

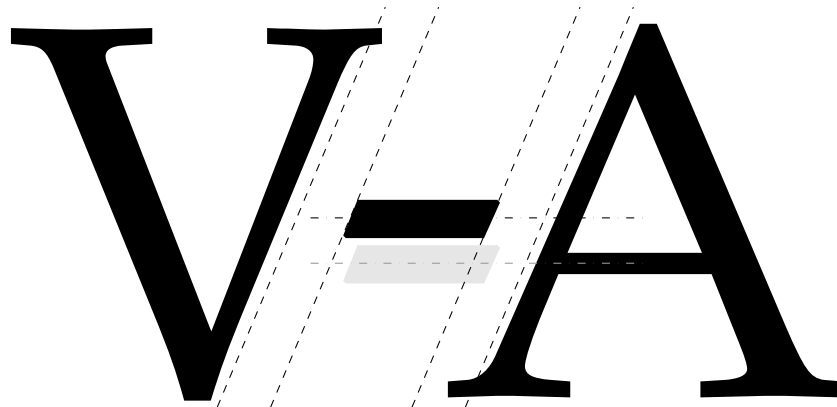
TypoG – Typographic Fine-Tuning

Ch. L. Spiel*

v0.4 2024/05/21

Abstract

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



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

* cspiel@users.sourceforge.org

Contents

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

1

Introduction 1

- 1.1 Overview 1
- 1.2 Prerequisites 1

2

Package Options 2

3

Macros and Environments 5

- 3.1 Setup and Reconfiguration 5
- 3.2 Information 6
 - 3.2.1 Font Information 6
 - 3.2.2 Paragraph- and Page-Breaking Trace 6
- 3.3 Hyphenation 10
- 3.4 Disable/Break Ligatures 13
- 3.5 Manual Italic Correction 14
- 3.6 Apply Extra Kerning 15
 - 3.6.1 Slash 16
 - 3.6.2 Hyphen 16
- 3.7 Raise Selected Characters 17
 - 3.7.1 Capital Hyphen 17
 - 3.7.2 Capital Dash 18
 - 3.7.3 Number Dash (Figure Dash) 18
 - 3.7.4 Multiplication Sign 19
 - 3.7.5 Guillemets 19
- 3.8 Vertically Adjust Label Items 21
- 3.9 Align Last Line 26
- 3.10 Fill Last Line 26
 - 3.10.1 Problem Definition 26
 - 3.10.2 Manual Changes 28
 - 3.10.3 Multi-Purpose Environments 29
 - 3.10.4 Specialized Environments 30

Table of Contents continued on next page.

- 3.11 Spacing [30](#)
 - 3.11.1 Looser/Tighter [30](#)
 - 3.11.2 Wide Space [32](#)
 - 3.11.3 Narrow Space [33](#)
- 3.12 Microtype Front-End [33](#)
 - 3.12.1 Tracking [34](#)
 - 3.12.2 Font Expansion [35](#)
 - 3.12.3 Character Protrusion [36](#)
- 3.13 Sloppy Paragraphs [37](#)
- 3.14 Vertically Partially-Tied Paragraphs [38](#)
- 3.15 Breakable Displayed Equations [41](#)
- 3.16 Setspace Front-End [43](#)
- 3.17 Smooth Ragged [46](#)

4

Other Packages for Fine L^AT_EX Typography [50](#)

A

Package Code [51](#)

- | | |
|--|--|
| A.1 Setup and Reconfiguration 55 | A.10 Fill Last Line 72 |
| A.2 Information 56 | A.11 Spacing 73 |
| A.3 Hyphenation 58 | A.12 Microtype Front-End 75 |
| A.4 Disable/Break Ligatures 59 | A.13 Sloppy Paragraphs 78 |
| A.5 Manual Italic Correction 59 | A.14 Vert. Tie Paragraphs 80 |
| A.6 Apply Extra Kerning 60 | A.15 Breakable Disp. Eqs. 85 |
| A.7 Raise Selected Characters 62 | A.16 Setspace Front-End 85 |
| A.8 Vert. Adjust Label Items 65 | A.17 Smooth Ragged 89 |
| A.9 Align Last Line 71 | |

B

typog-grep [95](#)

Change History [102](#)

References [103](#)

Index [106](#)

List of Tables

1	Hyphens and automatic hyphenation	11
2	Suggested raise amounts for <code>\figuredash</code>	19
3	Suggested raise amounts for guillemets	20
4	Label item adjustment suggestions	25
5	Spacing changes made by <code>loosespacing</code>	31
6	Spacing changes made by <code>tightspacing</code>	31
7	<code>\fontdimen<number></code> parameters	32
8	Comparison of some space sizes	34
9	Shrink values of <code>setfontshrink</code>	35
10	Stretch values of <code>setfontstretch</code>	35
11	Shrink and stretch values of <code>setfontexpand</code>	36
12	Parameter adjustments of <code>\slightlyloppy</code>	37
13	Partial paragraph line counts	41
14	Env. <code>breakabledisplay</code> and <code>\interdisplaylinepenalty</code>	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.

1.1 Overview

Package `typog` focuses on (micro-)typographic improvements.

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

Section 3.3 expands the hyphenation facilities of L^AT_EX.

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

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

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

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

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

Section 3.14 presents several special functions to avoid club or widow lines in a paragraph.

As a simple extension of displayed mathematical equations we define a breakable variant in Sec. 3.15.

Section 3.16 introduces the `setspace` front-end.

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

1.2 Prerequisites

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

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

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

2 Package Options

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

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

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

\usepackage[⟨OPTION⟩...]{typog}
```

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

`breakpenalty=⟨penalty⟩`

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

`debug, nodebug`

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

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

`ligaturekern=⟨dim⟩`

Set `⟨dim⟩` of the kern that is inserted to split a ligature in macro `\nolig`. See Sec. 3.4. Default value: $\frac{3}{1000}$ em.

`lowercaselabelitemadjustments={⟨dim1⟩,⟨dim2⟩,⟨dim3⟩,⟨dim4⟩}`

Vertical shifts `⟨dimN⟩` to apply to `\labelitem⟨N⟩`, where `⟨N⟩` is the nesting level of the `itemize` list starting at one. An empty `⟨dimN⟩` is equivalent to 0 pt. The adjustments apply to the lowercase setting (`\lowercaseadjustlabelitems`). See Sec. 3.8 (in particular subsection `»Setup«` and Tab. 4 on p. 25) and also configuration option `uppercaseadjustlabelitem`.

All four lengths default to 0 pt.

Important

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

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

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

SINCE V0.4

`mathitalicscorrection=<dim>`

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

`raise*=<dim>`

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

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

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

`raisecapitaldash=<dim>`

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

`raisecapitalhyphen=<dim>`

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

`raisecapitaltimes=<dim>`

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

`raisecapitalguillemets=<dim>`

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

`raiseguillemets=<dim>`

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

`raisefiguredash=<dim>`

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

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

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

Set the three limits, given in $\frac{1}{1000}$ em, of shrinkability and stretchability for the respective levels. They are used in `setfontshrink` (shrinklimits triple only), `setfontstretch` (stretchlimits triple only), and `setfontexpand` (both triples of limits). See Sec. 3.12.2.

New `<limit-#>` values replace old ones. If one or more limits of the triple should remain unchanged pass a `*` 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.

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

`slashkern=<dim>`

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

`textitalicscorrection=<dim>`

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

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

Set the outer spacing of all typewriter fonts if used in environment `set-tracking` as described in Sec. 3.12.1.

The argument `<outer-spacing>` gets passed to microtype's `\SetTracking` option `outer spacing` [21, Sec. 5.3]. If it contains commas, enclose the whole argument in curly braces. Default argument value: 300, 90, 60.

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

By default this option is unset.

`uppercaselabelitemadjustments={<dim1>, <dim2>, <dim3>, <dim4>}`

SINCE V0.4

Vertical shifts `<dimN>` to apply to `\labelitem<N>`, where `<N>` is the nesting level of the `itemize` list starting at one. An empty `<dimN>` is equivalent to 0 pt. The adjustments apply to the uppercase setting (`\uppercaseadjustlabelitems`). See Sec. 3.8 (in particular subsection ›Setup‹ and Tab. 4 on p. 25) and also configuration option `lowercaseadjustlabelitem`.

All four lengths default to 0 pt.

Important

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

3 Macros and Environments

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

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

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

3.1 Setup and Reconfiguration

`typogsetup (env.)`

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

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

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

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

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

Note

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

Use Cases

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

`\typogget`

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

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

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

Use Case

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

```

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

```

The latter only is useful inside of an `itemize` environment of course. Compare with the solution in Sec. 3.8 offered by `typog` since v0.4. ■

3.2 Information

Never forget: The visual output counts;
it must always be checked, [...].
— UDO WERMUTH [28]

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

We define some functions for introspection of the typesetting process.

3.2.1 Font Information

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

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

The call to `\fontsizeinfo` introduces a pair of macros to access the stored values. The unstarred version `\cs-name` expands to the lengths including their units (i. e., pt), the starred version `\cs-name*` omits the units. The separating slash is `\kernedslash`, which is introduced in Sec. 3.6.1.

Note

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

Use Cases

Colophon. ¶ Font test pages. ■

3.2.2 Paragraph- and Page-Breaking Trace

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

`typoginspectpar (env.)`

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

³ The line spacing simply is `\baselineskip`.

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

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

Option

tracingboxes[= $\langle size \rangle$]

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

Caution

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

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

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

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

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

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

or

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

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

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

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

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

Tips

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

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

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

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

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

to the respective command-line.

- If more trace information is needed just add `\tracing...` calls right after `\begin{typoginspect}` or `\begin{typoginspectpar}`.
- As the overhead of `\typoginspect` is relatively low, hairy parts of a document can permanently be furnished with them, for example, the Index.
- Any labeled part can treat their ids to `<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 [14, 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 [14, p. 99]:

$0 \mapsto \text{very loose}, \quad 1 \mapsto \text{loose}, \quad 2 \mapsto \text{decent}, \text{ and } \quad 3 \mapsto \text{tight}.$

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.

⁵ Reference 27 provides an exceptionally detailed discussion of the output of `\tracingparagraphs`.

4. The following two values only show up if `\lastlinefit ≠ 0`:
 - (a) The shortfall `␣` and
 - (b) glue `␣` or `g`.⁶
5. The best⁷ way to get here, i. e., `@@17` is via `␣->` breakpoint `@@ 16`. ■

Note

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

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

or

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

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

Investigating page-breaks. Use `\tracingpages=1` or the [typoginspect](#) environment to switch on tracing of T_EX's page-break information [14, 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.

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

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

8 See also the discussion of the T_EX output routines by SOLOMON [23].

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

$\text{T}_{\text{E}}\text{X}$'s and thus $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$'s hyphenation algorithm is highly sophisticated, yet the document author sometimes lacks convenient macros to solve seemingly trivial typographic tasks. For example, to hyphenate a compound word connected by a hyphen.

`\allowhyphenation`

$\text{T}_{\text{E}}\text{X}$ inhibits breaks of the component words by default. The following macro rectifies the problem.

```
\allowhyphenation
```

Macro `\allowhyphenation` re-enables automatic hyphenation after $\text{T}_{\text{E}}\text{X}$ has turned it off, for example, in the innocuous case of a hyphenated compound.

The admittedly simple rules when $\text{T}_{\text{E}}\text{X}$ auto-hyphenates and when not give rise to so many different, yet interesting cases that we devote Tab. 1 to them. The seemingly special cases shown there are not that uncommon, e. g., consider ›spin-½‹ 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. 11. ¶

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

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

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

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

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

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

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

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

A common trick to sweet-talk $\text{T}_{\text{E}}\text{X}$ into hyphenating the first word of a paragraph is to put `\hskip0pt` in front of it. ■

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

⁹ `\babelhyphenation` is the multi-lingual extension of $\text{T}_{\text{E}}\text{X}$'s `\hyphenation` and it is defined in package `babel` [5].

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

\LaTeX -Code	Result	Note
hyphenated-compound	hyphenated- compound	Most frequently used code; the <code>hyphen</code> expands to <code>\discretionary{-}{-}{-}</code> rendering the parts un-breakable
hyphenated\mbox{-}% compound	hyphenated-compound	Suppress hyphenation with the <code>\mbox</code> in the compound
\mbox{hyphenated-% compound}	hyphenated-compound	Avoid line break and thus hyphenation
hyphenated\hyp compound	hy- phen- ated- com- pound	Macro <code>\hyp</code> defined in package <code>hyphenat</code> [34]
hyphenated% \allowhyphenation-% compound	hy- phen- ated- compound	Macro <code>\allowhyphenation</code> of package <code>typog</code> ; only unblock hyphenation of the first part
hyphenated-% \allowhyphenation compound	hyphenated- com- pound	Macro <code>\allowhyphenation</code> of package <code>typog</code> ; only unblock hyphenation of the second part
hyphenated% \allowhyphenation \mbox{-}% compound	hy- phen- ated-compound	Macro <code>\allowhyphenation</code> of package <code>typog</code> ; hyphenate first part and keep the original <code>hyphen</code> unbreakable
hyphenated% \allowhyphenation-% \allowhyphenation compound	hy- phen- ated- com- pound	Macro <code>\allowhyphenation</code> of package <code>typog</code> ; hyphenate both parts, similar to <code>\hyp</code> shown above

times `\-` actually *is* the way to go.

Let us assume we mark up proper names with

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

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

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

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

\TeX never hyphenates the initial word in a paragraph and `\allowhyphenation` cannot help in this case. Start the paragraph with `\hskip 0pt` to enable hyphenation even for the first word.

Tip — Typewriter Fonts

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

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

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

`\breakpoint`
`\breakpoint*`

The empty discretionary construct [14, p. 95], `\discretionary{}{}{}{}`, is so helpful that it deserves its own macro – with a descriptive name.

```
\breakpoint
\breakpoint*
```

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

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

Use Case

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

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

`hyphenmin (env.)`
SINCE v0.3

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

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

environment.

```
\begin{hyphenmin}[\langle left-hyphen-minimum \rangle]{\langle hyphen-minimum \rangle}
...
\end{hyphenmin}
```

Without optional argument `hyphenmin` sets both `\lefthyphenmin` and `\righthyphenmin` to `\langle hyphen-minimum \rangle`. When called with an optional argument it sets `\lefthyphenmin` to `\langle left-hyphen-minimum \rangle` and `\righthyphenmin` to `\langle hyphen-minimum \rangle`.¹¹

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 `\langle hyphen-minimum \rangle` to disable hyphenation. ■

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 `{}`.

¹¹ The current values for `\lefthyphenmin` and `\righthyphenmin` in this document are 2 and 3, respectively.

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

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

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

Alternatively use `\texorpdfstring` [20, Sec. 4.1.2, p. 22]. ■

Use Cases

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

3.5 Manual Italic Correction

`\itcorr`
`\itcorr*`

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

```
\itcorr{\langle strength \rangle}
\itcorr*{\langle strength \rangle}
```

In text mode macro `\itcorr` inserts a kern whose width is proportional to `\fontdim1`, which is the font's italic correction. If `\fontdim1` happens to be zero (e.g. for an upright font), `\itcorr` uses the value set with `textitalicscorrection` instead of `\fontdim1`. The starred version always uses `textitalicscorrection`. In math mode macro `\itcorr` uses the value set with `mathitalicscorrection`¹² in both the starred and the unstarred form.

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

Use Cases

Stronger or weaker correction than `\/.` ¶ Correct a non-slanted or non-italicized font. ¶ Negative correction at the left-hand side¹³ of italics, i. e., compensate »shift-to-the-right effect« of italics. ¶ Positive correction at the left-hand side of italics, e.g., an opening parenthesis or square bracket followed by an italic `f` (before: 8, after: 7) or `y` (before: 4, after: 1) reaching far to the left below the baseline. ■

¹² Separate adjustments may be desirable if the math font's italics have markedly different slants.

¹³ 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 `\,`.

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

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

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

Correction of the right-hand side and $\alpha > 0$: A reasonable first guess of $\langle strength \rangle$ 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 $\langle strength \rangle$ and α are related. Moreover, it demonstrates how intricate italics correction is.

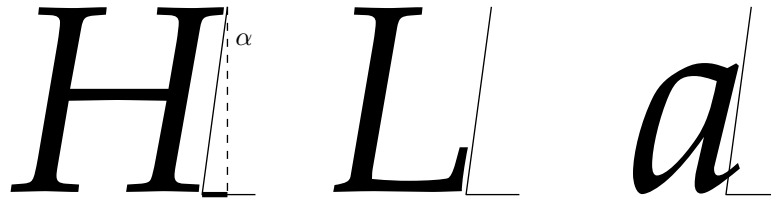


FIGURE 1: Some letters of an italics font. We use the capital $\text{\textit{H}}$ to measure the angle α between the plumb-line (drawn dashed) and a tangent to the rightmost parts of the glyph. The length of the plumb-line is proportional to $\langle strength \rangle$ and the short, thick part of the baseline symbolizes the resulting italics correction. ¶ The middle example, the capital $\text{\textit{L}}$, shares α with $\text{\textit{H}}$ but obviously needs a far smaller $\langle strength \rangle$ or even no correction at all. ¶ The $\text{\textit{a}}$ 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 $\text{\textit{a}}$ a larger α – despite being a member of the same font – but its serif adds as much to the width as the slanted stem.

3.6 Apply Extra Kerning

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

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

3.6.1 Slash

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

```
\kernedslash
\kernedslash*
```

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

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

Use Cases

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

3.6.2 Hyphen

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

```
\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` [34]) with `\kernedhyphen` and apply some kerning to left and to the right of it. The values `<left-kerning>` and `<right-kerning>` are multiplied with one thousandth of the current font's em to get the size of the kern.

The optional argument `<raise>`, also given in $\frac{1}{1000}$ em, allows to adjust the height of the hyphen similar to the macros described in Sec. 3.7. In text mode the special argument `*` for `<raise>` transfers the current value of `raisecapital-hyphen`. The default for `<raise>` is zero.

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

```
\leftkernedhyphen[<raise>]{<left-kerning>}
\leftkernedhyphen*[<raise>]{<left-kerning>}
\rightkernedhyphen[<raise>]{<right-kerning>}
\rightkernedhyphen*[<raise>]{<right-kerning>}
```

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

Use Cases

Composites in the form $\langle\textit{math}\rangle\text{-}\langle\textit{noun}\rangle$ in languages where nouns are capitalized. ¶
 Composites where one or both sides of the hyphen are typeset in different fonts, like, $\langle\textit{small-caps}\rangle\text{-}\langle\textit{roman}\rangle$. ■

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 $\backslash\text{labelitem}\langle N\rangle$ markers, bullets, stars, and even fancy dingbats. If these hyphens and dashes connect uppercase letters (or lining numerals) they sometimes appear to low; they disrespect the glyphs' symmetry axis. A similar situation arises if `itemize` list markers precede an uppercase letter, a lining numeral, or a big mathematical operator.

We introduce a set of macros for the most common cases that typeset these characters at a user definable, adjusted height above the baseline. Users can base their own definitions of raised characters on their associated dimensions.¹⁴

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

$\backslash\text{capitalhyphen}$
 $\backslash\text{capitalhyphen}^*$

In many fonts the height of the hyphen character ̣^2 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.

```
 $\backslash\text{capitalhyphen}$ 
 $\backslash\text{capitalhyphen}^*$ 
```

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 $\backslash\text{capitalhyphen}$ is determined by `raisecapitalhyphen`.

Use Cases

In languages that capitalize their nouns, the typical use-case is between an $\langle\textit{abbreviation}\rangle$ and a $\langle\textit{noun}\rangle$ when $\langle\textit{abbreviation}\rangle$ is a string of uppercase letters. The same holds true for a connection of an uppercase variable in mathematical mode and a $\langle\textit{noun}\rangle$ 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. ■

¹⁴ Also compare with Ex. 12 in Ref. 33 for an attempt to automate vertical alignment.

3.7.2 Capital Dash

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

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

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

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

Use Cases

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

`\capitalemdash`
`\capitalemdash*`

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

```
\capitalemdash
\capitalemdash*
```

Use Cases

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

3.7.3 Number Dash (Figure Dash)

`\figuredash`
`\figuredash*`

The en-dash often gets used as separator for numerical ranges. In most fonts it has the correct height above baseline for oldstyle numerals, e. g. 12–34–56–78, but with lining numerals – depending on the font – it may look like it suffers from »broken suspenders«: 12–34–56–78. The situation is similar to `\capitaldash` and `\capitalhyphen` discussed in Secs. 3.7.1 and 3.7.2.

```
\figuredash
\figuredash*
```

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

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

Other macros may be redefined with `\figuredash` for a consistent appearance of the copy, like, for example, `\citedash` (package `cite [3]`), or `\crefrangeconjunction` (package `cleveref [10]`).

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

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

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

* Free Bembo.

Use Case

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

3.7.4 Multiplication Sign – Times ^x

`\capitaltimes`

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

`\capitaltimes`

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

Use Case

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

3.7.5 Guillemets

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

For the implementation typog relies on the T1¹⁵ font encoding not on package babel.

¹⁵ Font encoding T1 can be forced via `\usepackage[T1]{fontenc}` in the document preamble.

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

Lowercase Versions.

<code>\singleguillemetleft</code>	<code>\singleguillemetright</code>
<code>\doubleguillemetleft</code>	<code>\doubleguillemetright</code>

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.

<code>\Singleguillemetleft</code>	<code>\Singleguillemetright</code>
<code>\Doubleguillemetleft</code>	<code>\Doubleguillemetright</code>

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

TABLE 3: Suggested values for raising guillemets of some selected fonts in multiples of 1 em.

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

Tip

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

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

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

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

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

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

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

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

Use Case

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

Anticipated Changes & Possible Extensions

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

3.8 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.¹⁶

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

ALL THREE SINCE V0.4

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

¹⁶ The exception being mathematics typeset as text via package `mathastext` [7].

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

The argument $\langle levels-to-adjust \rangle$ is a – possibly empty – comma separated list of the levels the respective adjustments are to be applied to. The levels themselves are given as *decimal* numbers, this is, 1, 2, 3, 4 or the special value \ast 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{\ast}
```

- ▷ 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}
{uppercase-labelitemadjustments={.1em}}
\renewcommand*\labelitemi{{\small$\rhd$}}
\begin{itemize}
\uppercaseadjustlabelitems{}
\item ...
\end{itemize}
\end{typogsetup}
```

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

Setup. To assist the user in finding the desired adjustments of the labels of `typog` provides macros that help setting up `lowercase label item adjustments` and `uppercase label item adjustments`. 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 $\frac{1}{2}$.¹⁷ The then correct height gets indicated by a thin horizontal line parallel to the baseline. Thus, at sufficiently high magnifications it is possible to judge whether a label gets typeset too high or too low with respect to this reference line.

Note

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

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

`\typogadjuststairs`

SINCE V0.4

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

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

Generate stairs of `⟨number-of-steps⟩` vertically shifted label items; use the next odd number, if `⟨number-of-steps⟩` is even. Draw a reference hairline at `⟨scale-factor⟩` times the height of `⟨sample⟩`, where `⟨scale-factor⟩` defaults to `.5`. The stairs start at a vertical shift of

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

and repeat up

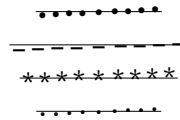
$$\frac{\langle \text{number-of-steps} \rangle - 1}{2} \times \langle \text{step-size} \rangle.$$

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

Example

Play ball!

¹⁷ The default factor of `.5` hearkens back to STRIZVER's suggestion that ›[b]ullets should be centered on either the cap height or x-height of the neighboring text, ...‹ [24, p. 220].



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

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

BOTH SINCE V0.4

```
\typoguppercaseadjustcheck[<scale-factor>=.5]{<sample>}
\typoglowercaseadjustcheck[<scale-factor>=.5]{<sample>}
```

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

Example

Uppercase check with `\typoguppercaseadjustcheck{ABCXYZ}`:

ABGH•—•QWYZ, 0123•—•4567

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

ace•—•mno, bdf•—•gkj, 0123•—•4567,

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

In Table 4 on p. 25 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 •, —, *, and ., respectively.

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

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

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

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

‡ In package kpfnts.

§ Contained in package mathpazo.

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

3.9 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` [22]) are in effect. For an instructive discussion consult Ch. 17, »Paragraph End«, of Ref. 11. The following environments adjust the last lines of paragraphs in different ways.

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

`lastlineraggedleftpar`
(env.)
`lastlineflushrightpar`
(env.)

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

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

Center the last lines of the paragraphs enclosed by this environment.¹⁸

`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. ■

3.10 Fill Last Line of a Paragraph

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

3.10.1 Problem Definition

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

1. `\parindent > 0` [30, O1]

¹⁸ Also compare the approach taken in Ref. 30.

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



The same problem arises with displayed math in a flush-left¹⁹ setting, e. g., `amsmath` [2] and option `fleqn`.²⁰

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



2. `\par indent = 0` [30, 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.²¹



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



The suggestions for the gap-width vary from two em to twice the width of a ›typical‹ `\par indent`²² for the gap [8].

Tip

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

¹⁹ The common practice of centering displayed equations does not call for the manipulations of a paragraph's last line discussed here.

²⁰ For displayed equations and `amsmath` the relevant parameter is `\math indent`.

²¹ Package `parskip` defines `\parskip` as 6 pt plus 2 pt for a base size of 10 pt.

²² For example, L^AT_EX's class `article` uses a `\par indent` of 25 pt.

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

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

Important

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

- Technically, the paragraphs need to contain enough glue (see for example Sec. 3.13) to achieve a low badness such that the desired paragraph end is deemed feasible by \TeX .
- Aesthetically, the paragraphs must be long enough to absorb the change in last-line fill level otherwise their gray-values visibly deviate from the average. ■

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

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`.

(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 [\[32\]](#). In turn the next-to-last lines may become unsightly. Counteract this degradation e. g. with recipes [2a](#) to [2c](#).

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

2. Uniform paragraph change.

(a) Vary spacing.

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

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

(b) Vary font tracking.

Enclose the paragraph in a `setfonttracking` group. See Sec. 3.12.1. Increase or decrease the tracking in steps of $\frac{1}{1000}$ em until the last line looks good.

(c) Vary font expansion.

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

3. A combination of any of the above items.

4. Some curveballs.

(a) If the paragraph already suffers from one of the problems that \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.*)

`prolongpar` (*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 line²³ or a club line²⁴ [14, p. 104 and 18]. (See also Sec. 3.14 for special functions to avoid clubs or widows.) ›Accidentally‹, they also change the length of the last line of the paragraph.

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

Environment `shortenpar` decreases the `\looseness` of the paragraph.²⁵ It performs well if the last line of the paragraph is short or the whole paragraph is loose.

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

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

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

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

25 Command `\looseness` is a \TeX primitive [14, p. 103n]. A thorough discussion of the interaction of `\linepenalty` and `\looseness` can be found in Ref. 29.

3.10.4 Specialized Environments

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

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

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

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

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

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

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

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

3.11 Spacing

90% of design is typography.
And the other 90% is whitespace.
— JEFFREY ZELDMAN

The functions described in this section rely only on plain \LaTeX . 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 [[17](#), p. 119]

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

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

`loosespacing (env.)`

`tightspacing (env.)`

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

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

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

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

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

The default level of `loosespacing` is 1.

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

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

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

TABLE 6: Adjustments made by environment `tightspacing` to `\spaceskip`. The mapping of $\langle level \rangle$ to the exact skip definitions are $1 \mapsto .9875^{+.0125}_{-.5}$, $2 \mapsto .975^{+.025}_{-.5}$, $3 \mapsto .95^{+.05}_{-.5}$, and $\geq 4 \mapsto .9^{+.1}_{-.5}$, where all factors scale with `\dimen2`, the current font's space-width.

The default level of `tightspacing` is 1.

Note

At a given $\langle level \rangle$ the changes of `loosespacing` are much larger than those of `tightspacing`. ■

Use Cases

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

3.11.2 Wide Space

The `\widespace` macro and its companion `\narrowospace` derive their appearances from several of the current font's `\fontdimen⟨number⟩`s. \TeX addresses the latter by integers, which is totally non-memnonic. Therefore, we play softball by first presenting Tab. 7 that associates the `\fontdimen⟨number⟩`s with their meanings and also reports on their current values (for this document).²⁶

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

TABLE 7: All \TeX font parameters normalized to the font's quad-size. The first column `#` states the index of the `\fontdimen` parameter: `⟨number⟩`. Column 2 presents short descriptions of `\fontdimen⟨number⟩`. As examples, the values for the current font are shown in column 3.

`\widespace`
`\widespace*`

STARRED FORM SINCE V0.2

Typeset a wide, sentence-ending space as if in `\nonfrenchspacing` mode. Consult Table 8 for a comparison of the various sizes.

`\widespace`
`\widespace*`

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

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

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

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

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

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

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

²⁶ The association is given in Appendix F (p. 433) of Ref. 14. For a concise and understandable explanation of the \TeX `\fontdimen` parameters consult Ref. 9.

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

Use Case

Useful as a sentence-ending space if, for example, the sentence ends in an abbreviation with a period or decimal number without trailing digits *and* the next sentence should be delimited in a clearer way. ¶ Open tight lines with a series of `\widespaces`.²⁷ ■

3.11.3 Narrow Space

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

`\narrowospace*`
SINCE V0.2

```
\narrowospace
\narrowospace*
```

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

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

if `\fontdimen7` is different from zero or otherwise

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

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

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

Use Case

Tighten loose lines with a series of `\narrowospaces`.²⁸ ■

3.12 Microtype Front-End

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

Important

All macros and environments introduced in this section require that package `microtype` [21] has been loaded, preferably *before* package `typog`

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

in the document preamble. ■

²⁷ See also »Investigating the badness of a paragraph« on Page 8.

²⁸ Footnote 27 again applies.

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

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

3.12.1 Tracking

Caution

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

`setfonttracking` (*env.*)

Override the default tracking for all fonts.

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

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

See Sec. 5.3, »Tracking«, and 7, »Letterspacing revisited«, in the documentation of `microtype` [21] for a detailed explanation.

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

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

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

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

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

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

Use Cases

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

3.12.2 Font Expansion

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

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

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

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

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

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

TABLE 10: Preconfigured values for stretch inside of environment setfontstretch as $\frac{1}{1000}$ em. Note that all shrink values are zero, so the fonts only can stretch.

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

Use Cases

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

setfontexpand (env.) Manipulate both, stretch and shrink values at the same time.

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

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

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

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

Notes

- Environment `setfontexpand` shares its `shrinklimits` with `setfontshrink` and its `stretchlimits` with `setfontstretch`.
 - These environments do not nail down any font’s expansion but only set up its available range. See Sec. 3.3, »Font Expansion«, in the microtype documentation [21].
- Moreover, a text may not »respond« neither to `setfontshrink`, `setfontstretch`, nor `setfontexpand` because \TeX already considers it optimal without expansion or within the previous expansion limits, e. g., those set at microtype load time as opposed to typog’s load time. ■

Use Cases

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

`nofontexpansion` (*env.*)

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

```
\begin{nofontexpansion} ... \end{nofontexpansion}
nofontexpand (alias)
```

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

Use Cases

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

3.12.3 Character Protrusion

`nocharprotrusion` (*env.*)

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

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

Use Cases

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

3.13 Sloppy Paragraphs

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

We define the macro `\slightlyloppy` and the associated environment, `slightlyloppypar`, with a user-selectable $\langle sloppiness \rangle$ parameter. The constructions recover the known settings `\fussy` ($\langle sloppiness \rangle = 0$) and `\sloppy` ($\langle sloppiness \rangle \geq 8$), and introduce seven intermediate $\langle sloppiness \rangle$ levels.²⁹ The default $\langle sloppiness \rangle$ is 1.

`\slightlyloppy`
`slightlyloppypar`
 (env.)

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

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

TABLE 12: Adjustments made by `\slightlyloppy` to various T_EX parameters at different levels of $\langle sloppiness \rangle$.

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

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

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

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

to prevent excessive stretchability in narrow lines.

Environment `slightlyloppypar[\langle sloppiness \rangle]` mimics L^AT_EX's `sloppy-`

²⁹ Also compare the findings for `\emergencystretch` in Ref. 28.

par, while offering the flexibility of `\slightlyloppy`.

Use Cases

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

3.14 Vertically Partially-Tied Paragraphs

L^AT_EX provides several macros and environments to tie material vertically – most prominently `samepage` and `minipage`.³⁰ Typog’s macros and environments constitute more sophisticated but weaker forms of these. They tie only the first or last couple of lines in a paragraph while the rest of the paragraph gets broken into pages by T_EX in the usual way.

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

Avoid a club line in each partial paragraph.

`\vtietop`

`vtietoppar (env.)`

```
\vtietop[⟨number-of-lines⟩]
\begin{vtietoppar}[⟨number-of-lines⟩] ... \end{vtietoppar}
```

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

Use Cases

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

`\splicevtietop`

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

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

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

Example for a description list and plain L^AT_EX:

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

```
\end{description}
```

³⁰ A valuable complement to these is package `needspace` [36] which takes a different approach and reliably works in *mixed* horizontal and vertical mode situations.

Alternatively with package `enumitem` [4]:

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

or shorter and with the default $\langle number-of-lines \rangle$, 3, using the `enumitem` style³¹ `vtietop`:

`vtietop` (*enumitem* key)

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

`\vtiebot`

Avoid a widow line in each partial paragraph.

`vtiebotpar` (*env.*)

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

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

`vtiebotdisp` (*env.*)

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

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

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

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

`vtiebotdisptoppar`
(*env.*)

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

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

³¹ The documentation of `enumitem` prosaically calls them ›keys‹ (Section 3) not ›styles‹.

Vertically tie the last $\langle\textit{before-disp-number-of-lines}\rangle$ in the paragraph before a display and the first $\langle\textit{after-disp-number-of-lines}\rangle$ in the paragraph after the display. Moreover, turn the paragraphs and the display into an un-breakable unit.³²

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

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

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

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

See also Sec. 3.10.3 for other methods to avoid club or widow lines.

Partial Paragraphs And Counting Lines. The top-of-paragraph ties, `\vtietop` and `vtietoppar` count $\langle\textit{number-of-lines}\rangle$ from the beginning of every partial paragraph. Each displayed math in the paragraph resets the count. The bottom-paragraph ties, `\vtiebot`, `vtiebotpar`, `\vtiebotdisp`, and `vtiebotdisppar` count backward from the end of each partial paragraph. Again, each displayed math in the paragraph resets the count. According to T_EX's rules, a displayed math formula always is counted as *three* lines no matter its contents. Table 13 summarizes these rules with the help of an example.

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`³³ or `\shortintertext`³⁴ it is desirable to make the display breakable. This is achieved with `\allowdisplaybreaks` or the environment `breakabledisplay` which will be described in Sec. 3.15. ■

Use Cases

Fix widows and orphans, e. g., those turned up by package `widows-and-orphans` [19]. ¶
Extend the typographic convention of »three to four lines instead of a single club or widow line« to a context-dependent number of lines that tries to keep all (well, dream on) the information together the reader needs at that particular point. ■

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

33 Introduced in package `amsmath` [2].

34 Defined in package `mathtools` [12].

TABLE 13: Exemplary, eight-line paragraph compounded of two partial paragraphs of three and two lines and a displayed math formula of arbitrary size sandwiched in between.

Continuous Line Number	Example Contents	$\backslash\text{vtietop}^\dagger$ Count	$\backslash\text{vtiebot}^\ddagger$ Count
1	Text line 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 ϵ -TeX's counting scheme of $\backslash\text{clubpenalties}$; it also holds for vtietoppar .

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

3.15 Breakable Displayed Equations

`breakabledisplay`
(*env.*)

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

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

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

Tips

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

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

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

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

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

Use Cases

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

3.16 Setspace Front-End

Package `setspace` [25] is a base hit when it comes to consistently setting the line skip for a document via the macro `\setstretch`. The interface of `\setstretch` though is unintuitive as it asks for an obscure factor. The L^AT_EX 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.³⁵ This is where the following macros go to bat.

Important

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

```
\usepackage{setspace}
```

in the document preamble. ■

`\setbaselineskip`
SINCE V0.3

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

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

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

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

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

Example

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

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

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

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

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

`\resetbaselineskip`
SINCE V0.3

Reset the `\baselineskip` to its original value.

```
\resetbaselineskip
```

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

`\setbaselineskippercentage`
SINCE V0.3

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

³⁵ To find out about the current font's size and the `\baselineskip` in printable form check out Sec. 3.2.1 on p. 6.

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

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

Example

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

```
\setbaselineskippercentage{125}
```

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

`\setleading`
SINCE V0.3

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

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

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

Example

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

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

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

`\setleadingpercentage`
SINCE V0.3

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

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

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

Example

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

```
\setleadingpercentage{25}
```

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

`\typogfontsize`
(dimen)
SINCE V0.3

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

The length `\typogfontsize` gets initialized at the end of the preamble to the default font's quad size:³⁶

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

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

36 For an overview of the various `\fontdimen<number>` parameters consult Tab. 7 on p. 32.

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

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

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

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

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

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

or

```
\setleadingpercentage{0}
```

Tip

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

Here are some forms that do work:

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

3.17 Smooth Ragged

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

Package `typog` implements a novel approach to typeset ragged paragraphs. Instead of setting the glue inside of a paragraph to zero and letting the line-widths vary accordingly [31] we prescribe the line-widths with $\text{T}_{\text{E}}\text{X}$'s `\parshape` primitive and leave alone the stretchability or shrinkability of the glue.

Caution

None of the following environments work inside of lists. ■

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

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

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

Options

leftskip= $\langle dim \rangle$

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

parindent= $\langle dim \rangle$

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

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

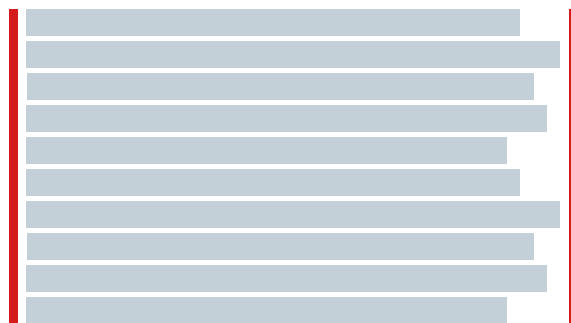
```
\begin{smoothrowrightpar}[\langle option... \rangle]
...
\end{smoothrowrightpar}
```

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

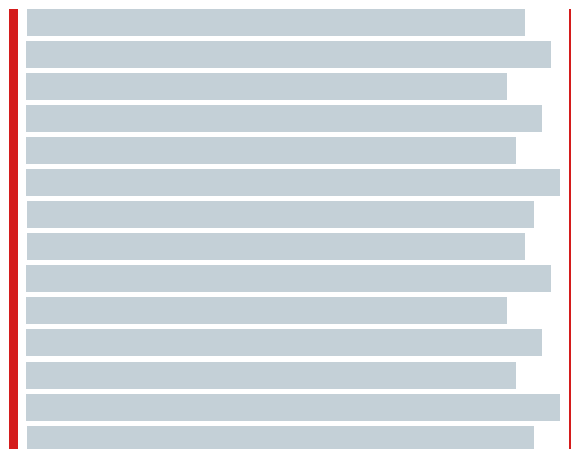
- The triplet generator repeats a *short line – long line – middle-length line* sequence. Shown below are two complete cycles.



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



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



smoothrowright
(env.)

Environment smoothrowright is the multi-paragraph version of smoothrow-

`raggedrightpar`. It takes the same optional arguments.

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

Options

`\linewidth=` $\langle dim \rangle$

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

Global Parameters

`\smoothraggedrightfuzzfactor=` $\langle factor \rangle$

The environment adds glue to every line-width³⁷ to achieve a more convincing »ragged appearance« and to reduce the number of overfull lines. The algorithm divides the smooth margin into 3, 5, or 7 parts depending on the chosen `\smoothraggedrightgenerator` (see below). The `\smoothraggedrightfuzzfactor` is the amount of glue of each line expressed as a multiple of the distance between the division points. The default of 1.0 means to add as much glue such that the lines just do not overlap (assuming justification is feasible).

`\smoothraggedrightgenerator`

Select a generator to use. Valid generator names:

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

The default generator is `triplet`.

`\smoothraggedrightleftskip=` $\langle dim \rangle$

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

`\smoothraggedrightparindent=` $\langle dim \rangle$

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

`\smoothraggedrightragwidth=` $\langle dim \rangle$

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

³⁷ The shortest line only gets stretchability, the longest only receives shrinkability. All other lines are both stretchable and shrinkable.

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

Use Cases

Replacement for `\RaggedRight` [22]. ¶ Design alternative for fully justified paragraphs if used with a small rag-width. ■

Anticipated Changes & Possible Extensions

Translate the code to `l3galley` which is part of the `l3experimental` package [15]. Galley code is supposed to work inside of lists, too. ■

4 Other Packages for Fine L^AT_EX Typography

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

- | | |
|-----------|--|
| enumitem | Flexible and consistent definition of all basic L ^A T _E X-list types plus in-line lists [4]. |
| geometry | Powerful and sophisticated setup of the page layout [26]. Best accompanied by layout [16] to visualize the page geometries. |
| hyphenat | Hyphens that do not inhibit further auto-hyphenation of a compound word [34]. |
| microtype | Fine control of spacing, tracking, sidebearings, character protrusion into the margins, font expansion, and much more [21].
See Section 3.12 for a front-end to microtype offered by this package.
See also KHIRIVICH's discussion [13]. |
| ragged2e | Improved versions of environments <code>raggedleft</code> , <code>raggedright</code> , and <code>center</code> [22]. |
| setspace | Consistently set the line-spacing of a document, i. e., control <code>\baselineskip</code> [25].
See Section 3.16 for a front-end to setspace offered by this package. |

A Package Code

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

```

1%<*package>
2\NeedsTeXFormat{LaTeX2e}[2005/12/01]
3\ProvidesPackage{typog}
4          [2024/05/21 v0.4 TypoGraphic extensions]
5
6\RequirePackage{etoolbox}
7\RequirePackage{everyhook}
8\RequirePackage{xkeyval}
9

```

Declarations of Lengths, Skips, etc.

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

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

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

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

\iftypog@debug Our switch for debug information.

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

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

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

\typog@typeout Our debug information printer.

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

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

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

\typog@trim@spaces Pull \tl_trim_spaces into the ›classic‹ namespace.

```

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

```

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

```

23 \newcommand{\typog@register@pdfsubstitute}[1]{%
24   \AtBeginDocument{%
25     \ifdefined\pdfstringdefDisableCommands
26       \pdfstringdefDisableCommands{#1}%
27     \fi}}
28
```

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`.

`\typog@microtype@preloaded`

```

29 \newif\iftypog@microtype@preloaded
30
```

`\typog@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.

```

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

```

`\iftypog@microtype@loaded`

```

43 \newif\iftypog@microtype@loaded
44
```

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

```

45 \AtBeginDocument{
46   \ifdefined\MT@MT
47     \typog@typeout{package microtype loaded}%
48     \typog@microtype@loadedtrue
49     \def\typog@require@microtype{\relax}
50   \else
51     \typog@microtype@loadedfalse
52     \def\typog@require@microtype
53       {\PackageError{typog}%
54               {package microtype not loaded}%

```



```

55                                     {require package microtype before package ty-
    pog}}}%
56   \fi
57 }
58

```

Our own state...

`nfig@mathitalicscorrection`

```

59 \newmuskip\typog@config@mathitalicscorrection
60

```

Actual `\labelitem<N>` corrections.

`og@adjust@labelitemi (dimen)`

```

61 \newdimen{\typog@adjust@labelitemi}

```

`g@adjust@labelitemii (dimen)`

```

62 \newdimen{\typog@adjust@labelitemii}

```

`@adjust@labelitemiii (dimen)`

```

63 \newdimen{\typog@adjust@labelitemiii}

```

`g@adjust@labelitemiv (dimen)`

```

64 \newdimen{\typog@adjust@labelitemiv}

```

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

`lowercase@labelitemi (dimen)`

```

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

```

`owercase@labelitemii (dimen)`

```

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

```

`wercase@labelitemiii (dimen)`

```

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

```

`owercase@labelitemiv (dimen)`

```

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

```

`uppercase@labelitemi (dimen)`

```

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

```

`ppercase@labelitemii (dimen)`

```

70 \newdimen{\typog@adjust@uppercase@labelitemii}

```

```

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

ppercase@labelitemiv (dimen)
72 \newdimen{\typog@adjust@uppercase@labelitemiv}
73

    Other lengths...

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

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

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

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

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

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

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

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

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

\typog@config@breakpenalty
83 \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.
84 \newlength{\typog@dim@unit}
85 \setlength{\typog@dim@unit}{.001em}

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

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

```

```

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

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

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

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

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

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

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

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

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

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

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

```

A.1 Setup and Reconfiguration

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

```

99 \NewDocumentEnvironment{typogsetup}{m}
100   {\def\typog@@arg{#1}%
101     \ifx\typog@@arg\empty
102       \typog@initialize@options
103     \else
104       \setkeys{typog}{#1}%
105     \fi
106     \ignorespaces}
107   {\ignorespacesafterend}

```

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

```

108 \NewDocumentCommand{\typogget}{m}{\csname typog@config@#1\endcsname}
109

```

A.2 Information

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

```

110 \ExplSyntaxOn
111 \newcommand*{\typog@round@dim@to@tenths}[1]
112   {\fp_to_decimal:n {round(10 * \dim_to_fp:n{#1} / 1\p@) / 10}}
113 \ExplSyntaxOff
114

```

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

```

115 \newcommand*{\typog@formatsizeinfo}[3]
116   {#1#3\kernedslash #2#3}
117

```

`\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.

```

118 \NewDocumentCommand{\fontsizeinfo}{s m}
119   {\global\expandafter\edef\csname typog@fontsize@#2\endcsname
120     {\typog@round@dim@to@tenths{\fontdimen6\font}}}%
121   \global\expandafter\edef\csname typog@linespacing@#2\endcsname
122     {\typog@round@dim@to@tenths{\baselineskip}}}%
123   \protected\expandafter\gdef\csname #2\endcsname
124     {\@ifnextchar*{\typog@formatsizeinfo
125                   {\csname typog@fontsize@#2\endcsname}%
126                   {\csname typog@linespacing@#2\endcsname}%
127                   }% no unit
128                   \ignorespaces % eat spaces after star
129                   \@gobble} % consume the star itself
130   {\typog@formatsizeinfo
131     {\csname typog@fontsize@#2\endcsname}%
132     {\csname typog@linespacing@#2\endcsname}%
133     {\,pt}% decorative unit 'pt'
134   }}}
135

```

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

```

136 \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.

```

137 \newcounter{typog@inspect@count}
138

```

`typoginspect` (*env.*)

```

139 \define@key[typog]{typoginspect}{tracingboxes}[\maxdimen]%
140     {\def\typog@@typoginspect@tracingboxes{#1}}
141 \NewDocumentEnvironment{typoginspect}{0}{ m}
142     {\def\typog@@typoginspect@tracingboxes{\m@ne}%
143     \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.

```

144     \edef\typog@@arg{#2}%
145     \ifx\typog@@arg\empty
146         \stepcounter{typog@inspect@count}%
147         \edef\typog@@id{\typog@default@inspect@id@prefix
148             \arabic{typog@inspect@count}}%
149     \else
150         \edef\typog@@id{\typog@trim@spaces{\typog@@arg}}%
151     \fi
152     \typeout{<typog-inspect>\space
153         id="\typog@@id"\space
154         job="\jobname"\space
155         line="\the\inputlineno"\space
156         page="\the\value{page}">}%

```

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

```

157     \hbadness=\m@ne
158     \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.

```

159     \tracingnone
160     \tracingpages=\@ne
161     \tracingparagraphs=\@ne
162     \showboxbreadth=\typog@@typoginspect@tracingboxes
163     \showboxdepth=\typog@@typoginspect@tracingboxes}
164 {\typeout{</typog-inspect>}}%
165 \ignorespacesafterend}

```

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

```

166 \NewDocumentEnvironment{typoginspectpar}{m}
167     {\typoginspect{#1}}
168     {\par\endtypoginspect}
169

```

A.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`.

```

170 \newcommand*{\typog@allowhyphenation}
171   {\ifvmode
172     \relax
173   \else
174     \nobreak
175     \hskip\z@skip
176   \fi}
177

```

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

```

178 \unless\ifdefined\allowhyphenation
179   \let\allowhyphenation=\typog@allowhyphenation
180 \fi
181

```

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

```

182 \NewDocumentCommand{\breakpoint}{s}
183   {\discretionary{}{}{}}%
184   \IfBooleanTF{#1}%
185     {\ignorespaces}%
186     {\typog@allowhyphenation}}
187

```

PDF-substitute definition

```

188 \typog@register@pdfsubstitute{
189   \def\breakpoint#1{\if*\detokenize{#1}\ignorespaces\fi}%
190 }
191

```

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

```

192 \NewDocumentEnvironment{hyphenmin}{o m}
193   {\lefthyphenmin=\IfNoValueTF{#1}{#2}{#1}%
194     \righthyphenmin=#2}
195   {}
196

```

A.4 Disable/Break Ligatures

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

```
197 \newcommand*{\typog@hyphen}{\char‘-’}
198
```

`\nolig`

```
199 \NewDocumentCommand{\nolig}{s o}
200   {\dimen0=\IfNoValueTF{#2}{\typog@config@ligaturekern}{#2\typog@dim@unit}%
201    \IfBooleanTF{#1}%
202     {\kern\dimen0\ignorespaces}%
203     {\discretionary{\typog@hyphen}{\kern\dimen0}%
204      \typog@allowhyphenation
205      \IfNoValueF{#2}{\ignorespaces}}}
206
```

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

```
207 \typog@register@pdfsubstitute{
208   \RenewExpandableDocumentCommand{\nolig}{s o m}{%
209     \ifx\typog@TYPOG#3\typog@TYPOG
210       \relax
211     \else
212       \ifx\relax#3\relax
213         \relax
214       \else
215         \PackageError{typog}
216                               {Missing third argument of \nolig}
217                               {Append empty group or \relax after macro in-
218                                vocation}
218       \fi
219     \fi}
220 }
221
```

A.5 Manual Italic Correction

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

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

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

```
224 \newcommand*{\typog@itcorr@text}[1]
225   {\def\typog@@strength{#1}%
226    \dimen0=\fontdimen1\font
227    \ifdim\dimen0=\z@
228      \typog@itcorr@text@unconditional{\typog@@strength}%
229    \else
230      \kern\typog@@strength\dimen0
231    \fi}
```

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

```
232 \newcommand*{\typog@itcorr@math}[1]
233   {\mkern#1\typog@config@mathitalicscorrection}
```

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

```
234 \NewDocumentCommand{\itcorr}{s m}
235   {\ifmmode
236     \typog@itcorr@math{#2}%
237   \else
238     \IfBooleanTF{#1}%
239       {\typog@itcorr@text{#2}}%
240       {\typog@itcorr@text@unconditional{#2}}%
241   \fi}
```

PDF-substitute definition

```
242 \typog@register@pdfsubstitute{
243   \RenewExpandableDocumentCommand{\itcorr}{s m}{}
244 }
245
```

A.6 Apply Extra Kerning

Slash

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

```
246 \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’`.

```
247 \NewDocumentCommand{\kernedslash}{s}
248   {\hspace*{\typog@config@slashkern}%
249   \typog@forwardslash
250   \IfBooleanTF{#1}%
251     {\hspace*{\typog@config@slashkern}\ignorespaces}%
252     {\typog@breakpoint\typog@allowhyphenation\hspace*{\typog@config@slashkern}}}
```

PDF-substitute definition

```
253 \typog@register@pdfsubstitute{
254   \def\kernedslash#1{\if*\detokenize{#1}/\ignorespaces\else/#1\fi}%
255 }
256
```


Hyphen**\kernedhyphen**

```

257 \NewDocumentCommand{\kernedhyphen}{s O{0} m m}
258 {
259   \mspace{\muexpr(#3 mu) * 18 / 1000}%
260   \raisebox{#2\typog@dim@unit}{\m@th\mathord{-}}%
261   \mspace{\muexpr(#4 mu) * 18 / 1000}%
262   \else
263     \def\typog@@auto{*}%
264     \def\typog@@optarg{#2}%
265     \hspace*{#3\typog@dim@unit}%
266     \raisebox{\ifx\typog@@optarg\typog@@auto
267               \typog@config@raisecapitalhyphen
268               \else
269                 \typog@@optarg\typog@dim@unit
270               \fi}{\typog@hyphen}%
271     \hspace{#4\typog@dim@unit}%
272     \IfBooleanT{#1}{\nobreak}%
273   \fi}

PDF-substitute definition

274 \typog@register@pdfsubstitute{
275   \RenewExpandableDocumentCommand{\kernedhyphen}{s o m m}{-}
276 }

```

One-argument shorthands.

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

```

277 \NewDocumentCommand{\leftkernedhyphen}{s O{0} m}
278 {
279   \IfBooleanTF{#1}%
280   {\kernedhyphen*{#2}{#3}{0}\ignorespaces}%
281   {\kernedhyphen[#2]{#3}{0}}}

PDF-substitute definition

281 \typog@register@pdfsubstitute{
282   \RenewExpandableDocumentCommand{\leftkernedhyphen}{s o m}{-}
283 }
284

```

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

```

285 \NewDocumentCommand{\rightkernedhyphen}{s O{0} m}
286 {
287   \IfBooleanTF{#1}%
288   {\kernedhyphen*{#2}{0}{#3}\ignorespaces}%
289   {\kernedhyphen[#2]{0}{#3}}}

PDF-substitute definition

289 \typog@register@pdfsubstitute{
290   \RenewExpandableDocumentCommand{\rightkernedhyphen}{s o m}{-}
291 }
292

```

A.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.

```
293 \newcommand*{\typog@breakpoint}
294   {\penalty\typog@config@breakpenalty}
```

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

```
295 \NewDocumentCommand{\capitalhyphen}{s}
296   {\raisebox{\typog@config@raisecapitalhyphen}{\typog@hyphen}%
297     \IfBooleanTF{#1}%
298       {\ignorespaces}%
299       {\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.

```
300 \typog@register@pdfsubstitute{
301   \def\capitalhyphen#1{%
302     \if*\detokenize{#1}%
303       -\ignorespaces
304     \else
305       -#1%
306     \fi}
307 }
308
```

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

```
309 \NewDocumentCommand{\capitalendash}{s}
310   {\raisebox{\typog@config@raisecapitaldash}{\textendash}%
311     \IfBooleanTF{#1}%
312       {\ignorespaces}%
313       {\typog@breakpoint\typog@allowhyphenation}}
314 \let\capitaldash=\capitalendash
```

PDF-substitute definition

```
315 \typog@register@pdfsubstitute{
316   \def\capitalendash#1{%
317     \if*\detokenize{#1}%
318       \textendash\ignorespaces
319     \else
320       \textendash#1%
321     \fi}
322 \let\capitaldash=\capitalendash
323 }
324
```

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

```
325 \NewDocumentCommand{\capitalemdash}{s}
326   {\raisebox{\typog@config@raisecapitaldash}{\textemdash}%
327     \IfBooleanTF{#1}%
328       {\ignorespaces}%
329       {\typog@breakpoint\typog@allowhyphenation}}}
```

PDF-substitute definition

```
330 \typog@register@pdfsubstitute{
331   \def\capitalemdash#1{%
332     \if*\detokenize{#1}%
333       \textemdash\ignorespaces
334     \else
335       \textemdash#1%
336     \fi}
337 }
338
```

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

```
339 \NewDocumentCommand{\figuredash}{s}
340   {\raisebox{\typog@config@raisefiguredash}{\textendash}%
341     \IfBooleanTF{#1}%
342       {\ignorespaces}%
343       {\typog@breakpoint\typog@allowhyphenation}}}
```

PDF-substitute definition

```
344 \typog@register@pdfsubstitute{\let\figuredash=\capitaldash}
345
```

`\capitaltimes`

```
346 \NewDocumentCommand{\capitaltimes}{}
347   {\ifmmode
348     \mathbin{\raisebox{\typog@config@raisecapitaltimes}{\m@th\times}}}%
349   \else
350     \raisebox{\typog@config@raisecapitaltimes}{\textttimes}%
351   \fi}
```

PDF-substitute definition

```
352 \typog@register@pdfsubstitute{
353   \RenewExpandableDocumentCommand{\capitaltimes}{}{\textttimes}
354 }
355
```

`\singleguillemetleft`

```
356 \NewDocumentCommand{\singleguillemetleft}{}
357   {\typog@allowhyphenation
358     \raisebox{\typog@config@raiseguillemets}{\guilsinglleft}}
```

PDF-substitute definition

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

`\singleguillemetright`

```
360 \NewDocumentCommand{\singleguillemetright}{}
361   {\raisebox{\typog@config@raiseguillemets}{\guilsinglright}%
362     \typog@allowhyphenation}
    PDF-substitute definition
363 \typog@register@pdfsubstitute{\let\singleguillemetright\guilsinglright}
```

`\doubleguillemetleft`

```
364 \NewDocumentCommand{\doubleguillemetleft}{}
365   {\typog@allowhyphenation
366     \raisebox{\typog@config@raiseguillemets}{\guillemotleft}}
    PDF-substitute definition
367 \typog@register@pdfsubstitute{\let\doubleguillemetleft\guillemotleft}
```

`\doubleguillemetright`

```
368 \NewDocumentCommand{\doubleguillemetright}{}
369   {\raisebox{\typog@config@raiseguillemets}{\guillemotright}%
370     \typog@allowhyphenation}
    PDF-substitute definition
371 \typog@register@pdfsubstitute{\let\doubleguillemetright\guillemotright}
```

`\Singleguillemetleft`

```
372 \NewDocumentCommand{\Singleguillemetleft}{}
373   {\typog@allowhyphenation
374     \raisebox{\typog@config@raisecapitalguillemets}{\guilsinglleft}}
    PDF-substitute definition
375 \typog@register@pdfsubstitute{\let\Singleguillemetleft\guilsinglleft}
```

`\Singleguillemetright`

```
376 \NewDocumentCommand{\Singleguillemetright}{}
377   {\raisebox{\typog@config@raisecapitalguillemets}{\guilsinglright}%
378     \typog@allowhyphenation}
    PDF-substitute definition
379 \typog@register@pdfsubstitute{\let\Singleguillemetright\guilsinglright}
```

`\Doubleguillemetleft`

```
380 \NewDocumentCommand{\Doubleguillemetleft}{}
381   {\typog@allowhyphenation
382     \raisebox{\typog@config@raisecapitalguillemets}{\guillemotleft}}
    PDF-substitute definition
383 \typog@register@pdfsubstitute{\let\Doubleguillemetleft\guillemotleft}
```

`\Doubleguillemetright`

```
384 \NewDocumentCommand{\Doubleguillemetright}{}
385   {\raisebox{\typog@config@raisecapitalguillemets}{\guillemotright}%
386     \typog@allowhyphenation}
    PDF-substitute definition
387 \typog@register@pdfsubstitute{\let\Doubleguillemetright\guillemotright}
388
```

A.8 Vert. Adjust Label Items

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

```

389 \newcommand*{\@typog@uppercase@adjust@labelitem}[1]
390   {\@typog@maybe@patch@itemize
391     \ifstrequal{#1}{*}
392       {\setlength{\typog@adjust@labelitemi}
393         {\typog@adjust@uppercase@labelitemi}
394         \setlength{\typog@adjust@labelitemii}
395           {\typog@adjust@uppercase@labelitemii}
396         \setlength{\typog@adjust@labelitemiii}
397           {\typog@adjust@uppercase@labelitemiii}
398         \setlength{\typog@adjust@labelitemiv}
399           {\typog@adjust@uppercase@labelitemiv}}
400     {\ifcase #1% 0
401       \relax % outside of any itemize environment
402     \or % 1
403       \setlength{\typog@adjust@labelitemi}
404         {\typog@adjust@uppercase@labelitemi}
405     \or % 2
406       \setlength{\typog@adjust@labelitemii}
407         {\typog@adjust@uppercase@labelitemii}
408     \or % 3
409       \setlength{\typog@adjust@labelitemiii}
410         {\typog@adjust@uppercase@labelitemiii}
411     \or % 4
412       \setlength{\typog@adjust@labelitemiv}
413         {\typog@adjust@uppercase@labelitemiv}
414     \else
415       \PackageError{typog}
416         {Itemize level out of range}
417         {Valid levels are 1, 2, 3, 4, and *}
418     \fi}}
419 
```

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

```

420 \newcommand*{\@typog@lowercase@adjust@labelitem}[1]
421   {\@typog@maybe@patch@itemize
422     \ifstrequal{#1}{*}
423       {\setlength{\typog@adjust@labelitemi}
424         {\typog@adjust@lowercase@labelitemi}
425         \setlength{\typog@adjust@labelitemii}
426           {\typog@adjust@lowercase@labelitemii}
427         \setlength{\typog@adjust@labelitemiii}
428           {\typog@adjust@lowercase@labelitemiii}
429         \setlength{\typog@adjust@labelitemiv}
430           {\typog@adjust@lowercase@labelitemiv}}
431     {\ifcase #1% 0
432       \relax % outside of any itemize environment

```

```

433         \or % 1
434             \setlength{\typog@adjust@labelitemi}
435                 {\typog@adjust@lowercase@labelitemi}
436         \or % 2
437             \setlength{\typog@adjust@labelitemii}
438                 {\typog@adjust@lowercase@labelitemii}
439         \or % 3
440             \setlength{\typog@adjust@labelitemiii}
441                 {\typog@adjust@lowercase@labelitemiii}
442         \or % 4
443             \setlength{\typog@adjust@labelitemiv}
444                 {\typog@adjust@lowercase@labelitemiv}
445         \else
446             \PackageError{typog}
447                 {Itemize level out of range}
448                 {Valid levels are 1, 2, 3, 4, and *}
449         \fi}}
450

```

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

```

451 \newcommand*{\@typog@noadjust@labelitem}[1]
452 {\ifstrequal{#1}{*}
453     {\setlength{\typog@adjust@labelitemi}{\z@}
454     \setlength{\typog@adjust@labelitemii}{\z@}
455     \setlength{\typog@adjust@labelitemiii}{\z@}
456     \setlength{\typog@adjust@labelitemiv}{\z@}}
457 {\ifcase #1% 0
458     \relax % outside of any itemize environment
459     \or % 1
460         \setlength{\typog@adjust@labelitemi}{\z@}
461     \or % 2
462         \setlength{\typog@adjust@labelitemii}{\z@}
463     \or % 3
464         \setlength{\typog@adjust@labelitemiii}{\z@}
465     \or % 4
466         \setlength{\typog@adjust@labelitemiv}{\z@}
467     \else
468         \PackageError{typog}
469             {Itemize level out of range}
470             {Valid levels are 1, 2, 3, 4, and *}
471     \fi}}
472

```

`\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`.

```

473 \NewDocumentCommand{\uppercaseadjustlabelitems}{m}
474 {\AfterPreamble{%
475     \ifblank{#1}

```

```

476             {\@typog@uppercase@adjust@labelitem{\@itemdepth}}
477             {\forcsvlist{\@typog@uppercase@adjust@labelitem}{#1}}%
478     \ignorespaces}}
479

```

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

```

480 \NewDocumentCommand{\lowercaseadjustlabelitems}{m}
481   {\AfterPreamble{%
482     \ifblank{#1}
483       {\@typog@lowercase@adjust@labelitem{\@itemdepth}}
484       {\forcsvlist{\@typog@lowercase@adjust@labelitem}{#1}}%
485     \ignorespaces}}
486

```

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

```

487 \NewDocumentCommand{\noadjustlabelitems}{m}
488   {\ifblank{#1}
489     {\@typog@noadjust@labelitem{\@itemdepth}}
490     {\forcsvlist{\@typog@noadjust@labelitem}{#1}}%
491   \ignorespaces}
492

```

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`.

```

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

```

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

```

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

```

Sneak in our own macro's name.

```

495   \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.

```

496 \expandafter\def\csname\@itemitem\endcsname
497     {\raisebox{\csname typog@adjust@labelitem\romannumeral\the\@itemdepth\endcsname
498              {\@typog@old@itemitem}}}
499

```

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.

```

500 \newcommand*{\@typog@patch@itemize}
501 {
502   \ifdefined\enit@itemize@i
503     \patchcmd{\enit@itemize@i}
504       {\expandafter}
505       {\@typog@itemize@patch\expandafter}
506       {\typog@debug@typeout{patching enumitem \string\enit@itemize@i\space
507         ceeded}}
508       {\PackageError{typog}
509         {Patching enumitem macro \string\enit@itemize@i\space
510         {}}
511       \else
512     \patchcmd{\itemize}
513       {\expandafter}
514       {\@typog@itemize@patch\expandafter}
515       {\typog@debug@typeout{patching \string\itemize\space suc-
516         ceeded}}
517       {\PackageError{typog}
518         {Patching plain LaTeX macro \string\itemize\space fai

```



```

516                                     {}
517   \fi}
518

```

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

```

519 \newbool{@typog@itemize@has@been@patched}
520 \newcommand*{@typog@maybe@patch@itemize}
521   {\ifbool{@typog@itemize@has@been@patched}
522     {\relax}
523     {@typog@patch@itemize
524       \booltrue{@typog@itemize@has@been@patched}}}
525

```

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.

```

526 \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.

```

527 \newcommand*{@typogadjuststairsfor}[5]

```

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

```

528   {\dimen0=1pt%

```

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

```

529   \count0=#3\relax
530   \unless\ifodd\count0
531     \advance\count0 by 1%
532   \fi

```

Set the iteration counter.

```

533   \setcounter{typog@@iteration}{1}%

```

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

```

534   \setbox0=\hbox{#4}%

```

Box 1 is the accumulator for the raised samples.

```

535   \setbox1=\hbox{}%

```

Build the stairs.

```

536   \loop
537     \ifnum\thetypog@@iteration=\numexpr\count0 / 2\relax
538       \dimen1=2\dimen0
539     \else
540       \dimen1=\dimen0
541     \fi
542     \dimen2=\dimexpr#2 * (\thetypog@@iteration - \count0 / 2)\re-
lax
543     \setbox1=\hbox{\unhbox1\raisebox{\dimen2}{\kern\dimen1 #5\kern\dimen1}}%

```

```

544 \addtocounter{typog@@iteration}{1}%
545 \unless\ifnum\thetypog@@iteration>\count0
546 \repeat

```

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

```

547 \mbox{\rlap{\raisebox{\fpeval{#1}\ht0}{\rule{\wd1}{\typog@hairline@width}}}\bo
548

```

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

```

549 \NewDocumentCommand{\typogadjuststairs}{0{.5} m m m}
550 {\begingroup
551 \unless\ifdim #2>\z@
552 \PackageError{typog}
553 {\string\typogadjuststairs\space non-positive step-
554 size}
555 {\step-size must be a positive dimension}%
556 \fi
557 \ifnum #3<1
558 \PackageError{typog}
559 {\string\typogadjuststairs\space too few number-
560 of-steps}
561 {\number-of-steps must at least be 1}%
562 \fi
563 \ifblank{#4}
564 {\PackageError{typog}
565 {sample must not be empty}
566 {supply either some uppercase or some low-
567 ercase letters}}
568 {\}%
569 \def\arraystretch{1}%
570 \begin{tabular}{@{}c@{}}
571 \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemi} \\
572 \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemii} \\
573 \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemiii} \\
574 \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemiv}
575 \end{tabular}
576 \endgroup}
577

```

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

```

575 \newcommand*{\typoguppercaseadjusted@labelitems}
576 {\hbox{\raisebox{\typog@adjust@uppercase@labelitemi}{\labelitemi}%
577 \raisebox{\typog@adjust@uppercase@labelitemii}{\labelitemii}%
578 \raisebox{\typog@adjust@uppercase@labelitemiii}{\labelitemiii}%
579 \raisebox{\typog@adjust@uppercase@labelitemiv}{\labelitemiv}}}

```

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

```

580 \NewDocumentCommand{\typoguppercaseadjustcheck}{0{.5} m}
581   {\setbox0=\hbox{#2}%
582    \setbox1=\typog@uppercase@adjusted@labelitems
583    \mbox{\rlap{\raisebox{\fpeval{#1}\ht0}
584              {\rule{\wd1}{\typog@hairline@width}}}%
585           \box1}}
586

```

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

```

587 \newcommand*{\typog@lowercase@adjusted@labelitems}
588   {\hbox{\raisebox{\typog@adjust@lowercase@labelitemi}{\labelitemi}%
589           \raisebox{\typog@adjust@lowercase@labelitemii}{\labelitemii}%
590           \raisebox{\typog@adjust@lowercase@labelitemiii}{\labelitemiii}%
591           \raisebox{\typog@adjust@lowercase@labelitemiv}{\labelitemiv}}}

```

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

```

592 \NewDocumentCommand{\typoglowercaseadjustcheck}{0{.5} m}
593   {\setbox0=\hbox{#2}%
594    \setbox1=\typog@lowercase@adjusted@labelitems
595    \mbox{\rlap{\raisebox{\fpeval{#1}\ht0}
596              {\rule{\wd1}{\typog@hairline@width}}}%
597           \box1}}
598

```

A.9 Align Last Line of a Paragraph

The code of environment `lastlinerraggedleftpar` has been inspired by macro `\lastlinerraggedleft` [35, Sec. 2].

`lastlinerraggedleftpar (env.)`

```

599 \NewDocumentEnvironment{lastlinerraggedleftpar}{}
600   {\lastlinefit=0%
601    \setlength{\leftskip}{\z@ \@plus 1fil}%
602    \setlength{\rightskip}{-\leftskip}%
603    \setlength{\parfillskip}{\leftskip}}
604   {\par}

```

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

```

605 \let\lastlineflushrightpar=\lastlinerraggedleftpar
606 \let\endlastlineflushrightpar=\endlastlinerraggedleftpar
607

```

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

```

608 \NewDocumentEnvironment{lastlinecenteredpar}{}
609   {\lastlinefit=0%
610    \setlength{\leftskip}{\z@ \@plus .5fil}%
611    \setlength{\rightskip}{-\leftskip}%
612    \setlength{\parfillskip}{\z@ \@plus 1fil}}
613   {\par}
614

```

A.10 Fill Last Line of a Paragraph

`shortenpar` (*env.*)

```
615 \NewDocumentEnvironment{shortenpar}{}
616 {\advance\looseness by -1
617 \ifnum\tracingparagraphs>0
618 \typeout{@ looseness \the\looseness}%
619 \fi}
620 {\par}
621
```

`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.

```
622 \NewDocumentEnvironment{prolongpar}{}
623 {\finalhyphendemerits=100000001
624 \advance\looseness by 1
625 \ifnum\tracingparagraphs>0
626 \typeout{@ looseness \the\looseness}%
627 \fi}
628 {\par}
629
```

`nextindentpar@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`.

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

`nextindentpar@nonzero@parindent`

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

`covernextindentpar` (*env.*)

```
632 \NewDocumentEnvironment{covernextindentpar}{o}
633 {\IfNoValueTF{#1}
634 {\ifdim\parindent=\z@
635 \dimen0=\dimexpr\linewidth - \typog@covernextindentpar@zero@parindent
636 \else
637 \dimen0=\dimexpr\linewidth - \typog@covernextindentpar@nonzero@parindent
638 \fi}
639 {\dimen0=\dimexpr\linewidth - (#1)}%
640 \parfillskip=\dimen0 \@minus \dimen0
641 \relax}
642 {\par}
643
```

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

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

`lastlinepar@nonzero@parindent`

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

`openlastlinepar` (*env.*) Compare with the suggestion in Ref. 30.

```

646 \NewDocumentEnvironment{openlastlinepar}{o}
647   {\IfNoValueTF{#1}
648     {\ifdim\parindent=\z@
649       \skip0=\typog@openlastlinepar@zero@parindent
650       \@plus 1fil
651       \@minus \typog@openlastlinepar@zero@parindent
652     }else
653       \skip0=\typog@openlastlinepar@nonzero@parindent
654       \@plus 1fil
655       \@minus \typog@openlastlinepar@nonzero@parindent
656     }fi}
657   {\dimen0=\dimexpr#1\relax
658     \skip0=\dimen0 \@plus 1fil \@minus \dimen0}
659   \parfillskip=\skip0}
660 {\par}
661
```

A.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.

```
662 \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.

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

`\widespace`

```

664 \NewDocumentCommand{\widespace}{s}
665   {\IfBooleanTF{#1}%
666     {\dimen0=\widespacescale\fontdimen2\font}%
667     {\ifdim\fontdimen7\font=\z@
668       \dimen0=\widespacescale\fontdimen2\font
669     }else
670       \dimen0=\dimexpr\fontdimen2\font +
671         \widespacestrength\fontdimen7\font
672     }fi}%
673   \hskip \glueexpr\dimen0
674     \@plus \widespacescale\fontdimen3\font
675     \@minus \widespacescale\fontdimen4\font
676   \ignorespaces}
677
```

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

```
678 \newcommand*{\narrowospacestrength}{.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`.

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

`\narrowspace`

```
680 \NewDocumentCommand{\narrowspace}{s}
681   {\IfBooleanTF{#1}%
682     {\dimen0=\narrowspacescale\fontdimen2\font}%
683     {\ifdim\fontdimen7\font=\z@
684       \dimen0=\narrowspacescale\fontdimen2\font
685       \else
686         \dimen0=\dimexpr\fontdimen2\font -
687           \narrowspacestrength\fontdimen7\font
688       \fi}%
689   \hskip \glueexpr\dimen0
690     \@plus \narrowspacescale\fontdimen3\font
691     \@minus \narrowspacescale\fontdimen4\font
692   \ignorespaces}
693
```

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

`loosespacing` (*env.*)

```
694 \NewDocumentEnvironment{loosespacing}{0{1}}
695   {\dimen2=\fontdimen2\font
696   \ifcase #1
697     \spaceskip=\z@
698     \or % 1      +5%
699     \spaceskip=1.05\dimen2 \@plus .5\dimen2 \@minus .1\dimen2
700     \or % 2      +10%
701     \spaceskip=1.1\dimen2 \@plus .5\dimen2 \@minus .1\dimen2
702     \or % 3      +20%
703     \spaceskip=1.2\dimen2 \@plus .6\dimen2 \@minus .2\dimen2
704     \else % >= 4  +30%
705     \spaceskip=1.3\dimen2 \@plus .8\dimen2 \@minus .3\dimen2
706     \fi
707   \ignorespaces}
708 {\ignorespacesafterend}
709
```

`tightspacing` (*env.*)

```
710 \NewDocumentEnvironment{tightspacing}{0{1}}
711   {\dimen2=\fontdimen2\font
712   \ifcase #1
713     \spaceskip=\z@
714     \or % 1      -1.25%
715     \spaceskip=.9875\dimen2 \@plus .0125\dimen2 \@minus .5\dimen2
716     \or % 2      -2.5%
717     \spaceskip=.975\dimen2 \@plus .025\dimen2 \@minus .5\dimen2
718     \or % 3      -5%
```

```

719     \spaceskip=.95\dimen2 \@plus .05\dimen2 \@minus .5\dimen2
720     \else % >= 4      -10%
721     \spaceskip=.9\dimen2 \@plus .1\dimen2 \@minus .5\dimen2
722     \fi
723     \ignorespaces}
724     {\ignorespacesafterend}
725

```

A.12 Microtype Front-End

Tracking

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

```

726 \NewDocumentEnvironment{setfonttracking}{m}
727   {\edef\MT@letterspace@{#1}%
728    \lsstyle
729    \ignorespaces}
730   {\ignorespacesafterend}
731

```

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.

```

732 \newcommand*{\typog@setup@font@expansion}
733   {\SetExpansion
734    [context = typog@shrink1,
735     shrink = \typog@shrink@i,
736     stretch = 0]%
737    {encoding = {*}}%
738    {}
739   \SetExpansion
740    [context = typog@shrink2,
741     shrink = \typog@shrink@ii,
742     stretch = 0]%
743    {encoding = {*}}%
744    {}
745   \SetExpansion
746    [context = typog@shrink3,
747     shrink = \typog@shrink@iii,
748     stretch = 0]%
749    {encoding = {*}}%
750    {}
751
752   \SetExpansion
753    [context = typog@stretch1,
754     shrink = 0,
755     stretch = \typog@stretch@i]%
756    {encoding = {*}}%
757    {}

```

```

758 \SetExpansion
759   [context = typog@stretch2,
760     shrink = 0,
761     stretch = \typog@stretch@ii]%
762   {encoding = {*}}%
763   {}
764 \SetExpansion
765   [context = typog@stretch3,
766     shrink = 0,
767     stretch = \typog@stretch@iii]%
768   {encoding = {*}}%
769   {}
770
771 \SetExpansion
772   [context = typog@expand1,
773     shrink = \typog@shrink@i,
774     stretch = \typog@stretch@i]%
775   {encoding = {*}}%
776   {}
777 \SetExpansion
778   [context = typog@expand2,
779     shrink = \typog@shrink@ii,
780     stretch = \typog@stretch@ii]%
781   {encoding = {*}}%
782   {}
783 \SetExpansion
784   [context = typog@expand3,
785     shrink = \typog@shrink@iii,
786     stretch = \typog@stretch@iii]%
787   {encoding = {*}}%
788   {}

```

`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.

```

789 \newcommand*{\typog@test@microtype@expansion@feature}
790   {\ifMT@expansion
791     \typog@typeout{microtype preloaded -- font expansion features avail-
792       able}%
793     \def\typog@require@microtype@expansion{\relax}
794     \typog@setup@font@expansion
795   \else
796     \PackageWarning{typog}{microtype preloaded,\space
797       but font expansion is disabled}%
798     \def\typog@require@microtype@expansion
799       {\PackageError{typog}
800         {microtype font expansion disabled}
801         {pass option 'expansion' to package microtype}}
802   \fi}

```

`require@microtype@expansion` We are all set for the initialization of the font expansion, however, we must

be careful in which (load-)state package microtype is in. Compare the code for `\typog@require@microtype` and `\typog@require@preloaded@microtype`.

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

```

802 \iftypog@microtype@preloaded
803   \typog@test@microtype@expansion@feature
804 \else
805   \def\typog@require@microtype@expansion
806     {\PackageError{typog}%
807       {package microtype not (pre-)loaded, %
808         which is required for typog's font expansion}%
809       {require package microtype before package typog}}
810 \fi
811
```

`setfontshrink (env.)`

```

812 \NewDocumentEnvironment{setfontshrink}{0{1}}
813 {\typog@require@microtype@expansion
814   \ifcase#1% 0
815     \relax
816   \or % 1
817     \microtypecontext{expansion=typog@shrink1}%
818   \or % 2
819     \microtypecontext{expansion=typog@shrink2}%
820   \else % >= 3
821     \microtypecontext{expansion=typog@shrink3}%
822   \fi
823   \ignorespaces}
824 {\ignorespacesafterend}
825
```

`setfontstretch (env.)`

```

826 \NewDocumentEnvironment{setfontstretch}{0{1}}
827 {\typog@require@microtype@expansion
828   \ifcase#1% 0
829     \relax
830   \or % 1
831     \microtypecontext{expansion=typog@stretch1}%
832   \or % 2
833     \microtypecontext{expansion=typog@stretch2}%
834   \else % >= 3
835     \microtypecontext{expansion=typog@stretch3}%
836   \fi
837   \ignorespaces}
838 {\ignorespacesafterend}
839
```

`setfontexpand (env.)`

```

840 \NewDocumentEnvironment{setfontexpand}{0{1}}
841 {\typog@require@microtype@expansion
842   \ifcase#1% 0
843     \relax
```

```

844 \or % 1
845   \microtypecontext{expansion=typog@expand1}%
846 \or % 2
847   \microtypecontext{expansion=typog@expand2}%
848 \else % >= 3
849   \microtypecontext{expansion=typog@expand3}%
850 \fi
851 \ignorespaces}
852 {\ignorespacesafterend}
853

```

`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.

```

854 \NewDocumentEnvironment{nofontexpansion}{}
855   {\ifdefined\microtypesetup
856     \microtypesetup{expansion=false}%
857   \fi
858   \ignorespaces}
859   {\ignorespacesafterend}

```

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

```

860 \let\nofontexpand=\nofontexpansion
861 \let\endnofontexpand=\endnofontexpansion
862

```

Character Protrusion

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

```

863 \NewDocumentEnvironment{nocharprotrusion}{}
864   {\ifdefined\microtypesetup
865     \microtypesetup{protrusion=false}%
866   \fi
867   \ignorespaces}
868   {\ignorespacesafterend}
869

```

A.13 Sloppy Paragraphs

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

```

870 \newcommand*{\typog@scaled@emergencystretch}[1]
871   {\emergencystretch=\ifdim\linewidth=\z@
872     #1%
873   \else
874     \dimexpr (#1) * \linewidth / \textwidth
875   \fi}
876

```

`\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.

```

877 \NewDocumentCommand{\slightlyloppy}{0{1}}
878   {\ifcase #1% 0
879     % \tolerance=200
880     % \emergencystretch=\z@
881     % \hfuzz=.1\p@
882     % \vfuzz=\hfuzz
883     \fussy
884   \or % 1
885     \pretolerance=165%
886     \tolerance=330%
887     \typog@scaled@emergencystretch{.375em}%
888     \hfuzz=.15\p@
889     \vfuzz=\hfuzz
890   \or % 2
891     \pretolerance=265%
892     \tolerance=530%
893     \typog@scaled@emergencystretch{.75em}%
894     \hfuzz=.15\p@
895     \vfuzz=\hfuzz
896   \or % 3
897     \pretolerance=435%
898     \tolerance=870%
899     \typog@scaled@emergencystretch{1.125em}%
900     \hfuzz=.2\p@
901     \vfuzz=\hfuzz
902   \or % 4
903     \pretolerance=705%
904     \tolerance=1410%
905     \typog@scaled@emergencystretch{1.5em}%
906     \hfuzz=.3\p@
907     \vfuzz=\hfuzz
908   \or % 5
909     \pretolerance=1155%
910     \tolerance=2310%
911     \typog@scaled@emergencystretch{1.875em}%
912     \hfuzz=.35\p@
913     \vfuzz=\hfuzz
914   \or % 6
915     \pretolerance=1880%
916     \tolerance=3760%
917     \typog@scaled@emergencystretch{2.25em}%
918     \hfuzz=.4\p@
919     \vfuzz=\hfuzz
920   \or % 7
921     \pretolerance=3065%
922     \tolerance=6130%
923     \typog@scaled@emergencystretch{2.625em}%
924     \hfuzz=.45\p@

```

```

925 \vfuzz=\hfuzz
926 \else % >= 8
927 % \tolerance=9999
928 % \emergencystretch=3em
929 % \hfuzz=.5\p@
930 % \vfuzz=\hfuzz
931 \sloppy
932 \fi
933 \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.)`

```

934 \NewDocumentEnvironment{slightlyloppypar}{0{1}}
935 {\par\slightlyloppy[#1]\ignorespaces}
936 {\par}
937

```

A.14 Vertically Partially-Tied Paragraphs

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

```

938 \ExplSyntaxOn
939 \newcommand*{\typog@geometric@mean}[2]
940 {\fp_to_int:n {sqrt((#1) * (#2))}}
941 \ExplSyntaxOff
942

```

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

```

943 \newcounter{typog@mean@penalty}
944
\vtietop
945 \NewDocumentCommand{\vtietop}{0{3}}
946 { \setcounter{typog@mean@penalty}
947   { \typog@geometric@mean{\@M}{\clubpenalty}}%
948   \typog@debug@typeout{vtietop: penalties \the\@M--\the\value{typog@mean@penalty}
949   -\the\clubpenalty}%
949   \unless\ifnum\clubpenalty<\@M
950     \PackageWarning{typog}{vtietop: clubpenalty=\the\clubpenalty\space>= 10000}%
951   \fi
952   \ifcase#1% 0
953     \relax
954   \or % 1
955     \relax
956   \or % 2
957     \clubpenalties 3
958     \@M
959     \value{typog@mean@penalty}
960     \clubpenalty
961   \or % 3
962     \clubpenalties 4
963     \@M \@M
964     \value{typog@mean@penalty}
965     \clubpenalty
966   \or % 4
967     \clubpenalties 5
968     \@M \@M \@M
969     \value{typog@mean@penalty}
970     \clubpenalty
971   \or % 5
972     \clubpenalties 6
973     \@M \@M \@M \@M
974     \value{typog@mean@penalty}
975     \clubpenalty
976   \or % 6
977     \clubpenalties 7
978     \@M \@M \@M \@M \@M
979     \value{typog@mean@penalty}
980     \clubpenalty
981   \or % 7
982     \clubpenalties 8
983     \@M \@M \@M \@M \@M
984     \value{typog@mean@penalty}
985     \clubpenalty
986   \or % 8
987     \clubpenalties 9
988     \@M \@M \@M \@M \@M
989     \value{typog@mean@penalty}
990     \clubpenalty

```

```

991 \else % >= 9
992   \clubpenalties 10
993   \@M \@M \@M \@M \@M \@M \@M \@M
994   \value{typog@mean@penalty}
995   \clubpenalty
996 \fi}
997

```

`vtietoppar (env.)`

```

998 \NewDocumentEnvironment{vtietoppar}{0{3}}
999   {\vtietop[#1]}
1000   {\par
1001     \ignorespacesafterend}
1002

```

`\splicevtietop`

```

1003 \NewDocumentCommand{\splicevtietop}{0{3}}
1004   {\let\typog@old@item=\@item
1005     \def\@item[##1]{\typog@old@item[##1]\vtietop[#1]}%
1006     \ignorespaces}
1007

```

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).

```

1008 \ifdefined\SetEnumitemKey
1009   \SetEnumitemKey{vtietop}{first=\splicevtietop}
1010 \fi
1011

```

`\vtiebot`

```

1012 \NewDocumentCommand{\vtiebot}{0{3}}
1013   {\setcounter{typog@mean@penalty}
1014     {\typog@geometric@mean{\@M}{\widowpenalty}}%
1015     \typog@debug@typeout{vtiebot: penalties \the\@M--\the\value{typog@mean@penalty}
1016       -\the\widowpenalty}%
1016     \unless\ifnum\widowpenalty<\@M
1017       \PackageWarning{typog}{vtiebot: widowpenalty=\the\widowpenalty\space>= 10000}
1018     \fi
1019     \ifcase#1% 0
1020       \relax
1021     \or % 1
1022       \relax
1023     \or % 2
1024       \widowpenalties 3
1025       \@M
1026       \value{typog@mean@penalty}
1027       \widowpenalty
1028     \or % 3
1029       \widowpenalties 4
1030       \@M \@M
1031       \value{typog@mean@penalty}
1032       \widowpenalty

```

```

1033 \or % 4
1034   \widowpenalties 5
1035     \@M \@M \@M
1036     \value{typog@mean@penalty}
1037     \widowpenalty
1038 \or % 5
1039   \widowpenalties 6
1040     \@M \@M \@M \@M
1041     \value{typog@mean@penalty}
1042     \widowpenalty
1043 \or % 6
1044   \widowpenalties 7
1045     \@M \@M \@M \@M \@M
1046     \value{typog@mean@penalty}
1047     \widowpenalty
1048 \or % 7
1049   \widowpenalties 8
1050     \@M \@M \@M \@M \@M \@M
1051     \value{typog@mean@penalty}
1052     \widowpenalty
1053 \or % 8
1054   \widowpenalties 9
1055     \@M \@M \@M \@M \@M \@M \@M
1056     \value{typog@mean@penalty}
1057     \widowpenalty
1058 \else % >= 9
1059   \widowpenalties 10
1060     \@M \@M \@M \@M \@M \@M \@M \@M
1061     \value{typog@mean@penalty}
1062     \widowpenalty
1063 \fi}
1064

```

vtiebotpar (*env.*)

```

1065 \NewDocumentEnvironment{vtiebotpar}{0{3}}
1066   {\vtiebot[#1]}
1067   {\par
1068     \ignorespacesafterend}
1069

```

\typog@vtiebotdisp

```

1070 \NewDocumentCommand{\typog@vtiebotdisp}{m}
1071   {\setcounter{typog@mean@penalty}
1072     {\typog@geometric@mean{\@M}{\displaywidowpenalty}}%
1073     \typog@debug@typeout{vtiebotdisp: penalties \the\@M--\the\value{typog@mean@pen
1074       -\the\displaywidowpenalty}}%
1075     \unless\ifnum\displaywidowpenalty<\@M
1076       \PackageWarning{typog}{vtiebotdisp: displaywidowpenalty=\the\displaywidowpen
1077       \fi
1078     \ifcase#1% 0
1079       \relax
1080     \or % 1

```

```

1080     \relax
1081   \or % 2
1082     \displaywidowpenalties 3
1083     \@M
1084     \value{typog@mean@penalty}
1085     \displaywidowpenalty
1086   \or % 3
1087     \displaywidowpenalties 4
1088     \@M \@M
1089     \value{typog@mean@penalty}
1090     \displaywidowpenalty
1091   \or % 4
1092     \displaywidowpenalties 5
1093     \@M \@M \@M
1094     \value{typog@mean@penalty}
1095     \displaywidowpenalty
1096   \or % 5
1097     \displaywidowpenalties 6
1098     \@M \@M \@M \@M
1099     \value{typog@mean@penalty}
1100     \displaywidowpenalty
1101   \or % 6
1102     \displaywidowpenalties 7
1103     \@M \@M \@M \@M \@M
1104     \value{typog@mean@penalty}
1105     \displaywidowpenalty
1106   \or % 7
1107     \displaywidowpenalties 8
1108     \@M \@M \@M \@M \@M \@M
1109     \value{typog@mean@penalty}
1110     \displaywidowpenalty
1111   \or % 8
1112     \displaywidowpenalties 9
1113     \@M \@M \@M \@M \@M \@M \@M
1114     \value{typog@mean@penalty}
1115     \displaywidowpenalty
1116   \else % >= 9
1117     \displaywidowpenalties 10
1118     \@M \@M \@M \@M \@M \@M \@M \@M
1119     \value{typog@mean@penalty}
1120     \displaywidowpenalty
1121   \fi}
1122

```

vtiebotdisp (*env.*)

```

1123 \NewDocumentEnvironment{vtiebotdisp}{0{3}}
1124 {\typog@vtiebotdisp{#1}}
1125 {\ignorespacesafterend}
1126

```

vtiebotdisptoppar (*env.*)

```

1127 \NewDocumentEnvironment{vtiebotdisptoppar}{0{3}o}

```



```

1128 {\postdisplaypenalty=\@M
1129 \predisplaypenalty=10001% in accordance with package ‘widows-
and-orphans’
1130 \edef\typog@@top@lines{\IfNoValueTF{#2}{#1}{#2}}%
1131 \edef\typog@@after@display@math{\vtietop[\typog@@top@lines]]}%
1132 \PushPostHook{display}{\aftergroup\typog@@after@display@math}%
1133 \vtiebotdisp[#1]}
1134 {\par
1135 \PopPostHook{display}%
1136 \ignorespacesafterend}
1137

```

A.15 Breakable Disp. Eqs.

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

```

1138 \newenvironment*{breakabledisplay}[1][3]
1139 {\allowdisplaybreaks[#1]}
1140 {\ignorespacesafterend}
1141

```

A.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.

```

1142 \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.

```

1143 \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.

```

1144 \ExplSyntaxOn
1145 \cs_new:Npn \typog@setbaselineskip #1
1146 {

```

Initialize our “emergency-stop” loop counter.

```

1147 \int_set:Nn \l_tmpa_int {1}
1148 \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.

```

1149 \dim_set:Nn \l_tmpa_dim {\glueexpr #1}
1150
1151 \typog@debug@typeout{\string\setbaselineskip:\space
initial\space baselineskip:\space \the\baselineskip}
1152 \typog@debug@typeout{\string\setbaselineskip:\space
target\space baselineskip:\space \dim_use:N \l_tmpa_dim}
1153
1154
1155

```

```

1156 \dim_compare:nNnTF {\baselineskip} > {\c_zero_dim}
1157 {}
1158 {
1159   \PackageError{typog}
1160     {\string\setbaselineskip:\space
1161      baselineskip\space not\space positive}
1162   {}
1163 }
1164
1165 \dim_compare:nNnTF {\l_tmpa_dim} > {\c_zero_dim}
1166 {}
1167 {
1168   \PackageError{typog}
1169     {\string\setbaselineskip:\space target\space
1170      baselineskip\space must\space be\space
1171      positive}
1172   {}
1173 }
1174
1175 \skip_if_eq:nNnTF {\l_tmpa_dim} {\glueexpr #1}
1176 {}
1177 {
1178   \PackageWarning{typog}
1179     {\string\setbaselineskip:\space argument\space
1180      is\space a\space skip;\space
1181      will\space ignore\space glue}
1182   {}
1183 }
1184
1185 \fp_set:Nn \l_tmpa_fp {\l_tmpa_dim / \baselineskip}
1186 \fp_until_do:nNnn {abs(\l_tmpa_dim / \baselineskip - 1)} <
1187   {\typog@setbaselineskip@relative@error}
1188 {
1189   \setstretch{\fp_use:N \l_tmpa_fp}
1190   \fp_set:Nn \l_tmpa_fp
1191     {\l_tmpa_fp * \l_tmpa_dim / \baselineskip}
1192
1193   \int_incr:N \l_tmpa_int
1194   \int_compare:nNnTF {\l_tmpa_int} > {\l_tmpb_int}
1195   {
1196     \PackageError{typog}
1197       {\string\setbaselineskip:\space excessive\space
1198        number\space of\space iterations:\space
1199        \int_use:N \l_tmpa_int\space >\space
1200        \int_use:N \l_tmpb_int}
1201     {}
1202   }
1203   {}
1204 }
1205
1206 \typog@debug@typeout{\string\setbaselineskip:\space
1207   final\space \string\setstretch\space argument:\space

```

```

1208 \fp_use:N \l_tmpa_fp}
1209 \typog@debug@typeout{\string\setbaselineskip:\space
1210 final\space baselineskip:\space \the\baselineskip}
1211 }
1212

```

`\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.

```

1213 \cs_new:Npn \setbaselineskip #1
1214 {
1215 \AfterPreamble{\typog@setbaselineskip{#1}}
1216 \ignorespaces
1217 }
1218

```

`\resetbaselineskip` Set the `\baselineskip` to `>neutral<`.

```

1219 \cs_new:Npn \resetbaselineskip
1220 {
1221 \AfterPreamble{\setstretch{1}}
1222 }
1223

```

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

```

1224 \dim_new:N \typogfontsize

```

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

```

1225 \AfterEndPreamble{
1226 \dim_set:Nn \typogfontsize {\fontdimen6\font}
1227 \typog@debug@typeout{\string\typogfontsize =
1228 \dim_use:N \typogfontsize\space
1229 (at\space begin\space of\space document)}
1230 }
1231

```

`\setbaselineskippercentage`

```

1232 \cs_new:Npn \setbaselineskippercentage #1
1233 {
1234   \AfterPreamble{
1235     \dim_compare:nNnTF {\typogfontsize} > {\c_zero_dim}
1236     {
1237       \typog@setbaselineskip{
1238         \fp_eval:n {(#1) / 100} \typogfontsize}
1239     }
1240     {
1241       \PackageError{typog}
1242         {\string\setbaselineskippercentage:\space
1243          \string\typogfontsize <= 0}
1244         {Maybe\space \string\typogfontsize\space
1245          is\space uninitialized?}
1246     }
1247   }
1248   \ignorespaces
1249 }
1250

```

\setleading

```

1251 \cs_new:Npn \setleading #1
1252 {
1253   \AfterPreamble{
1254     \dim_compare:nNnTF {\typogfontsize} > {\c_zero_dim}
1255     {
1256       \typog@setbaselineskip{\typogfontsize + \dimexpr #1}
1257     }
1258     {
1259       \PackageError{typog}
1260         {\string\setleading:\space
1261          \string\typogfontsize <= 0}
1262         {Maybe\space \string\typogfontsize\space
1263          is\space uninitialized?}
1264     }
1265   }
1266   \ignorespaces
1267 }
1268

```

\setleadingpercentage

```

1269 \cs_new:Npn \setleadingpercentage #1
1270 {
1271   \AfterPreamble{
1272     \dim_compare:nNnTF {\typogfontsize} > {\c_zero_dim}
1273     {
1274       \typog@setbaselineskip{
1275         \fp_eval:n {1 + (#1) / 100} \typogfontsize}
1276     }
1277     {
1278       \PackageError{typog}
1279         {\string\setleadingpercentage:\space

```

```

1280         \string\typogfontsize <= 0}
1281     {Maybe\space \string\typogfontsize\space
1282       is\space uninitialized?}
1283   }
1284 }
1285 \ignorespaces
1286 }
1287 \ExplSyntaxOff
1288

```

A.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.

```

1289 \ExplSyntaxOn
1290 \cs_new_eq:NN \typog@repeat \prg_replicate:nn
1291

```

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

```

1292 \newcommand*{\typog@mod}[2]{\int_mod:nn{#1}{#2}}
1293 \ExplSyntaxOff
1294

```

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

```

1295 \newcommand*{\typog@triplet@max@lines}{99}
1296

```

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

```

1297 \define@key[typog]{smoothraggedrightshapetriplet}{leftskip}%
1298     {\def\typog@@triplet@leftskip{#1}}
1299 \define@key[typog]{smoothraggedrightshapetriplet}{parindent}%
1300     {\def\typog@@triplet@parindent{#1}}
1301 \NewDocumentEnvironment{smoothraggedrightshapetriplet}{0}{m m m}
1302   {\def\typog@@triplet@leftskip{\z@}%
1303    \def\typog@@triplet@parindent{\z@}%
1304    \setkeys*{typog}{smoothraggedrightshapetriplet}{#1}%
1305    \skip0=\typog@@triplet@leftskip\relax
1306    \skip1=#2\relax
1307    \skip2=#3\relax
1308    \skip3=#4\relax
1309    \typog@debug@typeout{smoothraggedrightshapetriplet: skip0=\the\skip0}%
1310    \typog@debug@typeout{smoothraggedrightshapetriplet: skip1=\the\skip1}%
1311    \typog@debug@typeout{smoothraggedrightshapetriplet: skip2=\the\skip2}%
1312    \typog@debug@typeout{smoothraggedrightshapetriplet: skip3=\the\skip3}%
1313    \unless\ifnum\typog@mod{\typog@triplet@max@lines}{3}=0
1314      \PackageError{typog}
1315        {Line number of triplet generator\space

```

```

1316             (\typog@triplet@max@lines) not divisible by 3}
1317             {}
1318     \fi
1319     \edef\typog@@triplet@linespecs{%
1320         \glueexpr \skip0 + \typog@@triplet@parindent\relax
1321         \glueexpr \skip1 - \typog@@triplet@parindent\relax
1322         \skip0 \skip2 \skip0 \skip3
1323         \typog@repeat{\numexpr\typog@triplet@max@lines / 3 - 1}
1324         {\skip0 \skip1 \skip0 \skip2 \skip0 \skip3}}
1325     \parshape=\typog@triplet@max@lines\typog@@triplet@linespecs\relax
1326     {\par}
1327

```

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

```

1328 \newcommand*{\typog@quintuplet@max@lines}{95}
1329

```

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

```

1330 \define@key[typog]{smoothraggedrightshapequintuplet}{leftskip}{%
1331     {\def\typog@@quintuplet@leftskip{#1}}}
1332 \define@key[typog]{smoothraggedrightshapequintuplet}{parindent}{%
1333     {\def\typog@@quintuplet@parindent{#1}}}
1334 \NewDocumentEnvironment{smoothraggedrightshapequintuplet}{0}{m m m m m}{
1335     {\def\typog@@quintuplet@leftskip{\z@}%
1336     \def\typog@@quintuplet@parindent{\z@}%
1337     \setkeys*{typog}{smoothraggedrightshapequintuplet}{#1}%
1338     \skip0=\typog@@quintuplet@leftskip
1339     \skip1=#2\relax
1340     \skip2=#3\relax
1341     \skip3=#4\relax
1342     \skip4=#5\relax
1343     \skip5=#6\relax
1344     \typog@debug@typeout{smoothraggedrightshapequintuplet: skip0=\the\skip0}%
1345     \typog@debug@typeout{smoothraggedrightshapequintuplet: skip1=\the\skip1}%
1346     \typog@debug@typeout{smoothraggedrightshapequintuplet: skip2=\the\skip2}%
1347     \typog@debug@typeout{smoothraggedrightshapequintuplet: skip3=\the\skip3}%
1348     \typog@debug@typeout{smoothraggedrightshapequintuplet: skip4=\the\skip4}%
1349     \typog@debug@typeout{smoothraggedrightshapequintuplet: skip5=\the\skip5}%
1350     \unless\ifnum\typog@mod{\typog@quintuplet@max@lines}{5}=0
1351         \PackageError{typog}
1352             {Line number of quintuplet generator\space
1353             (\typog@quintuplet@max@lines) not divisible by 5}
1354         {}
1355     \fi
1356     \edef\typog@@quintuplet@linespecs{%
1357         \glueexpr \skip0 + \typog@@quintuplet@parindent\relax
1358         \glueexpr \skip1 - \typog@@quintuplet@parindent\relax
1359         \skip0 \skip2 \skip0 \skip3 \skip0 \skip4 \skip0 \skip5
1360         \typog@repeat{\numexpr\typog@quintuplet@max@lines / 5 - 1}
1361         {\skip0 \skip1 \skip0 \skip2 \skip0 \skip3 \skip0 \skip4 \skip0 \skip5}
1362     \parshape=\typog@quintuplet@max@lines\typog@@quintuplet@linespecs\relax}

```

```

1363   {\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.
1364 \newcommand*{\typog@septuplet@max@lines}{98}
1365

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

1366 \define@key[typog]{smoothraggedrightshapeseptuplet}{leftskip}{%
1367     {\def\typog@@septuplet@leftskip{#1}}
1368 \define@key[typog]{smoothraggedrightshapeseptuplet}{parindent}{%
1369     {\def\typog@@septuplet@parindent{#1}}
1370 \NewDocumentEnvironment{smoothraggedrightshapeseptuplet}{0}{ m m m m m m m m}
1371 {\def\typog@@septuplet@leftskip{\z@}%
1372  \def\typog@@septuplet@parindent{\z@}%
1373  \setkeys*{typog}{smoothraggedrightshapeseptuplet}{#1}%
1374  \skip0=\typog@@septuplet@leftskip
1375  \skip1=#2\relax
1376  \skip2=#3\relax
1377  \skip3=#4\relax
1378  \skip4=#5\relax
1379  \skip5=#6\relax
1380  \skip6=#7\relax
1381  \skip7=#8\relax
1382  \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip0=\the\skip0}%
1383  \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip1=\the\skip1}%
1384  \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip2=\the\skip2}%
1385  \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip3=\the\skip3}%
1386  \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip4=\the\skip4}%
1387  \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip5=\the\skip5}%
1388  \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip6=\the\skip6}%
1389  \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip7=\the\skip7}%
1390  \unless\ifnum\typog@mod{\typog@septuplet@max@lines}{7}=0
1391      \PackageError{typog}
1392          {Line number of septuplet generator\space
1393           (\typog@septuplet@max@lines) not divisible by 7}
1394      {}
1395  \fi
1396  \edef\typog@@septuplet@linespecs{%
1397      \glueexpr \skip0 + \typog@@septuplet@parindent\relax
1398      \glueexpr \skip1 - typog@@septuplet@parindent\relax
1399      \skip0 \skip2 \skip0 \skip3 \skip0 \skip4 \skip0 \skip5 \
1400      \typog@repeat{\numexpr\typog@septuplet@max@lines / 7 - 1}
1401      {\skip0 \skip1 \skip0 \skip2 \skip0 \skip3 \skip0 \skip4 \s
1402      \parshape=\typog@septuplet@max@lines\typog@@septuplet@linespecs\relax}
1403  {\par}
1404

smoothraggedrightfuzzfactor
1405 \newcommand*{\smoothraggedrightfuzzfactor}{1.0}

```

`\smoothraggedrightgenerator`

```
1406 \newcommand*{\smoothraggedrightgenerator}{triplet}
```

`\smoothraggedrightleftskip`

```
1407 \newlength{\smoothraggedrightleftskip}
```

`\smoothraggedrightparindent`

```
1408 \newlength{\smoothraggedrightparindent}
```

`\smoothraggedrighttragwidth`

```
1409 \newlength{\smoothraggedrighttragwidth}
1410 \setlength{\smoothraggedrighttragwidth}{2em}
1411
```

`\typog@fuzzwidth` (*dimen*)

```
1412 \newdimen{\typog@fuzzwidth}
1413
```

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

```
1414 \define@key[typog]{smoothraggedrightpar}{linewidth}%
1415     {\def\typog@@linewidth{\linewidth}{#1}}
1416
1417 \NewDocumentEnvironment{smoothraggedrightpar}{0{}}
1418     {\edef\typog@@linewidth{\linewidth}%
1419     \setkeys[typog]{smoothraggedrightpar}{#1}%

```

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

```
1420 \edef\typog@@generatorchoice{%
1421     \ifnum\pdf@strcmp{\smoothraggedrightgenerator}{triplet}=\z@
1422     0%
1423     \else
1424     \ifnum\pdf@strcmp{\smoothraggedrightgenerator}{quintuplet}=\z@
1425     1%
1426     \else
1427     \ifnum\pdf@strcmp{\smoothraggedrightgenerator}{septuplet}=\z@
1428     2%
1429     \else
1430     \PackageError{typog}
1431         {smoothraggedright: unknown generator name}
1432         {valid generator names are triplet, quin-
1433         tuplet, and septuplet}%
1433     \fi
1434     \fi
1435     \fi}%

```

Obey to the indentation prescribed by any list environment.

```
1436 \let\typog@@smoothraggedrightleftskip=\smoothraggedrightleftskip
1437 \ifnum\@listdepth>0
1438     \addtolength{\typog@@smoothraggedrightleftskip}{\leftmargin}%
1439 \fi

```


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

```

1440 \typog@fuzzwidth=\smootheraggedrightfuzzfactor\smootheraggedrightragwidth
      Now for the generator-specific code...
1441 \ifcase\typog@@generatorchoice
      generator=triplet produces a »short line – long line – middle length
line« sequence.
1442 \typog@fuzzwidth=.25\smootheraggedrightragwidth
1443 \typog@debug@typeout{smootheraggedright: generator=triplet, ty-
pog@fuzzwidth=\the\typog@fuzzwidth}%
1444 \smootheraggedrightshapetriplet[leftskip=\typog@@smootheraggedrightleftskip,
1445                                parindent=\glueexpr\smootheraggedrightparinden-
indent,
1446                                #1]%
1447 {\glueexpr \typog@@linewidth - \smootheraggedrightragwidth
1448   + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
1449 {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth}% (3)
1450 {\glueexpr (\typog@@linewidth * 2 - \smootheraggedrightrag-
width) / 2
1451   + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (2)
1452 \or
      generator=quintuplet.
1453 \typog@fuzzwidth=.125\smootheraggedrightragwidth
1454 \typog@debug@typeout{smootheraggedright: generator=quintuplet, ty-
pog@fuzzwidth=\the\typog@fuzzwidth}%
1455 \smootheraggedrightshapequintuplet[leftskip=\typog@@smootheraggedrightleftskip
1456                                parindent=\glueexpr\smootheraggedrightparinden-
indent,
1457                                #1]%
1458 {\glueexpr (\typog@@linewidth * 4 - \smootheraggedrightrag-
width * 3) / 4
1459   + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (2)
1460 {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth\relax}% (5)
1461 {\glueexpr (\typog@@linewidth * 2 - \smootheraggedrightrag-
width) / 2
1462   + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (3)
1463 {\glueexpr (\typog@@linewidth * 4 - \smootheraggedrightrag-
width) / 4
1464   + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (4)
1465 {\glueexpr \typog@@linewidth - \smootheraggedrightragwidth
1466   + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
1467 \or
      generator=septuplet.
      Permutation 3 – 6 – 1 – 5 – 2 – 7 – 4 looks »random« enough for our purposes.
1468 \typog@fuzzwidth=.08333\smootheraggedrightragwidth

```

```

1469 \typog@debug@typeout{smoothraggedright: generator=septuplet, ty-
pog@fuzzwidth=\the\typog@fuzzwidth}%
1470 \smoothraggedrightshapeseptuplet[leftskip=\typog@@smoothraggedrightleftskip,
1471 parindent=\glueexpr\smoothraggedrightparind
indent,
1472 #1]%
1473 {\glueexpr (\typog@@linewidth * 3 - \smoothraggedrightrag-
width * 2) / 3
1474 + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (3)
1475 {\glueexpr (\typog@@linewidth * 6 - \smoothraggedrightrag-
width) / 6
1476 + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (6)
1477 {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth +
1478 + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
1479 {\glueexpr (\typog@@linewidth * 3 - \smoothraggedrightrag-
width) / 3
1480 + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (5)
1481 {\glueexpr (\typog@@linewidth * 6 - \smoothraggedrightrag-
width * 5) / 6
1482 + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (2)
1483 {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth\relax}% (7)
1484 {\glueexpr (\typog@@linewidth * 2 - \smoothraggedrightrag-
width) / 2
1485 + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (4)
1486 \fi}
1487 {\ifcase\typog@@generatorchoice
1488 \endsmoothraggedrightshapetriplet
1489 \or
1490 \endsmoothraggedrightshapequintuplet
1491 \or
1492 \endsmoothraggedrightshapeseptuplet
1493 \fi}
1494

```

smoothraggedright (*env.*)

```

1495 \NewDocumentEnvironment{smoothraggedright}{0{}}
1496 {\PushPostHook{par}{\hskip-\parindent\smoothraggedrightpar[#1]\relax}}
1497 {\par\PopPostHook{par}}
1498

```

B typog-grep

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

B.1 NAME

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

B.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*.

B.3 DESCRIPTION

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

In the L^AT_EX source file the user brackets her text or code in a **typoginspect** environment:

```
\begin{typoginspect}{ID}
TEXT-OR-CODE-TO-INVESTIGATE
\end{typoginspect}
```

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

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

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

where all the capital-letter sequences are meta-variables and in particular *JOB-NAME* is the expansion of `\jobname`, *LINE-NUMBER* is the L^AT_EX source file line number of the beginning of the `typoginspect` environment, and *PAGE-NUMBER* is the page where the output of *TEXT-OR-CODE-TO-INVESTIGATE* occurs.

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

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

B.4 OPTIONS

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

-a, --all, --any

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

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

--color, colour WHEN

Colorize specific log contents for the matching *IDs*. The argument *WHEN* determines when to apply color: `always`, `never`, or `auto`. The setting `auto` checks whether standard output has been redirected. This is the default.

-C, --config *KEY=VALUE[:KEY=VALUE[:...]]*

Set one or more configuration *KEY* to *VALUE* pairs. See section B.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 B.5.2.

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

-h, --help

Display brief help then exit.

-i, --[no-]id

Print the actual ID-name that matched *REGEXP*. Control the appearance of the matching *ID* with configuration item *id-heading*.

-y, --[no-]ignore-case

Match *IDs* while ignoring case distinctions in patterns and data.

-j, --[no-]job-name

Print the `\jobname` that *latex* associated with the input file.

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

Print the line number where the `typoginspect` environment was encountered in the *L^AT_EX* source file.

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

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

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

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

-P, --[no-]pager

Redirect output from *stdout* to the configured pager.

--show-config

Show the default configuration and exit.

--show-encodings

Show all known encodings and exit.

-V, --version

Show version information and exit.

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

Match only whole words.

B.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.

B.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`

B.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

B.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.

B.7 CAVEATS

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

B.8 SEE ALSO

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

v0.1	General: Initial version. i	\setleadingpercentage: New macro. 88
v0.2	\narrowsspace: New macro. 74	\typogfontsize: New dimen. . 87
	\widespace: Add fallback if fontdimen7 is zero. Extend with a starred version. 73	v0.4
		\lowercaseadjustlabelitems: New macro. 67
		\noadjustlabelitems: New macro. 67
v0.3	hyphenmin: New environment. .. 58	\typogadjuststairs: New macro. 70
	\resetbaselineskip: New macro. 87	\typoglowercaseadjustcheck: New macro. 71
	\setbaselineskip: New macro. 87	\typoguppercaseadjustcheck: New macro. 70
	\setbaselineskippercentage: New macro. 87	\uppercaseadjustlabelitems: New macro. 66
	\setleading: New macro. 88	

References

- [1] ABRAHAMS, PAUL W., HARGREAVES, KATHRYN A., and KARL BERRY. *T_EX for the Impatient*. 2020, <http://tug.ctan.org/info/impatient/book.pdf>.
- [2] AMERICAN MATHEMATICAL SOCIETY and the L^AT_EX3 PROJECT TEAM. *Package amsmath*. 2020, <https://ctan.org/pkg/amsmath>.
- [3] ARSENEAU, DONALD. *Package cite*. 2015, <https://ctan.org/pkg/cite>.
- [4] BEZOS, JAVIER. *Package enumitem*. 2019, <https://ctan.org/pkg/enumitem>.
- [5] BEZOS, JAVIER. *Package babel*. 2021, <https://ctan.org/pkg/babel>.
The original author of package babel was J. L. BRAAMS.
- [6] BREITENLOHNER, PETER and the N_TS TEAM. *ε-T_EX*. 1998, https://mirrors.ctan.org/systems/doc/etex/etex_man.pdf.
- [7] BURNOL, JEAN-FRANÇOIS. *Package mathastext*. 2023, <https://ctan.org/pkg/mathastext>.
- [8] CARLISLE, DAVID. *Russian Paragraph Shapes*. Baskerville, 6(1), 13–15, 1996, <http://uk-tug-archive.tug.org/wp-installed-content/uploads/2008/12/61.pdf>.
- [9] CARLISLE, DAVID. *What do different \fontdimen<num> mean*. 2013–1–2, <https://tex.stackexchange.com/questions/88991/what-do-different-fontdimennum-mean>.
- [10] CUBITT, TOBY. *Package cleveref*. 2018, <https://ctan.org/pkg/cleveref>.
- [11] EIJKHOUT, VICTOR. *T_EX By Topic, A Texnician's Reference*. 2007, <https://www.eijkhout.net/tex/tex-by-topic.html>.
- [12] HØGHOLM, MORTEN, MADSEN, LARS and the L^AT_EX3 PROJECT TEAM. *Package mathtools*. 2020, <https://ctan.org/pkg/mathtools>.
- [13] KHIRIVICH, SIARHEI. *Tips on Writing a Thesis in L^AT_EX*. 2013, <http://www.khirevich.com/latex/microtype>.
- [14] KNUTH, DONALD ERVIN. *The T_EXbook*. Addison Wesley, Reading/MA, 1986.
- [15] THE L^AT_EX PROJECT. *Package l3experimental*. 2024, <https://ctan.org/pkg/l3experimental>.
- [16] 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.
- [17] MIDDENDORP, JAN. *Shaping Text*. BIS publishers, Amsterdam, 2014.

- [18] 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>.
- [19] MITTELBACH, FRANK. *Package widows-and-orphans*. 2020, <https://ctan.org/pkg/widows-and-orphans>.
- [20] RAHTZ, SEBASTIAN, and FRANK MITTELBACH. *Package hyperref*. 2020, <https://ctan.org/pkg/hyperref>. The package is maintained by the L^AT_EX3 Project Team.
- [21] SCHLICHT, ROBERT. *Package microtype*. 2020, <https://ctan.org/pkg/microtype>.
- [22] SCHRÖDER, MARTIN. *Package ragged2e*. 2019, <https://ctan.org/pkg/ragged2e>.
- [23] SOLOMON, DAVID. *Output Routines: Examples and Techniques. Part I: Introduction and Examples*. TUGboat, 11(1), 69–85, 1990, <http://www.tug.org/TUGboat/Articles/tb11-1/tb27salomon.pdf>.
- [24] STRIZVER, ILENE. *Type rules!: the designer’s guide to professional typography*, 4th ed. John Wiley & Sons, Hoboken/NJ, 2014.
- [25] TOBIN, GEOFFREY, and ROBIN FAIRBAIRNS. *Package setspace*. 2011, <https://ctan.org/pkg/setspace>.
- [26] UMEKI, HIDEO. *Package geometry*. 2020, <https://ctan.org/pkg/geometry>.
- [27] WERMUTH, UDO. *Tracing paragraphs*. TUGboat, 37(3), 358–373, 2016, <https://tug.org/TUGboat/tb37-3/tb117wermuth.pdf>.
- [28] WERMUTH, UDO. *The optimal value for \emergencystretch*. TUGboat, 38(1), 65–86, 2017, <https://tug.org/TUGboat/tb38-1/tb118wermuth.pdf>.
- [29] WERMUTH, UDO. *A note on \linepenalty*. TUGboat, 38(3), 400–414, 2017, <https://tug.org/TUGboat/tb38-3/tb120wermuth.pdf>.
- [30] WERMUTH, UDO. *Experiments with \parfillskip*. TUGboat, 39(3), 276–303, 2018, <https://tug.org/TUGboat/tb39-3/tb123wermuth-parfillskip.pdf>.
- [31] WERMUTH, UDO. *An attempt at ragged-right typesetting*. TUGboat, 41(1), 73–94, 2020, <https://tug.org/TUGboat/tb41-1/tb127wermuth-ragged.pdf>.
- [32] WERMUTH, UDO. Personal communication. August 2, 2022.
- [33] WERMUTH, UDO. *Vertical alignments in plain T_EX*. TUGboat, 44(3), 427–440, 2023, <https://tug.org/TUGboat/tb44-3/tb138wermuth-valign.pdf>.

- [34] WILSON, PETER. *Package hyphenat*. 2004, <https://ctan.org/pkg/hyphenat>. The package is maintained by W. ROBERTSON.
- [35] WILSON, PETER. *Glisterings*. TUGboat, 28(2), 229–232, 2007, <https://tug.org/TUGboat/tb28-2/tb89glister.pdf>.
- [36] WILSON, PETER. *Package needspace*. 2010, <https://ctan.org/pkg/needspace>. The package is maintained by W. ROBERTSON.

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used. Numbers typeset in boldface name pages in the user's manual.

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

A

`\allowhyphenation`, [10](#), [178](#)
`amsmath` (package), [41](#)

B

baseline skip, [43](#)
 bookmark, [13](#)
`breakabledisplay` (env.), [41](#), [1138](#)
`\breakpoint`, [12](#), [182](#)
`\breakpoint*`, [12](#)

C

`\capitaldash`, [18](#), [314](#), [322](#), [344](#)
`\capitaldash*`, [18](#)
`\capitalemdash`, [18](#), [325](#)
`\capitalemdash*`, [18](#)
`\capitalendash`, [18](#), [309](#)
`\capitalendash*`, [18](#)
`\capitalhyphen`, [17](#), [295](#)
`\capitalhyphen*`, [17](#)
`\capitaltimes`, [19](#), [346](#)
 configuration, [5](#)
`covernextindentpar` (env.), [30](#), [632](#)
`csquotes` (package), [21](#)

D

debug (option), [2](#)
 dimensions:
 `\typogfontsize`, [44](#), [1224](#),
 [1235](#), [1238](#), [1243](#), [1244](#), [1254](#),
 [1256](#), [1261](#), [1262](#), [1272](#), [1275](#),
 [1280](#), [1281](#)
`\Doubleguillemetleft`, [20](#), [380](#)
`\doubleguillemetleft`, [20](#), [364](#)
`\Doubleguillemetright`, [20](#), [384](#)
`\doubleguillemetright`, [20](#), [368](#)

E

enumitem-keys:
 `vtietop`, [39](#)
 environments:
 `breakabledisplay`, [41](#), [1138](#)
 `covernextindentpar`, [30](#), [632](#)
 `hyphenmin`, [12](#), [192](#)

`lastlinecenteredpar`, [26](#), [608](#)
`lastlineflushrightpar`, [26](#), [605](#)
`lastlinerraggedleftpar`, [26](#), [599](#)
`loosespacing`, [31](#), [694](#)
`nocharprotrusion`, [36](#), [863](#)
`nofontexpand`, [860](#)
`nofontexpansion`, [36](#), [854](#)
`openlastlinepar`, [30](#), [646](#)
`prolongpar`, [29](#), [622](#)
`setfontexpand`, [35](#), [840](#)
`setfontshrink`, [35](#), [812](#)
`setfontstretch`, [35](#), [826](#)
`setfonttracking`, [34](#), [726](#)
`shortenpar`, [29](#), [615](#)
`slightlyloppypar`, [37](#), [934](#)
`smoothraggedright`, [47](#), [1495](#)
`smoothraggedrightpar`, [46](#), [1414](#)
`smoothraggedrightshape-`
 `quintuplet`, [46](#), [1330](#)
`smoothraggedrightshape-`
 `septuplet`, [46](#), [1366](#)
`smoothraggedrightshape-`
 `triplet`, [46](#), [1297](#)
`tightspacing`, [31](#), [710](#)
`typoginspect`, [6](#), [139](#)
`typoginspectpar`, [6](#), [166](#)
`typogsetup`, [5](#), [99](#)
`vtiebotdisp`, [39](#), [1123](#)
`vtiebotdisptoppar`, [39](#), [1127](#)
`vtiebotpar`, [39](#), [1065](#)
`vtietoppar`, [38](#), [998](#)

F

`\figuredash`, [18](#), [339](#)
`\figuredash*`, [18](#)
 font
 ADF Baskervald, [20](#)
 ADF Accanthis, [25](#)
 ADF Venturis, [19](#), [25](#)
 Alegreya, [19](#)
 Arvo, [19](#)
 Bitter, [19](#)
 Clara, [19](#)
 CM Roman, [25](#)

Cochineal, [25](#)
 Coelacanth, [19](#)
 Crimson Pro, [19](#)
 Crimson Text, [19](#)
 Domitian, [25](#)
 Droid Serif, [19](#)
 EB Garamond, [19](#), [25](#)
 EB Garamond, [20](#)
 encoding, [19](#)
 Erewhon, [19](#)
 etbb, [25](#)
 expansion, [29](#), [35](#)
 Extended Charter, [25](#)
 fbb, [19](#)
 Gentium, [19](#), [20](#), [25](#)
 GFS Artemisia, [19](#)
 GFS Artemisia, [20](#)
 GFS Bodoni, [25](#)
 GFS Didot, [25](#)
 GFS Didot, [20](#)
 Ibarra Real Nova, [19](#)
 IBM Plex Serif, [25](#)
 information, [6](#)
 INRIA Serif, [19](#)
 KP Serif, [25](#)
 Libertine, [19](#)
 Libertinus Serif, [19](#), [20](#), [25](#)
 Libre Baskerville, [19](#)
 Libre Caslon, [19](#)
 Merriweather, [19](#), [20](#)
 ML Modern, [25](#)
 protrusion, [36](#)
 PT Serif, [19](#)
 Quattrocento, [19](#)
 Roboto Slab, [19](#)
 size, [6](#)
 Source Serif Pro, [19](#), [25](#)
 spacing, [28](#), [30](#)
 loose, [30](#)
 tight, [30](#)
 Spectral, [19](#), [25](#)
 STIX, [19](#), [25](#)
 T_EX Gyre Pagella, [19](#)
 tracking, [29](#), [34](#)
 TX Fonts, [19](#)
 URW Palladio, [ii](#), [25](#)
 Utopia Regular, [25](#)
 \fontsizeinfo, [6](#), [118](#)
 forlorn line
 club, [29](#), [38](#)
 display widow, [39](#)
 orphan, [29](#)

widow, [29](#), [39](#)

H

\hskip, [12](#)
 hyperref (package), [13](#)
 hyphenation, [10](#)
 empty discretionary, [12](#)
 first word, [12](#)
 re-enable automatic, [10](#)
 hyphenmin (env.), [12](#), [192](#)

I

information, [6](#)
 \itcorr, [11](#), [14](#), [234](#)
 \itcorr*, [14](#)

K

\kernedhyphen, [16](#), [257](#), [279](#), [280](#),
 [287](#), [288](#)
 \kernedhyphen*, [16](#)
 \kernedslash, [16](#), [116](#), [247](#)
 \kernedslash*, [16](#)
 kerning
 extra, [15](#)
 forward slash, [16](#)
 hyphen, [16](#)
 ligature, [13](#)

L

label items, [21](#)
 lastlinecenteredpar (env.), [26](#), [608](#)
 lastlineflushrightpar
 (env.), [26](#), [605](#)
 lastlineraggedleftpar
 (env.), [26](#), [599](#)
 leading, [43](#)
 \leftkernedhyphen, [16](#), [277](#), [282](#)
 \leftkernedhyphen*, [16](#)
 ligature, [13](#)
 ligaturekern (option), [2](#)
 line spacing, [6](#)
 list, [38](#)
 loosespacing (env.), [31](#), [694](#)
 \lowercaseadjustlabelitems, [21](#),
 [480](#)
 lowercaselabelitemadjustments (op-
 tion), [2](#)

M

mathitalicscorrection (option), [2](#)
 microtype (package), [33](#)

N

narrow space, [33](#)
`\narrow space`, [33](#), [680](#)
`\narrow space*`, [33](#)
`\narrow spacescale`, [679](#), [682](#),
[684](#), [690](#), [691](#)
`\narrow space strength`, [678](#), [687](#)
`\noadjustlabel items`, [21](#), [487](#)
`nocharprotrusion` (env.), [36](#), [863](#)
`nodebug` (option), [2](#)
`nofontexpand` (env.), [860](#)
`nofontexpansion` (env.), [36](#), [854](#)
`\nolig`, [13](#), [199](#)
`\nolig*`, [13](#)

O

`openlastlinepar` (env.), [30](#), [646](#)

P

package option
 `debug`, [2](#)
 `ligaturekern`, [2](#)
 `lowercaselabelitemadjustments`, [2](#)
 `mathitalicscorrection`, [2](#)
 `nodebug`, [2](#)
 `penalty`, [2](#)
 `raise*`, [3](#)
 `raisecapitaldash`, [3](#)
 `raisecapitalguillemets`, [3](#)
 `raisecapitalhyphen`, [3](#)
 `raisecapitaltimes`, [3](#)
 `raisefiguredash`, [3](#)
 `raiseguillemets`, [3](#)
 `shrinklimits`, [3](#)
 `slashkern`, [4](#)
 `stretchlimits`, [3](#)
 `textitalicscorrection`, [4](#)
 `trackingttspacing`, [4](#)
 `uppercaselabelitemadjustments`, [4](#)
package options, [2–4](#)
page break, [9](#), [41](#)
paragraph
 align last line, [26](#)
 centered, [26](#)
 flush right, [26](#)
 badness, [8](#)
 fill last line, [26](#)
 covernextindentpar, [30](#)
 `\linebreak`, [28](#)
 `\mbox`, [28](#)
 `openlastlinepar`, [30](#)
 `prolongpar`, [29](#)

`shortenpar`, [29](#)
 tie, [28](#)
 sloppy, [37](#)
 vertically tied, [38](#)

PDF, [13](#)

`penalty` (option), [2](#)

`prolongpar` (env.), [29](#), [622](#)

R

ragged right, [46](#)
`raise*` (option), [3](#)
`raisecapitaldash` (option), [3](#)
`raisecapitalguillemets` (option), [3](#)
`raisecapitalhyphen` (option), [3](#)
`raisecapitaltimes` (option), [3](#)
raised character, [17](#)
 en-dash, [18](#)
 guillemets, [19](#)
 hyphen, [17](#)
 multiplication sign, [19](#)
 number dash, [18](#)
`raisefiguredash` (option), [3](#)
`raiseguillemets` (option), [3](#)
reconfigure, [5](#)
`\resetbaselineskip`, [43](#), [1219](#)
`\rightkernedhyphen`, [16](#), [285](#), [290](#)
`\rightkernedhyphen*`, [16](#)

S

`\setbaselineskip`, [43](#), [1151](#), [1153](#),
[1160](#), [1169](#), [1179](#), [1197](#), [1206](#),
[1209](#), [1213](#)
`\setbaselineskippercentage`, [43](#),
[1232](#)
`setfontexpand` (env.), [35](#), [840](#)
`setfontshrink` (env.), [35](#), [812](#)
`setfontstretch` (env.), [35](#), [826](#)
`setfonttracking` (env.), [34](#), [726](#)
`\setleading`, [44](#), [1251](#)
`\setleadingpercentage`, [44](#), [1269](#)
setup, [5](#)
`shortenpar` (env.), [29](#), [615](#)
`shrinklimits` (option), [3](#)
`\Singleguillemetleft`, [20](#), [372](#)
`\singleguillemetleft`, [20](#), [356](#)
`\Singleguillemetright`, [20](#), [376](#)
`\singleguillemetright`, [20](#), [360](#)
`slashkern` (option), [4](#)
`\slightlyloppy`, [37](#), [877](#), [935](#)
`\slightlyloppypar` (env.), [37](#), [934](#)
`smoothraggedright` (env.), [47](#), [1495](#)

`\smootheraggedrightfuzzfactor`, [1405](#),
[1440](#)
`\smootheraggedrightgenerator`, [1406](#),
[1421](#), [1424](#), [1427](#)
`\smootheraggedrightleftskip`, [1407](#),
[1436](#)
`smootheraggedrightpar` (env.), [46](#),
[1414](#)
`\smootheraggedrightparindent`, [1408](#),
[1445](#), [1456](#), [1471](#)
`\smootheraggedrightragwidth`, [1409](#),
[1440](#), [1442](#), [1447](#), [1450](#), [1453](#),
[1458](#), [1461](#), [1463](#), [1465](#), [1468](#),
[1473](#), [1475](#), [1477](#), [1479](#), [1481](#), [1484](#)
`smootheraggedrightshape-`
`quintuplet` (env.), [46](#), [1330](#)
`smootheraggedrightshape-`
`septuplet` (env.), [46](#), [1366](#)
`smootheraggedrightshape-`
`triplet` (env.), [46](#), [1297](#)
`\splicevtietop`, [38](#), [1003](#)
`stretchlimits` (option), [3](#)

T

`textitalicscorrection` (option), [4](#)
`tightspacing` (env.), [31](#), [710](#)
`trackingttspace` (option), [4](#)
`\typogadjuststairs`, [23](#), [549](#)
`\typogadjuststairsfor`, [527](#),
[568](#), [569](#), [570](#), [571](#)
`\typogfontsize` (dim.), [44](#),
[1224](#), [1235](#), [1238](#), [1243](#),
[1244](#), [1254](#), [1256](#), [1261](#),
[1262](#), [1272](#), [1275](#), [1280](#), [1281](#)

`\typogget`, [5](#), [108](#)
`typoginspect` (env.), [6](#), [139](#)
`typoginspectpar` (env.), [6](#), [166](#)
`\typoglogo`, [11](#)
`\typoglowercaseadjustcheck`, [24](#),
[592](#)
`typogsetup` (env.), [5](#), [99](#)
`\typoguppercaseadjustcheck`, [24](#),
[580](#)

U

`\uppercaseadjustlabelitems`, [21](#),
[473](#)
`uppercaselabelitemadjustments` (op-
tion), [4](#)

V

`\vtiebot`, [39](#), [1012](#), [1066](#)
`\vtiebotdisp`, [1133](#)
`vtiebotdisp` (env.), [39](#), [1123](#)
`vtiebotdisptoppar` (env.), [39](#), [1127](#)
`vtiebotpar` (env.), [39](#), [1065](#)
`\vtietop`, [38](#), [945](#), [999](#), [1005](#), [1131](#)
`vtietop` (enumitem-key), [39](#)
`vtietoppar` (env.), [38](#), [998](#)

W

`wide space`, [32](#)
`\widespace`, [32](#), [664](#)
`\widespace*`, [32](#)
`\widespacescale`, [663](#), [666](#), [668](#),
[674](#), [675](#)
`\widespacestrength`, [662](#), [671](#)