`, where `<arg>` typically is a punctuation mark, to remove the right-hand space of any spaced en-dash macro and put exactly that amount of space on the right-hand side of `<arg>`. Table 2 compares the effects on the spacing.

TABLE 2: Comparison of uncorrected and corrected `\rightspacedendash`-comma combinations for a space width of $\frac{1}{3}$ em and a (exaggerated) `endashspace` of $\frac{1}{2}$ em. We have added “book struts” to the left and to the right of the results to indicate the width of the output.

Code	Result
<code>\rightspacedendash*, \space</code>	┆ - ,┆
<code>\rightspacedendash*\unskip, \space</code>	┆ -,┆
<code>\rightspacedendash*\swapedashskip,</code>	┆ -,┆

The four macros `\leftspacedemdash`, `\leftspacedemdash*`, `\rightspacedemdash`, and `\rightspacedemdash*` all expand to an em-dash `—` that is surrounded by whitespace. They differ in the ways they handle line breaking before or after the em-dash.

Let us pick up the alternatives explained in the section on en-dashes and adapt them to em-dashes. Note that by joining words with em-dashes T_EX *turns off* automatic hyphenation, which usually is the right thing to do to avoid confusion with dashes and hyphens. The spaced em-dash macros duplicate this behavior.

1. Tie the em-dashes to the insertion.

```
before\breakpoint\mbox{---}
insertion---after
```

Macro `\breakpoint` was introduced in Section 3.3.

```
before\leftspacedemdash
insertion\rightspacedemdash after
```

2. By default T_EX inserts breakpoints after em-dashes. So that we can just write

```
before---insertion---after
```

`\leftspacedemdash`
`\leftspacedemdash*`
`\rightspacedemdash`
`\rightspacedemdash*`
`\spacedemdash`
`\spacedemdash*`
ALL SINCE V0.5

The spaced solution with package `typog` looks like this:

```
before\spacedemdash
insertion\spacedemdash after
```

<code>\leftspacedemdash[⟨raise⟩]</code>	<code>\leftspaceddash</code> (alias)
<code>\leftspacedemdash*[⟨raise⟩]</code>	<code>\leftspaceddash*</code> (alias)
<code>\rightspacedemdash[⟨raise⟩]</code>	<code>\rightspaceddash</code> (alias)
<code>\rightspacedemdash*[⟨raise⟩]</code>	<code>\spacedemdash</code> (alias)
	<code>\spacedemdash*</code> (alias)

Typeset an unbreakable em-dash with any of the starred variants. The unstarred `\left...`-macros have breakpoints at their left-hand sides and the `\right...`-macros breakpoints at their right-hand sides.

The size of the space before and after the dash is configured with option `emdashspace`.

The optional argument `⟨raise⟩` allows adjusting the height of the em-dash above the baseline 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 `\spacedendash`.

Note

Alternatively use package `microtype` to achieve a similar effect. Macro `\SetExtraKerning` [25, Sec. 5.4] allows modifying the side-bearings of any glyph. ■

Tip

The macros rely on `\textemdash` and `\textendash` to typeset em-dashes and en-dashes. If different glyphs are desired redefine them (preferably locally).

Example with the fictitious macro `\fancydash`:

```
\let\origspacedemdash=\spacedemdash
\renewcommand*{\spacedemdash}
  {\begingroup
   \typogsetup{emdashspace=.2pt}%
   \let\textemdash=\fancydash
   \origspacedemdash
  \endgroup} ■
```


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.¹⁷

Caution

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

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

3.7.1 Capital Hyphen

`\capitalhyphen`
`\capitalhyphen*`

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

```
\capitalhyphen   \capitalhyphen*
```

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

Use Cases

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

¹⁷ Also compare with Ex. 12 in reference 37 for an attempt to automate vertical alignment.

3.7.2 Capital En-Dash and Capital Em-Dash

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

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

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

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

Use Cases

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

`\capitalemdash`
`\capitalemdash*`

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

```
\capitalemdash   \capitalemdash*
```

Use Cases

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

`\leftspacedcapitalendash` Combine raising an en-dash and adding some extra space around it.

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

```
\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 `\capitalendash` with those of `\spacedendash` as described in Sec. 3.6.3.

`\leftspacedcapitalemdash` Combine raising an em-dash and adding some extra space around it.

`\leftspacedcapitalemdash*`
`\rightspacedcapitalemdash`
`\rightspacedcapitalemdash*`
`\spacedcapitalemdash`
`\spacedcapitalemdash*`

```
\leftspacedcapitalemdash   \leftspacedcapitalemdash*
\rightspacedcapitalemdash   \spacedcapitalemdash (alias)
\rightspacedcapitalemdash*  \spacedcapitalemdash* (alias)
```

ALL SINCE V0.5

These macros unite the features of `\capitalemdash` with those of `\spacedemdash` as described in Sec. 3.6.3.

Tip

The macros rely on `\textemdash` and `\textendash` to typeset em-dashes and en-dashes. If different glyphs are desired redefine them (preferably locally).

See the [tip](#) on p. 23 for a code example. ■

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 3 summarizes some findings.

TABLE 3: 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.

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` [11]).

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 \times

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

```
\capitaltimes
```

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

Use Case

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

3.7.5 Guillemets

The sentence uses `\leftspaced-endash` and `\rightspaced-endash`.

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 T1¹⁸ font encoding not on package babel.

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

Lowercase Versions.

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

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

```
\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 4 summarizes some findings.

¹⁸ Font encoding T1 can be forced via `\usepackage[T1]{fontenc}` in the document preamble.

TABLE 4: 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 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 exhibit descenders (see left-hand side of Fig. 2). For all-uppercase or all-smallcaps phrases, e. g. in headlines the descending parts of these marks disturb the tight block structure and it may be preferable to have inverted marks that align with the baseline.

`\capitalinvertedexclamationmark` The macros `\capitalinvertedexclamationmark` and `\capitalinvertedquestionmark` 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 text¹⁹ or just for some stylistic alternatives offered by particularly well equipped fonts.

BOTH SINCE V0.5

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

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

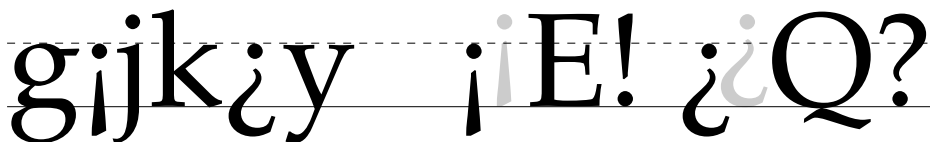


FIGURE 2: The left-hand sample shows the inverted exclamation marks and question marks wedged between some lowercase letters. The baseline is drawn with a solid line and the ex-height as well as the descender depth are indicated with dashed lines. ¶ On the right hand side we display the same marks in 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 `<number>` are raised by the same amount which is configurable with option `raiseinvertedmarks`, where `<number>` corresponds to the dimension at position `<number>`. If

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

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 near-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. 17) offers a solution that is almost portable across font changes and a $\langle strength \rangle$ of -1 is a good starting point. ■

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

```
\typog@inverted@exclamationmark@{\langle roman-numeral \rangle}
\typog@inverted@questionmark@{\langle roman-numeral \rangle}
```

where $\langle roman-numeral \rangle = i, ii, \text{ and } iii$ correspond to $\langle number \rangle = 1, 2, \text{ and } 3$. Their defaults are `\texttexclamdown` and `\textquestiondown` for all three numbers. 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.²⁰

`\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.

ALL THREE SINCE V0.5

```
\uppercaseadjustlabelitems{\langle levels-to-adjust \rangle}
\lowercaseadjustlabelitems{\langle levels-to-adjust \rangle}
\noadjustlabelitems{\langle levels-to-adjust \rangle}
```

Apply uppercase adjustment, lowercase adjustment or no adjustment to the labels in `itemize` environments at the $\langle levels-to-adjust \rangle$. The adjustment values

²⁰ The exception being mathematics typeset as text via package `mathastext` [8].

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 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.

```
\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 example uses this code. ■

Setup. To assist the user in finding the desired adjustments of the labels of typog provides macros that help setting up `lowercaselabelitemadjustments` and `uppercaselabelitemadjustments`. 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 $\frac{1}{2}$.²¹ 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.

`\Adjustedlabelitemi`
`\Adjustedlabelitemii`
`\Adjustedlabelitemiii`
`\Adjustedlabelitemiv`
`\adjustedlabelitemi`
`\adjustedlabelitemii`
`\adjustedlabelitemiii`
`\adjustedlabelitemiv`

```
\Adjustedlabelitemi      \adjustedlabelitemi
\Adjustedlabelitemii     \adjustedlabelitemii
\Adjustedlabelitemiii    \adjustedlabelitemiii
\Adjustedlabelitemiv     \adjustedlabelitemiv
```

ALL EIGHT SINCE V0.5

Typeset label items that are height-adjusted according to `uppercaselabelitemadjustments` or `lowercaselabelitemadjustments` either for uppercase (macros starting with a capital `A`) or lowercase (macros starting with 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 [23]. ■

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

```
\typogadjuststairs[<scale-factor>=.5]
                    {<step-size>} {<number-of-steps>}
                    {<sample>}
```

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

21 The default factor of $\frac{1}{2}$ hearkens back to STRIZVER’s suggestion that “[b]ullets should be centered on either the cap height or x-height of the neighboring text, [...]” [28, p. 220].

`\typogadjuststairs`
 SINCE V0.5

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

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

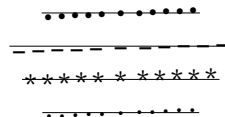
and repeat up to

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

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

Example

Play ball!



This is the result of `\typogadjuststairs{.25pt}{11}{ABC}` with the document's definitions of `\labelitem{N}`. The middle (6th) label item in each line is the uncorrected one. ■

`\typoguppercaseadjustcheck`
`\typoglowercaseadjustcheck`

BOTH SINCE V0.5

For a quick and easy check how the four label items vertically align *as configured* use `\typoguppercaseadjustcheck` and `\typoglowercaseadjustcheck`. Experienced users with a keen eye for type can apply these macros even in the initial setup. An accurate determination of uppercase-labelitemadjustments and lowercase-labelitemadjustments is preferably done at a high magnification (e. g., 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 $\langle scale-factor \rangle$ times the $\langle sample \rangle$'s actual height. The default $\langle scale-factor \rangle$ is .5. Both macros refer to the currently configured values for the uppercase or lowercase adjustments but they are independent of any settings done with `\uppercaseadjustlabelitems`, `\lowercaseadjustlabelitems`, or `\noadjustlabelitems`. Again, $\langle sample \rangle$ does not get printed.

Example

Uppercase check with `\typoguppercaseadjustcheck{ABCXYZ}`:

ABGH***QWYZ, 0123***4567

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

ace***mno, bdf***gjh, 0123***4567,

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

In Table 5 on p. 35 we collected some suggestions for adjustment values in the *default* case when the label items are not redefined by the user and expand like

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

They display as \cdot , --- , \ast , and \cdot .

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` [26]) are in effect. For an instructive discussion consult Ch. 17, “Paragraph End”, of reference 12. The following environments adjust the last lines of paragraphs in different ways.

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

`lastlinerraggedleftpar`
(*env.*)

`lastlineflushrightpar`
(*env.*)

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

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

Center the last lines of the paragraphs enclosed by this environment.²²

`lastlinecenteredpar`
(*env.*)

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

Use Cases

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

²² Also compare with the approach taken in reference 34.

TABLE 5: 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 relate to a document font size of 10 pt.

Font Name	Uppercase Adjustments				Lowercase Adjustments			
	1	2	3	4	1	2	3	4
ADF Accanthis	1.0	.75	1.125	1.125	.0	-.25	-.125	.0
ADF Venturis	.75	.5	.75	.875	-.125	-.25	-.125	-.125
Aleo	1.25	1.0	1.25	1.25	.0	-.25	.0	.125
Charis SIL	1.0	1.0	1.0	1.125	.0	-.25	.0	.125
CM Roman	1.0	.75	1.0	1.0	-.25	-.5	-.25	-.25
Domitian	1.0	.75	1.0	1.0	.0	-.25	-.25	.0
Cochineal	.75	.75	.75	.75	-.25	-.5	-.25	-.25
EB Garamond	1.0	.75	.75	1.0	-.25	-.5	-.25	-.25
etbb*	1.0	.75	1.0	1.0	-.25	-.5	-.25	-.25
Extended Charter†	1.0	.75	1.0	1.0	.0	-.25	.0	.0
Gentium Plus	.75	.5	.75	.75	-.25	-.5	-.25	.0
GFS Bodoni	1.0	.75	1.0	1.0	.0	-.25	-.125	.0
GFS Didot	1.25	1.0	1.25	1.25	.0	.0	.0	.0
IBM Plex Serif	1.125	1.0	1.125	1.25	.25	.25	.25	.25
KP Serif‡	1.0	.75	1.0	1.0	-.25	-.5	-.25	-.25
Libertinus Serif	.75	.5	.75	.875	-.25	-.5	-.25	-.25
ML Modern	1.	.75	1.0	1.0	-.25	-.5	-.25	-.25
Source Serif Pro	1.0	.75	1.0	1.0	.0	-.25	.0	.0
Spectral	1.0	.75	.75	1.0	.0	-.25	-.25	.0
STIX	1.0	.75	1.0	1.0	-.25	-.5	-.25	.0
URW Palladio§	1.0	.75	1.0	1.0	.0	.0	.0	.0
Utopia	.75	.5	.75	.75	-.25	-.5	-.25	-.25

* 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 kfonts.

§ Contained in package mathpazo.

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

Note

Depending on the font packages loaded (and on their load order) the three macros `\textendash`, `\textasteriskcentered`, `\textperiodcentered` may be redefined and thus expand to different glyphs which need different adjustment values than the ones mentioned in Table 5. To ensure a set of correction values works use macros `\typoguppercaseadjustcheck` or `\typoglowercaseadjustcheck` inside of the target document. ■

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 Description

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` [34, O1]

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



A similar issue occurs with displayed math in a flush left²³ setting, e. g., `amsmath` [2] and option `fleqn`.²⁴

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



2. `\parindent = 0` [34, 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.²⁵



²³ The common practice of centering displayed equations does not call for the manipulations of a paragraph's last line discussed here.

²⁴ For displayed equations and `amsmath` the relevant parameter is `\mathindent`.

²⁵ Package `parskip` defines `\parskip` as 6 pt plus 2 pt for a base size of 10 pt.

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



Recommended gap widths range from two ems to twice the typical `\parindent`²⁶ value [9].

Tip

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

\TeX evaluates paragraphs as a whole. Therefore, even changes intended for just the last line may affect the entire paragraph and potentially degrade its appearance.

Prudent users check the appearance of the problematic, original paragraph against one or more corrected versions—at a minimum, visually. Quantitative comparisons can be performed using `\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. ■

3.10.2 Manual Changes

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

1. End-of-paragraph intervention.

(a) Tie `\sim`

Tie the last words.

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

(b) `\mbox`

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

²⁶ For example, \LaTeX 's class article uses a `\parindent` of 25 pt.

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

(c) `\linebreak`

Add a `\linebreak` to the back part of the paragraph (approximately where the `\mbox` of item [1b](#) would start) in a way that the last line receives the desired length [\[36\]](#). As a result, the penultimate lines may become unsightly. Counteract this degradation e. g. with recipes [2a](#) to [2c](#).

Tying and the use of `\mbox` can be generalized. They are useful not only at the end of paragraphs but also effective for grouping logical units of sentences or inline equations, ensuring the key information remains in the reader's focus. Of course, binding text segments together must stop when overfull lines begin to appear.

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 *<level>* 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 line²⁷ or a club line²⁸ [\[17, p. 104 and 21\]](#). (See also Sec. [3.14](#)

²⁷ The last line of a paragraph becomes a 'widow' (ger. *Hurenkind*) if it starts the following page or column.

²⁸ The first line of a paragraph is called 'club' or 'orphan' (ger. *Schusterjunge*) if it appears at the bottom of the page or column.

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

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

Environment `shortenpar` decreases the `\looseness` of the paragraph.²⁹ This environment works well when the last line is short or the paragraph overall is loosely set.

```
\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 nearly 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 O1](#), i. e., a too short last line.

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

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

`openlastlinepar`
(*env.*)

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

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

It may resolve [case O2](#) as it attempts to prevent a completely filled line by introducing a partly unshrinkable `\parfillskip`. Without optional argument the threshold of unused last-line length is either $2\backslash\text{parindent}$ (if $\backslash\text{parindent} \neq 0$) or 2em (if $\backslash\text{parindent} = 0$). The optional argument $\langle dim \rangle$ directly sets the gap threshold.

Note that using this environment can succeed in avoiding a fully filled last line; however, the result might still fall into [case type O1](#).

²⁹ `\looseness` is a \TeX primitive [17, p. 103n]. A thorough discussion of the interaction of `\linepenalty` and `\looseness` can be found in reference 33.

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` [7, Sec. 3.8].³⁰ 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.

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

```
\begin{lastlinefitpar}[<value>] ... \end{lastlinefitpar}
```

The parameter range of `<value>` 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 `<value> = 0` requests no transfer (as if `\lastlinefit` is not used at all) and `<value> = 1000` initiates exact transfer of the spacing. The default `<value>` is 1000.

Note that different `<value>`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. ■

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

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 [20, p. 119]

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

On the one hand, they 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 is also important to ensure the monotonicity of the different `<level>`s. However, the strictly managed stretchability/

³⁰ 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 starting on p. 68.

shrinkability may lead to many overfull boxes with `\fussy`, or when applied to short lines.

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

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

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

Environment `loosespacing` increases the width of a space by the percentages given in Tab. 6.

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

TABLE 6: 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. 7.

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

TABLE 7: 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

Tighten spacing in narrow lines as, for example, in multi-column environments. ¶
 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⟨number⟩`s. T_EX addresses the latter by integers, which is totally non-memnonic. Therefore, we play softball by first presenting Tab. 8 that associates the `\fontdimen⟨number⟩`s with their meanings and also reports on their current values (for this document).³¹

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

TABLE 8: All T_EX font parameters normalized to the font's quad-size. The first column # states the index of the `\fontdimen⟨number⟩` parameter: `⟨number⟩`. Column 2 presents short descriptions of `\fontdimen⟨number⟩`. As examples, the values for the current font are shown in column 3 as percentages.

Typeset a wide, sentence-ending space as if in `\nonfrenchspacing` mode. Consult Table 9 for a comparison of the various sizes.

`\widespace`
`\widespace*`
STARRED FORM
SINCE V0.2

```
\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

$$\fontdimen2 + \widespacestrength \times \fontdimen7.$$

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

$$\widespacescale \times \fontdimen2$$

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

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

³¹ The association is given in Appendix F (p. 433) of reference 17. For a concise and understandable explanation of the T_EX `\fontdimen` parameters consult reference 10.

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

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`.³² ■

3.11.3 Narrow Space

`\narrowospace` Typeset a narrow space. Consult Table 9 for a comparison of the various sizes.

`\narrowospace*`
SINCE v0.2

```
\narrowospace \narrowospace*
```

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

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

if `\fontdimen7` is different from zero or otherwise

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

The starred version, `\narrowospace*`, unconditionally uses the `\fontdimen7 = 0` code-path. Refer to Table 8 for the meanings of the various `\fontdimen` parameters.

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

Use Cases

Tighten loose lines with a series of `\narrowospace`s.³³ ¶ Compensate the visual gap that may occur when switching from upright shape to italic shape. ■

TABLE 9: Exemplary comparison of standard `\space` versus `\narrowospace` and `\widespace`. All values are relative to the size of the current font's quad-size and shown as a percentage of it. `\narrowospace` and `\widespace` use the package's defaults. ¶ The upper values in the 'Width' column for `\narrowospace` and `\widespace` refer to the `\fontdimen7 ≠ 0` case and the lower ones to the `\fontdimen7 = 0` code-path.

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

32 See also "Investigating the badness of a paragraph" on Page 10.

33 Footnote 32 again applies.

3.12 Microtype Front-End

The manual uses `\spaced-`
`emdash` for the em-dash.

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

```
\usepackage[<microtype-options>...]{microtype}
\usepackage[<typog-options>...]{typog}
```

in the document preamble. ■

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` [25] 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 using `\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.

```
\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 10 and 11 summarize the values for stretch and shrink in these environments.

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

TABLE 10: 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 default
1	5	0	
2	10	0	
3	20	0	

TABLE 11: 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 12 gives an overview of the values associated with $\langle level \rangle$.

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

TABLE 12: 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 [25].

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

Experienced \LaTeX users know that `\sloppy` is more of a problem by itself and not really a viable solution of the “overfull box” syndrome.

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

We define the macro `\slightlyloppy` and the associated environment, `slightlyloppypar`, with a user-selectable $\langle sloppiness \rangle$ 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.³⁴ The default $\langle sloppiness \rangle$ is 1.

³⁴ Also compare with the findings for `\emergencystretch` in reference 32.

```

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

```

Table 13 summarizes the adjustments that `\slightlyloppy` makes depending on the `⟨sloppiness⟩` level.

TABLE 13: 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 `sloppypar[⟨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`.³⁵ 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 [7, 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 macro. ¶ 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` [5]:

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

or shorter and with the default *⟨number-of-lines⟩*, 3, using the `enumitem` style³⁶ `vtietop`:

³⁵ A valuable complement to these is package `needspace` [40] which takes a different approach and reliably works in *mixed* horizontal and vertical mode situations.

³⁶ 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 $\langle number-of-lines \rangle$ in a paragraph. Zero or one for $\langle number-of-lines \rangle$ are no-ops. Up to nine lines can be fused. The default is to link three lines. Avoid a display widow line in each partial paragraph.

`vtiebotdisp` (*env.*)

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

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

To use the environment, 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 $\langle before-disp-number-of-lines \rangle$ in the paragraph before a display and the first $\langle after-disp-number-of-lines \rangle$ in the paragraph after the display. Moreover, turn the paragraphs and the display into an unbreakable unit.³⁷

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

³⁷ 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 environment, 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 14 summarizes these rules with the help of an example.

TABLE 14: 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`³⁸ or `\shortintertext`³⁹ 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

Avoid widows and orphans, e. g., those turned up by package `widows-and-orphans` [22]. ¶
 Extend the convention of using three to four lines instead of a single club or widow line to a flexible, context-dependent rule that aims to keep all the relevant information together for the reader—ideally, at least. ■

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 results in fewer breaks in displayed equations, making `breakabledisplay` more suitable for automated page breaking.

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

Environment `breakabledisplay` simply passes on $\langle level \rangle$ to `\allowdisplaybreaks`. Table 15 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. Note, however, that `\displaybreak` is only effective (immediately) in front of `*^`. See the “User’s Guide for the `amsmath` Package” for a discussion [2, Sec. 3.9].
- 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}} ■
```

38 Introduced in package `amsmath` [2].

39 Defined in package `mathtools` [14].

TABLE 15: Penalties `\interdisplaylinepenalty` associated with different *⟨level⟩*s of environment `breakabledisplay`. Depending on the version of package `amsmath` the actual penalties may differ.

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

[†] 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` [29] 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.⁴⁰ 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{⟨baseline-skip⟩}
```

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

```
\setlength{\baselineskip}{⟨baseline-skip⟩}
```

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

⁴⁰ 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{12.5 pt}
```

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

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

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

`\resetbaselineskip`
SINCE V0.3

Reset the `\baselineskip` to its original value.

```
\resetbaselineskip
```

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

`\setbaselineskippercentage`
SINCE V0.3

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

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

Set `\baselineskip` to `\typogfontsize × <baselineskip-percentage>/100`.

Example

We modify the previous example and assume a point 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 point 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 point 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:⁴¹

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

which is also called its “nominal size” or its “point 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 point 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 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} ■
```

⁴¹ For an overview of the various `\fontdimen<number>` parameters consult Tab. 8 on p. 42.

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 typesetting ragged paragraphs. Instead of setting the glue inside a paragraph to zero and letting the line widths vary accordingly [35], we prescribe the line widths with T_EX's `\parshape` primitive and leave the stretchability and shrinkability of the glue unchanged.

Caution

None of the following environments work within lists (e.g. `description`, `enumerate`, or `itemize`). ■

<code>smoothraggedrightshapetriplet</code> (<i>env.</i>)	We introduce three environments that define three, five, or seven different line lengths (which T _E X will, of course, 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}[\option]...{\width1}{\width2}{\width3}
...
\end{smoothraggedrightshapetriplet}
\begin{smoothraggedrightshapequintuplet}[\option]...{\width1}...{\width5}
...
\end{smoothraggedrightshapequintuplet}
\begin{smoothraggedrightshapeseptuplet}[\option]...{\width1}...{\width7}
...
\end{smoothraggedrightshapeseptuplet}
```

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

Option `leftskip=\langle dim \rangle` sets the left margin of the smooth ragged paragraph to $\langle dim \rangle$; it is similar to the T_EX parameter `\leftskip`. Option `parindent=\langle dim \rangle` sets the first-line indent of the smooth ragged paragraph to $\langle dim \rangle$; it is similar to the T_EX parameter `\parindent`.

<code>smoothraggedrightpar</code> (<i>env.</i>)	Environment <code>smoothraggedrightpar</code> builds upon the three environments just described, using them as ‘generators’. It prescribes three, five, or seven relative line lengths that are known to yield convincing rags.
--	---

```
\begin{smoothraggedrightpar}[\option]...
...
\end{smoothraggedrightpar}
```

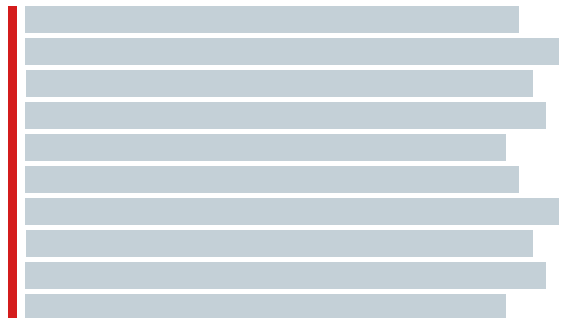

Typeset a single paragraph with a given rag width—set with the global parameter `smoothraggedrightwidth`—of the right margin, where with “rag width” we mean the length difference of the longest and the shortest lines. Select the generator with `smoothraggedrightgenerator`. The line lengths equally divide the ragged margin, i. e., they form an arithmetic sequence based on the generator size.

Option `linewidth=<dim>` overrides the length of the longest line with `<dim>`. The default line width is the current `\linewidth`.

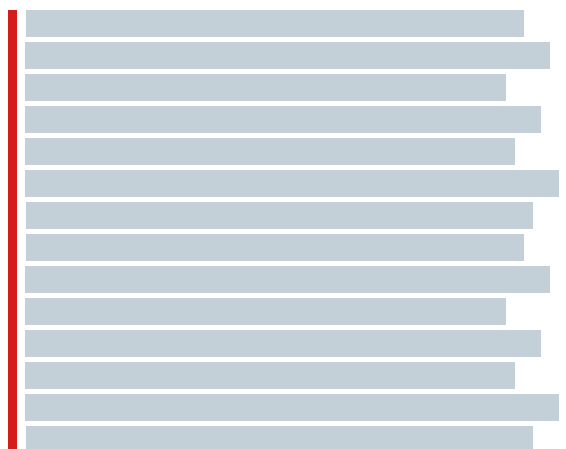
- The triplet generator repeats a *short line–long line–middle-length line* sequence. Shown below are two complete cycles.



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



- The septuplet generator uses a permutation that appears ‘random’; at the very least, it conceals the cycle boundaries effectively. Shown here are two of them.



The graphical representations of the smooth-ragged-right-paragraph shapes shown above are based on the actual values used by the environments and not made up only for pretty figures.

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

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

`\smoothraggedrightfuzzfactor` The environments `smoothraggedright` and `smoothraggedrightpar` add glue to every line width⁴² 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).

```
\renewcommand*{\smoothraggedrightfuzzfactor}{factor}
```

The `\smoothraggedrightfuzzfactor` is the amount of glue of each line expressed as a multiple of the distance between the division points. The default `<factor>` of 1.0 means that glue is added just enough to prevent the lines from overlapping (assuming justification is feasible).

`\smoothraggedrightgenerator` This macro selects the generator to use.

```
\renewcommand*{\smoothraggedrightgenerator}{generator}
```

Valid `<generator>` names are

- triplet,
- quintuplet, and
- septuplet.

The default generator is `triplet`.

`\smoothraggedrightleftskip` (skip) Value for `leftskip` to pass to the generator.

```
\smoothraggedrightleftskip=dim
```

Default: 0 pt.

`\smoothraggedrightparindent` (skip) Value for `parindent` to pass to the generator.

```
\smoothraggedrightparindent=dim
```

Default: 0 pt.

⁴² The shortest line only gets stretchability, the longest receives only shrinkability. All other lines are both stretchable and shrinkable.

`\smoothraggedright` `ragwidth` Value for the width of the ragged right margin.
(*skip*)

```
\smoothraggedrightragwidth=<dim>
```

Default: 2 em.⁴³

Use Cases

Replacement for `\RaggedRight` [26]. ¶ A design alternative to fully justified paragraphs when used with a small rag width. ■

Anticipated Changes & Possible Extensions

Translate the code to `l3galley` which is part of the `l3experimental` package [18]. Galley code is supposed to work inside of lists, too. ■

Sample Smooth-Ragged-Right Paragraph at Full Text-Width

Throughout this manual, we have demonstrated how `smoothraggedright` environments work for very narrow columns—namely, inside the document’s margins: all marginal notes were typeset with `smoothraggedright` environments (quintuplet generator, 1.5 em rag width, at footnote size in addition to using environments `slightlyloppy` and `loosespacing`). In this paragraph, we utilize it with the quintuplet generator and a rag width of only 7 pt in the body that is 355 pt 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 more easily. The sloppiness is minimal; that is, `\fussy` is in effect, and character protrusion into the margins is disabled. 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 of our examples are implemented as lists.

⁴³ This is also the default rag width of `ragged2e`’s `\RaggedRight` [26].

4 Limitations and Known Problems

Here is a list of some known problems with typog.

Interference with the package pdfcomment [16]. To ensure compatibility, typog *disables* the default string substitution inside PDF bookmarks for the following macros (including their starred variants) if *both* packages are loaded.

<code>\Adjustedlabelitemi</code>	32	<code>\kernedhyphen</code>	19
<code>\Adjustedlabelitemii</code>	32	<code>\kernedslash</code>	18
<code>\Adjustedlabelitemiii</code>	32	<code>\leftkernedhyphen</code>	19
<code>\Adjustedlabelitemiv</code>	32	<code>\leftspacedemdash</code>	23
<code>\adjustedlabelitemi</code>	32	<code>\leftspacedendash</code>	21
<code>\adjustedlabelitemii</code>	32	<code>\nolig</code>	16
<code>\adjustedlabelitemiii</code>	32	<code>\rightkernedhyphen</code>	19
<code>\adjustedlabelitemiv</code>	32	<code>\rightspacedemdash</code>	23
<code>\breakpoint</code>	14	<code>\rightspacedendash</code>	21
<code>\capitalemdash</code>	25	<code>\Singleguillemetleft</code>	27
<code>\capitalendash</code>	25	<code>\Singleguillemetright</code>	27
<code>\capitalhyphen</code>	24	<code>\singleguillemetleft</code>	27
<code>\capitaltimes</code>	27	<code>\singleguillemetright</code>	27
<code>\Doubleguillemetleft</code>	27	<code>\spacedcapitalemdash</code>	25
<code>\Doubleguillemetright</code>	27	<code>\spacedcapitalendash</code>	25
<code>\doubleguillemetleft</code>	27	<code>\spaceddash</code>	21
<code>\doubleguillemetright</code>	27	<code>\spacedemdash</code>	23
<code>\figuredash</code>	26	<code>\spacedendash</code>	21
<code>\itcorr</code>	17		

The sentence uses `\spaced-`
`endashes`.

Use `\texorpdfstring` – which is part of the package `hyperref` [24] – in the relevant sectioning macros to restore 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 produce 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^AT_EX list types, including inline lists [5].
- `geometry` Powerful and sophisticated configuration of page layout [30]. Best used together with `layout` [19] to visualize the page geometries.
- `hyphenat` Provides hyphens that do not inhibit further auto-hyphenation of compound words [38].
Also see Section 3.3 (pp. 11–15) for this package’s macros and environments that are related to hyphenation.
- `microtype` Fine control of spacing, tracking, sidebearings, character protrusion into the margins, font expansion, and more [25].
See Section 3.12 (pp. 44–46) for a front end to `microtype` provided by this package and also the discussion in reference 15.
- `ragged2e` Improved versions of the `raggedleft`, `raggedright`, and `center` environments [26].
- `setspace` Consistently sets the line spacing of a document (i. e., controls `\baselineskip`) [29].
See Section 3.16 (pp. 52–54) for a front end to `setspace` provided 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 typoginspect 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`. 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 *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 `LATEX` 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, \LaTeX 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 [7], 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. 40.

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.⁴⁴

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:

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

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.

⁴⁴ 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. 48 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           [2025/10/27 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\kern-.12em\textsl{y}poG}}
```

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

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

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

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

`\typog@typeout` Our debug information printer.

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

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

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

`\typog@trim@spaces` Pull `\tl_trim_spaces` into the ‘classic’ namespace.

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

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

```

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

```

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

`\iftypog@microtype@preloaded`

```

34 \newif\iftypog@microtype@preloaded
35

```

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

```

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

```

`\iftypog@microtype@loaded`

```

48 \newif\iftypog@microtype@loaded
49

```

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

```

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

```



```

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

```

Our own state ...

`\typog@config@mathitaliccorrection`

```

64 \newmuskip\typog@config@mathitaliccorrection
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@adjust@lowercase@labelitemi` (*dimen*)

```

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

```

`\typog@adjust@lowercase@labelitemii` (*dimen*)

```

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

```

<code>lowercase@labelitemiii</code> (<i>dimen</i>)	74 <code>\newdimen{\typog@adjust@lowercase@labelitemiii}</code>
<code>lowercase@labelitemiv</code> (<i>dimen</i>)	75 <code>\newdimen{\typog@adjust@lowercase@labelitemiv}</code>
<code>uppercase@labelitemi</code> (<i>dimen</i>)	76 <code>\newdimen{\typog@adjust@uppercase@labelitemi}</code>
<code>uppercase@labelitemii</code> (<i>dimen</i>)	77 <code>\newdimen{\typog@adjust@uppercase@labelitemii}</code>
<code>uppercase@labelitemiii</code> (<i>dimen</i>)	78 <code>\newdimen{\typog@adjust@uppercase@labelitemiii}</code>
<code>uppercase@labelitemiv</code> (<i>dimen</i>)	79 <code>\newdimen{\typog@adjust@uppercase@labelitemiv}</code>
	80
	Other lengths ...
<code>config@textitaliccorrection</code>	81 <code>\newlength{\typog@config@textitaliccorrection}</code>
<code>\typog@config@ligaturekern</code>	82 <code>\newlength{\typog@config@ligaturekern}</code>
<code>\typog@config@lowerslash</code>	83 <code>\newlength{\typog@config@lowerslash}</code>
<code>og@config@raisecapitaldash</code>	84 <code>\newlength{\typog@config@raisecapitaldash}</code>
<code>fig@raisecapitalguillemets</code>	85 <code>\newlength{\typog@config@raisecapitalguillemets}</code>
<code>@config@raisecapitalhyphen</code>	86 <code>\newlength{\typog@config@raisecapitalhyphen}</code>
<code>g@config@raisecapitaltimes</code>	87 <code>\newlength{\typog@config@raisecapitaltimes}</code>
<code>pog@config@raiseguillemets</code>	88 <code>\newlength{\typog@config@raiseguillemets}</code>
<code>pog@config@raisefiguredash</code>	89 <code>\newlength{\typog@config@raisefiguredash}</code>

```

\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@@typoginspect@tracingboxes{#1}}
173 \NewDocumentEnvironment{typoginspect}{0}{ m}
174   {\def\typog@@typoginspect@tracingboxes{\m@ne}%
175     \setkeys[typog]{typoginspect}{#1}%

```

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

```

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}
233   {\typog@config@ligaturekern}
234   {#2\typog@dim@unit}%
235   \IfBooleanTF{#1}%
236   {\kern\dimen0\ignorespaces}%
237   {\discretionary{\typog@hyphen}{\kern\dimen0}%
238    \typog@allowhyphenation
239    \IfNoValueF{#2}{\ignorespaces}}}
240

```

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

```

241 \typog@register@pdfsubstitute{
242   \RenewExpandableDocumentCommand{\nolig}{s o m}{%
243     \ifx\typog@TYPOG#3\typog@TYPOG
244       \relax
245     \else
246       \ifx\relax#3\relax
247         \relax
248       \else
249         \typog@missing@argument
250       \fi
251     \fi}
252 }
253

```

C.5 Manual Italic Correction

`\itcorr@text@unconditional` Fallback italic correction for text mode.

```

254 \newcommand*{\typog@itcorr@text@unconditional}[1]
255   {\kern#1\typog@config@textitaliccorrection}

```

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

```

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

```

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

```

264 \newcommand*{\typog@itcorr@math}[1]
265   {\mkern#1\typog@config@mathitaliccorrection}

```

`\itcorr` If the font has no italic 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.

```

266 \NewDocumentCommand{\itcorr}{s m}
267   {\ifmmode

```



```

268     \typog@itcorr@math{#2}%
269     \else
270     \IfBooleanTF{#1}%
271       {\typog@itcorr@text{#2}}%
272       {\typog@itcorr@text@unconditional{#2}}%
273     \fi}

```

PDF-substitute definition

```

274 \typog@register@pdfsubstitute{
275   \RenewExpandableDocumentCommand{\itcorr}{s m}{}
276 }
277

```

C.6 Apply Extra Kerning

Slash

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

```

278 \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`.

```

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

```

PDF-substitute definition

```

287 \typog@register@pdfsubstitute{
288   \def\kernedslash#1{\if* \detokenize{#1}/\ignorespaces\else/#1\fi}%
289 }
290

```

Hyphen

`\kernedhyphen`

```

291 \NewDocumentCommand{\kernedhyphen}{s O{0} m m}
292   {\ifmmode
293     \mspace{\muexpr{#3 mu} * 18 / 1000}%
294     \raisebox{#2\typog@dim@unit}{\mathord{-}}%
295     \mspace{\muexpr{#4 mu} * 18 / 1000}%
296   \else
297     \def\typog@@auto{*}%
298     \def\typog@@optarg{#2}%
299     \hspace*{#3\typog@dim@unit}%
300     \raisebox{\ifx\typog@@optarg\typog@@auto

```

```

301             \typog@config@raisecapitalhyphen
302             \else
303             \typog@optarg\typog@dim@unit
304             \fi}{\typog@hyphen}%
305 \hspace{#4\typog@dim@unit}%
306 \IfBooleanT{#1}{\nobreak}%
307 \fi}

```

PDF-substitute definition

```

308 \typog@register@pdfsubstitute{
309 \RenewExpandableDocumentCommand{\kernedhyphen}{s o m}{-}
310 }

```

One-argument shorthands.

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

```

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

```

PDF-substitute definition

```

315 \typog@register@pdfsubstitute{
316 \RenewExpandableDocumentCommand{\leftkernedhyphen}{s o m}{-}
317 }
318

```

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

```

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

```

PDF-substitute definition

```

323 \typog@register@pdfsubstitute{
324 \RenewExpandableDocumentCommand{\rightkernedhyphen}{s o m}{-}
325 }
326

```

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.

```

327 \newcommand*{\typog@wrap@endash}[3]
328  {#1\hspace{\typog@config@endashspace}%
329   #3%
330   #2\hspace{\typog@config@endashspace}}

```

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

```

331 \newcommand*{\typog@wrap@emdash}[3]

```

```

332  {#1\hspace{\typog@config@emdashspace}%
333  #3%
334  #2\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}
340     {\textendash}}
341    {\typog@wrap@endash{\nobreak}{\nobreak}
342     {\raisebox{#2}{\textendash}}}}
343   {\IfNoValueTF{#2}
344    {\typog@wrap@endash{\relax}{\nobreak}
345     {\textendash}}
346    {\typog@wrap@endash{\nobreak}{\nobreak}
347     {\raisebox{#2}{\textendash}}}}}%
348  \ignorespaces}
349 \let\leftspaceddash=\leftspacedendash

```

PDF-substitute definition

```

350 \typog@register@pdfsubstitute{
351  \RenewExpandableDocumentCommand{\leftspacedendash}{s o m}{%
352  \ifx\typog@TYPOG#3\typog@TYPOG
353  \textendash
354  \else
355  \ifx\relax#3\relax
356  \textendash
357  \else
358  \typog@missing@argument
359  \fi
360  \fi}
361  \let\leftspaceddash=\leftspacedendash
362 }
363

```

`\rightspacedendash` User-land macro for the right (aka closing) spaced en-dash. The unstarred variant introduces a breakpoint *after* the en-dash.

```

364 \NewDocumentCommand{\rightspacedendash}{s o}
365  {\IfBooleanTF{#1}
366   {\IfNoValueTF{#2}
367    {\typog@wrap@endash{\nobreak}{\nobreak}
368     {\textendash}}
369    {\typog@wrap@endash{\nobreak}{\nobreak}
370     {\raisebox{#2}{\textendash}}}}
371   {\IfNoValueTF{#2}
372    {\typog@wrap@endash{\nobreak}{\relax}
373     {\textendash}}
374    {\typog@wrap@endash{\nobreak}{\nobreak}

```

```

375             {\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

```

\swapedashskip

```

397 \NewDocumentCommand{\swapedashskip}{m}
398   {\skip0=\lastskip
399   \unskip
400   #1%
401   \hskip\skip0}
402

```

\leftspacedendash User-land macro for the spaced em-dash.

The two `\mboxes` turn off automatic hyphenation of the adjacent words. This ensures ‘compatibility’ with the en-dash `–`.

```

403 \NewDocumentCommand{\leftspacedemdash}{s o}
404   {\mbox{}%
405   \IfBooleanTF{#1}
406     {\IfNoValueTF{#2}
407       {\typog@wrap@emdash{\nobreak}{\nobreak}
408         {\textemdash}}
409       {\typog@wrap@emdash{\nobreak}{\nobreak}
410         {\raisebox{#2}{\textemdash}}}}
411     {\IfNoValueTF{#2}
412       {\typog@wrap@emdash{\relax}{\nobreak}
413         {\textemdash}}
414       {\typog@wrap@emdash{\nobreak}{\nobreak}
415         {\raisebox{#2}{\textemdash}}}}}%
416   \mbox{}%
417   \ignorespaces}

```

PDF-substitute definition

```

418 \typog@register@pdfsubstitute{
419 \RenewExpandableDocumentCommand{\leftspacedemdash}{s o m}{%
420   \ifx\typog@TYPOG#3\typog@TYPOG
421     \textemdash
422   \else
423     \ifx\relax#3\relax
424       \textemdash
425     \else
426       \typog@missing@argument
427     \fi
428   \fi}
429 }
430

```

`\rightspacedemdash` User-land macro for the right (aka closing) spaced em-dash. The unstarred variant introduces a breakpoint *after* the em-dash.

The two `\mboxes` turn off automatic hyphenation of the adjacent words. This ensures ‘compatibility’ with the em-dash `--`.

```

431 \NewDocumentCommand{\rightspacedemdash}{s o}
432   {\mbox}{%
433   \IfBooleanTF{#1}
434     {\IfNoValueTF{#2}
435       {\typog@wrap@emdash{\nobreak}{\nobreak}
436         {\textemdash}}
437       {\typog@wrap@emdash{\nobreak}{\nobreak}
438         {\raisebox{#2}{\textemdash}}}}
439     {\IfNoValueTF{#2}
440       {\typog@wrap@emdash{\nobreak}{\relax}
441         {\textemdash}}
442       {\typog@wrap@emdash{\nobreak}{\nobreak}
443         {\raisebox{#2}{\textemdash}}}}}%
444   \mbox}{%
445   \ignorespaces}
446 \let\spacedemdash=\rightspacedemdash

```

PDF-substitute definition

```

447 \typog@register@pdfsubstitute{
448 \RenewExpandableDocumentCommand{\rightspacedemdash}{s o m}{%
449   \ifx\typog@TYPOG#3\typog@TYPOG
450     \textemdash
451   \else
452     \ifx\relax#3\relax
453       \textemdash
454     \else
455       \typog@missing@argument
456     \fi
457   \fi
458   \ignorespaces}
459 \let\spacedemdash=\rightspacedemdash
460 }
461

```

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.

```
462 \newcommand*{\typog@breakpoint}
463   {\penalty\typog@config@breakpenalty}
```

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

```
464 \NewDocumentCommand{\capitalhyphen}{s}
465   {\raisebox{\typog@config@raisecapitalhyphen}{\typog@hyphen}%
466     \IfBooleanTF{#1}%
467       {\ignorespaces}%
468       {\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.

```
469 \typog@register@pdfsubstitute{
470   \def\capitalhyphen#1{%
471     \if*\detokenize{#1}%
472       -\ignorespaces
473     \else
474       -#1%
475     \fi}
476 }
477
```

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

```
478 \NewDocumentCommand{\capitalendash}{s}
479   {\raisebox{\typog@config@raisecapitaldash}{\textendash}%
480     \IfBooleanTF{#1}%
481       {\ignorespaces}%
482       {\typog@breakpoint\typog@allowhyphenation}}
483 \let\capitaldash=\capitalendash
```

PDF-substitute definition

```
484 \typog@register@pdfsubstitute{
485   \def\capitalendash#1{%
486     \if*\detokenize{#1}%
487       \textendash\ignorespaces
488     \else
489       \textendash#1%
490     \fi}
491 \let\capitaldash=\capitalendash
492 }
493
```

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

```
494 \NewDocumentCommand{\capitalemdash}{s}
495   {\raisebox{\typog@config@raisecapitaldash}{\textemdash}}%
496   \IfBooleanTF{#1}%
497     {\ignorespaces}%
498     {\typog@breakpoint\typog@allowhyphenation}}
```

PDF-substitute definition

```
499 \typog@register@pdfsubstitute{
500   \def\capitalemdash#1{%
501     \if*\detokenize{#1}%
502       \textemdash\ignorespaces
503     \else
504       \textemdash#1%
505     \fi}
506 }
507
```

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

```
508 \NewDocumentCommand{\leftspacedcapitalendash}{s}
509   {\IfBooleanTF{#1}%
510     {\typog@wrap@endash{\nobreak}{\nobreak}{\capitalendash*}}
511     {\typog@wrap@endash{\relax}{\nobreak}{\capitalendash}}}%
512   \ignorespaces}
513 \let\leftspacedcapitaldash=\leftspacedcapitalendash
```

PDF-substitute definition

```
514 \typog@register@pdfsubstitute{
515   \def\leftspacedcapitalendash#1{%
516     \if*\detokenize{#1}%
517       \textendash\ignorespaces
518     \else
519       \textendash#1%
520     \fi}
521   \let\leftspacedcapitaldash=\leftspacedcapitalendash
522 }
523
```

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

```
524 \NewDocumentCommand{\rightspacedcapitalendash}{s}
525   {\IfBooleanTF{#1}%
526     {\typog@wrap@endash{\nobreak}{\nobreak}{\capitalendash*}}
527     {\typog@wrap@endash{\nobreak}{\relax}{\capitalendash}}}%
528   \ignorespaces}
529 \let\rightspacedcapitaldash=\rightspacedcapitalendash
530 \let\spacedcapitalendash=\rightspacedcapitalendash
531 \let\spacedcapitaldash=\rightspacedcapitalendash
```

PDF-substitute definition

```

532 \typog@register@pdfsubstitute{
533   \def\rightrightdash#1{%
534     \if*\detokenize{#1}%
535       \textendash\ignorespaces
536     \else
537       \textendash#1%
538     \fi}
539 \let\rightrightdash=\rightrightdash
540 \let\spacedrightdash=\rightrightdash
541 \let\spacedrightdash=\rightrightdash
542 }
543

```

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

```

544 \NewDocumentCommand{\leftspacedcapitalemdash}{s}
545   {\IfBooleanTF{#1}%
546     {\typog@wrap@emdash{\nobreak}{\nobreak}{\capitalemdash*}}
547     {\typog@wrap@emdash{\relax}{\nobreak}{\capitalemdash}}}%
548   \ignorespaces}

```

PDF-substitute definition

```

549 \typog@register@pdfsubstitute{
550   \def\leftspacedcapitalemdash#1{%
551     \if*\detokenize{#1}%
552       \textemdash\ignorespaces
553     \else
554       \textemdash#1%
555     \fi}
556 }
557

```

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

```

558 \NewDocumentCommand{\rightspacedcapitalemdash}{s}
559   {\IfBooleanTF{#1}%
560     {\typog@wrap@emdash{\nobreak}{\nobreak}{\capitalemdash*}}
561     {\typog@wrap@emdash{\nobreak}{\relax}{\capitalemdash}}}%
562   \ignorespaces}
563 \let\spacedcapitalemdash=\rightspacedcapitalemdash

```

PDF-substitute definition

```

564 \typog@register@pdfsubstitute{
565   \def\rightrightdash#1{%
566     \if*\detokenize{#1}%
567       \textemdash\ignorespaces
568     \else
569       \textemdash#1%
570     \fi}
571 \let\spacedrightdash=\rightspacedcapitalemdash
572 }

```


573

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

```
574 \NewDocumentCommand{\figuredash}{s}
575   {\raisebox{\typog@config@raisefiguredash}{\textendash}%
576    \IfBooleanTF{#1}%
577     {\ignorespaces}%
578     {\typog@breakpoint\typog@allowhyphenation}}
```

PDF-substitute definition

```
579 \typog@register@pdfsubstitute{\let\figuredash=\capitaldash}
580
```

`\capitaltimes`

```
581 \NewDocumentCommand{\capitaltimes}{}
582   {\ifmmode
583     \mathbin{\raisebox{\typog@config@raisecapitaltimes}
584              {$\m@th\times$}}%
585     \else
586       \raisebox{\typog@config@raisecapitaltimes}{\texttimes}%
587     \fi}
```

PDF-substitute definition

```
588 \typog@register@pdfsubstitute{
589   \RenewExpandableDocumentCommand{\capitaltimes}{}{\texttimes}
590 }
591
```

`\singleguillemetleft`

```
592 \NewDocumentCommand{\singleguillemetleft}{}
593   {\typog@allowhyphenation
594    \raisebox{\typog@config@raiseguillemets}{\guilsinglleft}}
```

PDF-substitute definition

```
595 \typog@register@pdfsubstitute{
596   \let\singleguillemetleft\guilsinglleft
597 }
```

`\singleguillemetright`

```
598 \NewDocumentCommand{\singleguillemetright}{}
599   {\raisebox{\typog@config@raiseguillemets}{\guilsinglright}%
600    \typog@allowhyphenation}
```

PDF-substitute definition

```
601 \typog@register@pdfsubstitute{
602   \let\singleguillemetright\guilsinglright
603 }
```

`\doubleguillemetleft`

```
604 \NewDocumentCommand{\doubleguillemetleft}{}
605   {\typog@allowhyphenation
606    \raisebox{\typog@config@raiseguillemets}{\guillemotleft}}
```

PDF-substitute definition

```
607 \typog@register@pdfsubstitute{
608   \let\doubleguillemetleft\guillemotleft
609 }
```

\doubleguillemetright

```
610 \NewDocumentCommand{\doubleguillemetright}{}
611   {\raisebox{\typog@config@raiseguillemets}{\guillemotright}%
612    \typog@allowhyphenation}
```

PDF-substitute definition

```
613 \typog@register@pdfsubstitute{
614   \let\doubleguillemetright\guillemotright
615 }
```

\Singleguillemetleft

```
616 \NewDocumentCommand{\Singleguillemetleft}{}
617   {\typog@allowhyphenation
618    \raisebox{\typog@config@raisecapitalguillemets}
619             {\guilsinglleft}}
```

PDF-substitute definition

```
620 \typog@register@pdfsubstitute{
621   \let\Singleguillemetleft\guilsinglleft
622 }
```

\Singleguillemetright

```
623 \NewDocumentCommand{\Singleguillemetright}{}
624   {\raisebox{\typog@config@raisecapitalguillemets}
625             {\guilsinglright}%
626    \typog@allowhyphenation}
```

PDF-substitute definition

```
627 \typog@register@pdfsubstitute{
628   \let\Singleguillemetright\guilsinglright
629 }
```

\Doubleguillemetleft

```
630 \NewDocumentCommand{\Doubleguillemetleft}{}
631   {\typog@allowhyphenation
632    \raisebox{\typog@config@raisecapitalguillemets}
633             {\guillemotleft}}
```

PDF-substitute definition

```
634 \typog@register@pdfsubstitute{
635   \let\Doubleguillemetleft\guillemotleft
636 }
```

\Doubleguillemetright

```
637 \NewDocumentCommand{\Doubleguillemetright}{}
638   {\raisebox{\typog@config@raisecapitalguillemets}
639     {\guillemotright}%
640    \typog@allowhyphenation}
```

PDF-substitute definition

```

641 \typog@register@pdfsubstitute{
642   \let\Doubleguillemetright\guillemotright
643 }
644

```

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

```

\config@raiseinvertedmarks@i

```

```

645 \newlength{\typog@config@raiseinvertedmarks@i}

```

```

\config@raiseinvertedmarks@ii

```

```

646 \newlength{\typog@config@raiseinvertedmarks@ii}

```

```

\config@raiseinvertedmarks@iii

```

```

647 \newlength{\typog@config@raiseinvertedmarks@iii}

```

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

```

\inverted@exclamationmark@i

```

```

648 \newcommand*{\typog@inverted@exclamationmark@i}
649   {\textexclamdown}

```

```

\typog@inverted@questionmark@i

```

```

650 \newcommand*{\typog@inverted@questionmark@i}
651   {\textquestiondown}

```

```

\inverted@exclamationmark@ii

```

```

652 \newcommand*{\typog@inverted@exclamationmark@ii}
653   {\textexclamdown}

```

```

\typog@inverted@questionmark@ii

```

```

654 \newcommand*{\typog@inverted@questionmark@ii}
655   {\textquestiondown}

```

```

\inverted@exclamationmark@iii

```

```

656 \newcommand*{\typog@inverted@exclamationmark@iii}
657   {\textexclamdown}

```

```

\typog@inverted@questionmark@iii

```

```

658 \newcommand*{\typog@inverted@questionmark@iii}
659   {\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.

```

660 \newcommand*{\typog@raise@inverted@mark}[2]

```

```

661 {\letcs{\@tempa}{\typog@inverted@#1@\romannumeral#2}%
662 \letcs{\@tempb}{\typog@config@raise@inverted@marks@\romannumeral#2}%
663 \ifdim\@tempb=\z@
664   \setbox0=\hbox{\@tempa}%
665   \dimen0=\dp0
666 \else
667   \dimen0=\@tempb
668 \fi
669 \raisebox{\dimen0}{\@tempa}}

```

The user-side macros are almost trivial now.

capitalinvertedexclamationmark

```

670 \NewDocumentCommand{\capitalinvertedexclamationmark}{m}
671 {\typog@raise@inverted@mark{exclamationmark}{#1}%
672 \typog@allowhyphenation}

```

PDF-substitute definition

```

673 \typog@register@pdfsubstitute{
674 \def\capitalinvertedexclamationmark#1{%
675 \csname typog@inverted@exclamationmark@\romannumeral#1\endcsname
676 }
677 }

```

capitalinvertedquestionmark

```

678 \NewDocumentCommand{\capitalinvertedquestionmark}{m}
679 {\typog@raise@inverted@mark{questionmark}{#1}%
680 \typog@allowhyphenation}

```

PDF-substitute definition

```

681 \typog@register@pdfsubstitute{
682 \def\capitalinvertedquestionmark#1{%
683 \csname typog@inverted@questionmark@\romannumeral#1\endcsname
684 }
685 }

```

C.8 Vertically Adjusted Label Items

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

```

686 \newcommand*{\@typog@uppercase@adjust@labelitem}[1]
687 {\@typog@maybe@patch@itemize
688 \ifstrequal{#1}{*}
689   {\setlength{\typog@adjust@labelitemi}
690     {\typog@adjust@uppercase@labelitemi}%
691   \setlength{\typog@adjust@labelitemii}
692     {\typog@adjust@uppercase@labelitemii}%
693   \setlength{\typog@adjust@labelitemiii}
694     {\typog@adjust@uppercase@labelitemiii}%
695   \setlength{\typog@adjust@labelitemiv}
696     {\typog@adjust@uppercase@labelitemiv}}
697 {\ifcase #1% 0

```

```

698         \relax % outside of any itemize environment
699     \or % 1
700         \setlength{\typog@adjust@labelitemi}
701             {\typog@adjust@uppercase@labelitemi}%
702     \or % 2
703         \setlength{\typog@adjust@labelitemii}
704             {\typog@adjust@uppercase@labelitemii}%
705     \or % 3
706         \setlength{\typog@adjust@labelitemiii}
707             {\typog@adjust@uppercase@labelitemiii}%
708     \or % 4
709         \setlength{\typog@adjust@labelitemiv}
710             {\typog@adjust@uppercase@labelitemiv}%
711     \else
712         \PackageError{typog}
713             {Itemize level out of range}
714             {Valid levels are 1, 2, 3, 4, and *}
715     \fi}}
716

```

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

```

717 \newcommand*{\@typog@lowercase@adjust@labelitem}[1]
718   {\@typog@maybe@patch@itemize
719     \ifstrequal{#1}{*}
720       {\setlength{\typog@adjust@labelitemi}
721         {\typog@adjust@lowercase@labelitemi}%
722       \setlength{\typog@adjust@labelitemii}
723         {\typog@adjust@lowercase@labelitemii}%
724       \setlength{\typog@adjust@labelitemiii}
725         {\typog@adjust@lowercase@labelitemiii}%
726       \setlength{\typog@adjust@labelitemiv}
727         {\typog@adjust@lowercase@labelitemiv}}
728     {\ifcase #1% 0
729       \relax % outside of any itemize environment
730     \or % 1
731       \setlength{\typog@adjust@labelitemi}
732         {\typog@adjust@lowercase@labelitemi}%
733     \or % 2
734       \setlength{\typog@adjust@labelitemii}
735         {\typog@adjust@lowercase@labelitemii}%
736     \or % 3
737       \setlength{\typog@adjust@labelitemiii}
738         {\typog@adjust@lowercase@labelitemiii}%
739     \or % 4
740       \setlength{\typog@adjust@labelitemiv}
741         {\typog@adjust@lowercase@labelitemiv}%
742     \else
743       \PackageError{typog}
744         {Itemize level out of range}
745         {Valid levels are 1, 2, 3, 4, and *}
746     \fi}}

```

747

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

```

748 \newcommand*{\@typog@noadjust@labelitem}[1]
749   {\ifstrequal{#1}{*}
750     {\setlength{\typog@adjust@labelitemi}{\z@}%
751      \setlength{\typog@adjust@labelitemii}{\z@}%
752      \setlength{\typog@adjust@labelitemiii}{\z@}%
753      \setlength{\typog@adjust@labelitemiv}{\z@}}
754     {\ifcase #1% 0
755       \relax % outside of any itemize environment
756       \or % 1
757         \setlength{\typog@adjust@labelitemi}{\z@}%
758       \or % 2
759         \setlength{\typog@adjust@labelitemii}{\z@}%
760       \or % 3
761         \setlength{\typog@adjust@labelitemiii}{\z@}%
762       \or % 4
763         \setlength{\typog@adjust@labelitemiv}{\z@}%
764       \else
765         \PackageError{typog}
766           {Itemize level out of range}
767           {Valid levels are 1, 2, 3, 4, and *}
768       \fi}}
769

```

`\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`.

```

770 \NewDocumentCommand{\uppercaseadjustlabelitems}{m}
771   {\AfterPreamble{%
772     \ifblank{#1}
773       {\@typog@uppercase@adjust@labelitem{\@itemdepth}}
774       {\forcsvlist{\@typog@uppercase@adjust@labelitem}{#1}}%
775     \ignorespaces}}
776

```

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

```

777 \NewDocumentCommand{\lowercaseadjustlabelitems}{m}
778   {\AfterPreamble{%
779     \ifblank{#1}
780       {\@typog@lowercase@adjust@labelitem{\@itemdepth}}
781       {\forcsvlist{\@typog@lowercase@adjust@labelitem}{#1}}%
782     \ignorespaces}}
783

```

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

```

784 \NewDocumentCommand{\noadjustlabelitems}{m}
785   {\ifblank{#1}

```

```

786         {\@typog@noadjust@labelitem{\@itemdepth}}
787         {\forcsvlist{\@typog@noadjust@labelitem}{#1}}%
788     \ignorespaces}
789

```

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`.

```

790 \newcommand*{\@typog@itemize@patch}

```

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

```

791 {\letcs{\@typog@old@itemitem}{\@itemitem}

```

Sneak in our own macro's name.

```

792 \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.

```

793 \expandafter\def\csname\@itemitem\endcsname
794     {\raisebox{\csname typog@adjust@labelitem\romannumeral\the\@itemdepth\endcsname}
795      {\@typog@old@itemitem}}
796

```

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.

```

797 \newcommand*\@typog@patch@itemize
798   {\ifdefined\enit@itemize@i
799     \patchcmd{\enit@itemize@i
800               {\expandafter}
801               {\@typog@itemize@patch\expandafter}
802               {\typog@debug@typeout{patching enumitem \string\enit@itemize@i\space
803                                     ceeded}}}
804               {\PackageError{typog}
805               {Patching enumitem macro \string\enit@itemize@i\space
806               {}}}%
806   \else
807     \patchcmd{\itemize}
808               {\expandafter}
809               {\@typog@itemize@patch\expandafter}
810               {\typog@debug@typeout{patching \string\itemize\space suc-
811                                     ceeded}}}
812               {\PackageError{typog}
813               {Patching plain LaTeX macro \string\itemize\space fai
814               {}}}%
814   \fi}
815

```

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

```

816 \newbool{\@typog@itemize@has@been@patched}
817 \newcommand*\@typog@maybe@patch@itemize
818   {\ifbool{\@typog@itemize@has@been@patched}
819     {\relax}
820     {\@typog@patch@itemize
821     \booltrue{\@typog@itemize@has@been@patched}}}
822

```

Freestanding height-adjusted label items.

`\AdjustedLabelItem`

```

823 \NewDocumentCommand{\AdjustedLabelItem}{}

```



```
824 {\raisebox{\typog@adjust@uppercase@labelitemi}
825         {\labelitemi}}
```

`\adjustedlabelitemi`

```
826 \NewDocumentCommand{\adjustedlabelitemi}{}
827   {\raisebox{\typog@adjust@lowercase@labelitemi}
828         {\labelitemi}}
```

`\Adjustedlabelitemii`

```
829 \NewDocumentCommand{\Adjustedlabelitemii}{}
830   {\raisebox{\typog@adjust@uppercase@labelitemii}
831         {\labelitemii}}
```

`\adjustedlabelitemii`

```
832 \NewDocumentCommand{\adjustedlabelitemii}{}
833   {\raisebox{\typog@adjust@lowercase@labelitemii}
834         {\labelitemii}}
```

`\Adjustedlabelitemiii`

```
835 \NewDocumentCommand{\Adjustedlabelitemiii}{}
836   {\raisebox{\typog@adjust@uppercase@labelitemiii}
837         {\labelitemiii}}
```

`\adjustedlabelitemiii`

```
838 \NewDocumentCommand{\adjustedlabelitemiii}{}
839   {\raisebox{\typog@adjust@lowercase@labelitemiii}
840         {\labelitemiii}}
```

`\Adjustedlabelitemiv`

```
841 \NewDocumentCommand{\Adjustedlabelitemiv}{}
842   {\raisebox{\typog@adjust@uppercase@labelitemiv}
843         {\labelitemiv}}
```

`\adjustedlabelitemiv`

```
844 \NewDocumentCommand{\adjustedlabelitemiv}{}
845   {\raisebox{\typog@adjust@lowercase@labelitemiv}
846         {\labelitemiv}}
847
```

And now for their PDFsubstitutes.

```
848 \typog@register@pdfsubstitute{
849   \def\Adjustedlabelitemi{\labelitemi}
850   \def\adjustedlabelitemi{\labelitemi}
851   \def\Adjustedlabelitemii{\labelitemii}
852   \def\adjustedlabelitemii{\labelitemii}
853   \def\Adjustedlabelitemiii{\labelitemiii}
854   \def\adjustedlabelitemiii{\labelitemiii}
855   \def\Adjustedlabelitemiv{\labelitemiv}
856   \def\adjustedlabelitemiv{\labelitemiv}
857 }
858
```

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.

```
859 \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.

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

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

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

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

```
862 \count0=#3\relax
```

```
863 \unless\ifodd\count0
```

```
864 \advance\count0 by 1%
```

```
865 \fi
```

Set the iteration counter.

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

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

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

Box 1 is the accumulator for the raised samples.

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

Build the stairs.

```
869 \loop
```

```
870 \ifnum\thetypog@@iteration=\numexpr\count0 / 2\relax
```

```
871 \dimen1=3\dimen0
```

```
872 \else
```

```
873 \dimen1=\dimen0
```

```
874 \fi
```

```
875 \dimen2=\dimexpr#2 * (\thetypog@@iteration - \count0 / 2)\re-
```

```
lax
```

```
876 \setbox1=\hbox{\unhbox1\raisebox{\dimen2}{\kern\dimen1 #5\kern\dimen1}}%
```

```
877 \addtocounter{typog@@iteration}{1}%
```

```
878 \unless\ifnum\thetypog@@iteration>\count0
```

```
879 \repeat
```

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

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

```
881
```

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

```
882 \NewDocumentCommand{\typogadjuststairs}{0{.5} m m m}
```

```

883 {\begingroup
884   \unless\ifdim #2>\z@
885     \PackageError{typog}
886       {\string\typogadjuststairs\space non-positive step-
      size}
887       {step-size must be a positive dimension}%
888   \fi
889   \ifnum #3<1
890     \PackageError{typog}
891       {\string\typogadjuststairs\space too few number-
      of-steps}
892       {number-of-steps must at least be 1}%
893   \fi
894   \ifblank{#4}
895     {\PackageError{typog}
896       {sample must not be empty}
897       {supply either some uppercase or some low-
      ercase letters}}
898     {}%
899   \def\arraystretch{1}%
900   \begin{tabular}{@{}c@{}}
901     \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemi} \\\
902     \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemii} \\\
903     \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemiii} \\\
904     \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemiv}
905   \end{tabular}
906 \endgroup}
907

```

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

```

908 \newcommand*{\typoguppercaseadjusted@labelitems}
909   {\hbox{\raisebox{\typogadjust@uppercase@labelitemi}{\labelitemi}%
910         \raisebox{\typogadjust@uppercase@labelitemii}{\labelitemii}%
911         \raisebox{\typogadjust@uppercase@labelitemiii}{\labelitemiii}%
912         \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.

```

913 \NewDocumentCommand{\typoguppercaseadjustcheck}{0{.5} m}
914   {\setbox0=\hbox{#2}%
915    \setbox1=\typoguppercaseadjusted@labelitems
916    \mbox{\rlap{\raisebox{\fpeval{#1}\ht0}
917              {\rule{\wd1}{\typog@hairline@width}}}%
918          \box1}}
919

```

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

```

920 \newcommand*{\typoglowercaseadjusted@labelitems}

```

```

921 {\hbox{\raisebox{\typog@adjust@lowercase@labelitemi}{\labelitemi}%
922         \raisebox{\typog@adjust@lowercase@labelitemii}{\labelitemii}%
923         \raisebox{\typog@adjust@lowercase@labelitemiii}{\labelitemiii}%
924         \raisebox{\typog@adjust@lowercase@labelitemiv}{\labelitemiv}}}

```

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

```

925 \NewDocumentCommand{\typoglowercaseadjustcheck}{0{.5} m}
926   {\setbox0=\hbox{#2}%
927    \setbox1=\typog@lowercase@adjusted@labelitems
928    \mbox{\rlap{\raisebox{\fpeval{#1}\ht0}
929              {\rule{\wd1}{\typog@hairline@width}}}}%
930          \box1}}
931

```

C.9 Align Last Line of a Paragraph

The code of environment `lastlinerraggedleftpar` has been inspired by macro `\lastlinerraggedleft` [39, Sec. 2].

`lastlinerraggedleftpar` (*env.*)

```

932 \NewDocumentEnvironment{lastlinerraggedleftpar}{}
933   {\lastlinefit=0%
934    \setlength{\leftskip}{\z@ \@plus 1fil}%
935    \setlength{\rightskip}{-\leftskip}%
936    \setlength{\parfillskip}{\leftskip}}
937   {\par}

```

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

```

938 \let\lastlineflushrightpar=\lastlinerraggedleftpar
939 \let\endlastlineflushrightpar=\endlastlinerraggedleftpar
940

```

`lastlinecenteredpar` (*env.*) The code of environment `lastlinecenteredpar` has been inspired by *TeX By Topic* [12, Sec. 18.3.1].

```

941 \NewDocumentEnvironment{lastlinecenteredpar}{}
942   {\lastlinefit=0%
943    \setlength{\leftskip}{\z@ \@plus .5fil}%
944    \setlength{\rightskip}{-\leftskip}%
945    \setlength{\parfillskip}{\z@ \@plus 1fil}}
946   {\par}
947

```

C.10 Fill Last Line of a Paragraph

`shortenpar` (*env.*)

```

948 \NewDocumentEnvironment{shortenpar}{}
949   {\advance\looseness by -1
950    \ifnum\tracingparagraphs>0
951     \typeout{@ looseness \the\looseness}%
952    \fi}

```

```
953 {\par}
954
```

`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.

```
955 \NewDocumentEnvironment{prolongpar}{}
956 {\finalhyphendemerits=100000001
957 \advance\looseness by 1
958 \ifnum\tracingparagraphs>0
959 \typeout{@ looseness \the\looseness}%
960 \fi}
961 {\par}
962
```

`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`.

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

`ndentpar@nonzero@parindent`

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

`covernextindentpar` (*env.*)

```
965 \NewDocumentEnvironment{covernextindentpar}{o}
966 {\IfNoValueTF{#1}
967 {\ifdim\parindent=\z@
968 \dimen0=\dimexpr\linewidth - \typog@covernextindentpar@zero@parindent
969 \else
970 \dimen0=\dimexpr\linewidth - \typog@covernextindentpar@nonzero@parindent
971 \fi}
972 {\dimen0=\dimexpr\linewidth - (#1)}%
973 \parfillskip=\dimen0 \@minus \dimen0
974 \relax}
975 {\par}
976
```

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

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

`tlinepar@nonzero@parindent`

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

`openlastlinepar` (*env.*) Compare with the suggestion in reference [34](#).

```
979 \NewDocumentEnvironment{openlastlinepar}{o}
980 {\IfNoValueTF{#1}
981 {\ifdim\parindent=\z@
982 \skip0=\typog@openlastlinepar@zero@parindent
983 \@plus 1fil
984 \@minus \typog@openlastlinepar@zero@parindent
```

```

985     \else
986       \skip0=\typog@openlastlinepar@nonzero@parindent
987         \@plus 1fil
988         \@minus \typog@openlastlinepar@nonzero@parindent
989     \fi}
990   {\dimen0=\dimexpr#1\relax
991     \skip0=\dimen0 \@plus 1fil \@minus \dimen0}
992   \parfillskip=\skip0}
993 {\par}
994

```

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

```

995 \NewDocumentEnvironment{lastlinefitpar}{0{1000}}
996   {\lastlinefit=#1\relax}
997   {\par}
998

```

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.

```
999 \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.

```
1000 \newcommand*{\widespacescale}{1.125}
```

`\widespace`

```

1001 \NewDocumentCommand{\widespace}{s}
1002   {\IfBooleanTF{#1}%
1003     {\dimen0=\widespacescale\fontdimen2\font}%
1004     {\ifdim\fontdimen7\font=\z@
1005       \dimen0=\widespacescale\fontdimen2\font
1006       \else
1007         \dimen0=\dimexpr\fontdimen2\font +
1008           \widespacestrength\fontdimen7\font
1009       \fi}%
1010   \hskip \glueexpr\dimen0
1011     \@plus \widespacescale\fontdimen3\font
1012     \@minus \widespacescale\fontdimen4\font
1013   \ignorespaces}
1014

```

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

```
1015 \newcommand*{\narrowpacestrength}{.5}
```

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

```
1016 \newcommand*\narrowspacescale{.9375}
```

`\narrowspace`

```
1017 \NewDocumentCommand{\narrowspace}{s}
1018   {\IfBooleanTF{#1}%
1019     {\dimen0=\narrowspacescale\fontdimen2\font}%
1020     {\ifdim\fontdimen7\font=\z@
1021       \dimen0=\narrowspacescale\fontdimen2\font
1022       \else
1023         \dimen0=\dimexpr\fontdimen2\font -
1024           \narrowpacestrength\fontdimen7\font
1025       \fi}%
1026   \hskip \glueexpr\dimen0
1027     \@plus \narrowspacescale\fontdimen3\font
1028     \@minus \narrowspacescale\fontdimen4\font
1029   \ignorespaces}
1030
```

See also: TeX by Topic [12, ch. 20, p. 185–190].

`loosespacing` (*env.*)

```
1031 \NewDocumentEnvironment{loosespacing}{0{1}}
1032   {\dimen2=\fontdimen2\font
1033   \ifcase #1
1034     \spaceskip=\z@
1035     \or % 1      +5%
1036     \spaceskip=1.05\dimen2 \@plus .5\dimen2 \@minus .1\dimen2
1037     \or % 2      +10%
1038     \spaceskip=1.1\dimen2 \@plus .5\dimen2 \@minus .1\dimen2
1039     \or % 3      +20%
1040     \spaceskip=1.2\dimen2 \@plus .6\dimen2 \@minus .2\dimen2
1041     \else % >= 4  +30%
1042     \spaceskip=1.3\dimen2 \@plus .8\dimen2 \@minus .3\dimen2
1043     \fi
1044   \ignorespaces}
1045   {\ignorespacesafterend}
1046
```

`tightspacing` (*env.*)

```
1047 \NewDocumentEnvironment{tightspacing}{0{1}}
1048   {\dimen2=\fontdimen2\font
1049   \ifcase #1
1050     \spaceskip=\z@
1051     \or % 1      -1.25%
1052     \spaceskip=.9875\dimen2 \@plus .0125\dimen2 \@minus .5\dimen2
1053     \or % 2      -2.5%
1054     \spaceskip=.975\dimen2 \@plus .025\dimen2 \@minus .5\dimen2
1055     \or % 3      -5%
```

```

1056     \spaceskip=.95\dimen2 \@plus .05\dimen2 \@minus .5\dimen2
1057     \else % >= 4    -10%
1058     \spaceskip=.9\dimen2 \@plus .1\dimen2 \@minus .5\dimen2
1059     \fi
1060     \ignorespaces}
1061     {\ignorespacesafterend}
1062

```

C.12 Microtype Front-End

Tracking

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

```

1063 \NewDocumentEnvironment{setfonttracking}{m}
1064   {\edef\MT@letterspace@{#1}%
1065     \lsstyle
1066     \ignorespaces}
1067   {\ignorespacesafterend}
1068

```

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.

```

1069 \newcommand*{\typog@setup@font@expansion}
1070   {\SetExpansion
1071     [context = typog@shrink1,
1072     shrink = \typog@shrink@i,
1073     stretch = 0]%
1074     {encoding = {*}}%
1075     {}
1076   \SetExpansion
1077     [context = typog@shrink2,
1078     shrink = \typog@shrink@ii,
1079     stretch = 0]%
1080     {encoding = {*}}%
1081     {}
1082   \SetExpansion
1083     [context = typog@shrink3,
1084     shrink = \typog@shrink@iii,
1085     stretch = 0]%
1086     {encoding = {*}}%
1087     {}
1088
1089   \SetExpansion
1090     [context = typog@stretch1,
1091     shrink = 0,
1092     stretch = \typog@stretch@i]%
1093     {encoding = {*}}%
1094     {}

```



```

1095 \SetExpansion
1096   [context = typog@stretch2,
1097     shrink = 0,
1098     stretch = \typog@stretch@ii]%
1099   {encoding = {*}}%
1100   {}
1101 \SetExpansion
1102   [context = typog@stretch3,
1103     shrink = 0,
1104     stretch = \typog@stretch@iii]%
1105   {encoding = {*}}%
1106   {}
1107
1108 \SetExpansion
1109   [context = typog@expand1,
1110     shrink = \typog@shrink@i,
1111     stretch = \typog@stretch@i]%
1112   {encoding = {*}}%
1113   {}
1114 \SetExpansion
1115   [context = typog@expand2,
1116     shrink = \typog@shrink@ii,
1117     stretch = \typog@stretch@ii]%
1118   {encoding = {*}}%
1119   {}
1120 \SetExpansion
1121   [context = typog@expand3,
1122     shrink = \typog@shrink@iii,
1123     stretch = \typog@stretch@iii]%
1124   {encoding = {*}}%
1125   {}

```

`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.

```

1126 \newcommand*{\typog@test@microtype@expansion@feature}
1127   {\ifMT@expansion
1128     \typog@typeout{microtype preloaded -- font expansion features avail-
1129     able}%
1129     \def\typog@require@microtype@expansion{\relax}
1130     \typog@setup@font@expansion
1131     \else
1132     \PackageWarning{typog}{microtype preloaded,\space
1133     but font expansion is disabled}%
1134     \def\typog@require@microtype@expansion
1135     {\PackageError{typog}
1136     {microtype font expansion disabled}
1137     {pass option 'expansion' to package microtype}}
1138     \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 with the code for `\typog@require@microtype` and `\typog@require@preloaded@microtype`.

Initialize our own flag and setup meaningful messages for later feature checks.

```

1139 \iftypog@microtype@preloaded
1140 \typog@test@microtype@expansion@feature
1141 \else
1142 \def\typog@require@microtype@expansion
1143   {\PackageError{typog}%
1144     {package microtype not (pre-)loaded, %
1145     which is required for typog's font expansion}%
1146     {require package microtype before package typog}}
1147 \fi
1148

```

`setfontshrink` (*env.*)

```

1149 \NewDocumentEnvironment{setfontshrink}{0{1}}
1150 {\typog@require@microtype@expansion
1151   \ifcase#1% 0
1152     \relax
1153   \or % 1
1154     \microtypecontext{expansion=typog@shrink1}%
1155   \or % 2
1156     \microtypecontext{expansion=typog@shrink2}%
1157   \else % >= 3
1158     \microtypecontext{expansion=typog@shrink3}%
1159   \fi
1160   \ignorespaces}
1161 {\ignorespacesafterend}
1162

```

`setfontstretch` (*env.*)

```

1163 \NewDocumentEnvironment{setfontstretch}{0{1}}
1164 {\typog@require@microtype@expansion
1165   \ifcase#1% 0
1166     \relax
1167   \or % 1
1168     \microtypecontext{expansion=typog@stretch1}%
1169   \or % 2
1170     \microtypecontext{expansion=typog@stretch2}%
1171   \else % >= 3
1172     \microtypecontext{expansion=typog@stretch3}%
1173   \fi
1174   \ignorespaces}
1175 {\ignorespacesafterend}
1176

```

`setfontexpand` (*env.*)

```

1177 \NewDocumentEnvironment{setfontexpand}{0{1}}
1178 {\typog@require@microtype@expansion
1179   \ifcase#1% 0
1180     \relax

```

```

1181 \or % 1
1182   \microtypecontext{expansion=typog@expand1}%
1183 \or % 2
1184   \microtypecontext{expansion=typog@expand2}%
1185 \else % >= 3
1186   \microtypecontext{expansion=typog@expand3}%
1187 \fi
1188 \ignorespaces}
1189 {\ignorespacesafterend}
1190

```

`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.

```

1191 \NewDocumentEnvironment{nofontexpansion}{}
1192   {\ifdefined\microtypesetup
1193     \microtypesetup{expansion=false}%
1194   \fi
1195   \ignorespaces}
1196   {\ignorespacesafterend}

```

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

```

1197 \let\nofontexpand=\nofontexpansion
1198 \let\endnofontexpand=\endnofontexpansion
1199

```

Character Protrusion

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

```

1200 \NewDocumentEnvironment{nocharprotrusion}{}
1201   {\ifdefined\microtypesetup
1202     \microtypesetup{protrusion=false}%
1203   \fi
1204   \ignorespaces}
1205   {\ignorespacesafterend}
1206

```

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`.

```

1207 \newcommand*{\typog@scaled@emergencystretch}[1]
1208   {\emergencystretch=\ifdim\linewidth=\z@
1209     #1%
1210   \else
1211     \dimexpr (#1) * \linewidth / \textwidth
1212   \fi}
1213

```

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

```

1214 \NewDocumentCommand{\slightlyloppy}{0{1}}
1215   {\ifcase #1% 0
1216     % \tolerance=200
1217     % \emergencystretch=\z@
1218     % \hfuzz=.1\p@
1219     % \vfuzz=\hfuzz
1220     \fussy
1221   \or % 1
1222     \pretolerance=165%
1223     \tolerance=330%
1224     \typog@scaled@emergencystretch{.375em}%
1225     \hfuzz=.15\p@
1226     \vfuzz=\hfuzz
1227   \or % 2
1228     \pretolerance=265%
1229     \tolerance=530%
1230     \typog@scaled@emergencystretch{.75em}%
1231     \hfuzz=.15\p@
1232     \vfuzz=\hfuzz
1233   \or % 3
1234     \pretolerance=435%
1235     \tolerance=870%
1236     \typog@scaled@emergencystretch{1.125em}%
1237     \hfuzz=.2\p@
1238     \vfuzz=\hfuzz
1239   \or % 4
1240     \pretolerance=705%
1241     \tolerance=1410%
1242     \typog@scaled@emergencystretch{1.5em}%
1243     \hfuzz=.3\p@
1244     \vfuzz=\hfuzz
1245   \or % 5
1246     \pretolerance=1155%
1247     \tolerance=2310%
1248     \typog@scaled@emergencystretch{1.875em}%
1249     \hfuzz=.35\p@
1250     \vfuzz=\hfuzz
1251   \or % 6
1252     \pretolerance=1880%
1253     \tolerance=3760%
1254     \typog@scaled@emergencystretch{2.25em}%
1255     \hfuzz=.4\p@
1256     \vfuzz=\hfuzz
1257   \or % 7
1258     \pretolerance=3065%
1259     \tolerance=6130%
1260     \typog@scaled@emergencystretch{2.625em}%
1261     \hfuzz=.45\p@

```

```

1262     \vfuzz=\hfuzz
1263     \else % >= 8
1264         % \tolerance=9999
1265         % \emergencystretch=3em
1266         % \hfuzz=.5\p@
1267         % \vfuzz=\hfuzz
1268     \sloppy
1269     \fi
1270     \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.)`

```

1271 \NewDocumentEnvironment{slightlyloppypar}{0{1}}
1272   {\par\sloppypar[#1]\ignorespaces}
1273   {\par}
1274

```

C.14 Vertically Partially-Tied Paragraphs

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

```

1275 \ExplSyntaxOn
1276 \newcommand*{\typog@geometric@mean}[2]
1277     {\fp_to_int:n {sqrt((#1) * (#2))}}
1278 \ExplSyntaxOff
1279

```

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

```

1280 \newcounter{typog@mean@penalty}
1281

```

`\vtietop`

```

1282 \NewDocumentCommand{\vtietop}{0{3}}
1283   {\setcounter{typog@mean@penalty}
1284     {\typog@geometric@mean{\@M}{\clubpenalty}}%
1285   \typog@debug@typeout{vtietop: penalties \the\@M--\the\value{typog@mean@penalty}
-\the\clubpenalty}%
1286   \unless\ifnum\clubpenalty<\@M
1287     \PackageWarning{typog}{vtietop: clubpenalty=\the\clubpenalty\space>= 10000}%
1288   \fi
1289   \ifcase#1% 0
1290     \relax
1291   \or % 1
1292     \relax
1293   \or % 2
1294     \clubpenalties 3
1295       \@M
1296       \value{typog@mean@penalty}
1297       \clubpenalty
1298   \or % 3
1299     \clubpenalties 4
1300       \@M \@M
1301       \value{typog@mean@penalty}
1302       \clubpenalty
1303   \or % 4
1304     \clubpenalties 5
1305       \@M \@M \@M
1306       \value{typog@mean@penalty}
1307       \clubpenalty
1308   \or % 5
1309     \clubpenalties 6
1310       \@M \@M \@M \@M
1311       \value{typog@mean@penalty}
1312       \clubpenalty
1313   \or % 6
1314     \clubpenalties 7
1315       \@M \@M \@M \@M \@M
1316       \value{typog@mean@penalty}
1317       \clubpenalty
1318   \or % 7
1319     \clubpenalties 8
1320       \@M \@M \@M \@M \@M
1321       \value{typog@mean@penalty}
1322       \clubpenalty
1323   \or % 8
1324     \clubpenalties 9
1325       \@M \@M \@M \@M \@M
1326       \value{typog@mean@penalty}
1327       \clubpenalty
1328   \else % >= 9
1329     \clubpenalties 10
1330       \@M \@M \@M \@M \@M \@M \@M \@M \@M
1331       \value{typog@mean@penalty}

```

```

1332         \clubpenalty
1333     \fi}
1334

```

`vtietoppar` (*env.*)

```

1335 \NewDocumentEnvironment{vtietoppar}{0{3}}
1336   {\vtietop[#1]}
1337   {\par
1338     \ignorespacesafterend}
1339

```

`\splicevtietop`

```

1340 \NewDocumentCommand{\splicevtietop}{0{3}}
1341   {\let\typog@old@item=\@item
1342     \def\@item[##1]{\typog@old@item[##1]\vtietop[#1]}%
1343     \ignorespaces}
1344

```

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).

```

1345 \ifdefined\SetEnumitemKey
1346   \SetEnumitemKey{vtietop}{first=\splicevtietop}
1347 \fi
1348

```

`\vtiebot`

```

1349 \NewDocumentCommand{\vtiebot}{0{3}}
1350   {\setcounter{typog@mean@penalty}
1351     {\typog@geometric@mean{\@M}{\widowpenalty}}%
1352     \typog@debug@typeout{vtiebot: penalties \the\@M--\the\value{typog@mean@penalty}
1353       -\the\widowpenalty}%
1354     \unless\ifnum\widowpenalty<\@M
1355       \PackageWarning{typog}{vtiebot: widowpenalty=\the\widowpenalty\space>= 10000}
1356     \fi
1357     \ifcase#1% 0
1358       \relax
1359     \or % 1
1360       \relax
1361     \or % 2
1362       \widowpenalties 3
1363       \@M
1364       \value{typog@mean@penalty}
1365       \widowpenalty
1366     \or % 3
1367       \widowpenalties 4
1368       \@M \@M
1369       \value{typog@mean@penalty}
1370       \widowpenalty
1371     \or % 4
1372       \widowpenalties 5
1373       \@M \@M \@M
1374       \value{typog@mean@penalty}

```

```

1374         \widowpenalty
1375     \or % 5
1376         \widowpenalties 6
1377             \@M \@M \@M \@M
1378             \value{typog@mean@penalty}
1379             \widowpenalty
1380     \or % 6
1381         \widowpenalties 7
1382             \@M \@M \@M \@M \@M
1383             \value{typog@mean@penalty}
1384             \widowpenalty
1385     \or % 7
1386         \widowpenalties 8
1387             \@M \@M \@M \@M \@M \@M
1388             \value{typog@mean@penalty}
1389             \widowpenalty
1390     \or % 8
1391         \widowpenalties 9
1392             \@M \@M \@M \@M \@M \@M \@M
1393             \value{typog@mean@penalty}
1394             \widowpenalty
1395     \else % >= 9
1396         \widowpenalties 10
1397             \@M \@M \@M \@M \@M \@M \@M \@M
1398             \value{typog@mean@penalty}
1399             \widowpenalty
1400     \fi}
1401

```

vtiebotpar (*env.*)

```

1402 \NewDocumentEnvironment{vtiebotpar}{0{3}}
1403   {\vtiebot[#1]}
1404   {\par
1405     \ignorespacesafterend}
1406

```

\typog@vtiebotdisp

```

1407 \NewDocumentCommand{\typog@vtiebotdisp}{m}
1408   {\setcounter{typog@mean@penalty}
1409     {\typog@geometric@mean{\@M}{\displaywidowpenalty}}%
1410     \typog@debug@typeout{vtiebotdisp: penalties \the\@M--\the\value{typog@mean@pen
1411     -\the\displaywidowpenalty}%
1412     \unless\ifnum\displaywidowpenalty<\@M
1413       \PackageWarning{typog}{vtiebotdisp: displaywidowpenalty=\the\displaywidowpen
1414     \fi
1415     \ifcase#1% 0
1416       \relax
1417     \or % 1
1418       \relax
1419     \or % 2
1420       \displaywidowpenalties 3
1421       \@M

```



```

1421         \value{typog@mean@penalty}
1422         \displaywidowpenalty
1423     \or % 3
1424         \displaywidowpenalties 4
1425         \@M \@M
1426         \value{typog@mean@penalty}
1427         \displaywidowpenalty
1428     \or % 4
1429         \displaywidowpenalties 5
1430         \@M \@M \@M
1431         \value{typog@mean@penalty}
1432         \displaywidowpenalty
1433     \or % 5
1434         \displaywidowpenalties 6
1435         \@M \@M \@M \@M
1436         \value{typog@mean@penalty}
1437         \displaywidowpenalty
1438     \or % 6
1439         \displaywidowpenalties 7
1440         \@M \@M \@M \@M \@M
1441         \value{typog@mean@penalty}
1442         \displaywidowpenalty
1443     \or % 7
1444         \displaywidowpenalties 8
1445         \@M \@M \@M \@M \@M \@M
1446         \value{typog@mean@penalty}
1447         \displaywidowpenalty
1448     \or % 8
1449         \displaywidowpenalties 9
1450         \@M \@M \@M \@M \@M \@M \@M
1451         \value{typog@mean@penalty}
1452         \displaywidowpenalty
1453     \else % >= 9
1454         \displaywidowpenalties 10
1455         \@M \@M \@M \@M \@M \@M \@M \@M
1456         \value{typog@mean@penalty}
1457         \displaywidowpenalty
1458     \fi}
1459

```

`vtiebotdisp` (*env.*)

```

1460 \NewDocumentEnvironment{vtiebotdisp}{0{3}}
1461   {\typog@vtiebotdisp{#1}}
1462   {\ignorespacesafterend}
1463

```

`vtiebotdisptoppar` (*env.*)

```

1464 \NewDocumentEnvironment{vtiebotdisptoppar}{0{3}o}
1465   {\postdisplaypenalty=\@M
1466   \predisplaypenalty=10001% in accordance with package ‘widows-
and-orphans’
1467   \edef\typog@top@lines{\IfNoValueTF{#2}{#1}{#2}}%

```

```

1468 \edef\typog@@after@display@math{\vtietop[\typog@@top@lines]}%
1469 \PushPostHook{display}{\aftergroup\typog@@after@display@math}%
1470 \vtiebotdisp[#1]}
1471 {\par
1472 \PopPostHook{display}%
1473 \ignorespacesafterend}
1474

```

C.15 Breakable Displayed Equations

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

```

1475 \newenvironment*{breakabledisplay}[1][3]
1476 {\allowdisplaybreaks[#1]}
1477 {\ignorespacesafterend}
1478

```

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.

```

1479 \newcommand*{\typog@setbaselineskip@iter@limit}{10}

```

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

```

1480 \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.

```

1481 \ExplSyntaxOn
1482 \cs_new:Npn \typog@setbaselineskip #1
1483 {

```

Initialize our “emergency-stop” loop counter.

```

1484 \int_set:Nn \l_tmpa_int {1}
1485 \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.

```

1486 \dim_set:Nn \l_tmpa_dim {\glueexpr #1}
1487
1488 \typog@debug@typeout{\string\setbaselineskip:\space
1489 initial\space baselineskip:\space \the\baselineskip}
1490 \typog@debug@typeout{\string\setbaselineskip:\space
1491 target\space baselineskip:\space \dim_use:N \l_tmpa_dim}
1492
1493 \dim_compare:nNnTF {\baselineskip} > {\c_zero_dim}
1494 {}
1495 {
1496 \PackageError{typog}

```

```

1497         {\string\setbaselineskip:\space
1498          baselineskip\space not\space positive}
1499         {}
1500     }
1501
1502 \dim_compare:nNnTF {\l_tmpa_dim} > {\c_zero_dim}
1503 {}
1504 {
1505     \PackageError{typog}
1506         {\string\setbaselineskip:\space target\space
1507          baselineskip\space must\space be\space
1508          positive}
1509         {}
1510 }
1511
1512 \skip_if_eq:nnTF {\l_tmpa_dim} {\glueexpr #1}
1513 {}
1514 {
1515     \PackageWarning{typog}
1516         {\string\setbaselineskip:\space argument\space
1517          is\space a\space skip;\space
1518          will\space ignore\space glue}
1519         {}
1520 }
1521
1522 \fp_set:Nn \l_tmpa_fp {\l_tmpa_dim / \baselineskip}
1523 \fp_until_do:nNnn {abs(\l_tmpa_dim / \baselineskip - 1)} <
1524     {\typog@setbaselineskip@relative@error}
1525 {
1526     \setstretch{\fp_use:N \l_tmpa_fp}
1527     \fp_set:Nn \l_tmpa_fp
1528         {\l_tmpa_fp * \l_tmpa_dim / \baselineskip}
1529
1530     \int_incr:N \l_tmpa_int
1531     \int_compare:nNnTF {\l_tmpa_int} > {\l_tmpb_int}
1532     {
1533         \PackageError{typog}
1534             {\string\setbaselineskip:\space excessive\space
1535              number\space of\space iterations:\space
1536              \int_use:N \l_tmpa_int\space >\space
1537              \int_use:N \l_tmpb_int}
1538             {}
1539     }
1540     {}
1541 }
1542
1543 \typog@debug@typeout{\string\setbaselineskip:\space
1544   final\space \string\setstretch\space argument:\space
1545   \fp_use:N \l_tmpa_fp}
1546 \typog@debug@typeout{\string\setbaselineskip:\space
1547   final\space baselineskip:\space \the\baselineskip}
1548 }

```

1549

`\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.

```
1550 \cs_new:Npn \setbaselineskip #1
1551 {
1552   \AfterPreamble{\typog@setbaselineskip{#1}}
1553   \ignorespaces
1554 }
1555
```

`\resetbaselineskip` Set the `\baselineskip` to ‘neutral’.

```
1556 \cs_new:Npn \resetbaselineskip
1557 {
1558   \AfterPreamble{\setstretch{1}}
1559 }
1560
```

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

```
1561 \dim_new:N \typogfontsize

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

1562 \AfterEndPreamble{
1563   \dim_set:Nn \typogfontsize {\fontdimen6\font}
1564   \typog@debug@typeout{\string\typogfontsize =
1565     \dim_use:N \typogfontsize\space
1566     (at\space begin\space of\space document)}
1567 }
1568
```

`\setbaselineskippercentage`

```
1569 \cs_new:Npn \setbaselineskippercentage #1
1570 {
1571   \AfterPreamble{
1572     \dim_compare:nNnTF {\typogfontsize} > {\c_zero_dim}
1573     {
1574       \typog@setbaselineskip{
```

```

1575     \fp_eval:n {(#1) / 100} \typogfontsize}
1576   }
1577   {
1578     \PackageError{typog}
1579       {\string\setbaselineskippercentage:\space
1580        \string\typogfontsize <= 0}
1581       {Maybe\space \string\typogfontsize\space
1582        is\space uninitialized?}
1583   }
1584 }
1585 \ignorespaces
1586 }
1587

```

`\setleading`

```

1588 \cs_new:Npn \setleading #1
1589 {
1590   \AfterPreamble{
1591     \dim_compare:nNnTF {\typogfontsize} > {\c_zero_dim}
1592     {
1593       \typog@setbaselineskip{\typogfontsize + \dimexpr #1}
1594     }
1595     {
1596       \PackageError{typog}
1597         {\string\setleading:\space
1598          \string\typogfontsize <= 0}
1599         {Maybe\space \string\typogfontsize\space
1600          is\space uninitialized?}
1601     }
1602   }
1603   \ignorespaces
1604 }
1605

```

`\setleadingpercentage`

```

1606 \cs_new:Npn \setleadingpercentage #1
1607 {
1608   \AfterPreamble{
1609     \dim_compare:nNnTF {\typogfontsize} > {\c_zero_dim}
1610     {
1611       \typog@setbaselineskip{
1612         \fp_eval:n {1 + (#1) / 100} \typogfontsize}
1613     }
1614     {
1615       \PackageError{typog}
1616         {\string\setleadingpercentage:\space
1617          \string\typogfontsize <= 0}
1618         {Maybe\space \string\typogfontsize\space
1619          is\space uninitialized?}
1620     }
1621   }
1622   \ignorespaces

```

```

1623 }
1624 \ExplSyntaxOff
1625

```

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.

```

1626 \ExplSyntaxOn
1627 \cs_new_eq:NN \typog@repeat \prg_replicate:nn
1628

```

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

```

1629 \newcommand*{\typog@mod}[2]{\int_mod:nn{#1}{#2}}
1630 \ExplSyntaxOff
1631

```

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

```

1632 \newcommand*{\typog@triplet@max@lines}{99}
1633

```

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

```

1634 \define@key[typog]{smoothraggedrightshapetriplet}{leftskip}%
1635         {\def\typog@@triplet@leftskip{#1}}
1636 \define@key[typog]{smoothraggedrightshapetriplet}{parindent}%
1637         {\def\typog@@triplet@parindent{#1}}
1638 \NewDocumentEnvironment{smoothraggedrightshapetriplet}{0}{ m m m}
1639   {\def\typog@@triplet@leftskip{\z@}%
1640    \def\typog@@triplet@parindent{\z@}%
1641    \setkeys* [typog]{smoothraggedrightshapetriplet}{#1}%
1642    \skip0=\typog@@triplet@leftskip\relax
1643    \skip1=#2\relax
1644    \skip2=#3\relax
1645    \skip3=#4\relax
1646    \typog@debug@typeout{smoothraggedrightshapetriplet: skip0=\the\skip0}%
1647    \typog@debug@typeout{smoothraggedrightshapetriplet: skip1=\the\skip1}%
1648    \typog@debug@typeout{smoothraggedrightshapetriplet: skip2=\the\skip2}%
1649    \typog@debug@typeout{smoothraggedrightshapetriplet: skip3=\the\skip3}%
1650    \unless\ifnum\typog@mod{\typog@triplet@max@lines}{3}=0
1651      \PackageError{typog}
1652        {Line number of triplet generator\space
1653         (\typog@triplet@max@lines) not divisible by 3}
1654      {}
1655    \fi
1656    \edef\typog@@triplet@linespecs{%
1657      \glueexpr \skip0 + \typog@@triplet@parindent\relax
1658      \glueexpr \skip1 - \typog@@triplet@parindent\relax

```

```

1659             \skip0 \skip2 \skip0 \skip3
1660     \typog@repeat{\numexpr\typog@triplet@max@lines / 3 - 1}
1661             {\skip0 \skip1 \skip0 \skip2 \skip0 \skip3}}
1662     \parshape=\typog@triplet@max@lines\typog@@triplet@linespecs\relax
1663     {\par}
1664

```

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

```

1665 \newcommand*{\typog@quintuplet@max@lines}{95}
1666

```

`\smoothraggedrightshapequintuplet (env.)` Engine for 5-line repetitions.

```

1667 \define@key[typog]{smoothraggedrightshapequintuplet}{leftskip}
1668     {\def\typog@@quintuplet@leftskip{#1}}
1669 \define@key[typog]{smoothraggedrightshapequintuplet}{parindent}
1670     {\def\typog@@quintuplet@parindent{#1}}
1671 \NewDocumentEnvironment{smoothraggedrightshapequintuplet}{0}{ m m m m m}
1672     {\def\typog@@quintuplet@leftskip{\z@}%
1673     \def\typog@@quintuplet@parindent{\z@}%
1674     \setkeys* [typog]{smoothraggedrightshapequintuplet}{#1}%
1675     \skip0=\typog@@quintuplet@leftskip\relax
1676     \skip1=#2\relax
1677     \skip2=#3\relax
1678     \skip3=#4\relax
1679     \skip4=#5\relax
1680     \skip5=#6\relax
1681     \typog@debug@typeout{smoothraggedrightshapequintuplet: skip0=\the\skip0}%
1682     \typog@debug@typeout{smoothraggedrightshapequintuplet: skip1=\the\skip1}%
1683     \typog@debug@typeout{smoothraggedrightshapequintuplet: skip2=\the\skip2}%
1684     \typog@debug@typeout{smoothraggedrightshapequintuplet: skip3=\the\skip3}%
1685     \typog@debug@typeout{smoothraggedrightshapequintuplet: skip4=\the\skip4}%
1686     \typog@debug@typeout{smoothraggedrightshapequintuplet: skip5=\the\skip5}%
1687     \unless\ifnum\typog@mod{\typog@quintuplet@max@lines}{5}=0
1688     \PackageError{typog}
1689         {Line number of quintuplet generator\space
1690         (\typog@quintuplet@max@lines) not divisible by 5}
1691     {}
1692     \fi
1693     \edef\typog@@quintuplet@linespecs{%
1694     \glueexpr \skip0 + \typog@@quintuplet@parindent\relax
1695     \glueexpr \skip1 - \typog@@quintuplet@parindent\relax
1696     \skip0 \skip2 \skip0 \skip3
1697     \skip0 \skip4 \skip0 \skip5
1698     \typog@repeat{\numexpr\typog@quintuplet@max@lines / 5 - 1}
1699     {\skip0 \skip1 \skip0 \skip2 \skip0 \skip3
1700     \skip0 \skip4 \skip0 \skip5}}
1701     \parshape=\typog@quintuplet@max@lines\typog@@quintuplet@linespecs\relax
1702     {\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.

```
1703 \newcommand*{\typog@septuplet@max@lines}{98}
1704
```

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

```
1705 \define@key[typog]{smoothraggedrightshapeseptuplet}{leftskip}%
1706     {\def\typog@@septuplet@leftskip{#1}}
1707 \define@key[typog]{smoothraggedrightshapeseptuplet}{parindent}%
1708     {\def\typog@@septuplet@parindent{#1}}
1709 \NewDocumentEnvironment{smoothraggedrightshapeseptuplet}{0}{ m m m m m m m}
1710     {\def\typog@@septuplet@leftskip{z@}%
1711     \def\typog@@septuplet@parindent{z@}%
1712     \setkeys*[typog]{smoothraggedrightshapeseptuplet}{#1}%
1713     \skip0=\typog@@septuplet@leftskip\relax
1714     \skip1=#2\relax
1715     \skip2=#3\relax
1716     \skip3=#4\relax
1717     \skip4=#5\relax
1718     \skip5=#6\relax
1719     \skip6=#7\relax
1720     \skip7=#8\relax
1721     \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip0=\the\skip0}%
1722     \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip1=\the\skip1}%
1723     \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip2=\the\skip2}%
1724     \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip3=\the\skip3}%
1725     \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip4=\the\skip4}%
1726     \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip5=\the\skip5}%
1727     \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip6=\the\skip6}%
1728     \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip7=\the\skip7}%
1729     \unless\ifnum\typog@mod{\typog@septuplet@max@lines}{7}=0
1730         \PackageError{typog}
1731             {Line number of septuplet generator\space
1732             (\typog@septuplet@max@lines) not divisible by 7}
1733         {}
1734     \fi
1735     \edef\typog@@septuplet@linespecs{%
1736         \glueexpr \skip0 + \typog@@septuplet@parindent\relax
1737         \glueexpr \skip1 - \typog@@septuplet@parindent\relax
1738         \skip0 \skip2 \skip0 \skip3 \skip0 \skip4
1739         \skip0 \skip5 \skip0 \skip6 \skip0 \skip7
1740         \typog@repeat{\numexpr\typog@septuplet@max@lines / 7 - 1}
1741             {\skip0 \skip1 \skip0 \skip2 \skip0 \skip3 \skip0 \skip4
1742             \skip0 \skip5 \skip0 \skip6 \skip0 \skip7}}
1743     \parshape=\typog@septuplet@max@lines\typog@@septuplet@linespecs\relax
1744     {\par}
1745
```

smoothraggedrightfuzzfactor

```
1746 \newcommand*{\smoothraggedrightfuzzfactor}{1.0}
```

smoothraggedrightgenerator

```
1747 \newcommand*{\smoothraggedrightgenerator}{triplet}
```


`\smoothraggedrightleftskip` (*skip*)

```
1748 \newlength{\smoothraggedrightleftskip}
```

`\smoothraggedrightparindent` (*skip*)

```
1749 \newlength{\smoothraggedrightparindent}
```

`\smoothraggedrightragwidth` (*skip*)

```
1750 \newlength{\smoothraggedrightragwidth}
1751 \setlength{\smoothraggedrightragwidth}{2em}
1752
```

`\typog@fuzzwidth` (*dimen*)

```
1753 \newdimen{\typog@fuzzwidth}
1754
```

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

```
1755 \define@key[typog]{smoothraggedrightpar}{linewidth}{%
1756     {\def\typog@@linewidth{#1}}
1757
```

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

```
1758 \ExplSyntaxOn
1759 \cs_new:Npn \typog@generator@index:n #1 {
1760     \str_case:nnTF {#1}
1761         {
1762             {triplet} {0}
1763             {quintuplet} {1}
1764             {septuplet} {2}
1765         }
1766         {}
1767         {-1}
1768 }
```

```
1769 \cs_generate_variant:Nn \typog@generator@index:n {e}
1770 \let\typog@generator@index=\typog@generator@index:e
1771 \ExplSyntaxOff
1772
```

```
1773 \NewDocumentEnvironment{smoothraggedrightpar}{0{}}
```

```
1774     {\edef\typog@@linewidth{\linewidth}%
1775     \setkeys[typog]{smoothraggedrightpar}{#1}%
1776     \edef\typog@@generatorchoice{\typog@generator@index{\smoothraggedrightgenerato
```

Obey to the indentation prescribed by any list environment.

```
1777     \let\typog@@smoothraggedrightleftskip=\smoothraggedrightleftskip
1778     \ifnum\@listdepth>0
1779         \addtolength{\typog@@smoothraggedrightleftskip}{\leftmargin}%
1780     \fi
```

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

```

1781 \typog@fuzzwidth=\smoothraggedrightfuzzfactor\smoothraggedrightragwidth
      Now for the generator-specific code ...
1782 \ifcase\typog@@generatorchoice
      generator=triplet produces a “short line – long line – middle length
line” sequence.
1783 \typog@fuzzwidth=.25\smoothraggedrightragwidth
1784 \typog@debug@typeout{smoothraggedright: generator=triplet, ty-
pog@fuzzwidth=\the\typog@fuzzwidth}%
1785 \smoothraggedrightshapetriplet[leftskip=\typog@@smoothraggedrightleftskip,
1786 \parindent=\glueexpr\smoothraggedrightparinden
      indent,
1787 \hspace{#1}%
1788 {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth
1789 + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
1790 {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth}% (3)
1791 {\glueexpr (\typog@@linewidth * 2 - \smoothraggedrightrag-
width) / 2
1792 + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (2)
1793 \or
      generator=quintuplet.
1794 \typog@fuzzwidth=.125\smoothraggedrightragwidth
1795 \typog@debug@typeout{smoothraggedright: generator=quintuplet, ty-
pog@fuzzwidth=\the\typog@fuzzwidth}%
1796 \smoothraggedrightshapequintuplet[leftskip=\typog@@smoothraggedrightleftskip
1797 \parindent=\glueexpr\smoothraggedrightparinden
      indent,
1798 \hspace{#1}%
1799 {\glueexpr (\typog@@linewidth * 4 - \smoothraggedrightrag-
width * 3) / 4
1800 + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (2)
1801 {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth\relax}% (5)
1802 {\glueexpr (\typog@@linewidth * 2 - \smoothraggedrightrag-
width) / 2
1803 + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (3)
1804 {\glueexpr (\typog@@linewidth * 4 - \smoothraggedrightrag-
width) / 4
1805 + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (4)
1806 {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth
1807 + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
1808 \or
      generator=septuplet.
      Permutation 3 – 6 – 1 – 5 – 2 – 7 – 4 looks ‘random’ enough for our purposes.
1809 \typog@fuzzwidth=.08333\smoothraggedrightragwidth

```

```

1810     \typog@debug@typeout{smoothraggedright: generator=septuplet, ty-
pog@fuzzwidth=\the\typog@fuzzwidth}%
1811     \smoothraggedrightshapeseptuplet[leftskip=\typog@smoothraggedrightleftskip,
1812                                     parindent=\glueexpr\smoothraggedrightparind
indent,
1813                                     #1]%
1814     {\glueexpr (\typog@@linewidth * 3 - \smoothraggedrightrag-
width * 2) / 3
1815         + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (3)
1816     {\glueexpr (\typog@@linewidth * 6 - \smoothraggedrightrag-
width) / 6
1817         + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (6)
1818     {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth +
1819         + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
1820     {\glueexpr (\typog@@linewidth * 3 - \smoothraggedrightrag-
width) / 3
1821         + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (5)
1822     {\glueexpr (\typog@@linewidth * 6 - \smoothraggedrightrag-
width * 5) / 6
1823         + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (2)
1824     {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth\relax}% (7)
1825     {\glueexpr (\typog@@linewidth * 2 - \smoothraggedrightrag-
width) / 2
1826         + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (4)
1827     \else
1828     \PackageError{typog}
1829         {smoothraggedright: unknown generator name (\smooth-
raggedrightgenerator)}
1830         {valid generator names are triplet, quintuplet, and sep-
tuplet}
1831     \fi}
1832 {\ifcase\typog@@generatorchoice
1833     \endsmoothraggedrightshapetriplet
1834     \or
1835     \endsmoothraggedrightshapequintuplet
1836     \or
1837     \endsmoothraggedrightshapeseptuplet
1838     \fi}
1839

```

smoothraggedright (*env.*)

```

1840 \NewDocumentEnvironment{smoothraggedright}{0{}}
1841 {\PushPostHook{par}{\hskip-\parindent\smoothraggedrightpar[#1]\relax}}
1842 {\par\PopPostHook{par}}
1843

```

Change History

v0.1

General: Initial version. [i](#)

v0.2

`\narrowospace`: New macro. [102](#)

`\widespace`: Add fallback if `\fontdimen7` is zero. Extend with a starred version. [101](#)

v0.3

`hyphenmin`: New environment. [78](#)

`\resetbaselineskip`: New macro. [115](#)

`\setbaselineskip`: New macro. [115](#)

`\setbaselineskippercentage`: New macro. [115](#)

`\setleading`: New macro. [116](#)

`\setleadingpercentage`: New macro. [116](#)

`\typogfontsize`: New dimen. [115](#)

v0.4

`\lowercaseadjustlabelitems`: New macro. [93](#)

`\noadjustlabelitems`: New macro. [93](#)

`\typogadjuststairs`: New macro. [97](#)

`\typoggetnth`: New macro. [75](#)

`\typoglowercaseadjustcheck`: New macro. [99](#)

`\typoguppercaseadjustcheck`: New macro. [98](#)

`\uppercaseadjustlabelitems`: New macro. [93](#)

v0.4a

`smoothraggedrightshapeseptuplet`: Add missing backslash to `\typog@@septuplet@parindent` in local macro `\typog@@septuplet@linespecs` which rendered the septuplet generator unusable. [119](#)

`\typog@register@pdfsubstitute`: Using `\pdfstringdefDisableCommands` breaks package `pdfcomment`. So we just refrain from using the macro if `pdfcomment` was loaded. [71](#)

v0.5

General: **[BACKWARD INCOMPATIBLE CHANGE]** Rename package option `mathitalicscorrection` to `mathitaliccorrection`. [74](#)

[BACKWARD INCOMPATIBLE CHANGE] Rename package option `textitalicscorrection` to `textitaliccorrection`. [74](#)

Rewrite parser for package option `lowercaselabelitemadjustments` to accept a star as a placeholder. [74](#)

Rewrite parser for package option `uppercaselabelitemadjustments` to accept a star as a placeholder. [74](#)

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

`\adjustedlabelitemi`: New macro. [96](#)

`\Adjustedlabelitemii`: New macro. [96](#)

`\adjustedlabelitemii`: New macro. [96](#)

`\Adjustedlabelitemiii`: New macro. [96](#)

`\adjustedlabelitemiii`: New macro. [96](#)

`\Adjustedlabelitemiv`: New macro. [96](#)

`\adjustedlabelitemiv`: New macro. [96](#)
`\capitalinvertedexclamationmark`: New macro. [91](#)
`\capitalinvertedquestionmark`: New macro. [91](#)
`\kernedslash`: Allow for lowering (or raising) `\kernedslash` with package option `lowerslash`. [80](#)
`lastlinefitpar`: New environment. [101](#)
`\leftspacedcapitalemdash`: New macro. [87](#)
`\leftspacedcapitalendash`: New macro. [86](#)
`\leftspacedendash`: New macro. [82, 83](#)
`\rightspacedcapitalemdash`: New macro. [87](#)
`\rightspacedcapitalendash`: New macro. [86](#)
`\rightspacedemdash`: New macro. [84](#)
`\rightspacedendash`: New macro. [82](#)
`\swapendashskip`: New macro. [83](#)
`\typoglogo`: Make macro independent of typog setup. [70](#)

References

- [1] ABRAHAMS, PAUL W., KATHRYN A. HARGREAVES, 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] *The Chicago Manual of Style*, 17th ed. The University of Chicago Press, Chicago/IL, 2017.
- [5] BEZOS, JAVIER. *Package enumitem*. 2019, <https://ctan.org/pkg/enumitem>.
- [6] BEZOS, JAVIER. *Package babel*. 2021, <https://ctan.org/pkg/babel>. The original author of package babel was J. L. BRAAMS.
- [7] BREITENLOHNER, PETER and the N_TS TEAM. *e-T_EX*. 1998, https://mirrors.ctan.org/systems/doc/etex/etex_man.pdf.
- [8] BURNOL, JEAN-FRANÇOIS. *Package mathastext*. 2023, <https://ctan.org/pkg/mathastext>.
- [9] 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>.
- [10] CARLISLE, DAVID. *What do different \fontdimen<num> mean*. 2013–1–2, <https://tex.stackexchange.com/questions/88991/what-do-different-fontdimennum-mean>.
- [11] CUBITT, TOBY. *Package cleveref*. 2018, <https://ctan.org/pkg/cleveref>.
- [12] EIJKHOUT, VICTOR. *T_EX By Topic, A Texnician's Reference*. 2007, <https://www.eijkhout.net/tex/tex-by-topic.html>.
- [13] FORSSMAN, FRIEDRICH, and RALF DE JONG. *Detailtypographie*. 4. Aufl. Verlag Hermann Schmitz, Mainz, 2008.
- [14] HØGHOLM, MORTEN, LARS MADSEN, and the L^AT_EX₃ PROJECT TEAM. *Package mathtools*. 2020, <https://ctan.org/pkg/mathtools>.
- [15] KHIREVICH, SIARHEI. *Tips on Writing a Thesis in L^AT_EX*. 2013, <https://www.khirevich.com/latex/microtype>.
- [16] KLEBER, JOSEF. *Package pdfcomment*. 2018, <https://ctan.org/pkg/pdfcomment>.
- [17] KNUTH, DONALD ERVIN. *The T_EXbook*. Addison Wesley, Reading/MA, 1986.

- [18] THE L^AT_EX PROJECT. *Package l3experimental*. 2024, <https://ctan.org/pkg/l3experimental>.
- [19] 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.
- [20] MIDDENDORP, JAN. *Shaping Text*. BIS publishers, Amsterdam, 2014.
- [21] 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>.
- [22] MITTELBACH, FRANK. *Package widows-and-orphans*. 2020, <https://ctan.org/pkg/widows-and-orphans>.
- [23] NIEDERBERGER, CLEMENS. *Package tasks*. 2022, <https://github.com/cgnieder/tasks>.
- [24] RAHTZ, SEBASTIAN, and FRANK MITTELBACH. *Package hyperref*. 2020, <https://ctan.org/pkg/hyperref>. The package is maintained by the L^AT_EX3 Project Team.
- [25] SCHLICHT, ROBERT. *Package microtype*. 2020, <https://ctan.org/pkg/microtype>.
- [26] SCHRÖDER, MARTIN. *Package ragged2e*. 2019, <https://ctan.org/pkg/ragged2e>.
- [27] 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>.
- [28] STRIZVER, ILENE. *Type rules!: the designer's guide to professional typography*, 4th ed. John Wiley & Sons, Hoboken/NJ, 2014.
- [29] TOBIN, GEOFFREY, and ROBIN FAIRBAIRNS. *Package setspace*. 2011, <https://ctan.org/pkg/setspace>.
- [30] UMEKI, HIDEO. *Package geometry*. 2020, <https://ctan.org/pkg/geometry>.
- [31] WERMUTH, UDO. *Tracing paragraphs*. TUGboat, 37(3), 358–373, 2016, <https://tug.org/TUGboat/tb37-3/tb117wermuth.pdf>.
- [32] WERMUTH, UDO. *The optimal value for \emergencystretch*. TUGboat, 38(1), 65–86, 2017, <https://tug.org/TUGboat/tb38-1/tb118wermuth.pdf>.
- [33] WERMUTH, UDO. *A note on \linepenalty*. TUGboat, 38(3), 400–414, 2017, <https://tug.org/TUGboat/tb38-3/tb120wermuth.pdf>.
- [34] WERMUTH, UDO. *Experiments with \parfillskip*. TUGboat, 39(3), 276–303, 2018, <https://tug.org/TUGboat/tb39-3/tb123wermuth-parfillskip.pdf>.

- [35] WERMUTH, UDO. *An attempt at ragged-right typesetting*. TUGboat, 41(1), 73–94, 2020, <https://tug.org/TUGboat/tb41-1/tb127wermuth-ragged.pdf>.
- [36] WERMUTH, UDO. Personal communication. August 2, 2022.
- [37] WERMUTH, UDO. *Vertical alignments in plain T_EX*. TUGboat, 44(3), 427–440, 2023, <https://tug.org/TUGboat/tb44-3/tb138wermuth-valign.pdf>.
- [38] WILSON, PETER. *Package hyphenat*. 2004, <https://ctan.org/pkg/hyphenat>. The package is maintained by W. ROBERTSON.
- [39] WILSON, PETER. *Glisterings*. TUGboat, 28(2), 229–232, 2007, <https://tug.org/TUGboat/tb28-2/tb89glister.pdf>.
- [40] WILSON, PETER. *Package needspace*. 2010, <https://ctan.org/pkg/needspace>. The package is maintained by W. ROBERTSON.

Index

Es ist schon alles gesagt,
nur noch nicht von allen.
— KARL VALENTIN

Numbers written in *italic style* refer to definitions; numbers in *regular style* refer to pages where an entry is used. References to code lines are prefixed with ‘*l*’. Again, *italic style* refers to definitions and *regular style* to lines where the code is used.

A

`\Adjustedlabelitemi` 32, *l823*, *l849*
`\adjustedlabelitemi` 32, *l826*, *l850*
`\Adjustedlabelitemii` 32, *l829*,
l851
`\adjustedlabelitemii` 32, *l832*,
l852
`\Adjustedlabelitemiii` 32, *l835*,
l853
`\adjustedlabelitemiii` 32, *l838*,
l854
`\Adjustedlabelitemiv` 32, *l841*,
l855
`\adjustedlabelitemiv` 32, *l844*,
l856
`\allowhyphenation` 11, *l210*
`amsmath` (package) 51

B

`baseline skip` 52
`bookmark` 16, 21
`breakabledisplay` (env.) 51, *l1475*
`breakpenalty` (option) 2, 16, 18, 24–26
`\breakpoint` 14, *l214*
`\breakpoint*` 14

C

`\capitaldash` 25, *l483*, *l491*, *l579*
`\capitaldash*` 25
`\capitalemdash` 25, *l494*, *l546*, *l547*,
l560, *l561*
`\capitalemdash*` 25
`\capitalendash` 25, *l478*, *l510*, *l511*,
l526, *l527*
`\capitalendash*` 25
`\capitalhyphen` 24, *l464*
`\capitalhyphen*` 24
`\capitalinvertedexclamationmark` 29,
l670

`\capitalinvertedquestionmark` 29,
l678
`\capitaltimes` 27, *l581*
code reference 70–122
configuration 6
`covernextindentpar` (env.) 39, *l965*
`csquotes` (package) 28

D

`debug` (option) 2
dimensions:
`\smoothraggedrightleftskip` 57,
l1748, *l1777*
`\smoothraggedrightparindent` 57,
l1749, *l1786*, *l1797*, *l1812*
`\smoothraggedrightragwidth` 58,
l1750, *l1781*, *l1783*, *l1788*, *l1791*,
l1794, *l1799*, *l1802*, *l1804*, *l1806*,
l1809, *l1814*, *l1816*, *l1818*, *l1820*,
l1822, *l1825*
`\typogfontsize` 54, *l1561*, *l1572*,
l1575, *l1580*, *l1581*, *l1591*, *l1593*,
l1598, *l1599*, *l1609*, *l1612*, *l1617*,
l1618
`\Doubleguillemetleft` 27, *l630*
`\doubleguillemetleft` 27, *l604*
`\Doubleguillemetright` 27, *l637*
`\doubleguillemetright` 27, *l610*

E

`emdashspace` (option) 2
`endashspace` (option) 2
enumitem-keys:
`vtietop` 49
environments:
`breakabledisplay` 51, *l1475*
`covernextindentpar` 39, *l965*
`hyphenmin` 14, *l224*
`lastlinecenteredpar` 34, *l941*
`lastlinefitpar` 40, *l995*

- lastlineflushrightpar 34, ℓ938
- lastlineraggedleftpar 34, ℓ932
- looespacing 41, ℓ1031
- nocharprotrusion 46, ℓ1200
- nofontexpand 46, ℓ1197
- nofontexpansion 46, ℓ1191
- openlastlinepar 39, ℓ979
- prolongpar 38, ℓ955
- setfontexpand 45, ℓ1177
- setfontshrink 45, ℓ1149
- setfontstretch 45, ℓ1163
- setfonttracking 44, ℓ1063
- shortenpar 38, ℓ948
- slightlyloppypar 46, ℓ1271
- smoothraggedright 57, ℓ1840
- smoothraggedrightpar 55, ℓ1755
- smoothraggedrightshape-
quintuplet 55, ℓ1667
- smoothraggedrightshape-
septuplet 55, ℓ1705
- smoothraggedrightshape-
triplet 55, ℓ1634
- tightspacing 41, ℓ1047
- typoginspect 8, ℓ171
- typoginspectpar 8, ℓ198
- typogsetup 6, ℓ107
- vtiebotdisp 49, ℓ1460
- vtiebotdisptoppar 49, ℓ1464
- vtiebotpar 49, ℓ1402
- vtietoppar 48, ℓ1335
- extra spacing 18
- F**
- \figuredash 26, ℓ574
- \figuredash* 26
- font
 - encoding 27
 - expansion 38, 45
 - information 7
 - protrusion 46
 - size 7
 - spacing 38, 40
 - loose 40
 - tight 40
 - tracking 38, 44
 - typeface
 - ADF Baskervald 28
 - ADF Accanthis 35
 - ADF Venturis 26, 35
 - Alegreya 26
 - Aleo 35
 - Arvo 26
 - Bitter 26
 - Charis SIL 26, 28, 35
 - Clara 26
 - CM Roman 35
 - Cochineal 19, 35
 - Coelacanth 26
 - Crimson Pro 26
 - Crimson Text 26
 - Domitian 35
 - Droid Serif 26
 - EB Garamond 26, 28, 35
 - Erewhon 26
 - etbb 35
 - Extended Charter 35
 - fbf 26
 - Gentium 26, 28
 - Gentium Plus 35
 - GFS Artemisia 26, 28
 - GFS Bodoni 35
 - GFS Didot 28, 35
 - Ibarra Real Nova 26
 - IBM Plex Serif 35
 - INRIA Serif 26
 - KP Serif 35
 - Libertine 26
 - Libertinus Serif 26, 28, 35
 - Libre Baskerville 26
 - Libre Caslon 26
 - Merriweather 26, 28
 - ML Modern 35
 - PT Serif 26
 - Quattrocento 26
 - Roboto Slab 26
 - Source Serif Pro 26, 35
 - Spectral 26, 35
 - STIX 26, 35
 - T_EX Gyre Pagella 26
 - TX Fonts 26
 - URW Palladio ii, 29, 35
 - Utopia 35
- \fontsizeinfo 7, ℓ150
- forlorn line
 - club 38, 48
 - display widow 49
 - orphan 38
 - widow 38, 49

H

hyperref (package) 16, 21
 hyphenation 11
 empty discretionary 14
 first word 14
 \hskip 14
 re-enable automatic 11
 hyphenmin (env.) 14, ℓ224

I

information 7
 inverted
 exclamation mark 29
 question mark 29
 \itcorr 17, 30, ℓ266
 \itcorr* 17

K

\kernedhyphen 19, ℓ291, ℓ313, ℓ314,
 ℓ321, ℓ322
 \kernedhyphen* 19
 \kernedslash 18, ℓ148, ℓ279
 \kernedslash* 18
 kerning
 extra 18
 forward slash 18
 hyphen 19
 ligature 16
 known problems 59

L

label items 30
 lastlinecenteredpar (env.) 34, ℓ941
 \lastlinefit 10, 40, 68
 lastlinefitpar (env.) 40, 68, ℓ995
 lastlineflushrightpar (env.) 34,
 ℓ938
 lastlineraggedleftpar (env.) 34,
 ℓ932
 leading 52
 \leftkernedhyphen 19, ℓ311, ℓ316
 \leftkernedhyphen* 19
 \leftspacedcapitaldash 25, ℓ513,
 ℓ521
 \leftspacedcapitaldash* 25
 \leftspacedcapitalemdash 25,
 ℓ544
 \leftspacedcapitalemdash* 25
 \leftspacedcapitalendash 25,
 ℓ508
 \leftspacedcapitalendash* 25
 \leftspacedddash 21, ℓ349, ℓ361

\leftspacedddash* 21
 \leftspacedemdash 22, ℓ403, ℓ419
 \leftspacedemdash* 22
 \leftspacedendash 21, ℓ336, ℓ403
 \leftspacedendash* 21
 ligature 16
 ligaturekern (option) 2, 16
 limitations 59
 line spacing 7
 list 48
 loosespacing (env.) 41, ℓ1031
 lowercaselabelitem-
 adjustments (option) 2, 30
 \lowercaseadjustlabelitems 30,
 ℓ777
 lowerslash (option) 3, 18

M

mathitaliccorrection (option) 3,
 17
 microtype (package) 44

N

narrow space 43
 \narrow space 43, ℓ1017
 \narrow space* 43
 \narrow spacescale 43, ℓ1016, ℓ1019,
 ℓ1021, ℓ1027, ℓ1028
 \narrow space strength 43, ℓ1015,
 ℓ1024
 \noadjustlabelitems 30, ℓ784
 nocharprotrusion (env.) 46, ℓ1200
 nodebug (option) 2
 nofontexpand (env.) 46, ℓ1197
 nofontexpansion (env.) 46, ℓ1191
 \nolig 16, ℓ231
 \nolig* 16

O

openlastlinepar (env.) 39, ℓ979

P

package code 70–122
 package option 2–5
 breakpenalty 2, 16, 18, 24–26
 debug 2
 emdashspace 2
 endashspace 2
 ligaturekern 2, 16
 lowercaselabelitem-
 adjustments 2, 30
 lowerslash 3, 18

- mathitaliccorrection 3, 17
 - nodebug 2
 - raise* 3
 - raisecapitaldash 3, 25
 - raisecapitalguillemets 3, 27
 - raisecapitalhyphen 3, 19
 - raisecapitaltimes 3, 27
 - raisefiguredash 3, 26
 - raiseguillemets 3, 27
 - raiseinvertedmarks 3, 29
 - shrinklimits 4, 45
 - slashkern 4, 18
 - stretchlimits 4, 45
 - textitaliccorrection 4, 17
 - trackingttspacing 4, 44
 - uppercaselabelitem-
adjustments 4, 30
 - page break 11, 51
 - paragraph
 - align last line 34
 - centered 34
 - flush right 34
 - badness 10
 - fill last line 36
 - covernextindentpar 39
 - \linebreak 38
 - \mbox 37
 - openlastlinepar 39
 - prolongpar 38
 - shortenpar 38
 - tie 37
 - sloppy 46
 - vertically tied 48
 - PDF 16, 21
 - prolongpar (env.) 38, *l955*
- Q**
- quick reference v, 1
- R**
- ragged right 55
 - raise* (option) 3
 - raisecapitaldash (option) 3, 25
 - raisecapitalguillemets (option) 3, 27
 - raisecapitalhyphen (option) 3, 19
 - raisecapitaltimes (option) 3, 27
 - raised character 24
 - em-dash 25
 - en-dash 25
 - guillemets 27
 - hyphen 24
 - inverted exclamation mark 29
 - inverted question mark 29
 - multiplication sign 27
 - number dash 26
 - raisefiguredash (option) 3, 26
 - raiseguillemets (option) 3, 27
 - raiseinvertedmarks (option) 3, 29
 - reconfigure 6
 - \resetbaselineskip 53, *l1556*
 - \rightkernedhyphen 19, *l319*, *l324*
 - \rightkernedhyphen* 19
 - \rightspacedcapitaldash 25, *l529*, *l539*
 - \rightspacedcapitaldash* 25
 - \rightspacedcapitalemdash 25, *l558*
 - \rightspacedcapitalemdash* 25
 - \rightspacedcapitalendash 25, *l524*
 - \rightspacedcapitalendash* 25
 - \rightspaceddash 21, *l377*, *l392*
 - \rightspaceddash* 21
 - \rightspacedemdash 22, *l431*
 - \rightspacedemdash* 22
 - \rightspacedendash 21, *l364*
 - \rightspacedendash* 21
- S**
- \setbaselineskip 52, *l1488*, *l1490*, *l1497*, *l1506*, *l1516*, *l1534*, *l1543*, *l1546*, *l1550*
 - \setbaselineskippercentage 53, *l1569*
 - setfontexpand (env.) 45, *l1177*
 - setfontshrink (env.) 45, *l1149*
 - setfontstretch (env.) 45, *l1163*
 - setfonttracking (env.) 44, *l1063*
 - \setleading 53, *l1588*
 - \setleadingpercentage 53, *l1606*
 - setup 6
 - shortenpar (env.) 38, *l948*
 - shrinklimits (option) 4, 45
 - \Singleguillemetleft 27, *l616*
 - \singleguillemetleft 27, *l592*
 - \Singleguillemetright 27, *l623*
 - \singleguillemetright 27, *l598*
 - slashkern (option) 4, 18
 - \slightlyloppy 46, *l1214*, *l1272*
 - slightlyloppypar (env.) 46, *l1271*
 - smoothraggedright (env.) 57, *l1840*
 - \smoothraggedrightfuzzfactor 57, *l1746*, *l1781*

- `\smoothraggedrightgenerator` 57, [l1747](#), [l1776](#), [l1829](#)
 - `\smoothraggedrightleftskip` (skip) 57, [l1748](#), [l1777](#)
 - `smoothraggedrightpar` (env.) 55, [l1755](#)
 - `\smoothraggedrightparindent` (skip) 57, [l1749](#), [l1786](#), [l1797](#), [l1812](#)
 - `\smoothraggedrightragwidth` (skip) 58, [l1750](#), [l1781](#), [l1783](#), [l1788](#), [l1791](#), [l1794](#), [l1799](#), [l1802](#), [l1804](#), [l1806](#), [l1809](#), [l1814](#), [l1816](#), [l1818](#), [l1820](#), [l1822](#), [l1825](#)
 - `smoothraggedrightshape-quintuplet` (env.) 55, [l1667](#)
 - `smoothraggedrightshape-septuplet` (env.) 55, [l1705](#)
 - `smoothraggedrightshapetriplet` (env.) 55, [l1634](#)
 - spaced character
 - em-dash 22
 - en-dash 21
 - `\spacedcapitaldash` [l531](#), [l541](#)
 - `\spacedcapitalemdash` 25, [l563](#), [l571](#)
 - `\spacedcapitalemdash*` 25
 - `\spacedcapitalendash` [l530](#), [l540](#)
 - `\spaceddash` 21, [l379](#), [l394](#)
 - `\spaceddash*` 21
 - `\spacedemdash` 22, [l446](#), [l459](#)
 - `\spacedemdash*` 22
 - `\spacedendash` 21, [l378](#), [l393](#)
 - `\spacedendash*` 21
 - `\splicevtietop` 48, [l1340](#)
 - `stretchlimits` (option) 4, 45
 - `\swapendashskip` 22, [l397](#)
- T**
- `\textemdash` 23, 26
 - `\textendash` 23, 26
 - `\texttexclamdown` 30
 - `textitaliccorrection` (option) 4, [l17](#)
- `\textquestiondown` 30
 - `tightspacing` (env.) 41, [l1047](#)
 - `trackingttspacing` (option) 4, 44
 - `\typogadjuststairs` 32, [l882](#)
 - `\typogadjuststairsfor` [l860](#)
 - `\typogfontsize` (dim.) 54, [l1561](#), [l1572](#), [l1575](#), [l1580](#), [l1581](#), [l1591](#), [l1593](#), [l1598](#), [l1599](#), [l1609](#), [l1612](#), [l1617](#), [l1618](#)
 - `\typogget` 6, [l116](#)
 - `\typoggetnth` 7, [l119](#)
 - `typoginspect` (env.) 8, [l171](#)
 - `typoginspectpar` (env.) 8, [l198](#)
 - `\typoglogo` [l11](#)
 - `\typoglouercaseadjustcheck` 33, [l925](#)
 - `typogsetup` (env.) 6, [l107](#)
 - `\typoguppercaseadjustcheck` 33, [l913](#)
- U**
- `uppercaselabelitem-adjustments` (option) 4, 30
 - `\uppercaseadjustlabelitems` 30, [l770](#)
- V**
- `\vtiebot` 49, [l1349](#), [l1403](#)
 - `\vtiebotdisp` [l1470](#)
 - `vtiebotdisp` (env.) 49, [l1460](#)
 - `vtiebotdisptoppar` (env.) 49, [l1464](#)
 - `vtiebotpar` (env.) 49, [l1402](#)
 - `\vtietop` 48, [l1282](#), [l1336](#), [l1342](#), [l1468](#)
 - `vtietop` (enumitem-key) 49
 - `vtietoppar` (env.) 48, [l1335](#)
- W**
- wide space 42
 - `\widespace` 42, [l1001](#)
 - `\widespace*` 42
 - `\widespacescale` 42, [l1000](#), [l1003](#), [l1005](#), [l1011](#), [l1012](#)
 - `\widespacestrength` 42, [l999](#), [l1008](#)