

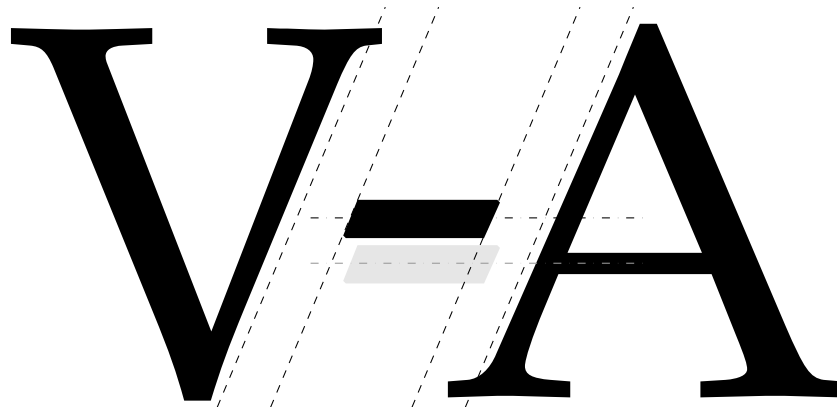
TypoG – Typographic Fine-Tuning

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 7
- 3.3 Hyphenation 10
- 3.4 Disable/Break Ligatures 14
- 3.5 Manual Italic Correction 15
- 3.6 Apply Extra Kerning 16
 - 3.6.1 Slash 16
 - 3.6.2 Hyphen 17
- 3.7 Raise Selected Characters 17
 - 3.7.1 Capital Hyphen 18
 - 3.7.2 Capital Dash 18
 - 3.7.3 Number Dash (Figure Dash) 19
 - 3.7.4 Multiplication Sign 20
 - 3.7.5 Guillemets 20
- 3.8 Vertically Adjust Label Items 22
- 3.9 Align Last Line 25
- 3.10 Fill Last Line 27
 - 3.10.1 Problem Definition 27
 - 3.10.2 Manual Changes 29
 - 3.10.3 Multi-Purpose Environments 30
 - 3.10.4 Specialized Environments 30

Table of Contents continued on next page.

- 3.11 Spacing [31](#)
 - 3.11.1 Looser/Tighter [31](#)
 - 3.11.2 Wide Space [32](#)
 - 3.11.3 Narrow Space [33](#)
- 3.12 Microtype Front-End [34](#)
 - 3.12.1 Tracking [35](#)
 - 3.12.2 Font Expansion [35](#)
 - 3.12.3 Character Protrusion [37](#)
- 3.13 Sloppy Paragraphs [37](#)
- 3.14 Vertically Partially-Tied Paragraphs [39](#)
- 3.15 Breakable Displayed Equations [42](#)
- 3.16 Setspace Front-End [44](#)
- 3.17 Smooth Ragged [47](#)

4

Other Packages for Fine L^AT_EX Typography [51](#)

A

Package Code [52](#)

- | | |
|--------------------------------------------------|----------------------------------------------|
| A.1 Setup and Reconfiguration 56 | A.10 Fill Last Line 73 |
| A.2 Information 57 | A.11 Spacing 75 |
| A.3 Hyphenation 59 | A.12 Microtype Front-End 76 |
| A.4 Disable/Break Ligatures 60 | A.13 Sloppy Paragraphs 80 |
| A.5 Manual Italic Correction 61 | A.14 Vert. Tie Paragraphs 82 |
| A.6 Apply Extra Kerning 62 | A.15 Breakable Disp. Eqs. 87 |
| A.7 Raise Selected Characters 63 | A.16 Setspace Front-End 87 |
| A.8 Vert. Adjust Label Items 66 | A.17 Smooth Ragged 91 |
| A.9 Align Last Line 73 | |

B

typog-grep [97](#)

Change History [104](#)

References [105](#)

Index [108](#)

List of Tables

- 1 Hyphens and automatic hyphenation 12
- 2 Suggested raise amounts for `\figuredash` 19
- 3 Suggested raise amounts for guillemets 21
- 4 Label item adjustment suggestions 26
- 5 Spacing changes made by `loosespacing` 32
- 6 Spacing changes made by `tightspacing` 32
- 7 `\fontdimen<number>` parameters 33
- 8 Comparison of some space sizes 34
- 9 Shrink values of `setfontshrink` 36
- 10 Stretch values of `setfontstretch` 36
- 11 Shrink and stretch values of `setfontexpand` 36
- 12 Parameter adjustments of `\slightlyloppy` 38
- 13 Partial paragraph line counts 42
- 14 Env. `breakabledisplay` and `\interdisplaylinepenalty` 43

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: $\frac{33}{1000}$ 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. 26) 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 `\capitaltimes`. 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 $\frac{1}{48}$ 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: $\frac{5}{1000}$ 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: $\frac{2}{1000}$ 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. 26) and also configuration option `lowercaseadjustlabelitem`.

All four lengths default to 0 pt.

Important

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

3 Macros and Environments

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

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

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

3.1 Setup and Reconfiguration

`typogsetup` (*env.*)

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

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

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

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

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

Note

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

Use Cases

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

`\typogget`

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

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

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

Use Case

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

```

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

```

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

`\typoggetnth`
SINCE v0.4

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

```
\typoggetnth{<key>}{<index>}
```

Retrieve the configuration value – which is a comma-separated list – that is associated with `<key>` and return the item having the position `<index>`. Index into the list either from left-to-right with positive indices starting at 1 up to the length of the list, or from right-to-left with negative indices starting at `-1` down to the negative length.

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

3.2 Information

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

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

Note

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

Use Cases

Colophon. ¶ Font test pages. ■

3.2.2 Paragraph- and Page-Breaking Trace

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

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

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

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

Option**tracingboxes**[=`\langle size \rangle`]

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

Caution

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

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

```
<typog-inspect_id="\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.

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).

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


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

or

```
perl -ne '$a=0 if /<\/typog-inspect>/; \
          print $_ if $a; \
          $a=1 if /<typog-inspect_id="..."/' \
          < jobname.log
```
- The companion program `typog-grep` is tailored to extract the information marked up by `typoginspect` and `typoginspectpar` even if the environments are nested.

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

Tips

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

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

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

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

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

to the 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

$$@@\langle breakpoint\text{-}number \rangle \text{ line } \langle line\text{-}number \rangle . \langle suffix \rangle$$

of each line in the paragraph, where for $\langle suffix \rangle$ the following mapping holds [14, p. 99]:

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

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

Note

When package `microtype`’s font expansion 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.,

$$\backslash T1/erewhon-LF/m/n/9/@/@ (-13) \dots$$

or

$$\backslash T1/erewhon-LF/m/n/9/@/@/10ls (+7) \dots$$

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

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

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

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

Investigating page-breaks. Use `\tracingpages=1` or the `typoginspect` environment to switch on tracing of T_EX's page-break information [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.
2. The current goal height `g` is 553.9 pt. If the initial goal height was 598 pt we can deduce that some space for other vertical material was subtracted.
3. The badness `b` of this vertical break is horrendous which is expected for the first lines on a page since breaks so early are rightfully considered infinitely bad.
4. The penalty `p` at this point actually is a bonus.
5. As the badness is 10000 the cost for a break is calculated to 100000. ■

3.3 Hyphenation

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

`\allowhyphenation`

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

```
\allowhyphenation
```

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

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

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

Use Cases

All examples from the bottom of Tab. 1 on p. 12. ¶

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

$\text{Halbgruppe, Transformations}\text{\allowhyphenation}\text{\mbox{-}}\text{\,---}$

The first part, $\text{\>Transformations}\text{\<}$ 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 :

$\text{Einslement}\text{\allowhyphenation}\text{\rlap{,}}\text{\footnote{...}}$

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

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

$\text{\marginpar}\text{\allowhyphenation Kontakttransformationen}$

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

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

Let us assume we mark up proper names with

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

and we want to have hyphenatable $\text{\>ABELSche Gruppe}\text{\<}$ or $\text{\>EUKLIDischer Vektorraum}\text{\<}$ without dropping the markup. To that end we define commands that insert a hyphenation point at the right place:

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

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.

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

Tip – Typewriter Fonts

Sometimes it is desired to get a hyphenatable typewriter font. L^AT_EX suppresses any hyphenation for fonts in `\ttfamily` by un-defining their `\hyphenchar`s. 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 L^AT_EX macro `\allowbreak` is not only that the former has a starred form, but the penalty associated with `\breakpoint` is the current¹⁰ `\exhyphenpenalty`, whereas `\allowbreak` statically assigns a zero penalty.

Use Case

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

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

`hyphenmin (env.)`
SINCE V0.3

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

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

Without optional argument `hyphenmin` sets both `\lefthyphenmin` and `\righthyphenmin` to `⟨hyphen-minimum⟩`. When called with an optional argument it sets `\lefthyphenmin` to `⟨left-hyphen-minimum⟩` and `\righthyphenmin` to `⟨hyphen-minimum⟩`.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 `⟨hyphen-minimum⟩` to disable hyphenation. ■

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

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

3.4 Disable/Break Ligatures

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

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

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

Use Cases

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

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

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

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

Important – hyperref bookmarks

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

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

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

L^AT_EX will 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` The italic correction offered by \TeX or \LaTeX sometimes needs a helping hand.
`\itcorr*`

```
\itcorr{<strength>}
\itcorr*{<strength>}
```

In text mode macro `\itcorr` inserts a kern whose width is proportional to `\fontdim1`, which is the font's italic correction. If `\fontdim1` happens to be zero (e.g. for an upright font), `\itcorr` uses the value set with `textitalicscorrection` instead of `\fontdim1`. The starred version always uses `textitalicscorrection`. In math mode macro `\itcorr` uses the value set with `mathitalicscorrection`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: `<strength>` can be negative and fractional `<strength>`s are allowed.

Use Cases

Stronger or weaker correction than `\.` ¶ Correct a non-slanted or non-italicized font. ¶ Negative correction at the left-hand 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. ■

The `<strength>` 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, `\itcorr{<strength>}` calculates

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

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

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

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

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 `\,^`.

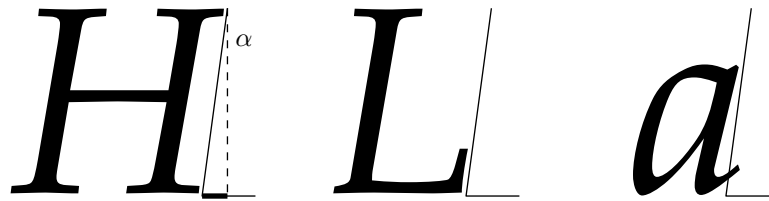


FIGURE 1: Some letters of an italics font. We use the capital `\H` to measure the angle α between the plumb-line (drawn dashed) and a tangent to the rightmost parts of the glyph. The length of the plumb-line is proportional to `\strength` and the short, thick part of the baseline symbolizes the resulting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

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

3.6 Apply Extra Kerning

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

3.6.1 Slash

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

`\kernedslash*`

```
\kernedslash
\kernedslash*
```

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

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: `\year_1/\year_2`. ¶ 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`
`\kernedhyphen*` Macros `\kernedhyphen*` and `\kernedhyphen` expand to a hyphen (·) with given kerning to its left and to its right.

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

Typeset an unbreakable hyphen with `\kernedhyphen*` or a breakable hyphen (like `\hyp` of package `hyphenat` [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 `raisecapitalhyphen`. The default for `⟨raise⟩` is zero.

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

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

Use Cases

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

3.7 Raise Selected Characters

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

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

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

Caution

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

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

3.7.1 Capital Hyphen

`\capitalhyphen`
`\capitalhyphen*`

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

```
\capitalhyphen
\capitalhyphen*
```

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

Use Cases

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

3.7.2 Capital Dash

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

The situation of the en-dash $\text{̣}^{\text{̣}}$ is almost identical to the one of the hyphen character $\text{̣}^{\text{̣}}$ 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 $\text{̣}^{\text{̣}}$. 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*
```

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

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

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
fbf*, Source Serif Pro	.05
Libre Baskerville, Crimson Pro, Erewhon, Droid Serif	.0667
GFS Artemisia, Libre Caslon, Coelacanth, Crimson Pro, Crimson Text, T _E X Gyre Pagella, Quattrocento, TX Fonts, ADF Venturis, and many more	.1

* Free Bembo.

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

Use Case

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

3.7.4 Multiplication Sign – Times \times

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

```
\capitaltimes
```

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

Use Case

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

3.7.5 Guillemets

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.

Lowercase Versions.

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

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

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

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


`\Singleguillemetleft`
`\Singleguillemetright`
`\Doubleguillemetleft`
`\Doubleguillemetright`

Uppercase Versions.

```

\Singleguillemetleft   \Singleguillemetright
\Doubleguillemetleft  \Doubleguillemetright

```

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

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

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

Tip

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

```

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

```

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

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

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

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

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

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

Use Case

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

Anticipated Changes & Possible Extensions

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

3.8 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

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

ALL THREE SINCE V0.4

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

Apply uppercase adjustment, lowercase adjustment or no adjustment to the labels in `itemize` environments at the `<levels-to-adjust>`. The adjustment values themselves, this is the vertical shifts are configured with options `uppercase-labelitemadjustments` and `lowercaselabelitemadjustments`. They

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

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

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

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

The macros assume a $\langle correct \rangle$ 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

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

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

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

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

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

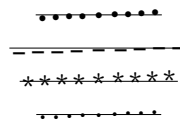
and repeat up

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

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

Example

Play ball!



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

`\typoguppercaseadjustcheck`
`\typoglowercaseadjustcheck`
BOTH SINCE V0.4

For a quick and easy check how the four label items vertically align *as configured* use `\typoguppercaseadjustcheck` and `\typoglowercaseadjustcheck`. Experienced users with a keen eye for type can apply these

macros even in the initial setup. An accurate determination of `uppercase-labelitemadjustments` and `lowercaselabelitemadjustments` is preferably done at a high magnification (400% to 600% on a 100 dpi screen) with a representative sample of initial letters.

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

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

Example

Uppercase check with `\typoguppercaseadjustcheck{ABCXYZ}`:

```
ABGH•••QWYZ, 0123•••4567
```

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

```
ace•••mno, bdf•••gjh, 0123•••4567,
```

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

In Table 4 on p. 26 we collected some suggestions for adjustment values in the *default* case when the label items are not redefined by the user and expand like

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

They display as \bullet , --- , \ast , and --- , respectively.

3.9 Align Last Line of a Paragraph

The usual algorithms of L^AT_EX typeset the last line of a paragraph flush with the left margin unless `center`, `raggedleft` or `Centering`, `FlushRight` (package `ragged2e` [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 `lastlinerraggedleftpar` adjusts the various skips such that the last lines of the paragraphs gets typeset flush with the right margin.

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

`lastlinerraggedleftpar`
(env.)

`lastlineflushrightpar`
(env.)

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

Font Name	Uppercase Adjustments				Lowercase Adjustments			
	1	2	3	4	1	2	3	4
ADF Accanthis	.5	1.5	-1.5	1.125	-.75	.25	-2.75	-.125
ADF Venturis	.0	1.0	.75	1.0	-.75	.0	-.25	.0
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.

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

`lastlinecenteredpar`
(*env.*)

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

```
\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]

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



The same problem arises with displayed math in a flush-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 `\parindent`; a typical suggestion being twice the `\parindent`.



¹⁸ Also compare the approach taken in Ref. 30.

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 `\mathindent`.

2. `\parindent = 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‹ `\parindent`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.

\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. ■

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

²² For example, \LaTeX 's class `article` uses a `\parindent` of 25 pt.

3.10.2 Manual Changes

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

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

2. Uniform paragraph change.
 - (a) Vary spacing.
Modify the inter-word spacing, for example, with the macros introduced in Sec. [3.11.1](#).
Enclose the paragraph in either `loosespacing` or `tightspacing`. Increase the spacing *<level>* until the last line gets the desired length.
 - (b) Vary font tracking.
Enclose the paragraph in a `setfonttracking` group. See Sec. [3.12.1](#). Increase or decrease the tracking in steps of $\frac{1}{1000}$ em until the last line looks good.
 - (c) Vary font expansion.
Enclose the paragraph in a `setfontexpand` group. See Sec. [3.12.2](#).
3. A combination of any of the above items.
4. Some curveballs.
 - (a) If the paragraph already suffers from one of the problems that $\text{T}_{\text{E}}\text{X}$ addresses with `\doublehyphendemerits`, `\finalhyphendemerits`, or `\adjdemerits`, crank up one or all of these values to 10000 and observe whether the length of last line changes in the desired direction.
 - (b) If any influential microtype features have been enabled try with one more more of them *disabled*. See, e. g., environment `nofontexpansion` in Sec. [3.12.2](#).

3.10.3 Multi-Purpose Environments

`shortenpar` (*env.*) The two environments `shortenpar` and `prolongpar` can be employed in quite general situations when a paragraph should be typeset one line longer or shorter, e. g., to avoid a widow 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.

3.10.4 Specialized Environments

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

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

```
\begin{covernextindentpar} [<dim>]
...
\end{covernextindentpar}
```

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

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

```
\begin{openlastlinepar} [<dim>]
...
\end{openlastlinepar}
```

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.

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 L^AT_EX. No extra packages are required. Compare to the microtype-based functionality of [Sec. 3.12](#).

3.11.1 Looser or Tighter Spacing

Never try to adjust lines by squeezing or stretching the tracking.
Go for the subtle solution: adjust word spacing instead.
— JAN MIDDENDORP [[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 $\langle looseness \rangle$ or $\langle tightness \rangle$, 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](#).

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](#).

The default level of `tightspacing` is 1.

$\langle level \rangle$	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 $1 \mapsto 1.05_{-.1}^{+.5}$, $2 \mapsto 1.1_{-.1}^{+.5}$, $3 \mapsto 1.2_{-.2}^{+.6}$, and $\geq 4 \mapsto 1.3_{-.3}^{+.8}$, where all factors scale with `\dimen2`, the current font's space-width.

$\langle level \rangle$	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 $1 \mapsto .9875_{-.5}^{+.0125}$, $2 \mapsto .975_{-.5}^{+.025}$, $3 \mapsto .95_{-.5}^{+.05}$, and $\geq 4 \mapsto .9_{-.5}^{+.1}$, where all factors scale with `\dimen2`, the current font's space-width.

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. T_EX addresses the latter by integers, which is totally non-memnonic. Therefore, we play softball by first presenting Tab. 7 that associates the `\fontdimen<number>`s with their meanings and also reports on their current values (for this document).²⁶

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

`\widespace`
`\widespace*`

STARRED FORM SINCE V0.2

```
\widespace
\widespace*
```

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

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

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

#	Description	Value
1	Slant per 1 pt height	0
2	Interword space width	23.3
3	Interword stretch	11.6
4	Interword shrink	7.8
5	x^7 height	47.5
6	$\backslash\text{quad}$ height	100
7	Extra space width	3.9

TABLE 7: All $\text{T}_\text{E}_\text{X}$ font parameters normalized to the font's quad-size. The first column $\#^7$ states the index of the $\backslash\text{fontdimen}$ parameter: $\langle number \rangle$. Column 2 presents short descriptions of $\backslash\text{fontdimen}\langle number \rangle$. As examples, the values for the current font are shown in column 3.

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

If $\backslash\text{fontdimen}7$ happens to be zero $\backslash\text{widestospace}$ uses

$$\backslash\text{widestpacescale} \times \backslash\text{fontdimen}2$$

as width instead, where $\backslash\text{widestpacescale}$ defaults to 1.125. The stretchability and shrinkability of $\backslash\text{widestospace}$ always are scaled with $\backslash\text{widestpacescale}$. The $\backslash\text{widestpacescale}$ too can be redefined by the user to achieve different effects.

The starred form, $\backslash\text{widestospace}^*$, unconditionally uses the $\backslash\text{fontdimen}7 = 0$ 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 $\backslash\text{widestpaces}$.²⁷ ■

3.11.3 Narrow Space

$\backslash\text{narrowospace}$
 $\backslash\text{narrowospace}^*$
 SINCE v0.2

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

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

The unstarred macro $\backslash\text{narrowospace}$ inserts a narrow space with the width

$$\backslash\text{fontdimen}2 - \backslash\text{narrowospacestrength} \times \backslash\text{fontdimen}7$$

if $\backslash\text{fontdimen}7$ is different from zero or otherwise

$$\backslash\text{narrowospace} \times \backslash\text{fontdimen}2.$$

The starred version, $\backslash\text{narrowospace}^*$, unconditionally uses the $\backslash\text{fontdimen}7 =$

²⁷ See also »Investigating the badness of a paragraph« on Page 9.

The sentence that ends with ›1.« uses $\backslash\text{widestospace}$ after the period.

0 code-path. Refer to Table 7 for the meanings of the various `\fontdimen` parameters.

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

Use Case

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

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

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

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. ■

28 Footnote 27 again applies.

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}{\langle delta \rangle}
...
\end{setfonttracking}
```

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

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`.

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

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

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

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

Use Cases

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

`setfontexpand` (*env.*)

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

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

Table 11 gives an overview of the values associated with $\langle level \rangle$.

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

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

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

Notes

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

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

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

```
\slightlyloppy[<loppiness>]
\begin{slightlyloppypar}[<loppiness>]
...
\end{slightlyloppypar}
```

Table 12 summarizes the adjustments that `\slightlyloppy` makes depending on the `<loppiness>` level.

Environment `slightlyloppypar[<loppiness>]` mimics L^AT_EX's `sloppypar`, while offering the flexibility of `\slightlyloppy`.

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

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

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

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

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

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

to prevent excessive stretchability in narrow lines.

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}
```

Alternatively with package `enumitem` [4]:

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.

```

\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.*)

```

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

Zero or one for $\langle before\text{-}disp\text{-}number\text{-}of\text{-}lines \rangle$ as well as $\langle after\text{-}disp\text{-}number\text{-}of\text{-}lines \rangle$ 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 $\langle number\text{-}of\text{-}lines \rangle$ from the beginning of every partial paragraph. Each displayed math in the paragraph resets the count. The bottom-paragraph ties, `\vtiebot`, `vtiebotpar`, `\vtiebotdisp`, and `vtiebotdisp-par` count backward from the end of each partial paragraph. Again, each displayed math in the paragraph resets the count. According to TeX's rules, a displayed math formula always is counted as *three* lines no matter its contents. Table 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 concremented 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 - $\text{T}_\text{E}_\text{X}$'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 - $\text{T}_\text{E}_\text{X}$'s line counts of $\backslash\text{widowpenalties}$ and $\backslash\text{displaywidowpenalties}$ on which the functions of this package are based.

3.15 Breakable Displayed Equations

`breakabledisplay`
(*env.*)

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

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

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

Tips

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

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

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

[†] 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. 33.

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

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

An alternative to relying on the design size is using the actual size of an uppercase letter:

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

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

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

or

```
\setleadingpercentage{0}
```

Tip

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

Here are some forms that do work:

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

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 \TeX 's `\parshape` primitive and leave alone the stretchability or shrinkability of the glue.

Caution

None of the following environments work inside of lists. ■

<code>smoothraggedrightshapetriplet</code> (env.)	We introduce three environments that set three, five, or seven different line-lengths (which \TeX of course will repeat for paragraphs longer than three, five, or seven lines): <code>smoothraggedrightshapetriplet</code> , <code>smoothraggedrightshapequintuplet</code> , and <code>smoothraggedrightshapeseptuplet</code> ; they work for paragraph lengths up to 99, 95, and 98 lines, respectively.
<code>smoothraggedrightshapequintuplet</code> (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 $\langle widthI \rangle$, $I = 1, \dots, N$ is a skip, i. e., a dimen that can include some glue.

Options

leftskip= $\langle dim \rangle$

Set the left margin for the smooth ragged paragraph to $\langle dim \rangle$. Similar to the \TeX parameter `\leftskip`.

parindent= $\langle dim \rangle$

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

<code>smoothraggedrightpar</code> (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.
---------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

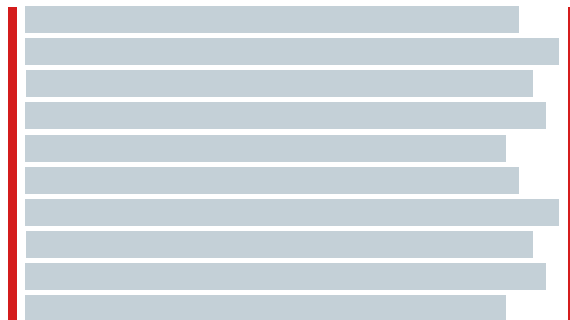
```
\begin{smoothraggedrightpar}[\langle option... \rangle]
...
\end{smoothraggedrightpar}
```

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

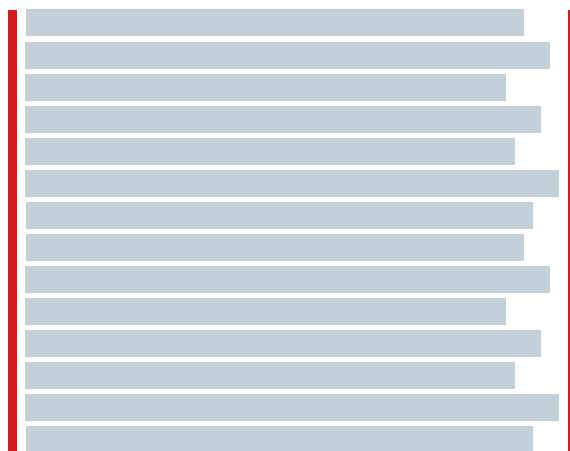
- The triplet generator repeats a *short line – long line – middle-length line* sequence. Shown below are two complete cycles.



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



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



smoothraggedright
(env.)

Environment smoothraggedright is the multi-paragraph version of smooth-

`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 marginal 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^AT_EX-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 KHIREVICH's discussion [13].
- `ragged2e` Improved versions of environments `raggedleft`, `raggedright`, and `center` [22].
- `setspace` Consistently set the line-spacing of a document, i. e., control `\baselineskip` [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`.

`\ftypog@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}}
41 \fi
42

```

`\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-
56   pog}}}%
57   \fi
58

```

Our own state...

config@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

```

`\typoggetnth` Access the n^{th} element of a list-like configuration item's value. The indexing either starts at 1 and goes up to n , or at -1 and then goes down to $-n$. We catch all out-of-bounds errors.

```

110 \ExplSyntaxOn
111 \NewDocumentCommand{\typoggetnth}{m m}
112 {
113   \clist_set:Nx \l_tmpa_clist
114     {\csname typog@config@#1\endcsname}
115   \typog@typeout {typoggetnth...}
116   \clist_log:N \l_tmpa_clist
117
118   \int_set:Nn \l_tmpa_int {#2}
119   \int_set:Nn \l_tmpb_int {\clist_count:N \l_tmpa_clist}
120   \int_compare:nT {\l_tmpa_int = 0}
121     {\PackageError {typog}
122       {Index\space equal\space to\space zero}
123       {Use\space a\space nonzero\space index.}}
124   \int_compare:nT {\l_tmpa_int > \l_tmpb_int}
125     {\PackageError {typog}
126       {Index\space out\space of\space range}
127       {Range\space is\space 1..\l_tmpb_int.\space
128         Index\space is\space \l_tmpa_int.}}
129   \int_compare:nT {\l_tmpa_int < -\l_tmpb_int}
130     {\PackageError {typog}
131       {Index\space out\space of\space range}
132       {Range\space is\space -\l_tmpb_int..-1.\space
133         Index\space is\space \l_tmpa_int.}}
134
135   \clist_item:Nn {\l_tmpa_clist} {\l_tmpa_int}
136 }
137 \ExplSyntaxOff
138

```

A.2 Information

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

```

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

```

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

```

144 \newcommand*{\typog@formatsizeinfo}[3]
145   {#1#3\kerned\slash #2#3}
146

```

`\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.

```

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

```

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

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

`\typog@inspect@count` Counter to supply unique number and in turn $\langle id \rangle$ for those `typoinspect` environments that were not identified by the user.

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

```

`typoginspect (env.)`

```

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

```

173   \edef\typog@@arg{#2}%
174   \ifx\typog@@arg\empty
175     \stepcounter{typog@inspect@count}%
176     \edef\typog@@id{\typog@default@inspect@id@prefix
177                   \arabic{typog@inspect@count}}%
178   \else

```

```

179     \edef\typog@@id{\typog@trim@spaces{\typog@@arg}}%
180     \fi
181     \typeout{<typog-inspect\space
182             id="\typog@@id"\space
183             job="\jobname"\space
184             line="\the\inputlineno"\space
185             page="\the\value{page}">}%

```

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

```

186     \hbadness=\m@ne
187     \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.

```

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

```

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

```

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

```

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`.

```

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

```

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

```

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

```

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

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

PDF-substitute definition

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

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

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

A.4 Disable/Break Ligatures

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

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

`\nolig`

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

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

```
236 \typog@register@pdfsubstitute{
237   \RenewExpandableDocumentCommand{\nolig}{s o m}{%
238     \ifx\typog@TYPOG#3\typog@TYPOG
239       \relax
240     \else
241       \ifx\relax#3\relax
242         \relax
243       \else
244         \PackageError{typog}
245           {Missing third argument of \nolig}
```



```

246             {Append empty group or \relax after macro in-
      vocation}
247         \fi
248     \fi}
249 }
250

```

A.5 Manual Italic Correction

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

```

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

```

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

```

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

```

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

```

261 \newcommand*{\typog@itcorr@math}[1]
262   {\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.

```

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

```

PDF-substitute definition

```

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

```

A.6 Apply Extra Kerning

Slash

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

```
275 \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’`.

```
276 \NewDocumentCommand{\kernedslash}{s}
277   {\hspace*{\typog@config@slashkern}%
278    \typog@forwardslash
279    \IfBooleanTF{#1}%
280     {\hspace*{\typog@config@slashkern}\ignorespaces}%
281     {\typog@breakpoint\typog@allowhyphenation\hspace*{\typog@config@slashkern}}}
```

PDF-substitute definition

```
282 \typog@register@pdfsubstitute{
283   \def\kernedslash#1{\if* \detokenize{#1}/\ignorespaces\else/#1\fi}%
284 }
285
```

Hyphen

`\kernedhyphen`

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

PDF-substitute definition

```
303 \typog@register@pdfsubstitute{
304   \RenewExpandableDocumentCommand{\kernedhyphen}{s o m m}{-}
305 }
```

One-argument shorthands.

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

```
306 \NewDocumentCommand{\leftkernedhyphen}{s O{0} m}
307   {\IfBooleanTF{#1}%
308     {\kernedhyphen*[#2]{#3}{0}\ignorespaces}%
309     {\kernedhyphen[#2]{#3}{0}}}
```

PDF-substitute definition

```
310 \typog@register@pdfsubstitute{
311   \RenewExpandableDocumentCommand{\leftkernedhyphen}{s o m}{-}
312 }
313
```

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

```
314 \NewDocumentCommand{\rightkernedhyphen}{s O{0} m}
315   {\IfBooleanTF{#1}%
316     {\kernedhyphen*[#2]{0}{#3}\ignorespaces}%
317     {\kernedhyphen[#2]{0}{#3}}}
```

PDF-substitute definition

```
318 \typog@register@pdfsubstitute{
319   \RenewExpandableDocumentCommand{\rightkernedhyphen}{s o m}{-}
320 }
321
```

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.

```
322 \newcommand*{\typog@breakpoint}
323   {\penalty\typog@config@breakpenalty}
```

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

```
324 \NewDocumentCommand{\capitalhyphen}{s}
325   {\raisebox{\typog@config@raisecapitalhyphen}{\typog@hyphen}%
326     \IfBooleanTF{#1}%
327     {\ignorespaces}%
328     {\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.

```
329 \typog@register@pdfsubstitute{
330   \def\capitalhyphen#1{%
331     \if*\detokenize{#1}%
332     -\ignorespaces
333     \else
```

```

334     -#1%
335     \fi}
336 }
337

```

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

```

338 \NewDocumentCommand{\capitalendash}{s}
339   {\raisebox{\typog@config@raisecapitaldash}{\textendash}%
340    \IfBooleanTF{#1}%
341     {\ignorespaces}%
342     {\typog@breakpoint\typog@allowhyphenation}}}
343 \let\capitaldash=\capitalendash

```

PDF-substitute definition

```

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

```

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

```

354 \NewDocumentCommand{\capitalemdash}{s}
355   {\raisebox{\typog@config@raisecapitaldash}{\textemdash}%
356    \IfBooleanTF{#1}%
357     {\ignorespaces}%
358     {\typog@breakpoint\typog@allowhyphenation}}}

```

PDF-substitute definition

```

359 \typog@register@pdfsubstitute{
360   \def\capitalemdash#1{%
361     \if*\detokenize{#1}%
362       \textemdash\ignorespaces
363     \else
364       \textemdash#1%
365     \fi}
366 }
367

```

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

```

368 \NewDocumentCommand{\figuredash}{s}
369   {\raisebox{\typog@config@raisefiguredash}{\textendash}%
370    \IfBooleanTF{#1}%
371     {\ignorespaces}%
372     {\typog@breakpoint\typog@allowhyphenation}}}

```

PDF-substitute definition

```
373 \typog@register@pdfsubstitute{\let\figuredash=\capitaldash}
374
```

\capitaltimes

```
375 \NewDocumentCommand{\capitaltimes}{}
376   {\ifmmode
377     \mathbin{\raisebox{\typog@config@raisecapitaltimes}{\m@th\times}}%
378     \else
379     \raisebox{\typog@config@raisecapitaltimes}{\texttimes}%
380     \fi}
```

PDF-substitute definition

```
381 \typog@register@pdfsubstitute{
382   \RenewExpandableDocumentCommand{\capitaltimes}{}{\texttimes}
383 }
384
```

\singleguillemetleft

```
385 \NewDocumentCommand{\singleguillemetleft}{}
386   {\typog@allowhyphenation
387     \raisebox{\typog@config@raiseguillemets}{\guilsinglleft}}
```

PDF-substitute definition

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

\singleguillemetright

```
389 \NewDocumentCommand{\singleguillemetright}{}
390   {\raisebox{\typog@config@raiseguillemets}{\guilsinglright}%
391     \typog@allowhyphenation}
```

PDF-substitute definition

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

\doubleguillemetleft

```
393 \NewDocumentCommand{\doubleguillemetleft}{}
394   {\typog@allowhyphenation
395     \raisebox{\typog@config@raiseguillemets}{\guillemotleft}}
```

PDF-substitute definition

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

\doubleguillemetright

```
397 \NewDocumentCommand{\doubleguillemetright}{}
398   {\raisebox{\typog@config@raiseguillemets}{\guillemotright}%
399     \typog@allowhyphenation}
```

PDF-substitute definition

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

\Singleguillemetleft

```
401 \NewDocumentCommand{\Singleguillemetleft}{}
402   {\typog@allowhyphenation
403     \raisebox{\typog@config@raisecapitalguillemets}{\guilsinglleft}}
```

PDF-substitute definition

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

\Singleguillemetright

```
405 \NewDocumentCommand{\Singleguillemetright}{}
406 {\raisebox{\typog@config@raisecapitalguillemets}{\guilsinglright}%
407 \typog@allowhyphenation}
```

PDF-substitute definition

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

\Doubleguillemetleft

```
409 \NewDocumentCommand{\Doubleguillemetleft}{}
410 {\typog@allowhyphenation
411 \raisebox{\typog@config@raisecapitalguillemets}{\guillemotleft}}
```

PDF-substitute definition

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

\Doubleguillemetright

```
413 \NewDocumentCommand{\Doubleguillemetright}{}
414 {\raisebox{\typog@config@raisecapitalguillemets}{\guillemotright}%
415 \typog@allowhyphenation}
```

PDF-substitute definition

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

```
417
```

A.8 Vert. Adjust Label Items

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

```
418 \newcommand*{@typog@uppercase@adjust@labelitem}[1]
419 {\@typog@maybe@patch@itemize
420 \ifstrequal{#1}{*}
421 {\setlength{\typog@adjust@labelitemi}
422 {\typog@adjust@uppercase@labelitemi}
423 \setlength{\typog@adjust@labelitemii}
424 {\typog@adjust@uppercase@labelitemiii}
425 \setlength{\typog@adjust@labelitemiii}
426 {\typog@adjust@uppercase@labelitemiii}
427 \setlength{\typog@adjust@labelitemiv}
428 {\typog@adjust@uppercase@labelitemiv}}
429 {\ifcase #1% 0
430 \relax % outside of any itemize environment
431 \or % 1
432 \setlength{\typog@adjust@labelitemi}
433 {\typog@adjust@uppercase@labelitemi}
434 \or % 2
435 \setlength{\typog@adjust@labelitemii}
436 {\typog@adjust@uppercase@labelitemii}}
```

```

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

```

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

```

449 \newcommand*{\@typog@lowercase@adjust@labelitem}[1]
450   {\@typog@maybe@patch@itemize
451     \ifstrequal{#1}{*}
452       {\setlength{\typog@adjust@labelitemi}
453         {\typog@adjust@lowercase@labelitemi}
454         \setlength{\typog@adjust@labelitemii}
455           {\typog@adjust@lowercase@labelitemii}
456         \setlength{\typog@adjust@labelitemiii}
457           {\typog@adjust@lowercase@labelitemiii}
458         \setlength{\typog@adjust@labelitemiv}
459           {\typog@adjust@lowercase@labelitemiv}}
460       {\ifcase #1% 0
461         \relax % outside of any itemize environment
462         \or % 1
463             \setlength{\typog@adjust@labelitemi}
464                 {\typog@adjust@lowercase@labelitemi}
465         \or % 2
466             \setlength{\typog@adjust@labelitemii}
467                 {\typog@adjust@lowercase@labelitemii}
468         \or % 3
469             \setlength{\typog@adjust@labelitemiii}
470                 {\typog@adjust@lowercase@labelitemiii}
471         \or % 4
472             \setlength{\typog@adjust@labelitemiv}
473                 {\typog@adjust@lowercase@labelitemiv}
474         \else
475             \PackageError{typog}
476                 {Itemize level out of range}
477                 {Valid levels are 1, 2, 3, 4, and *}
478         \fi}}
479

```

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

```

480 \newcommand*{\@typog@noadjust@labelitem}[1]
481   {\ifstrequal{#1}{*}

```

```

482         {\setlength{\typog@adjust@labelitemi}{\z@}
483          \setlength{\typog@adjust@labelitemii}{\z@}
484          \setlength{\typog@adjust@labelitemiii}{\z@}
485          \setlength{\typog@adjust@labelitemiv}{\z@}}
486     {\ifcase #1% 0
487      \relax % outside of any itemize environment
488      \or % 1
489       \setlength{\typog@adjust@labelitemi}{\z@}
490      \or % 2
491       \setlength{\typog@adjust@labelitemii}{\z@}
492      \or % 3
493       \setlength{\typog@adjust@labelitemiii}{\z@}
494      \or % 4
495       \setlength{\typog@adjust@labelitemiv}{\z@}
496      \else
497       \PackageError{typog}
498         {Itemize level out of range}
499         {Valid levels are 1, 2, 3, 4, and *}
500     \fi}}
501

```

`\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`.

```

502 \NewDocumentCommand{\uppercaseadjustlabelitems}{m}
503   {\AfterPreamble{%
504     \ifblank{#1}
505       {\@typog@uppercase@adjust@labelitem{\@itemdepth}}
506       {\forcsvlist{\@typog@uppercase@adjust@labelitem}{#1}}%
507     \ignorespaces}}
508

```

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

```

509 \NewDocumentCommand{\lowercaseadjustlabelitems}{m}
510   {\AfterPreamble{%
511     \ifblank{#1}
512       {\@typog@lowercase@adjust@labelitem{\@itemdepth}}
513       {\forcsvlist{\@typog@lowercase@adjust@labelitem}{#1}}%
514     \ignorespaces}}
515

```

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

```

516 \NewDocumentCommand{\noadjustlabelitems}{m}
517   {\ifblank{#1}
518     {\@typog@noadjust@labelitem{\@itemdepth}}
519     {\forcsvlist{\@typog@noadjust@labelitem}{#1}}%
520   \ignorespaces}
521

```


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

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

```
% \def\itemize{%
%   \ifnum \@itemdepth > \thr@@
%     \@toodeep
%   \else
%     \advance\@itemdepth\@ne
%     \edef\@itemitem{label\item\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 \LaTeX 's or package `enumitem`'s `\itemize`.

```
522 \newcommand*\@typog@itemize@patch
```

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

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

Sneak in our own macro's name.

```
524 \edef\@itemitem{@typog@label\item\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.

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

```
526   {\raisebox{\csname typog@adjust@label\item\romannumeral\the\@itemdepth\endcsname
```

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

```
528
```

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.

```

529 \newcommand*{\@typog@patch@itemize}
530   {\ifdefined\enit@itemize@i
531     \patchcmd{\enit@itemize@i
532               {\expandafter}
533               {\@typog@itemize@patch\expandafter}
534               {\typog@debug@typeout{patching enumitem \string\enit@itemize@i\space
535                                     ceeded}}}
536               {\PackageError{typog}
537                       {Patching enumitem macro \string\enit@itemize@i\space
538                       {}}
539     \else
540     \patchcmd{\itemize}
541               {\expandafter}
542               {\@typog@itemize@patch\expandafter}
543               {\typog@debug@typeout{patching \string\itemize\space suc-
544                                     ceeded}}}
545               {\PackageError{typog}
546                       {Patching plain LaTeX macro \string\itemize\space fai
547                       {}}
548     \fi}
549

```

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

```

548 \newbool{\@typog@itemize@has@been@patched}
549 \newcommand*{\@typog@maybe@patch@itemize}
550   {\ifbool{\@typog@itemize@has@been@patched}
551     {\relax}
552     {\@typog@patch@itemize
553     \booltrue{\@typog@itemize@has@been@patched}}}
554

```

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.

```
555 \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.

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

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

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

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

```
558 \count0=#3\relax
559 \unless\ifodd\count0
560 \advance\count0 by 1%
561 \fi
```

Set the iteration counter.

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

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

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

Box 1 is the accumulator for the raised samples.

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

Build the stairs.

```
565 \loop
566 \ifnum\thetypog@@iteration=\numexpr\count0 / 2\relax
567 \dimen1=2\dimen0
568 \else
569 \dimen1=\dimen0
570 \fi
571 \dimen2=\dimexpr#2 * (\thetypog@@iteration - \count0 / 2)\re-
\ax
572 \setbox1=\hbox{\unhbox1\raisebox{\dimen2}{\kern\dimen1 #5\kern\dimen1}}%
573 \addtocounter{typog@@iteration}{1}%
574 \unless\ifnum\thetypog@@iteration>\count0
575 \repeat
```

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

```
576 \mbox{\rlap{\raisebox{\fpeval{#1}\ht0}{\rule{\wd1}{\typog@hairline@width}}}\bo
577
```

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

```
578 \NewDocumentCommand{\typogadjuststairs}{0{.5} m m m}
579 {\begingroup
580 \unless\ifdim #2>\z@
581 \PackageError{typog}
582 {\string\typogadjuststairs\space non-positive step-
size}
583 {\step-size must be a positive dimension}%
584 \fi
585 \ifnum #3<1
586 \PackageError{typog}
587 {\string\typogadjuststairs\space too few number-
of-steps}
588 {\number-of-steps must at least be 1}%
```

```

589 \fi
590 \ifblank{#4}
591     {\PackageError{typog}
592         {sample must not be empty}
593         {supply either some uppercase or some low-
ercase letters}}
594     }%
595 \def\arraystretch{1}%
596 \begin{tabular}{@{}c@{}}
597     \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemi} \\
598     \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemii} \\
599     \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemiii} \\
600     \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemiv}
601 \end{tabular}
602 \endgroup}
603

```

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

```

604 \newcommand*{\typog@uppercase@adjusted@labelitems}
605 {\hbox{\raisebox{\typog@adjust@uppercase@labelitemi}{\labelitemi}%
606         \raisebox{\typog@adjust@uppercase@labelitemii}{\labelitemii}%
607         \raisebox{\typog@adjust@uppercase@labelitemiii}{\labelitemiii}%
608         \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.

```

609 \NewDocumentCommand{\typoguppercaseadjustcheck}{0{.5} m}
610 {\setbox0=\hbox{#2}%
611  \setbox1=\typog@uppercase@adjusted@labelitems
612  \mbox{\rlap{\raisebox{\fpeval{#1}\ht0}
613           {\rule{\wd1}{\typog@hairline@width}}}%
614         \box1}}
615

```

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

```

616 \newcommand*{\typog@lowercase@adjusted@labelitems}
617 {\hbox{\raisebox{\typog@adjust@lowercase@labelitemi}{\labelitemi}%
618         \raisebox{\typog@adjust@lowercase@labelitemii}{\labelitemii}%
619         \raisebox{\typog@adjust@lowercase@labelitemiii}{\labelitemiii}%
620         \raisebox{\typog@adjust@lowercase@labelitemiv}{\labelitemiv}}}

```

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

```

621 \NewDocumentCommand{\typoglowercaseadjustcheck}{0{.5} m}
622 {\setbox0=\hbox{#2}%
623  \setbox1=\typog@lowercase@adjusted@labelitems
624  \mbox{\rlap{\raisebox{\fpeval{#1}\ht0}
625           {\rule{\wd1}{\typog@hairline@width}}}%
626         \box1}}
627

```

A.9 Align Last Line of a Paragraph

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

`lastlineraggedleftpar` (*env.*)

```
628 \NewDocumentEnvironment{lastlineraggedleftpar}{}
629   {\lastlinefit=0%
630    \setlength{\leftskip}{\z@ \@plus 1fil}%
631    \setlength{\rightskip}{-\leftskip}%
632    \setlength{\parfillskip}{\leftskip}}
633   {\par}
```

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

```
634 \let\lastlineflushrightpar=\lastlineraggedleftpar
635 \let\endlastlineflushrightpar=\endlastlineraggedleftpar
636
```

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

```
637 \NewDocumentEnvironment{lastlinecenteredpar}{}
638   {\lastlinefit=0%
639    \setlength{\leftskip}{\z@ \@plus .5fil}%
640    \setlength{\rightskip}{-\leftskip}%
641    \setlength{\parfillskip}{\z@ \@plus 1fil}}
642   {\par}
643
```

A.10 Fill Last Line of a Paragraph

`shortenpar` (*env.*)

```
644 \NewDocumentEnvironment{shortenpar}{}
645   {\advance\looseness by -1
646    \ifnum\tracingparagraphs>0
647     \typeout{@ looseness \the\looseness}%
648    \fi}
649   {\par}
650
```

`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.

```
651 \NewDocumentEnvironment{prolongpar}{}
652   {\finalhyphendemerits=100000001
653    \advance\looseness by 1
654    \ifnum\tracingparagraphs>0
655     \typeout{@ looseness \the\looseness}%
656    \fi}
657   {\par}
658
```

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

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

`ndentpar@nonzero@parindent`

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

`covernextindentpar (env.)`

```
661 \NewDocumentEnvironment{covernextindentpar}{o}
662   {\IfNoValueTF{#1}
663     {\ifdim\parindent=\z@
664       \dimen0=\dimexpr\linewidth - \typog@covernextindentpar@zero@parindent
665       \else
666         \dimen0=\dimexpr\linewidth - \typog@covernextindentpar@nonzero@parindent
667       \fi}
668     {\dimen0=\dimexpr\linewidth - (#1)}%
669     \parfillskip=\dimen0 \@minus \dimen0
670     \relax}
671   {\par}
672
```

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

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

`tlinepar@nonzero@parindent`

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

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

```
675 \NewDocumentEnvironment{openlastlinepar}{o}
676   {\IfNoValueTF{#1}
677     {\ifdim\parindent=\z@
678       \skip0=\typog@openlastlinepar@zero@parindent
679       \@plus 1fil
680       \@minus \typog@openlastlinepar@zero@parindent
681     \else
682       \skip0=\typog@openlastlinepar@nonzero@parindent
683       \@plus 1fil
684       \@minus \typog@openlastlinepar@nonzero@parindent
685     \fi}
686     {\dimen0=\dimexpr#1\relax
687     \skip0=\dimen0 \@plus 1fil \@minus \dimen0}
688     \parfillskip=\skip0}
689   {\par}
690
```

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.

```
691 \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.

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

`\widespace`

```
693 \NewDocumentCommand{\widespace}{s}
694   {\IfBooleanTF{#1}%
695     {\dimen0=\widespacescale\fontdimen2\font}%
696     {\ifdim\fontdimen7\font=\z@
697       \dimen0=\widespacescale\fontdimen2\font
698       \else
699         \dimen0=\dimexpr\fontdimen2\font +
700           \widespacestrength\fontdimen7\font
701       \fi}%
702   \hskip \glueexpr\dimen0
703         \@plus \widespacescale\fontdimen3\font
704         \@minus \widespacescale\fontdimen4\font
705   \ignorespaces}
706
```

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

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

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

```
708 \newcommand*{\narrowpacescale}{.9375}
```

`\narrowpace`

```
709 \NewDocumentCommand{\narrowpace}{s}
710   {\IfBooleanTF{#1}%
711     {\dimen0=\narrowpacescale\fontdimen2\font}%
712     {\ifdim\fontdimen7\font=\z@
713       \dimen0=\narrowpacescale\fontdimen2\font
714       \else
715         \dimen0=\dimexpr\fontdimen2\font -
716           \narrowpacestrength\fontdimen7\font
717       \fi}%
718   \hskip \glueexpr\dimen0
719         \@plus \narrowpacescale\fontdimen3\font
```

```

720         \@minus \narrowspacescale\fontdimen4\font
721     \ignorespaces}
722

```

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

looespacing (*env.*)

```

723 \NewDocumentEnvironment{looespacing}{0{1}}
724   {\dimen2=\fontdimen2\font
725   \ifcase #1
726     \spaceskip=\z@
727   \or % 1      +5%
728     \spaceskip=1.05\dimen2 \@plus .5\dimen2 \@minus .1\dimen2
729   \or % 2      +10%
730     \spaceskip=1.1\dimen2 \@plus .5\dimen2 \@minus .1\dimen2
731   \or % 3      +20%
732     \spaceskip=1.2\dimen2 \@plus .6\dimen2 \@minus .2\dimen2
733   \else % >= 4 +30%
734     \spaceskip=1.3\dimen2 \@plus .8\dimen2 \@minus .3\dimen2
735   \fi
736   \ignorespaces}
737   {\ignorespacesafterend}
738

```

tightspacing (*env.*)

```

739 \NewDocumentEnvironment{tightspacing}{0{1}}
740   {\dimen2=\fontdimen2\font
741   \ifcase #1
742     \spaceskip=\z@
743   \or % 1      -1.25%
744     \spaceskip=.9875\dimen2 \@plus .0125\dimen2 \@minus .5\dimen2
745   \or % 2      -2.5%
746     \spaceskip=.975\dimen2 \@plus .025\dimen2 \@minus .5\dimen2
747   \or % 3      -5%
748     \spaceskip=.95\dimen2 \@plus .05\dimen2 \@minus .5\dimen2
749   \else % >= 4 -10%
750     \spaceskip=.9\dimen2 \@plus .1\dimen2 \@minus .5\dimen2
751   \fi
752   \ignorespaces}
753   {\ignorespacesafterend}
754

```

A.12 Microtype Front-End

Tracking

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

```

755 \NewDocumentEnvironment{setfonttracking}{m}
756   {\edef\MT@letterspace@{#1}%
757   \lsstyle
758   \ignorespaces}
759   {\ignorespacesafterend}

```


760

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.

```

761 \newcommand*{\typog@setup@font@expansion}
762   {\SetExpansion
763     [context = typog@shrink1,
764     shrink = \typog@shrink@i,
765     stretch = 0]%
766     {encoding = {*}}%
767     {}
768   \SetExpansion
769     [context = typog@shrink2,
770     shrink = \typog@shrink@ii,
771     stretch = 0]%
772     {encoding = {*}}%
773     {}
774   \SetExpansion
775     [context = typog@shrink3,
776     shrink = \typog@shrink@iii,
777     stretch = 0]%
778     {encoding = {*}}%
779     {}
780
781   \SetExpansion
782     [context = typog@stretch1,
783     shrink = 0,
784     stretch = \typog@stretch@i]%
785     {encoding = {*}}%
786     {}
787   \SetExpansion
788     [context = typog@stretch2,
789     shrink = 0,
790     stretch = \typog@stretch@ii]%
791     {encoding = {*}}%
792     {}
793   \SetExpansion
794     [context = typog@stretch3,
795     shrink = 0,
796     stretch = \typog@stretch@iii]%
797     {encoding = {*}}%
798     {}
799
800   \SetExpansion
801     [context = typog@expand1,
802     shrink = \typog@shrink@i,
803     stretch = \typog@stretch@i]%
804     {encoding = {*}}%

```

```

805     {}
806   \SetExpansion
807     [context = typog@expand2,
808       shrink = \typog@shrink@ii,
809       stretch = \typog@stretch@ii]%
810     {encoding = {*}}%
811     {}
812   \SetExpansion
813     [context = typog@expand3,
814       shrink = \typog@shrink@iii,
815       stretch = \typog@stretch@iii]%
816     {encoding = {*}}%
817     {}

```

`\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.

```

818 \newcommand*{\typog@test@microtype@expansion@feature}
819   {\ifMT@expansion
820     \typog@typeout{microtype preloaded -- font expansion features avail-
821       able}%
822     \def\typog@require@microtype@expansion{\relax}
823     \typog@setup@font@expansion
824     \else
825       \PackageWarning{typog}{microtype preloaded,\space
826         but font expansion is disabled}%
827     \def\typog@require@microtype@expansion
828       {\PackageError{typog}
829         {microtype font expansion disabled}
830         {pass option 'expansion' to package microtype}}
831   \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.

```

831 \iftypog@microtype@preloaded
832   \typog@test@microtype@expansion@feature
833 \else
834   \def\typog@require@microtype@expansion
835     {\PackageError{typog}%
836       {package microtype not (pre-)loaded, %
837         which is required for typog's font expansion}%
838     {require package microtype before package typog}}
839 \fi
840

```

`setfontshrink` (*env.*)

```

841 \NewDocumentEnvironment{setfontshrink}{0{1}}
842   {\typog@require@microtype@expansion

```

```

843 \ifcase#1% 0
844   \relax
845 \or % 1
846   \microtypecontext{expansion=typog@shrink1}%
847 \or % 2
848   \microtypecontext{expansion=typog@shrink2}%
849 \else % >= 3
850   \microtypecontext{expansion=typog@shrink3}%
851 \fi
852 \ignorespaces}
853 {\ignorespacesafterend}
854

```

`setfontstretch` (*env.*)

```

855 \NewDocumentEnvironment{setfontstretch}{0{1}}
856 {\typog@require@microtype@expansion
857   \ifcase#1% 0
858     \relax
859   \or % 1
860     \microtypecontext{expansion=typog@stretch1}%
861   \or % 2
862     \microtypecontext{expansion=typog@stretch2}%
863   \else % >= 3
864     \microtypecontext{expansion=typog@stretch3}%
865   \fi
866   \ignorespaces}
867 {\ignorespacesafterend}
868

```

`setfontexpand` (*env.*)

```

869 \NewDocumentEnvironment{setfontexpand}{0{1}}
870 {\typog@require@microtype@expansion
871   \ifcase#1% 0
872     \relax
873   \or % 1
874     \microtypecontext{expansion=typog@expand1}%
875   \or % 2
876     \microtypecontext{expansion=typog@expand2}%
877   \else % >= 3
878     \microtypecontext{expansion=typog@expand3}%
879   \fi
880   \ignorespaces}
881 {\ignorespacesafterend}
882

```

`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.

```

883 \NewDocumentEnvironment{nofontexpansion}{}
884 {\ifdefined\microtypesetup
885   \microtypesetup{expansion=false}%

```

```

886   \fi
887   \ignorespaces}
888   {\ignorespacesafterend}

```

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

```

889 \let\nofontexpand=\nofontexpansion
890 \let\endnofontexpand=\endnofontexpansion
891

```

Character Protrusion

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

```

892 \NewDocumentEnvironment{nocharprotrusion}{}
893   {\ifdefined\microtypesetup
894     \microtypesetup{protrusion=false}%
895     \fi
896     \ignorespaces}
897   {\ignorespacesafterend}
898

```

A.13 Sloppy Paragraphs

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

```

899 \newcommand*{\typog@scaled@emergencystretch}[1]
900   {\emergencystretch=\ifdim\linewidth=\z@
901     #1%
902     \else
903       \dimexpr (#1) * \linewidth / \textwidth
904     \fi}
905

```

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

```

906 \NewDocumentCommand{\slightlyloppy}{0{1}}
907   {\ifcase #1% 0
908     % \tolerance=200
909     % \emergencystretch=\z@
910     % \hfuzz=.1\p@
911     % \vfuzz=\hfuzz
912     \fussy
913   \or % 1
914     \pretolerance=165%
915     \tolerance=330%
916     \typog@scaled@emergencystretch{.375em}%
917     \hfuzz=.15\p@
918     \vfuzz=\hfuzz
919   \or % 2
920     \pretolerance=265%

```

```
921     \tolerance=530%
922     \typog@scaled@emergencystretch{.75em}%
923     \hfuzz=.15\p@
924     \vfuzz=\hfuzz
925   \or % 3
926     \pretolerance=435%
927     \tolerance=870%
928     \typog@scaled@emergencystretch{1.125em}%
929     \hfuzz=.2\p@
930     \vfuzz=\hfuzz
931   \or % 4
932     \pretolerance=705%
933     \tolerance=1410%
934     \typog@scaled@emergencystretch{1.5em}%
935     \hfuzz=.3\p@
936     \vfuzz=\hfuzz
937   \or % 5
938     \pretolerance=1155%
939     \tolerance=2310%
940     \typog@scaled@emergencystretch{1.875em}%
941     \hfuzz=.35\p@
942     \vfuzz=\hfuzz
943   \or % 6
944     \pretolerance=1880%
945     \tolerance=3760%
946     \typog@scaled@emergencystretch{2.25em}%
947     \hfuzz=.4\p@
948     \vfuzz=\hfuzz
949   \or % 7
950     \pretolerance=3065%
951     \tolerance=6130%
952     \typog@scaled@emergencystretch{2.625em}%
953     \hfuzz=.45\p@
954     \vfuzz=\hfuzz
955   \else % >= 8
956     % \tolerance=9999
957     % \emergencystretch=3em
958     % \hfuzz=.5\p@
959     % \vfuzz=\hfuzz
960     \sloppy
961   \fi
962   \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.*)

```
963 \NewDocumentEnvironment{slightlyloppypar}{0{1}}
964   {\par\slightlyloppy[#1]\ignorespaces}
965   {\par}
966
```

A.14 Vertically Partially-Tied Paragraphs

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

```
967 \ExplSyntaxOn
968 \newcommand*{\typog@geometric@mean}[2]
969   {\fp_to_int:n {sqrt((#1) * (#2))}}
970 \ExplSyntaxOff
971
```

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

```
972 \newcounter{typog@mean@penalty}
973
```

`\vtietop`

```
974 \NewDocumentCommand{\vtietop}{0{3}}
975   {\setcounter{typog@mean@penalty}
976     {\typog@geometric@mean{\@M}{\clubpenalty}}}%
977   \typog@debug@typeout{vtietop: penalties \the\@M--\the\value{typog@mean@penalty}
-\the\clubpenalty}%
```

```

978 \unless\ifnum\clubpenalty<\@M
979 \PackageWarning{typog}{vtietop: clubpenalty=\the\clubpenalty\space>= 10000}%
980 \fi
981 \ifcase#1% 0
982 \relax
983 \or % 1
984 \relax
985 \or % 2
986 \clubpenalties 3
987 \@M
988 \value{typog@mean@penalty}
989 \clubpenalty
990 \or % 3
991 \clubpenalties 4
992 \@M \@M
993 \value{typog@mean@penalty}
994 \clubpenalty
995 \or % 4
996 \clubpenalties 5
997 \@M \@M \@M
998 \value{typog@mean@penalty}
999 \clubpenalty
1000 \or % 5
1001 \clubpenalties 6
1002 \@M \@M \@M \@M
1003 \value{typog@mean@penalty}
1004 \clubpenalty
1005 \or % 6
1006 \clubpenalties 7
1007 \@M \@M \@M \@M \@M
1008 \value{typog@mean@penalty}
1009 \clubpenalty
1010 \or % 7
1011 \clubpenalties 8
1012 \@M \@M \@M \@M \@M
1013 \value{typog@mean@penalty}
1014 \clubpenalty
1015 \or % 8
1016 \clubpenalties 9
1017 \@M \@M \@M \@M \@M
1018 \value{typog@mean@penalty}
1019 \clubpenalty
1020 \else % >= 9
1021 \clubpenalties 10
1022 \@M \@M \@M \@M \@M \@M \@M \@M
1023 \value{typog@mean@penalty}
1024 \clubpenalty
1025 \fi}
1026

```

vtietoppar (*env.*)

```
1027 \NewDocumentEnvironment{vtietoppar}{0{3}}
```

```

1028 {\vtietop[#1]}
1029 {\par
1030   \ignorespacesafterend}
1031

```

`\splicevtietop`

```

1032 \NewDocumentCommand{\splicevtietop}{0{3}}
1033   {\let\typog@old@item=\@item
1034    \def\@item[##1]{\typog@old@item[##1]\vtietop[#1]}%
1035    \ignorespaces}
1036

```

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).

```

1037 \ifdefined\SetEnumitemKey
1038   \SetEnumitemKey{vtietop}{first=\splicevtietop}
1039 \fi
1040

```

`\vtiebot`

```

1041 \NewDocumentCommand{\vtiebot}{0{3}}
1042   {\setcounter{typog@mean@penalty}
1043    {\typog@geometric@mean{\@M}{\widowpenalty}}%
1044    \typog@debug@typeout{vtiebot: penalties \the\@M--\the\value{typog@mean@penalty}
1045     -\the\widowpenalty}%
1046    \unless\ifnum\widowpenalty<\@M
1047      \PackageWarning{typog}{vtiebot: widowpenalty=\the\widowpenalty\space>= 10000}
1048    \fi
1049    \ifcase#1% 0
1050      \relax
1051    \or % 1
1052      \relax
1053    \or % 2
1054      \widowpenalties 3
1055      \@M
1056      \value{typog@mean@penalty}
1057      \widowpenalty
1058    \or % 3
1059      \widowpenalties 4
1060      \@M \@M
1061      \value{typog@mean@penalty}
1062      \widowpenalty
1063    \or % 4
1064      \widowpenalties 5
1065      \@M \@M \@M
1066      \value{typog@mean@penalty}
1067      \widowpenalty
1068    \or % 5
1069      \widowpenalties 6
1070      \@M \@M \@M \@M
1071      \value{typog@mean@penalty}
1072      \widowpenalty

```



```

1072 \or % 6
1073   \widowpenalties 7
1074     \@M \@M \@M \@M \@M
1075     \value{typog@mean@penalty}
1076     \widowpenalty
1077 \or % 7
1078   \widowpenalties 8
1079     \@M \@M \@M \@M \@M \@M
1080     \value{typog@mean@penalty}
1081     \widowpenalty
1082 \or % 8
1083   \widowpenalties 9
1084     \@M \@M \@M \@M \@M \@M \@M
1085     \value{typog@mean@penalty}
1086     \widowpenalty
1087 \else % >= 9
1088   \widowpenalties 10
1089     \@M \@M \@M \@M \@M \@M \@M \@M
1090     \value{typog@mean@penalty}
1091     \widowpenalty
1092 \fi}
1093

```

vtiebotpar (*env.*)

```

1094 \NewDocumentEnvironment{vtiebotpar}{0{3}}
1095   {\vtiebot[#1]}
1096   {\par
1097     \ignorespacesafterend}
1098

```

\typog@vtiebotdisp

```

1099 \NewDocumentCommand{\typog@vtiebotdisp}{m}
1100   {\setcounter{typog@mean@penalty}
1101     {\typog@geometric@mean{\@M}{\displaywidowpenalty}}%
1102   \typog@debug@typeout{vtiebotdisp: penalties \the\@M--\the\value{typog@mean@pen}
1103     -\the\displaywidowpenalty}%
1104   \unless\ifnum\displaywidowpenalty<\@M
1105     \PackageWarning{typog}{vtiebotdisp: displaywidowpenalty=\the\displaywidowpen}
1106   \fi
1107   \ifcase#1% 0
1108     \relax
1109   \or % 1
1110     \relax
1111   \or % 2
1112     \displaywidowpenalties 3
1113       \@M
1114       \value{typog@mean@penalty}
1115       \displaywidowpenalty
1116   \or % 3
1117     \displaywidowpenalties 4
1118       \@M \@M
1119       \value{typog@mean@penalty}

```

```

1119     \displaywidowpenalty
1120 \or % 4
1121     \displaywidowpenalties 5
1122     \@M \@M \@M
1123     \value{typog@mean@penalty}
1124     \displaywidowpenalty
1125 \or % 5
1126     \displaywidowpenalties 6
1127     \@M \@M \@M \@M
1128     \value{typog@mean@penalty}
1129     \displaywidowpenalty
1130 \or % 6
1131     \displaywidowpenalties 7
1132     \@M \@M \@M \@M \@M
1133     \value{typog@mean@penalty}
1134     \displaywidowpenalty
1135 \or % 7
1136     \displaywidowpenalties 8
1137     \@M \@M \@M \@M \@M \@M
1138     \value{typog@mean@penalty}
1139     \displaywidowpenalty
1140 \or % 8
1141     \displaywidowpenalties 9
1142     \@M \@M \@M \@M \@M \@M \@M
1143     \value{typog@mean@penalty}
1144     \displaywidowpenalty
1145 \else % >= 9
1146     \displaywidowpenalties 10
1147     \@M \@M \@M \@M \@M \@M \@M \@M
1148     \value{typog@mean@penalty}
1149     \displaywidowpenalty
1150 \fi}
1151

```

vtiebotdisp (*env.*)

```

1152 \NewDocumentEnvironment{vtiebotdisp}{0{3}}
1153 {\typog@vtiebotdisp{#1}}
1154 {\ignorespacesafterend}
1155

```

vtiebotdisptoppar (*env.*)

```

1156 \NewDocumentEnvironment{vtiebotdisptoppar}{0{3}o}
1157 {\postdisplaypenalty=\@M
1158 \predisplaypenalty=10001% in accordance with package ‘widows-
and-orphans’
1159 \edef\typog@@@top@lines{\IfNoValueTF{#2}{#1}{#2}}%
1160 \edef\typog@@@after@display@math{\vtietop[\typog@@@top@lines]}%
1161 \PushPostHook{display}{\aftergroup\typog@@@after@display@math}%
1162 \vtiebotdisp[#1]}
1163 {\par
1164 \PopPostHook{display}%
1165 \ignorespacesafterend}

```

1166

A.15 Breakable Disp. Eqs.

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

```
1167 \newenvironment*{breakabledisplay}[1][3]
1168   {\allowdisplaybreaks[#1]}
1169   {\ignorespacesafterend}
1170
```

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.

```
1171 \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.

```
1172 \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.

```
1173 \ExplSyntaxOn
1174 \cs_new:Npn \typog@setbaselineskip #1
1175 {
```

Initialize our “emergency-stop” loop counter.

```
1176   \int_set:Nn \l_tmpa_int {1}
1177   \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`.

```
1178   \dim_set:Nn \l_tmpa_dim {\glueexpr #1}
1179
1180   \typog@debug@typeout{\string\setbaselineskip:\space
1181     initial\space baselineskip:\space \the\baselineskip}
1182   \typog@debug@typeout{\string\setbaselineskip:\space
1183     target\space baselineskip:\space \dim_use:N \l_tmpa_dim}
1184
1185   \dim_compare:nNnTF {\baselineskip} > {\c_zero_dim}
1186   {}
1187   {
1188     \PackageError{typog}
1189       {\string\setbaselineskip:\space
1190         baselineskip\space not\space positive}
1191     {}
1192   }
1193
1194   \dim_compare:nNnTF {\l_tmpa_dim} > {\c_zero_dim}
```

```

1195 {}
1196 {
1197   \PackageError{typog}
1198     {\string\setbaselineskip:\space target\space
1199      baselineskip\space must\space be\space
1200      positive}
1201   {}
1202 }
1203
1204 \skip_if_eq:nnTF {\l_tmpa_dim} {\glueexpr #1}
1205 {}
1206 {
1207   \PackageWarning{typog}
1208     {\string\setbaselineskip:\space argument\space
1209      is\space a\space skip;\space
1210      will\space ignore\space glue}
1211   {}
1212 }
1213
1214 \fp_set:Nn \l_tmpa_fp {\l_tmpa_dim / \baselineskip}
1215 \fp_until_do:nNnn {abs(\l_tmpa_dim / \baselineskip - 1)} <
1216   {\typog@setbaselineskip@relative@error}
1217 {
1218   \setstretch{\fp_use:N \l_tmpa_fp}
1219   \fp_set:Nn \l_tmpa_fp
1220     {\l_tmpa_fp * \l_tmpa_dim / \baselineskip}
1221
1222   \int_incr:N \l_tmpa_int
1223   \int_compare:nNnTF {\l_tmpa_int} > {\l_tmpb_int}
1224   {
1225     \PackageError{typog}
1226       {\string\setbaselineskip:\space excessive\space
1227        number\space of\space iterations:\space
1228        \int_use:N \l_tmpa_int\space >\space
1229        \int_use:N \l_tmpb_int}
1230     {}
1231   }
1232   {}
1233 }
1234
1235 \typog@debug@typeout{\string\setbaselineskip:\space
1236   final\space \string\setstretch\space argument:\space
1237   \fp_use:N \l_tmpa_fp}
1238 \typog@debug@typeout{\string\setbaselineskip:\space
1239   final\space baselineskip:\space \the\baselineskip}
1240 }
1241

```

`\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.

```
1242 \cs_new:Npn \setbaselineskip #1
1243 {
1244   \AfterPreamble{\typog@setbaselineskip{#1}}
1245   \ignorespaces
1246 }
1247
```

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

```
1248 \cs_new:Npn \resetbaselineskip
1249 {
1250   \AfterPreamble{\setstretch{1}}
1251 }
1252
```

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

```
1253 \dim_new:N \typogfontsize
```

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

```
1254 \AfterEndPreamble{
1255   \dim_set:Nn \typogfontsize {\fontdimen6\font}
1256   \typog@debug@typeout{\string\typogfontsize =
1257     \dim_use:N \typogfontsize\space
1258     (at\space begin\space of\space document)}
1259 }
1260
```

`\setbaselineskippercentage`

```
1261 \cs_new:Npn \setbaselineskippercentage #1
1262 {
1263   \AfterPreamble{
1264     \dim_compare:nNnTF {\typogfontsize} > {\c_zero_dim}
1265     {
1266       \typog@setbaselineskip{
1267         \fp_eval:n {(#1) / 100} \typogfontsize}
1268     }

```

```

1269   {
1270     \PackageError{typog}
1271       {\string\setbaselineskippercentage:\space
1272        \string\typogfontsize <= 0}
1273       {Maybe\space \string\typogfontsize\space
1274        is\space uninitialized?}
1275   }
1276 }
1277 \ignorespaces
1278 }
1279

```

`\setleading`

```

1280 \cs_new:Npn \setleading #1
1281 {
1282   \AfterPreamble{
1283     \dim_compare:nNnTF {\typogfontsize} > {\c_zero_dim}
1284     {
1285       \typog@setbaselineskip{\typogfontsize + \dimexpr #1}
1286     }
1287     {
1288       \PackageError{typog}
1289         {\string\setleading:\space
1290          \string\typogfontsize <= 0}
1291         {Maybe\space \string\typogfontsize\space
1292          is\space uninitialized?}
1293     }
1294   }
1295   \ignorespaces
1296 }
1297

```

`\setleadingpercentage`

```

1298 \cs_new:Npn \setleadingpercentage #1
1299 {
1300   \AfterPreamble{
1301     \dim_compare:nNnTF {\typogfontsize} > {\c_zero_dim}
1302     {
1303       \typog@setbaselineskip{
1304         \fp_eval:n {1 + (#1) / 100} \typogfontsize}
1305     }
1306     {
1307       \PackageError{typog}
1308         {\string\setleadingpercentage:\space
1309          \string\typogfontsize <= 0}
1310         {Maybe\space \string\typogfontsize\space
1311          is\space uninitialized?}
1312     }
1313   }
1314   \ignorespaces
1315 }
1316 \ExplSyntaxOff

```

1317

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.

```
1318 \ExplSyntaxOn
1319 \cs_new_eq:NN \typog@repeat \prg_replicate:nn
1320
```

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

```
1321 \newcommand*{\typog@mod}[2]{\int_mod:nn{#1}{#2}}
1322 \ExplSyntaxOff
1323
```

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

```
1324 \newcommand*{\typog@triplet@max@lines}{99}
1325
```

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

```
1326 \define@key[typog]{smoothraggedrightshapetriplet}{leftskip}%
1327     {\def\typog@@triplet@leftskip{#1}}
1328 \define@key[typog]{smoothraggedrightshapetriplet}{parindent}%
1329     {\def\typog@@triplet@parindent{#1}}
1330 \NewDocumentEnvironment{smoothraggedrightshapetriplet}{0}{ m m m}
1331   {\def\typog@@triplet@leftskip{\z@}%
1332    \def\typog@@triplet@parindent{\z@}%
1333    \setkeys* [typog]{smoothraggedrightshapetriplet}{#1}%
1334    \skip0=\typog@@triplet@leftskip\relax
1335    \skip1=#2\relax
1336    \skip2=#3\relax
1337    \skip3=#4\relax
1338    \typog@debug@typeout{smoothraggedrightshapetriplet: skip0=\the\skip0}%
1339    \typog@debug@typeout{smoothraggedrightshapetriplet: skip1=\the\skip1}%
1340    \typog@debug@typeout{smoothraggedrightshapetriplet: skip2=\the\skip2}%
1341    \typog@debug@typeout{smoothraggedrightshapetriplet: skip3=\the\skip3}%
1342    \unless\ifnum\typog@mod{\typog@triplet@max@lines}{3}=0
1343      \PackageError{typog}
1344        {Line number of triplet generator\space
1345         (\typog@triplet@max@lines) not divisible by 3}
1346      {}
1347    \fi
1348    \edef\typog@@triplet@linespecs{%
1349      \glueexpr \skip0 + \typog@@triplet@parindent\relax
1350      \glueexpr \skip1 - \typog@@triplet@parindent\relax
1351      \skip0 \skip2 \skip0 \skip3
1352    \typog@repeat{\numexpr\typog@triplet@max@lines / 3 - 1}
```

```

1353             {\skip0 \skip1 \skip0 \skip2 \skip0 \skip3}}
1354   \parshape=\typog@triplet@max@lines\typog@triplet@linespecs\relax
1355   {\par}
1356

```

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

```

1357 \newcommand*{\typog@quintuplet@max@lines}{95}
1358

```

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

```

1359 \define@key[typog]{smoothraggedrightshapequintuplet}{leftskip}
1360   {\def\typog@@quintuplet@leftskip{#1}}
1361 \define@key[typog]{smoothraggedrightshapequintuplet}{parindent}
1362   {\def\typog@@quintuplet@parindent{#1}}
1363 \NewDocumentEnvironment{smoothraggedrightshapequintuplet}{0{} m m m m m}
1364   {\def\typog@@quintuplet@leftskip{\z@}%
1365   \def\typog@@quintuplet@parindent{\z@}%
1366   \setkeys*{typog}{smoothraggedrightshapequintuplet}{#1}%
1367   \skip0=\typog@@quintuplet@leftskip
1368   \skip1=#2\relax
1369   \skip2=#3\relax
1370   \skip3=#4\relax
1371   \skip4=#5\relax
1372   \skip5=#6\relax
1373   \typog@debug@typeout{smoothraggedrightshapequintuplet: skip0=\the\skip0}%
1374   \typog@debug@typeout{smoothraggedrightshapequintuplet: skip1=\the\skip1}%
1375   \typog@debug@typeout{smoothraggedrightshapequintuplet: skip2=\the\skip2}%
1376   \typog@debug@typeout{smoothraggedrightshapequintuplet: skip3=\the\skip3}%
1377   \typog@debug@typeout{smoothraggedrightshapequintuplet: skip4=\the\skip4}%
1378   \typog@debug@typeout{smoothraggedrightshapequintuplet: skip5=\the\skip5}%
1379   \unless\ifnum\typog@mod{\typog@quintuplet@max@lines}{5}=0
1380     \PackageError{typog}
1381       {Line number of quintuplet generator\space
1382       (\typog@quintuplet@max@lines) not divisible by 5}
1383     {}
1384   \fi
1385   \edef\typog@@quintuplet@linespecs{%
1386     \glueexpr \skip0 + \typog@@quintuplet@parindent\relax
1387     \glueexpr \skip1 - \typog@@quintuplet@parindent\relax
1388     \skip0 \skip2 \skip0 \skip3 \skip0 \skip4 \skip0 \skip5
1389     \typog@repeat{\numexpr\typog@quintuplet@max@lines / 5 - 1}
1390     {\skip0 \skip1 \skip0 \skip2 \skip0 \skip3 \skip0 \skip4 \s
1391     \parshape=\typog@quintuplet@max@lines\typog@@quintuplet@linespecs\relax}
1392   {\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.

```

1393 \newcommand*{\typog@septuplet@max@lines}{98}
1394

```


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

```

1395 \define@key[typog]{smoothraggedrightshapeseptuplet}{leftskip}%
1396         {\def\typog@septuplet@leftskip{#1}}
1397 \define@key[typog]{smoothraggedrightshapeseptuplet}{parindent}%
1398         {\def\typog@septuplet@parindent{#1}}
1399 \NewDocumentEnvironment{smoothraggedrightshapeseptuplet}{0}{ m m m m m m m}
1400 {\def\typog@septuplet@leftskip{z}%
1401  \def\typog@septuplet@parindent{z}%
1402  \setkeys* [typog]{smoothraggedrightshapeseptuplet}{#1}%
1403  \skip0=\typog@septuplet@leftskip
1404  \skip1=#2\relax
1405  \skip2=#3\relax
1406  \skip3=#4\relax
1407  \skip4=#5\relax
1408  \skip5=#6\relax
1409  \skip6=#7\relax
1410  \skip7=#8\relax
1411  \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip0=\the\skip0}%
1412  \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip1=\the\skip1}%
1413  \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip2=\the\skip2}%
1414  \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip3=\the\skip3}%
1415  \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip4=\the\skip4}%
1416  \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip5=\the\skip5}%
1417  \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip6=\the\skip6}%
1418  \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip7=\the\skip7}%
1419  \unless\ifnum\typog@mod{\typog@septuplet@max@lines}{7}=0
1420    \PackageError{typog}
1421      {Line number of septuplet generator\space
1422       (\typog@septuplet@max@lines) not divisible by 7}
1423    {}
1424  \fi
1425  \edef\typog@septuplet@linespecs{%
1426    \glueexpr \skip0 + \typog@septuplet@parindent\relax
1427      \glueexpr \skip1 - typog@septuplet@parindent\relax
1428        \skip0 \skip2 \skip0 \skip3 \skip0 \skip4 \skip0 \skip5 \
1429    \typog@repeat{\numexpr\typog@septuplet@max@lines / 7 - 1}
1430      {\skip0 \skip1 \skip0 \skip2 \skip0 \skip3 \skip0 \skip4 \s
1431    \parshape=\typog@septuplet@max@lines\typog@septuplet@linespecs\relax}
1432  {\par}
1433

```

smoothraggedrightfuzzfactor

```
1434 \newcommand*{\smoothraggedrightfuzzfactor}{1.0}
```

smoothraggedrightgenerator

```
1435 \newcommand*{\smoothraggedrightgenerator}{triplet}
```

\smoothraggedrightleftskip

```
1436 \newlength{\smoothraggedrightleftskip}
```

smoothraggedrightparindent

```
1437 \newlength{\smoothraggedrightparindent}
```

`\smoothraggedrightragwidth`

```
1438 \newlength{\smoothraggedrightragwidth}
1439 \setlength{\smoothraggedrightragwidth}{2em}
1440
```

`\typog@fuzzwidth (dimen)`

```
1441 \newdimen{\typog@fuzzwidth}
1442
```

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

```
1443 \define@key[typog]{smoothraggedrightpar}{linewidth}%
1444     {\def\typog@linewidth{#1}}
1445
1446 \NewDocumentEnvironment{smoothraggedrightpar}{0{}}
1447     {\edef\typog@linewidth{\linewidth}%
1448     \setkeys[typog]{smoothraggedrightpar}{#1}%
```

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

```
1449     \edef\typog@@generatorchoice{%
1450     \ifnum\pdf@strcmp{\smoothraggedrightgenerator}{triplet}=\z@
1451     0%
1452     \else
1453     \ifnum\pdf@strcmp{\smoothraggedrightgenerator}{quintuplet}=\z@
1454     1%
1455     \else
1456     \ifnum\pdf@strcmp{\smoothraggedrightgenerator}{septuplet}=\z@
1457     2%
1458     \else
1459     \PackageError{typog}
1460         {smoothraggedright: unknown generator name}
1461         {valid generator names are triplet, quin-
1462         tuplet, and septuplet}%
1462     \fi
1463     \fi
1464     \fi}%
```

Obey to the indentation prescribed by any list environment.

```
1465     \let\typog@@smoothraggedrightleftskip=\smoothraggedrightleftskip
1466     \ifnum\@listdepth>0
1467     \addtolength{\typog@@smoothraggedrightleftskip}{\leftmargin}%
1468     \fi
```

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

```
1469     \typog@fuzzwidth=\smoothraggedrightfuzzfactor\smoothraggedrightragwidth
```

Now for the generator-specific code...

```
1470     \ifcase\typog@@generatorchoice
```

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

```

1471     \typog@fuzzwidth=.25\smoothraggedrightragwidth
1472     \typog@debug@typeout{smoothraggedright: generator=triplet, ty-
pog@fuzzwidth=\the\typog@fuzzwidth}%
1473     \smoothraggedrightshapetriplet[leftskip=\typog@@smoothraggedrightleftskip,
1474     parindent=\glueexpr\smoothraggedrightparinden
indent,
1475     #1]%
1476     {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth
1477     + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
1478     {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth}% (3)
1479     {\glueexpr (\typog@@linewidth * 2 - \smoothraggedrightrag-
width) / 2
1480     + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (2)
1481     \or
generator=quintuplet.
1482     \typog@fuzzwidth=.125\smoothraggedrightragwidth
1483     \typog@debug@typeout{smoothraggedright: generator=quintuplet, ty-
pog@fuzzwidth=\the\typog@fuzzwidth}%
1484     \smoothraggedrightshapequintuplet[leftskip=\typog@@smoothraggedrightleftskip,
1485     parindent=\glueexpr\smoothraggedrightparinden
indent,
1486     #1]%
1487     {\glueexpr (\typog@@linewidth * 4 - \smoothraggedrightrag-
width * 3) / 4
1488     + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (2)
1489     {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth\relax}% (5)
1490     {\glueexpr (\typog@@linewidth * 2 - \smoothraggedrightrag-
width) / 2
1491     + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (3)
1492     {\glueexpr (\typog@@linewidth * 4 - \smoothraggedrightrag-
width) / 4
1493     + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
nus \typog@fuzzwidth\relax}% (4)
1494     {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth
1495     + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
1496     \or
generator=septuplet.
Permutation 3 - 6 - 1 - 5 - 2 - 7 - 4 looks ›random‹ enough for our purposes.
1497     \typog@fuzzwidth=.08333\smoothraggedrightragwidth
1498     \typog@debug@typeout{smoothraggedright: generator=septuplet, ty-
pog@fuzzwidth=\the\typog@fuzzwidth}%
1499     \smoothraggedrightshapeseptuplet[leftskip=\typog@@smoothraggedrightleftskip,
1500     parindent=\glueexpr\smoothraggedrightparinden
indent,
1501     #1]%
1502     {\glueexpr (\typog@@linewidth * 3 - \smoothraggedrightrag-
width * 2) / 3

```

```

1503         + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
      nus \typog@fuzzwidth\relax}% (3)
1504     {\glueexpr (\typog@@linewidth * 6 - \smoothraggedrightrag-
      width) / 6
1505         + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
      nus \typog@fuzzwidth\relax}% (6)
1506     {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth +
1507         + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
1508     {\glueexpr (\typog@@linewidth * 3 - \smoothraggedrightrag-
      width) / 3
1509         + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
      nus \typog@fuzzwidth\relax}% (5)
1510     {\glueexpr (\typog@@linewidth * 6 - \smoothraggedrightrag-
      width * 5) / 6
1511         + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
      nus \typog@fuzzwidth\relax}% (2)
1512     {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth\relax}% (7)
1513     {\glueexpr (\typog@@linewidth * 2 - \smoothraggedrightrag-
      width) / 2
1514         + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
      nus \typog@fuzzwidth\relax}% (4)
1515     \fi}
1516     {\ifcase\typog@@generatorchoice
1517         \endsmoothraggedrightshapetriplet
1518     \or
1519         \endsmoothraggedrightshapequintuplet
1520     \or
1521         \endsmoothraggedrightshapeseptuplet
1522     \fi}
1523

```

smoothraggedright (*env.*)

```

1524 \NewDocumentEnvironment{smoothraggedright}{0{}}
1525   {\PushPostHook{par}{\hskip-\parindent\smoothraggedrightpar[#1]\relax}}
1526   {\par\PopPostHook{par}}
1527

```

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 `LATEX` source file.

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

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

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

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

-P, --[no-]pager

Redirect output from *stdout* to the configured pager.

--show-config

Show the default configuration and exit.

--show-encodings

Show all known encodings and exit.

-V, --version

Show version information and exit.

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

Match only whole words.

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)`

Change History

v0.1		macro.	90
	General: Initial version.	i	
v0.2		<code>\narrowospace</code> : New macro.	75
	<code>\widespace</code> : Add fallback if		
	<code>\fontdimen7</code> is zero. Extend		
	with a starred version.	75	
v0.3		<code>hyphenmin</code> : New environment. . .	60
	<code>\resetbaselineskip</code> : New		
	macro.	89	
	<code>\setbaselineskip</code> : New macro.	88	
	<code>\setbaselineskippercentage</code> :		
	New macro.	89	
	<code>\setleading</code> : New macro.	90	
	<code>\setleadingpercentage</code> : New		
		macro.	90
		<code>\typogfontsize</code> : New dimen. . .	89
v0.4		<code>\lowercaseadjustlabelitems</code> :	
	New macro.	68	
	<code>\noadjustlabelitems</code> : New		
	macro.	68	
	<code>\typogadjuststairs</code> : New		
	macro.	71	
	<code>\typoggetnth</code> : New macro.	57	
	<code>\typoglowercaseadjustcheck</code> :		
	New macro.	72	
	<code>\typoguppercaseadjustcheck</code> :		
	New macro.	72	
	<code>\uppercaseadjustlabelitems</code> :		
	New macro.	68	

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] KHIREVICH, 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. We prefix all references to code lines with >ℓ<.

A

\allowhyphenation *10*, ℓ207
amsmath (package) *42*

B

baseline skip *44*
bookmark *14*
breakabledisplay (env.) *42*, ℓ1167
\breakpoint *13*, ℓ211
\breakpoint* *13*

C

\capitaldash *18*, ℓ343, ℓ351, ℓ373
\capitaldash* *18*
\capitalemdash *18*, ℓ354
\capitalemdash* *18*
\capitalendash *18*, ℓ338
\capitalendash* *18*
\capitalhyphen *18*, ℓ324
\capitalhyphen* *18*
\capitaltimes *20*, ℓ375
configuration *5*
covernextindentpar (env.) *30*, ℓ661
csquotes (package) *22*

D

debug (option) *2*
dimensions:
 \typogfontsize *45*, ℓ1253,
 ℓ1264, ℓ1267, ℓ1272, ℓ1273,
 ℓ1283, ℓ1285, ℓ1290, ℓ1291,
 ℓ1301, ℓ1304, ℓ1309, ℓ1310
\Doubleguillemetleft *21*, ℓ409
\doubleguillemetleft *20*, ℓ393
\Doubleguillemetright *21*, ℓ413
\doubleguillemetright *20*, ℓ397

E

enumitem-keys:
 vtietop *40*
environments:
 breakabledisplay *42*, ℓ1167
 covernextindentpar *30*, ℓ661
 hyphenmin *13*, ℓ221

lastlinecenteredpar *27*, ℓ637
lastlineflushrightpar *25*,
 ℓ634
lastlineraggedleftpar *25*,
 ℓ628
looespacing *31*, ℓ723
nocharprotrusion *37*, ℓ892
nofontexpand ℓ889
nofontexpansion *37*, ℓ883
openlastlinepar *30*, ℓ675
prolongpar *30*, ℓ651
setfontexpand *36*, ℓ869
setfontshrink *35*, ℓ841
setfontstretch *35*, ℓ855
setfonttracking *35*, ℓ755
shortenpar *30*, ℓ644
slightlyloppypar *37*, ℓ963
smoothraggedright *48*, ℓ1524
smoothraggedrightpar *47*,
 ℓ1443
smoothraggedrightshape-
 quintuplet *47*, ℓ1359
smoothraggedrightshape-
 septuplet *47*, ℓ1395
smoothraggedrightshape-
 triplet *47*, ℓ1326
tightspacing *31*, ℓ739
typoginspect *7*, ℓ168
typoginspectpar *7*, ℓ195
typogsetup *5*, ℓ99
vtiebotdisp *40*, ℓ1152
vtiebotdisptoppar *40*, ℓ1156
vtiebotpar *40*, ℓ1094
vtietoppar *39*, ℓ1027

F

\figuredash *19*, ℓ368
\figuredash* *19*
font
 encoding *20*
 expansion *29*, *35*
 information *6*
 protrusion *37*
 size *6*

- spacing 29, 31
 - loose 31
 - tight 31
- tracking 29, 35
- typeface
 - ADF Baskervald 21
 - ADF Accanthis 26
 - ADF Venturis 19, 26
 - Alegreya 19
 - Arvo 19
 - Bitter 19
 - Clara 19
 - CM Roman 26
 - Cochineal 26
 - Coelacanth 19
 - Crimson Pro 19
 - Crimson Text 19
 - Domitian 26
 - Droid Serif 19
 - EB Garamond 19, 21, 26
 - Erewhon 19
 - etbb 26
 - Extended Charter 26
 - fbf 19
 - Gentium 19, 21, 26
 - GFS Artemisia 19, 21
 - GFS Bodoni 26
 - GFS Didot 21, 26
 - Ibarra Real Nova 19
 - IBM Plex Serif 26
 - INRIA Serif 19
 - KP Serif 26
 - Libertine 19
 - Libertinus Serif 19, 21, 26
 - Libre Baskerville 19
 - Libre Caslon 19
 - Merriweather 19, 21
 - ML Modern 26
 - PT Serif 19
 - Quattrocento 19
 - Roboto Slab 19
 - Source Serif Pro 19, 26
 - Spectral 19, 26
 - STIX 19, 26
 - TeX Gyre Pagella 19
 - TX Fonts 19
 - URW Palladio ii, 26
 - Utopia Regular 26
- `\fontsizeinfo` 6, [ℓ147](#)
- forlorn line
 - club 30, 39
 - display widow 40
 - orphan 30
 - widow 30, 40
- H**
 - hyperref (package) 14
 - hyphenation 10
 - empty discretionary 13
 - first word 11
 - `\hskip` 11
 - re-enable automatic 10
 - hyphenmin (env.) 13, [ℓ221](#)
- I**
 - information 6
 - `\itcorr` [ℓ11](#), [15](#), [ℓ263](#)
 - `\itcorr*` 15
- K**
 - `\kernedhyphen` 17, [ℓ286](#), [ℓ308](#), [ℓ309](#), [ℓ316](#), [ℓ317](#)
 - `\kernedhyphen*` 17
 - `\kernedslash` 16, [ℓ145](#), [ℓ276](#)
 - `\kernedslash*` 16
 - kerning
 - extra 16
 - forward slash 16
 - hyphen 17
 - ligature 14
- L**
 - label items 22
 - `lastlinecenteredpar` (env.) 27, [ℓ637](#)
 - `lastlineflushrightpar` (env.) 25, [ℓ634](#)
 - `lastlineraggedleftpar` (env.) 25, [ℓ628](#)
 - leading 44
 - `\leftkernedhyphen` 17, [ℓ306](#), [ℓ311](#)
 - `\leftkernedhyphen*` 17
 - ligature 14
 - ligaturekern (option) 2
 - line spacing 6
 - list 39
 - `loosespacing` (env.) 31, [ℓ723](#)
 - `\lowercaseadjustlabelitems` 22, [ℓ509](#)
 - `lowercaselabelitemadjustments` (option) 2
- M**
 - `mathitalicscorrection` (option) 2
 - microtype (package) 34

N

narrow space 33
 \narrow space 33, [ℓ709](#)
 \narrow space* 33
 \narrow spacescale [ℓ708](#), [ℓ711](#),
 [ℓ713](#), [ℓ719](#), [ℓ720](#)
 \narrow space strength [ℓ707](#), [ℓ716](#)
 \noadjustlabelitems 22, [ℓ516](#)
 nocharprotrusion (env.) 37, [ℓ892](#)
 nodebug (option) 2
 nofontexpand (env.) [ℓ889](#)
 nofontexpansion (env.) 37, [ℓ883](#)
 \nolig 14, [ℓ228](#)
 \nolig* 14

O

openlastlinepar (env.) 30, [ℓ675](#)

P

package option
 debug 2
 ligaturekern 2
 lowercaselabelitem-
 adjustments 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
 uppercaselabelitem-
 adjustments 4
 package options 2–4
 page break 10, 42
 paragraph
 align last line 25
 centered 27
 flush right 25
 badness 9
 fill last line 27
 covernextindentpar 30
 \linebreak 29
 \mbox 29

openlastlinepar 30
 prolongpar 30
 shortenpar 30
 tie 29
 sloppy 37
 vertically tied 39

PDF 14

penalty (option) 2
 prolongpar (env.) 30, [ℓ651](#)

R

ragged right 47
 raise* (option) 3
 raisecapitaldash (option) 3
 raisecapitalguillemets (option) 3
 raisecapitalhyphen (option) 3
 raisecapitaltimes (option) 3
 raised character 17
 en-dash 18
 guillemets 20
 hyphen 18
 multiplication sign 20
 number dash 19
 raisefiguredash (option) 3
 raiseguillemets (option) 3
 reconfigure 5
 \resetbaselineskip 44, [ℓ1248](#)
 \rightkernedhyphen 17, [ℓ314](#), [ℓ319](#)
 \rightkernedhyphen* 17

S

\setbaselineskip 44, [ℓ1180](#),
 [ℓ1182](#), [ℓ1189](#), [ℓ1198](#), [ℓ1208](#),
 [ℓ1226](#), [ℓ1235](#), [ℓ1238](#), [ℓ1242](#)
 \setbaselineskippercentage 44,
 [ℓ1261](#)
 setfontexpand (env.) 36, [ℓ869](#)
 setfontshrink (env.) 35, [ℓ841](#)
 setfontstretch (env.) 35, [ℓ855](#)
 setfonttracking (env.) 35, [ℓ755](#)
 \setleading 45, [ℓ1280](#)
 \setleadingpercentage 45, [ℓ1298](#)
 setup 5
 shortenpar (env.) 30, [ℓ644](#)
 shrinklimits (option) 3
 \Singleguillemetleft 21, [ℓ401](#)
 \singleguillemetleft 20, [ℓ385](#)
 \Singleguillemetright 21, [ℓ405](#)
 \singleguillemetright 20, [ℓ389](#)
 slashkern (option) 4
 \slightlyloppy 37, [ℓ906](#), [ℓ964](#)
 slightlyloppypar (env.) 37, [ℓ963](#)

- smoothraggedright (env.) [48](#), [ℓ1524](#)
 \smoothraggedrightfuzzfactor [ℓ1434](#),
 [ℓ1469](#)
 \smoothraggedrightgenerator [ℓ1435](#),
 [ℓ1450](#), [ℓ1453](#), [ℓ1456](#)
 \smoothraggedrightleftskip [ℓ1436](#),
 [ℓ1465](#)
 smoothraggedrightpar (env.) [47](#),
 [ℓ1443](#)
 \smoothraggedrightparindent [ℓ1437](#),
 [ℓ1474](#), [ℓ1485](#), [ℓ1500](#)
 \smoothraggedrightragwidth [ℓ1438](#),
 [ℓ1469](#), [ℓ1471](#), [ℓ1476](#), [ℓ1479](#),
 [ℓ1482](#), [ℓ1487](#), [ℓ1490](#), [ℓ1492](#),
 [ℓ1494](#), [ℓ1497](#), [ℓ1502](#), [ℓ1504](#),
 [ℓ1506](#), [ℓ1508](#), [ℓ1510](#), [ℓ1513](#)
 smoothraggedrightshape-
 quintuplet (env.) [47](#), [ℓ1359](#)
 smoothraggedrightshape-
 septuplet (env.) [47](#), [ℓ1395](#)
 smoothraggedrightshape-
 triplet (env.) [47](#), [ℓ1326](#)
 \splicevtietop [39](#), [ℓ1032](#)
 stretchlimits (option) [3](#)
- T**
- textitalicscorrection (option) [4](#)
 tightspacing (env.) [31](#), [ℓ739](#)
 trackingttspacing (option) [4](#)
 \typogadjuststairs [24](#), [ℓ578](#)
 \typogadjuststairsfor [ℓ556](#),
 [ℓ597](#), [ℓ598](#), [ℓ599](#), [ℓ600](#)
 \typogfontsize (dim.) [45](#),
 [ℓ1253](#), [ℓ1264](#), [ℓ1267](#), [ℓ1272](#),
 [ℓ1273](#), [ℓ1283](#), [ℓ1285](#), [ℓ1290](#),
 [ℓ1291](#), [ℓ1301](#), [ℓ1304](#), [ℓ1309](#), [ℓ1310](#)
- \typogget [5](#), [ℓ108](#)
 \typoggetnth [6](#), [ℓ110](#)
 typoginspect (env.) [7](#), [ℓ168](#)
 typoginspectpar (env.) [7](#), [ℓ195](#)
 \typoglogo [ℓ11](#)
 \typoglowercaseadjustcheck [24](#),
 [ℓ621](#)
 typogsetup (env.) [5](#), [ℓ99](#)
 \typoguppercaseadjustcheck [24](#),
 [ℓ609](#)
- U**
- \uppercaseadjustlabelitem [22](#),
 [ℓ502](#)
 uppercaseadjustlabelitemadjustments (op-
 tion) [4](#)
- V**
- \vtiebot [40](#), [ℓ1041](#), [ℓ1095](#)
 \vtiebotdisp [ℓ1162](#)
 vtiebotdisp (env.) [40](#), [ℓ1152](#)
 vtiebotdisptoppar (env.) [40](#), [ℓ1156](#)
 vtiebotpar (env.) [40](#), [ℓ1094](#)
 \vtietop [39](#), [ℓ974](#), [ℓ1028](#), [ℓ1034](#), [ℓ1160](#)
 vtietop (enumitem-key) [40](#)
 vtietoppar (env.) [39](#), [ℓ1027](#)
- W**
- wide space [32](#)
 \widespace [32](#), [ℓ693](#)
 \widespace* [32](#)
 \widespacescale [ℓ692](#), [ℓ695](#),
 [ℓ697](#), [ℓ703](#), [ℓ704](#)
 \widespacestrength [ℓ691](#), [ℓ700](#)