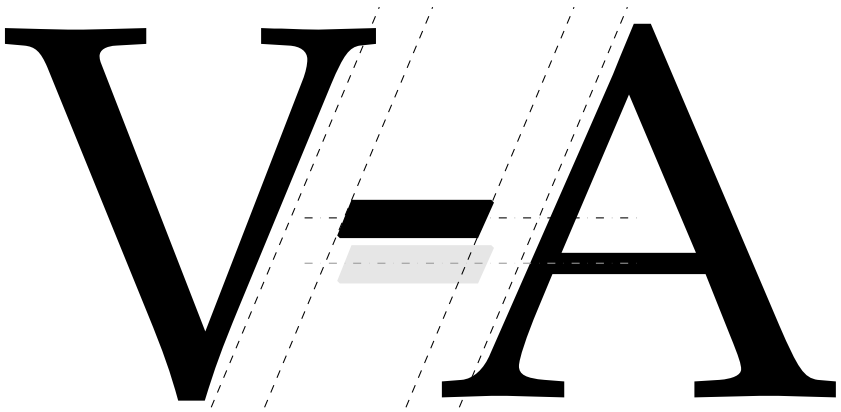
TypoG – Typographic Fine-Tuning

Ch. L. Spiel*

v0.5 2025/05/18

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.



* cspiel@users.sourceforge.org

Contents

Hoffentlich wird es nicht so schlimm, wie es schon ist!
— Karl Valentin

Table of Contents continued on next page.

The font sample on the title page was generated with the help of METAPOST using »URW Palladio«.

Table of Contents continued on next page.

List of Tables

Quick Reference

This is an alphabetically sorted list of all user macros and environments defined by package typog along with the page numbers of their descriptions. A list of all package options can be found on pages 3 to 6. The Index on pages 128 to 132 may provide some more detailed insights.

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

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

A

B

C

1 Introduction

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

LaTeX is the beginning of good typesetting – not the end. This package provides some tools for even better looking documents. When applied correctly its effects appear subtle and inconspicuous.

1.1 Overview

Package typog focuses on (micro-)typographic improvements.

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

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

Section 3.3 expands the hyphenation facilities of LaTeX.

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

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

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

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

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

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

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

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

1.2 Prerequisites

Package typog requires e-TEX; it relies on the LATEX3 interface. Parts of it are based on package microtype. However, if the functionality is not used, typog can be used without microtype. The same holds true for the setspace front-end.

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

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.«.

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

`breakpenalty=`⟨*penalty*⟩
: Penalty for a line break at various points. Default value: 50; initialized by the current `\exhyphenpenalty`: 50.

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

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

`emdashspace=`⟨*glue*⟩
SINCE V0.5
: Set the horizontal skip that is inserted before and after the em-dash in macros `\spacedemdash` and `\spacedcapitalemdash` to ⟨*glue*⟩. Default value: $^{56}/_{1000}$ em.

`endashspace=`⟨*glue*⟩
SINCE V0.5
: Set the horizontal skip that is inserted before and after the en-dash in macros `\spacedendash` and `\spacedcapitalendash` to ⟨*glue*⟩. Default value: $^{200}/_{1000}$ em plus $^{100}/_{1000}$ em minus $^{67}/_{1000}$ em.

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

`lowercaselabelitemadjustments=`{⟨*dim-1*⟩, ..., ⟨*dim-4*⟩}
SINCE V0.4
: Vertical shifts ⟨*dim-N*⟩ to apply to `\labelitem`⟨*N*⟩, where ⟨*N*⟩= 1, 2, 3, or 4 is the nesting level of the `itemize` list. Empty list elements are ignored. The special value ␣* instructs typog to preserve ⟨*dim-N*⟩ at that position. The adjustments apply to the lowercase setting (`\lowercaseadjustlabelitems`). See Sec. 3.8 (in particular subsection ›Setup‹ and Tab. 5 on p. 35) and also configuration option `uppercaseadjustlabelitem`.

  All four lengths default to 0 pt.

Important

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

`lowerslash=`⟨*dim*⟩

Since v0.5

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

`mathitalicscorrection=`⟨*dim*⟩

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

`raise*=`⟨*dim*⟩

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

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

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

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

`raisecapitaldash=`⟨*dim*⟩

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

`raisecapitalguillemets=`⟨*dim*⟩

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

`raisecapitalhyphen=`⟨*dim*⟩

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

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.

`raisefiguredash=`⟨*dim*⟩

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

`raiseguillemets=`⟨*dim*⟩

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

`raiseinvertedmarks={`⟨*dim-1*⟩, ⟨*dim-2*⟩, ⟨*dim-3*⟩`}`

Since v0.5

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

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

mark and \capitalinvertedquestionmark raise their associated inverted exclamation marks and inverted question marks. Each dimension corresponds to the optional indices of the macros. See Sec. 3.7.6.

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

`shrinklimits={`⟨*limit-1*⟩, ⟨*limit-2*⟩, ⟨*limit-3*⟩`}` microtype req.
`stretchlimits={`⟨*limit-1*⟩, ⟨*limit-2*⟩, ⟨*limit-3*⟩`}` microtype req.

Set the three limits, given in ¹⁄₁₀₀₀ em, of shrinkability and stretchability for the 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.

`slashkern=`⟨*dim*⟩

Set the size of the kerns before and after \kernedslash. See Sec. 3.6.1. Default value: ⁵⁰⁄₁₀₀₀ em.

`textitalicscorrection=`⟨*dim*⟩

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

`trackingttspacing={`⟨*outer-spacing*⟩`}` microtype req.

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

The argument ⟨*outer-spacing*⟩ gets passed to microtype's `\SetTracking` option `outer spacing` [25, Sec. 5.3]. If it contains commas, enclose the whole argument in curly braces. Default argument value: 300, 90, 60.

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

By default this option is unset.

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

Since v0.4

Vertical shifts ⟨*dim-N*⟩ to apply to `\labelitem`⟨*N*⟩, where ⟨*N*⟩=1, 2, 3, or 4 is the nesting level of the `itemize` list. Empty list elements are ignored. The special value ⁎ instructs typog to preserve ⟨*dim-N*⟩ at that position. The adjustments apply to the uppercase setting (`\uppercaseadjustlabelitems`). See Sec. 3.8 (in particular subsection ›Setup‹ and Tab. 5 on p. 35) 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 `\lower-caseadjustlabelitems` 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 ⟨*keys*⟩. An empty argument of `typogsetup` resets all ⟨*keys*⟩ to their default values.

```
\begin{typogsetup}{⟨keys⟩} ... \end{typogsetup}
```

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

```
\begin{typogsetup}{⟨keys⟩}
  ...
\end{typogsetup}
```

where ⟨*keys*⟩ have the same format as the package options described in Sec. 2.

Note

Use `\PassOptionsToPackage{`⟨*keys*⟩`}{typog}` to pass ⟨*keys*⟩ to typog before loading it and `\typogsetup{`⟨*keys*⟩`}` 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{⟨key⟩}
```

Retrieve the configuration value that is associated with ⟨*key*⟩. For a list of available ⟨*key*⟩s see Sec. 2.

Use Case

Raise glyphs by the same amount as configured with typog.

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

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

`\typoggetnth` SINCE V0.4

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

```
\typoggetnth{⟨csname⟩}{⟨key⟩}{⟨index⟩}
\typoggetnth{⟨dimen-register⟩}{⟨key⟩}{⟨index⟩}
```

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

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

Note

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

3.2 Information

Never forget: The visual output counts;
it must always be checked, […].
— UDO WERMUTH [32]

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

We define some functions for introspection of the typesetting process.

3.2.1 Font Information

`\fontsizeinfo`

Capture the font size[2] and line spacing[3] at the point where `\fontsizeinfo` *is called* in macro ⟨*cs-name*⟩. Both dimensions are measured in points (pt) and the results are rounded to tenths.

```
\fontsizeinfo{⟨cs-name⟩}
```

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

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

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

Note

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

Use Cases

Colophon. ¶ Font test pages. ■

3.2.2 Paragraph- and Page-Breaking Trace

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

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

```
\begin{typoginspect}[⟨option⟩]{⟨id⟩} ... \end{typoginspect}
\begin{typoginspectpar}[⟨option⟩]{⟨id⟩}
  ...
\end{typoginspectpar}
```

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

Option

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

Caution

The end-of-trace marker sometimes gets placed too early and the trace seems truncated. LaTeX reliably logs the requested trace information, but the write operations for trace data and `\immediate\write` which is used to print the end-tag are not synchronized. The workaround in such a situation is to enclose more text in the `typoginspect` environment (respecting the nesting of other environments of course). ■

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

```
<typog-inspect␣id="⟨id⟩"␣job="⟨jobname⟩"␣line="⟨line-number⟩"␣page="⟨page-number⟩">
  ...
</typog-inspect>
```

where the ⟨*id*⟩ is the user-supplied, unique[4] identifier of the group, ⟨*jobname*⟩ is the value of `\jobname`, ⟨*line-number*⟩ records the `\inputlineno` of the `\begin` of the group, and ⟨*page-number*⟩ gets replaced with the current value of the page counter.

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 A.

Tips

- It may be necessary to run whatever LaTeX 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 LaTeX.

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

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

  to the command-line.

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

4 It has turned out advantageous to use unique ⟨*id*⟩s. However, ⟨*id*⟩s are *not required* to be distinct.

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

Investigating the badness of a paragraph. It is generally unnecessary to determine the *exact* classification of a paragraph's badness [17, p. 97n], though the curious user can switch on logging of TEX's line-break information with `\tracing-paragraphs=1`[5] or simply use the typoginspect environment and check the suffixes

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

of each line in the paragraph, where for ⟨*suffix*⟩ the following mapping holds [17, p. 99]:

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

Example

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

1. The feasible breakpoint ␣@@␣ number 17 in the paragraph leads to
2. ␣line␣ 15, which is the loose ␣.1␣ last ␣-␣ line of the paragraph.
3. Up to this breakpoint the paragraph has picked up total demerits ␣t␣ of 142289.
4. The following two values only show up if `\lastlinefit` ≠ 0:
   (a) The shortfall ␣s␣ and
   (b) glue ␣a␣ or ␣g␣.[6]
5. The best[7] way to get here, i. e., @@17 is via ␣->␣ breakpoint ␣@@␣ 16. ■

Note

All of our guillemets were raised by 33⁄1000 em.

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

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

or

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

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

5 Reference 31 provides an exceptionally detailed discussion of the output of `\tracingparagraphs`.

6 The author is unaware of any descriptions of ␣s␣, ␣a␣, or ␣g␣ and the interested reader is referred to the source code, e. g., *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 TEX's page-break information [17, p. 112n].[8]

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

`%% goal height=`⟨*text-height*⟩`, max depth=`⟨*max-depth*⟩

where ⟨*text-height*⟩ is the total height TEX 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 TEX 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

TEX's and thus LATEX'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`

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

```
\allowhyphenation
```

Macro `\allowhyphenation` re-enables automatic hyphenation after TEX has turned it off, e. g. in a hyphenated compound.

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 ›spin-½‹

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

which is coded as `\mbox{spin-\textfrac{1}{2}}`. A line break between the text and the fraction would garble the term.

Use Cases

- All examples from the bottom of Tab. 1 on p. 14.
- Mend line breaks of index-entries in a narrow index:

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

  The first part, ›Transformations‹ is allowed to be hyphenated, but a break after the hyphen is prohibited as it results in a prowling em-dash at the beginning of the next line.
- Re-enable hyphenation when a macro decays into a `\hbox`:

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

  where `\rlap` is equivalent to something like `\makebox[0pt]{#1\hss}`.
- Use `\allowhyphenation` to turn on hyphenation of the first word of a paragraph as, e. g., in a narrow index or a `\marginpar`:

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

  A common trick to sweet-talk TEX into hyphenating the first word of a paragraph is to put `\hskip0pt` in front of it.
- Enable automatic hyphenation in the rare but weird cases when TEX does not hyphenate a word that is hyphenatable despite the result is an overfull box.[9] ■

Whenever using `\-`, the short-hand form of `\discretionary{-}{}{}`, authors writing in a foreign language should reconsider whether it really beats `\hyphenation` or `\babelhyphenation`[10] in the particular situation. However, sometimes `\-` actually *is* the way to go.

Let us assume we mark up proper names with

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

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

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

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

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

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

10 `\babelhyphenation` is the multi-lingual extension of TEX's `\hyphenation` and it is defined in package babel [6].

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

LATEX-Code	Result	Note
`hyphenated-compound`	hyphenated-compound	Most frequently used code; the hyphen `-` expands to `\discretionary{-}{}{-}` rendering the parts un-breakable
`hyphenated\mbox{-}%` `compound`	hyphenated-compound	Suppress hyphenation with the `\mbox` in the compound
`\mbox{hyphenated-%` `compound}`	hyphenated-compound	Avoid line break and thus hyphenation
`hyphenated\hyp` `compound`	hy- phen- ated- com- pound	Macro `\hyp` defined in package hyphenat [38]

`hyphenated%`
`\allowhyphenation-%`
`compound`
	hy-
	phen-
	ated-
	compound
		Macro `\allowhyphenation` of package typog; only unblock hyphenation of the first part

`hyphenated-%`
`\allowhyphenation`
`compound`
	hyphenated-
	com-
	pound
		Macro `\allowhyphenation` of package typog; only unblock hyphenation of the second part

`hyphenated%`
`\allowhyphenation`
`\mbox{-}%`
`compound`
	hy-
	phen-
	ated-compound
		Macro `\allowhyphenation` of package typog; hyphenate first part and keep the original hyphen unbreakable

`hyphenated%`
`\allowhyphenation-%`
`\allowhyphenation`
`compound`
	hy-
	phen-
	ated-
	com-
	pound
		Macro `\allowhyphenation` of package typog; hyphenate both parts, similar to `\hyp` shown above

***Tip* — Typewriter Fonts**

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

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

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

■

`\breakpoint`
`\breakpoint*`

The empty discretionary construct [17, p. 95], `\discretionary{}{}{}`, is helpful. It deserves its own macro – with a descriptive name.

```
\breakpoint    \breakpoint*
```

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

The difference between `\breakpoint` and the LaTeX macro `\allowbreak` is not only that the former has a starred form, but the penalty associated with `\breakpoint` is the current[11] `\exhyphenpenalty`, whereas `\allowbreak` statically assigns a zero penalty.

Use Case

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

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

■

hyphenmin (*env.*)
SINCE V0.3

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

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

Without optional argument `hyphenmin` sets both `\lefthyphenmin` and `\righthyphenmin` to ⟨*hyphen-minimum*⟩. When called with an optional argument it sets `\lefthyphenmin` to ⟨*left-hyphen-minimum*⟩ and `\righthyphenmin` to ⟨*hyphen-minimum*⟩.[12]

Use Case

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

11 At this point in the document `\exhyphenpenalty=50` holds.
12 The current values for `\lefthyphenmin` and `\righthyphenmin` in this document are 2 and 3.

3.4 Disable/Break Ligatures

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

```
\nolig*[⟨kerning⟩]
```

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 ⟨*kerning*⟩ 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[⟨kerning⟩]
```

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*[⟨kerning⟩]\relax    \nolig[⟨kerning⟩]\relax
\nolig*[⟨kerning⟩]{}        \nolig[⟨kerning⟩]{}
```

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

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

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

Use Cases

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

3.5 Manual Italic Correction

\itcorr
\itcorr*

The italic correction offered by TEX or LATEX sometimes needs a helping hand.

```
\itcorr{⟨strength⟩}    \itcorr*{⟨strength⟩}
```

In text mode macro \itcorr inserts a kern whose width is proportional to \fontdim1, which is the font's italic correction. If \fontdim1 happens to be zero (e. g. for an upright font), \itcorr uses the value set with textitalicscorrection instead of \fontdim1. The starred version always uses textitalicscorrection. In math mode macro \itcorr uses the value set with mathitalicscorrection[13] in both the starred and the unstarred form.

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

Use Cases

Stronger or weaker correction than \/. ¶ Correct a non-slanted or non-italicized font. ¶ Negative correction at the left-hand side[14] 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 $\llcorner f \lrcorner$ (before: 8, after: 7) or $\llcorner y \lrcorner$ (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

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

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

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

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

13 Separate adjustments may be desirable if the math font's italics have markedly different slants.
14 Groff has the machinery for left-italic-correction. Its font-metrics files support per glyph left-italic-correction values and users can access them conveniently via $\llcorner$\,,$\lrcorner$.

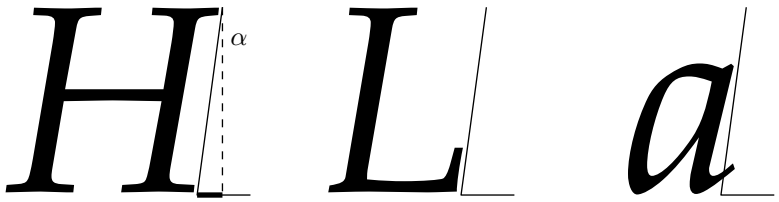


Figure 1: Some letters of an italics font. We use the capital ␣H␣ to measure the angle α between the plumb-line (drawn dashed) and a tangent to the rightmost parts of the glyph. The length of the plumb-line is proportional to ⟨*strength*⟩ and the short, thick part of the baseline symbolizes the resulting 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 TEX 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 `lastlinecenteredpar`.

3.6 Apply Extra Kerning or Spacing

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

3.6.1 Slash

`\kernedslash`
`\kernedslash*`

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

```
\kernedslash   \kernedslash*
```

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

Option `lowerslash`
introduced in v0.5

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

Tip

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

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

Generic solutions could build upon `\fontdimen5` and `\fontdimen6` (see Tab. 8 on p. 42) or measure the height and depth of the slash-glyph (abstracted

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

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

3.6.2 Hyphen

\kernedhyphen
\kernedhyphen⋆

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

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

Typeset an unbreakable hyphen with \kernedhyphen⋆ or a breakable hyphen (like \hyp of package hyphenat [38]) with \kernedhyphen and apply some kerning to left and to the right of it. The values ⟨*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 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.

\leftkernedhyphen
\leftkernedhyphen⋆
\rightkernedhyphen
\rightkernedhyphen⋆

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*⟩. ■

3.6.3 En-Dash and Em-Dash

The macros introduced in this sub-section allow for fine-tuning the white-space before and after en-dashes and em-dashes. Both sets of macros are designed for specific uses. The spaced en-dash macros are intended to be used where an en-dash is surrounded by normal spaces:

$$\langle WORD\rangle\,\_\text{-}\,\_\langle WORD\rangle.$$

The possible typographic improvement lies with the reduction of the width of the space around the en-dash with respect to the normal space.[15] On the other hand, the em-dash macros assume that there is no space between the ⟨*WORD*⟩s and the em-dash:

$$\langle WORD\rangle\text{—}\langle WORD\rangle.$$

The spaced em-dash macros add some (hairline-) space between the em-dash and the ⟨*WORD*⟩s, which may prevent the em-dash from running right into its neighbors.[16]

Line Breaking at Dashes. Package typog offers spaced en- and em-dashes that have their breakpoints on the left-hand side or on the right-hand side (or none for any of the starred versions) and indicates this in the names of the macros with `left` and `right`. All spaced dashes put equal amounts of space on both sides of their dashes, though; ›left‹ and ›right‹ refers to the breakpoints.

The insertion, »among …«, is separated by spaced en-dashes.

A single dash is used – among many other cases – to indicate a pause (e. g. announcing an action to be continued), instead of a comma, or even instead of a period. The typographic rules favor either no line break at all or a break at the right-hand side [4, 6.82] of the dash, which also is TEX's default.

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

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

   ```
   before␣--~insertion~--␣after
   ```

   For this solution typog offers `\leftspacedendash` and `\rightspaced-endash`:

   ```
   before\leftspacedendash
   insertion\rightspacedendash after
   ```

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

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

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

```
before~--␣insertion~--␣after
```

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

```
before\spacedendash
insertion\spacedendash after
```

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

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

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

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

Configure the size of the space before and after the dash with option endashspace.

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

***Important* — hyperref bookmarks**

If any of \leftspacedendash, \rightspacedendash, \spacedendash, \spaceddash, or \spacedemdash (see below) – whether starred or unstarred – occur in an argument that is processed with package hyperref for inclusion into the document's PDF-bookmarks an additional argument is necessary to parse the macro. This argument either is \relax or the empty group ␣{}␣.

We showcase the modifications of the plain calls for macro \spaceddash. They apply to all macros that generate spaced en- and em-dashes.

\spaceddash*[⟨*raise*⟩]\relax	\spaceddash[⟨*raise*⟩]\relax
\spaceddash*[⟨*raise*⟩]{}	\spaceddash[⟨*raise*⟩]{}

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

LaTeX will trip with »Undefined control sequence« on `\typog@missing@argument` if the extra argument is not passed.

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

`\swapendashskip`
Since v0.5

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

`\swapendashskip{`⟨*arg*⟩`}`

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

Table 2: Comparison of uncorrected and corrected `\rightspaced-endash-comma` combinations for a space width of ⅓ em and a (exaggerated) `endashspace` of ½ em. We have added »book struts« to the left and to the right of the results to indicate the width of the output.

Code	Result
`\rightspacedendash*,\space`	⌋ – ,⌊
`\rightspacedendash*\unskip,\space`	⌋ –,⌊
`\rightspacedendash*\swapendashskip,`	⌋ –,⌊

`\leftspacedemdash`
`\leftspacedemdash*`
`\rightspacedemdash`
`\rightspacedemdash*`
`\spacedemdash`
`\spacedemdash*`
All since v0.5

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

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

1. Tie the em-dashes to the insertion.

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

   Macro `\breakpoint` was introduced in Section 3.3.

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

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

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

The spaced solution with package typog like this:

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

`\leftspacedemdash[`⟨*raise*⟩`]`	`\leftspaceddash` (alias)
`\leftspacedemdash*[`⟨*raise*⟩`]`	`\leftspaceddash*` (alias)
`\rightspacedemdash[`⟨*raise*⟩`]`	`\rightspaceddash` (alias)
	`\spacedemdash` (alias)
`\rightspacedemdash*[`⟨*raise*⟩`]`	`\spacedemdash*` (alias)

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

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

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

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

Note

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

Tip

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

Example with the fictitious macro `\fancydash`:

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

3.7 Raise Selected Characters

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

We introduce a set of macros for the most common cases that typeset these characters at a user definable, adjusted height above the baseline. Readers familiar with OpenType fonts will be reminded of the `case` feature. Users can base their own definitions of raised characters on their associated dimensions.[17]

Caution

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

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

3.7.1 Capital Hyphen

`\capitalhyphen`
`\capitalhyphen*`

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

```
\capitalhyphen   \capitalhyphen*
```

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

Use Cases

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

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

3.7.2 Capital En-Dash and Capital Em-Dash

\capitalendash
\capitalendash*
\capitaldash
\capitaldash*

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

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

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

Use Cases

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

\capitalemdash
\capitalemdash*

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

```
\capitalemdash    \capitalemdash*
```

Use Cases

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

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

All since v0.5

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

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

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

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

All since v0.5

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

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

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

Tip

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

See the tip on p. 23 for a code example.

3.7.3 Number Dash (Figure Dash)

`\figuredash`
`\figuredash*`

`\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 3 summarizes some findings.

TABLE 3: Suggested values for raising `\figuredash`, which actually is an en-dash, between lining numerals of some selected fonts in multiples of 1 em.

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

* Free Bembo.

Other macros may be redefined with `\figuredash` for a consistent appearance of the copy, like, for example, `\citedash` (package cite [3]), or `\crefrangeconjunction` (package cleveref [11]).

Use Case

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

3.7.4 Multiplication Sign – Times ×

\capitaltimes

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

```
\capitaltimes
```

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

Use Case

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

3.7.5 Guillemets

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

For the implementation typog relies on the T1[18] font encoding not on package babel.

\singleguillemetleft
\singleguillemetright
\doubleguillemetleft
\doubleguillemetright

Lowercase Versions.

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

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

\Singleguillemetleft
\Singleguillemetright
\Doubleguillemetleft
\Doubleguillemetright

Uppercase Versions.

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

The companion set of single, double, left, and right quotes corrected for uppercase letters or lining numerals is \Singleguillemetleft and \Singleguillemetright and \Doubleguillemetleft and \doubleguillemetright. Mnemonic: These macros start with an uppercase letter. Their height above the baseline is adjusted with raisecapitalguillemets. Values of .025 em to .075 em are typical for fonts that need this kind of correction. Table 4 summarizes some findings.

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

Table 4: Suggested values for raising guillemets of some selected fonts in multiples of 1 em.

Font	Uppercase	Lowercase
EB Garamond, Libertinus, Merriweather, and many more	.05	.0
Gentium	.05	.025
GFS Artemisia, GFS Didot	.0625	.05
Charis SIL	.0667	.0333
ADF Baskervald	.0667	.04

Tip

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

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

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

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

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

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

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

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

Use Case

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

Anticipated Changes & Possible Extensions

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

3.7.6 Inverted Exclamation Mark and Inverted Question Mark

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

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

`\capitalinvertedexclamationmark`
`\capitalinvertedquestionmark`
Both since v0.5

The macros `\capitalinvertedexclamationmark` and `\capitalinvertedquestionmark` simultaneously provide bottom-aligned inverted marks for up to three different sets of exclamation marks or question marks, e. g., for uppercase, med-caps, and small-caps text[19] or just for some stylistic alternatives offered by particularly well equipped fonts.

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

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

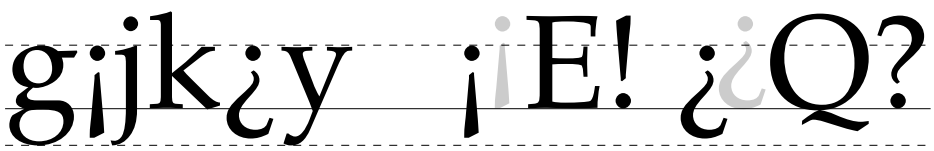


Figure 2: The left-hand sample shows the inverted exclamation marks and question marks wedged between some lowercase letters. The baseline is drawn with a solid lineand the ex-height as well as the descender depth are indicated with dashed lines. ¶ On the right hand side we display the same marks in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

Inverted exclamation and inverted question marks by the same ⟨*number*⟩ are raised by the same amount which is configurable with option `raiseinvertedmarks`, where ⟨*number*⟩ corresponds to the dimension at position ⟨*number*⟩. If

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

the dimension is equal to 0 pt the marks are auto-raised as shown in Fig. 2. This may amount to different lengths for the inverted exclamation mark and inverted question mark even at the same ⟨*number*⟩. A nonzero dimension raises the marks by exactly that length. To get a (quasi-)zero length use e. g. 1 sp.

Tip

The letters ¡V¿, ¡W¿, and ¡Y¿ following an inverted question mark may need some manual kerning as well as ¡J¿ following an inverted exclamation mark. Macro `\itcorr` (p. 17) offers a solution that is almost portable across font changes and a ⟨*strength*⟩ of −1 is a good starting point. ■

The user-side macros `\capitalinvertedexclamationmark{`⟨*number*⟩`}` and `\capitalinvertedquestionmark{`⟨*number*⟩`}` call the parameter-less, internal macros

```
\typog@inverted@exclamationmark@⟨roman-numeral⟩
\typog@inverted@questionmark@⟨roman-numeral⟩
```

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

3.8 Vertically Adjust Label Items of Environment `itemize`

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

The symbols that LaTeX uses to distinguish the items of `itemize` lists do not always align well in the vertical direction with the following text. Sometimes the label is too low, especially if followed by an uppercase (initial) letter. In rare occasions the label is placed too far above the baseline. If any label has been taken from a math-font vertical alignment with the text font is almost purely accidental.[20]

`\uppercaseadjustlabelitems`
`\lowercaseadjustlabelitems`
`\noadjustlabelitems`
ALL THREE SINCE V0.4

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: `dimen`s) to shift the labels up or down.

```
\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

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

themselves, this is the vertical shifts are configured with options `uppercaselabelitemadjustments` and `lowercaselabelitemadjustments`. They are doubly font dependent: on the one hand the font where the label itself comes from and on the other hand the font of the copy.

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

Example

With the flexible syntax the following settings are possible.

- Correct all `itemize` labels for uppercase letters.

```
\uppercaseadjustlabelitems{*}
```

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

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

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

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

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

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

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

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

The macros assume a ›correct‹ height that they derive from the measured height of a sample text scaled by a user-defined factor, which defaults to ½.[21] The then correct height gets indicated by a thin horizontal line parallel to the baseline. Thus, at sufficiently high magnifications it is possible to judge whether a label gets typeset too high or too low with respect to this reference line.

Note

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

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

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

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

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

Typeset label items that are height-adjusted according to `uppercaselabelitemadjustments` or `lowercaselabelitemadjustments` either for uppercase (macros starting with a capital ‚A') or lowercase (macros starting with a small ‚a'). These macros can be used anywhere. They are *not* controlled by `\uppercaseadjustlabelitems`, `\lowercaseadjustlabelitems`, nor `\noadjustlabelitems`.

Use Cases

Free, user-defined lists. ¶ Injection of typog's height-adjustment functionality in other packages as, for example, tasks [23]. ■

`\typogadjuststairs`
SINCE V0.4

To get a quick overview how the four `itemize` labels align vertically `\typogadjuststairs` draws them at user-defined steps, typically ¼ pt, ⅓ pt, or ½ pt. It ignores any existing adjustments and in that way can be utilized as a first configuration step or, for a small ⟨*step-size*⟩ and a high ⟨*number-of-steps*⟩, for an easy refinement.

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

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

21 The default factor of .5 hearkens back to STRIZVER'S suggestion that »[b]ullets should be centered on either the cap height or x-height of the neighboring text, ...« [28, p. 220].

times the height of ⟨*sample*⟩, where ⟨*scale-factor*⟩ defaults to .5. The stairs start at a vertical shift of

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

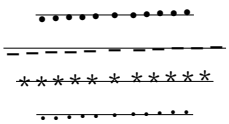
and repeat up

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

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

Example

Play ball!



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

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

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 `uppercaselabelitemadjustments` 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•–⁎·gjy, 0123•–⁎·4567,

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

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

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

They display as •, –, *, and ⋅.

3.9 Align Last Line of a Paragraph

The usual algorithms of LaTeX typeset the last line of a paragraph flush with the left margin unless `center`, `raggedleft` or `Centering`, `FlushRight` (package ragged2e [26]) are in effect. For an instructive discussion consult Ch. 17, »Paragraph End«, of Ref. 12. The following environments adjust the last lines of paragraphs in different ways.

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

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

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

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

`lastlinecenteredpar` (*env.*)

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

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

Use Cases

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

22 Also compare with the approach taken in Ref. 34.

TABLE 5: Some suggested values for the vertical adjustments of label items. The table assumes that the default definitions (of class article) for `\labelitem`⟨*N*⟩ are in effect. The `itemize`-list levels `i`, `ii`, `iii`, and `iv` are referred to with $N = 1, 2, 3, 4$. All lengths are given as printer points (pt) and relate to a document font size of 10 pt.

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

* EDWARD TUFTE's Bembo in package ETbb. Note the two initial capital letters in the filename.
† Found in package XCharter. Again note the two initial capitals in the filename.
‡ In package kpfonts.
§ Contained in package mathpazo.
‖ Utopia is available through package fourier or package mathdesign. For the latter pass option `adobe-utopia` to the package.

Note

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

3.10 Fill Last Line of a Paragraph

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

3.10.1 Problem 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 [34, O1]
   If the last line of a paragraph is shorter than the `\parindent` of the following paragraph a visual gap tears open.

   The same problem arises with displayed math in a flush-left[23] setting, e. g., amsmath [2] and option `fleqn`.[24]
   A possible remedy is to reflow the paragraph in a way that its last line is clearly wider than `\parindent`; a typical suggestion being twice the `\parindent`.

2. `\parindent` = 0 [34, O2]
   If the last line of a paragraph is completely filled with text, i. e., flush with the right margin, it may become hard to spot the start of the following paragraph unless `\parskip` is large.[25]

23 The common practice of centering displayed equations does not call for the manipulations of a paragraph's last line discussed here.
24 For displayed equations and amsmath the relevant parameter is `\mathindent`.
25 Package parskip defines `\parskip` as 6 pt plus 2 pt for a base size of 10 pt.

A possible, more legible solution is to reformat the paragraph 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`[26] for the gap [9].

Tip

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

TEX 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.

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

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

3.10.2 Manual Changes

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

1. End-of-paragraph intervention.
   (a) Tie ␣~
   Tie the last words.
   The problem with the tie may be a hyphenation of one of the words that participates in the tie. The next item avoids this disadvantage.
   (b) `\mbox`
   Join the last words or inline equation at the end of the paragraph with an `\mbox`.

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

(c) `\linebreak`

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

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

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

3.10.3 Multi-Purpose Environments

`shortenpar` (*env.*)
`prolongpar` (*env.*)

The two environments `shortenpar` and `prolongpar` can be employed in quite general situations when a paragraph should be typeset one line longer or shorter, e. g., to avoid a widow line[27] or a club line[28] [17, p. 104 and 21]. (See also Sec. 3.14

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

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

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

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

Environment `shortenpar` decreases the `\looseness` of the paragraph.[29] 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\parindent` (if `\parindent` ≠ 0), 2em (if `\parindent` = 0), or ⟨*dim*⟩ if called with an optional argument.

`openlastlinepar` (*env.*)

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

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

It may resolve case O2 as it attempts to prevent a completely filled line by introducing a partly unshrinkable `\parfillskip`. Without optional argument the threshold of unused last-line length is either `2\parindent` (if `\parindent` ≠ 0) or 2em (if `\parindent` = 0). The optional argument ⟨*dim*⟩ 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.

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

3.10.5 Consistent Spacing of Last Line

Since e-TEX the paragraph-breaking algorithm can be instructed to match the spacing of the last line of a paragraph to the next-to-last line with the built-in `\lastlinefit` [7, Sec. 3.8].[30] Package typog wraps this control in an environment to limit the scope as its effects are not always beneficial to the appearance of the body.

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

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

```
\begin{lastlinefitpar}[⟨value⟩] ... \end{lastlinefitpar}
```

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

Note that different ⟨*value*⟩s can lead to different breakpoints in the affected paragraph. e-TEX still considers the *whole* paragraph when calculating the optimal line breaks.

Use Cases

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

3.11 Spacing

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

The functions described in this section rely only on plain LATEX. No extra packages are required. Compare to the microtype-based functionality of Sec. 3.12.

3.11.1 Looser or Tighter Spacing

Never try to adjust lines by squeezing or stretching the tracking.
Go for the subtle solution: adjust word spacing instead.
— JAN MIDDENDORP [20, 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 ⟨*level*⟩s. However, the strictly managed stretchability/shrinkability may lead to many overfull boxes with `\fussy` or when applied to short lines.

30 The actual algorithm is somewhat involved. So, for ease of reference, we have included the relevant part of the e-TEX-manual as Appendix B on p. 68.

loosespacing (*env.*)
tightspacing (*env.*)

Environments `loosespacing` and `tightspacing` introduce four ⟨*level*⟩s of ›looseness‹ or ›tightness‹, where ⟨*level*⟩ = 0 disables the functionalities. The higher the ⟨*level*⟩ the looser or tighter the text will by typeset.

```
\begin{loosespacing}[⟨level⟩] ... \end{loosespacing}
```

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

⟨*level*⟩	Adjustment %	Note
0	n/a	neutral
1	+5	default
2	+10	
3	+20	
≥ 4	+30	

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

The default level of `loosespacing` is 1.

```
\begin{tightspacing}[⟨level⟩] ... \end{tightspacing}
```

Environment `tightspacing` decreases the width of a space by the percentages given in Tab. 7.

⟨*level*⟩	Adjustment %	Note
0	n/a	neutral
1	−1.25	default
2	−2.5	
3	−5	
≥ 4	−10	

TABLE 7: Adjustments made by environment `tightspacing` to `\spaceskip`. The mapping of ⟨*level*⟩ to the exact skip definitions are $1 \mapsto .9875^{+.0125}_{-.5}$, $2 \mapsto .975^{+.025}_{-.5}$, $3 \mapsto .95^{+.05}_{-.5}$, and $\geq 4 \mapsto .9^{+.1}_{-.5}$, where all factors scale with `\dimen2`, the current font's space-width.

The default level of `tightspacing` is 1.

Note

At a given ⟨*level*⟩ 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 \narrowspace derive their appearances from several of the current font's \fontdimen⟨*number*⟩s. TEX addresses the latter by integers, which is totally non-memnonic. Therefore, we play softball by first presenting Tab. 8 that associates the \fontdimen⟨*number*⟩s with their meanings and also reports on their current values (for this document).[31]

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

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

\widespace
\widespace*
STARRED FORM SINCE V0.2

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

```
\widespace    \widespace*
```

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

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

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

The sentence that ends with ›1.‹ uses \widespace after the period.

If \fontdimen7 happens to be zero \widespace uses

$$\texttt{\widespacescale} \times \texttt{\fontdimen2}$$

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

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

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

Use Case

Useful as a sentence-ending space if, for example, the sentence ends in an abbreviation with a period or decimal number without trailing digits *and* the next sentence should be delimited in a clearer way. ¶ Open tight lines with a series of \widespaces.[32] ■

3.11.3 Narrow Space

\narrowspace
\narrowspace*
Since v0.2

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

```
\narrowspace    \narrowspace*
```

The unstarred macro \narrowspace inserts a narrow space with the width

$$\texttt{\fontdimen2} - \texttt{\narrowspacestrength} \times \texttt{\fontdimen7}$$

if \fontdimen7 is different from zero or otherwise

$$\texttt{\narrowspacescale} \times \texttt{\fontdimen2}.$$

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

The stretchability and shrinkability of \narrowspace always get scaled with \narrowspacescale. Both factors, \narrowspacestrength and \narrowspacescale can be redefined by the user; their defaults are .5 and .9375.

Use Case

Tighten loose lines with a series of \narrowspaces.[33] ■

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

Macro	Width	Stretch	Shrink
\narrowspace	21.4 21.8	10.9	7.3
\space	23.3	11.6	7.8
\widespace	27.2 26.2	13.1	8.7

32 See also »Investigating the badness of a paragraph« on Page 11.
33 Footnote 32 again applies.

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

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

in the document preamble. ■

3.12.1 Tracking

Caution

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

`setfonttracking` (*env.*) Override the default tracking for all fonts.

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

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

See Sec. 5.3, ›Tracking‹, and 7, »Letterspacing revisited«, in the documentation of microtype [25] for a detailed explanation.

For font combinations involving monospaced fonts (TEX 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 `\smash`ed symbols of inline math. – Think of integrals. ¶ Control the length of the last line in a paragraph. ■

3.12.2 Font Expansion

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

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

```
\begin{setfontshrink}[⟨level⟩] ... \end{setfontshrink}
\begin{setfontstretch}[⟨level⟩] ... \end{setfontstretch}
```

A ⟨*level*⟩ of zero is a no-op. Tables 10 and 11 summarize the values for `stretch` and `shrink` in these environments.

⟨*level*⟩	`stretch`	`shrink`	Note
0	n/a	n/a	no operation
1	0	5	default
2	0	10	
3	0	20	

TABLE 10: Preconfigured values for `shrink` inside of environment `setfontshrink` as ⅟1000 em. Note that all `stretch` values are zero, so the fonts only can shrink.

⟨*level*⟩	`stretch`	`shrink`	
0	n/a	n/a	no operation
1	5	0	default
2	10	0	
3	20	0	

TABLE 11: Preconfigured values for `stretch` inside of environment `setfontstretch` as ⅟1000 em. Note that all `shrink` values are zero, so the fonts only can stretch.

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

Use Cases

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

`setfontexpand` (*env.*)

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

```
\begin{setfontexpand}[⟨level⟩] ... \end{setfontexpand}
```

Table 12 gives an overview of the values associated with ⟨*level*⟩.

⟨*level*⟩	`stretch`	`shrink`	Note
0	n/a	n/a	no operation
1	5	5	default
2	10	10	
3	20	20	

TABLE 12: Preconfigured values for `shrink` and `stretch` inside of environment `setfontexpand` as ⅟1000 em. Note that both `shrink` and `stretch` values are nonzero, so the fonts can shrink or expand.

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

Notes

- Environment `setfontexpand` shares its `shrinklimits` with `setfontshrink` and its `stretchlimits` with `setfontstretch`.
- These environments do not nail down any font's expansion but only set up its available range. See Sec. 3.3, »Font Expansion«, in the microtype documentation [25].

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

Use Cases

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

`nofontexpansion` (*env.*)
`nofontexpand` (*env.*)

Disable the microtype feature ›expansion‹ inside of the environment.

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

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

Use Cases

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

3.12.3 Character Protrusion

`nocharprotrusion` (*env.*)

Disable the microtype feature ›protrusion‹ inside of the environment.

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

Use Cases

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

3.13 Sloppy Paragraphs

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

`\slightlysloppy`
`slightlysloppypar` (*env.*)

We define the macro `\slightlysloppy` and the associated environment, `slightlysloppypar`, with a user-selectable ⟨*sloppiness*⟩ parameter. The constructions recover the known settings `\fussy` (⟨*sloppiness*⟩ = 0) and `\sloppy` (⟨*sloppiness*⟩ ≥ 8), and introduce seven intermediate ⟨*sloppiness*⟩ levels.[34] The default ⟨*sloppiness*⟩ is 1.

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

```
\slightlysloppy[⟨sloppiness⟩]
\begin{slightlysloppypar}[⟨sloppiness⟩]
  ...
\end{slightlysloppypar}
```

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

TABLE 13: Adjustments made by `\slightlysloppy` to various TEX parameters at different levels of ⟨*sloppiness*⟩.

⟨*sloppiness*⟩	`\tolerance`	`\hfuzz` `\vfuzz` pt	`\emergencystretch` *G* em	Note
0	200	.1	0	TEX: `\fussy`
1	330†	.15	.375‡	default
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: `\sloppy`

† 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

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

to prevent excessive stretchability in narrow lines.

Environment `slightlysloppypar[⟨sloppiness⟩]` mimics LATEX's `sloppypar`, while offering the flexibility of `\slightlysloppy`.

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

LaTeX provides several macros and environments to tie material vertically – most prominently `samepage` and `minipage`.[35] Typog's macros and environments constitute more sophisticated but weaker forms of these. They tie only the first or last couple of lines in a paragraph while the rest of the paragraph gets broken into pages by TeX in the usual way.

The macros and environments described in this section locally set e-TeX penalty arrays [7, Sec. 3.8]. In addition the environments `vtietoppar`, `vtiebotpar`, and `vtiebotdisptoppar` explicitly issue a `\par` at the end of the group.

`\vtietop` Avoid a club line in each partial paragraph.

`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 LaTeX:

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

Alternatively with package enumitem [5]:

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

or shorter and with the default ⟨*number-of-lines*⟩, 3, using the enumitem style[36] `vtietop`:

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

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

vtietop (*enumitem key*)

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

\vtiebot Avoid a widow line in each partial paragraph.

vtiebotpar (*env.*)

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

Vertically tie the last ⟨*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.

vtiebotdisp (*env.*) Avoid a display widow line in each partial paragraph.

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

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

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

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

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

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

Vertically tie the last ⟨*before-disp-number-of-lines*⟩ in the paragraph before a display and the first ⟨*after-disp-number-of-lines*⟩ in the paragraph after the display. Moreover, turn the paragraphs and the display into an un-breakable unit.[37]

Zero or one for ⟨*before-disp-number-of-lines*⟩ as well as ⟨*after-disp-number-of-lines*⟩ are no-ops for the paragraph. Up to nine lines each can be fused.

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

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

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

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

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

Partial Paragraphs And Counting Lines. The top-of-paragraph ties, `\vtietop` and `vtietoppar` count ⟨*number-of-lines*⟩ from the beginning of every partial paragraph. Each displayed math in the paragraph resets the count. The bottom-paragraph ties, `\vtiebot`, `vtiebotpar`, `\vtiebotdisp`, and `vtiebotdisppar` count backward from the end of each partial paragraph. Again, each displayed math in the paragraph resets the count. According to TEX's rules, a displayed math formula always is counted as *three* lines no matter its contents. Table 14 summarizes these rules with the help of an example.

TABLE 14: Exemplary, eight-line paragraph compounded of two partial paragraphs of three and two lines and a displayed math formula of arbitrary size sandwiched in between.

Continuous Line Number	Example Contents	`\vtietop`† Count	`\vtiebot`‡ Count
1	Text line$_1$	1	3
2	Text line$_2$	2	2
3	Text line$_3$	3	1
4 5 6	Display math		
7	Text line$_4$	1	2
8	Text line$_5$	2	1

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

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

Tips

- The environments can be combined to arrive at paragraphs that simultaneously are protected against club lines and (display) widow lines.
- For very long derivations that are not interrupted and thus made breakable with the help of `\intertext`[38] or `\shortintertext`[39] it is desirable to make the display breakable. This is achieved with `\allowdisplaybreaks` or the environment `breakabledisplay` which will be described in Sec. 3.15. ■

Use Cases

Avoid widows and orphans, e. g., those turned up by package widows-and-orphans [22]. ¶ Extend the typographic convention of »three to four lines instead of a single club or widow line« to a context-dependent number of lines that tries to keep all (well, dream on) the information together the reader needs at that particular point. ■

3.15 Breakable Displayed Equations

`breakabledisplay` (*env.*)

Package amsmath offers `\allowdisplaybreaks` to render displayed equations breakable at each of their lines. Environment `\breakabledisplay` is a wrapper around it which limits the macro's influence to the environment. Furthermore, the default ⟨*level*⟩ 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}[⟨level⟩]
  ...
\end{breakabledisplay}
```

Environment `breakabledisplay` simply passes on ⟨*level*⟩ to `\allowdisplaybreaks`. Table 15 shows the default penalties that amsmath associated with each of the ⟨*level*⟩s.

Tips

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

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

38 Introduced in package amsmath [2].
39 Defined in package mathtools [14].

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

⟨*level*⟩	`\interdisplay-linepenalty`	Note
0	10000	no operation
1	9999	
2	6999	
3	2999	default
4	0†	

† This is the default of `\allowdisplaybreaks`.

Use Cases

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

3.16 Setspace Front-End

Package setspace [29] is a base hit when it comes to consistently setting the line skip for a document via the macro `\setstretch`. The interface of `\setstretch` though is unintuitive as it asks for an obscure factor. The LaTeX user however prefers to keep her eyes on the ball and set the line skip directly (e. g. 12.5 pt) or the lines' leading to a length or percentage of the font's size.[40] This is where the following macros go to bat.

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

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.

40 To find out about the current font's size and the `\baselineskip` in printable form check out Sec. 3.2.1 on p. 8.

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.

```
\setbaselineskippercentage{⟨baselineskip-percentage⟩}
```

Set `\baselineskip` to `\typogfontsize` × ⟨*baselineskip-percentage*⟩/100.

Example

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

```
\setbaselineskippercentage{125}
```

which sets `\baselineskip` to 10 pt × 125/100 = 12.5 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 10pt + 2.5pt = 12.5pt. ■

`\setleadingpercentage` SINCE V0.3

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

```
\setleadingpercentage{⟨leading-percentage⟩}
```

Set `\baselineskip` to `\typogfontsize` × (1 + ⟨*leading-percentage*⟩/100).

Example

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

```
\setleadingpercentage{25}
```

which sets `\baselineskip` to 10 pt × (1 + 25/100) = 12.5 pt. ■

`\typogfontsize` (*dimen*) SINCE V0.3

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

The length `\typogfontsize` gets initialized at the end of the preamble to the default font's quad size:[41]

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

which is also called its »nominal size« or its »design size«. This assignment can be repeated at any point in the document to record a reference font's size. To set just `\typogfontsize` without changing the current font, encapsulate the font change in a group and export the new value:

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

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

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

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

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

or

```
\setleadingpercentage{0}
```

Tip

All macros in this section actually accept expressions of their argument types, though the sick rules of TEX ⟨*dimen*⟩- and ⟨*skip*⟩-expressions apply.

Here are some forms that do work:

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

■

41 For an overview of the various `\fontdimen`⟨*number*⟩ parameters consult Tab. 8 on p. 42.

3.17 Smooth Ragged

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

Package typog implements a novel approach to typeset ragged paragraphs. Instead of setting the glue inside of a paragraph to zero and letting the line-widths vary accordingly [35] 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. ■

`smoothraggedrightshapetriplet` (*env.*)
`smoothraggedrightshapequintuplet` (*env.*)
`smoothraggedrightshapeseptuplet` (*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): `smoothraggedrightshapetriplet`, `smoothraggedrightshapequintuplet`, and `smoothraggedrightshapeseptuplet`; they work for paragraph lengths up to 99, 95, and 98 lines.

```
\begin{smoothraggedrightshapetriplet}[⟨option⟩...]{⟨width1⟩}{⟨width2⟩}{⟨width3⟩}
  ...
\end{smoothraggedrightshapetriplet}
\begin{smoothraggedrightshapequintuplet}[⟨option⟩...]{⟨width1⟩}...{⟨width5⟩}
  ...
\end{smoothraggedrightshapequintuplet}
\beginsmoothraggedrightshapeseptuplet[⟨option⟩...]{⟨width1⟩}...{⟨width7⟩}
  ...
\end{smoothraggedrightshapeseptuplet}
```

The environments take $N = 3$, 5, or 7 mandatory line-width parameters, where each ⟨*width1*⟩, $I = 1, \ldots, N$ is a skip, i. e., a dimen that can include some glue.

Options

`leftskip`=⟨*dim*⟩
: Set the left margin for the smooth ragged paragraph to ⟨*dim*⟩. Similar to the TEX parameter `\leftskip`.

`parindent`=⟨*dim*⟩
: Set the first-line indent for the smooth ragged paragraph to ⟨*dim*⟩. Similar to the TEX parameter `\parindent`.

`smoothraggedrightpar` (*env.*)

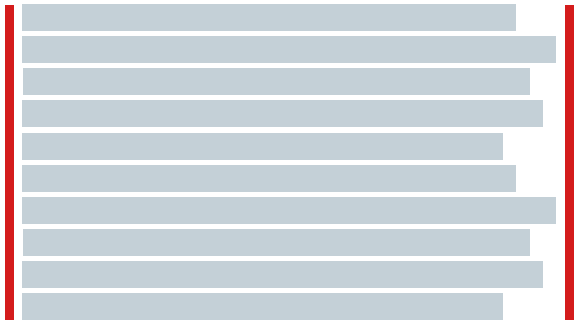
Environment `smoothraggedrightpar` builds upon the three generators. It typesets a single paragraph with a given ⟨*ragwidth*⟩ of the ragged, right margin, where the rag width is the length-difference of the longest and the shortest lines.

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

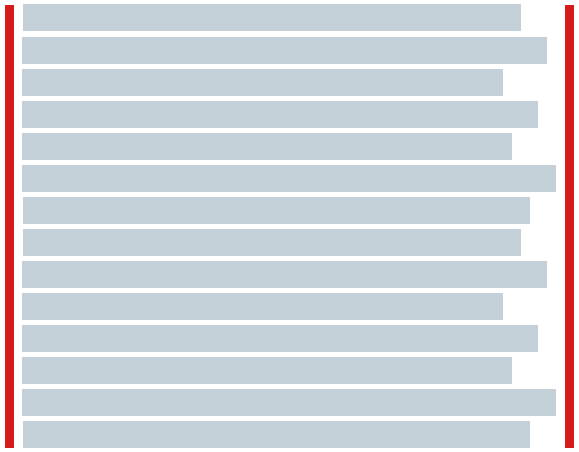
The line lengths equally divide the ragged margin, i. e., they are arithmetic means with respect to the generator size.

— 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}[⟨option⟩...]
  ...
\end{smoothraggedright}
```

Options

`linewidth`=⟨*dim*⟩
: Override the length of the longest line. The default line-width is `\linewidth`.

Global Parameters

`\smoothraggedrightfuzzfactor`=⟨*factor*⟩
: The environment adds glue to every line-width[42] 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`=⟨*dim*⟩
: Value for `leftskip` to pass to the generator. Default: 0 pt.

`\smoothraggedrightparindent`=⟨*dim*⟩
: Value for `parindent` to pass to the generator. Default: 0 pt.

`\smoothraggedrightragwidth`=⟨*dim*⟩
: Value for the width of the ragged right margin. Default: 2 em.

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

Throughout this manual we have demonstrated how `smoothraggedright` environments work for very narrow columns namely inside of the document's margins: all maginal notes were typeset inside of `\smoothraggedright` environments (quintuplet generator, 1.5 em rag width, at footnote size in addition using environments `slightlysloppy` 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` [26]. ¶ 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 [18]. Galley code is supposed to work inside of lists, too. ■

4 Limitations and Known Problems

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

Interference with package pdfcomment [16]. To play well with pdfcomment package typog *turns off* the default string substitution inside of PDF-bookmarks for the following macros (including their starred variants) if *both* packages are loaded.

The sentence uses \spaced-endashes.

Use \texorpdfstring – which is part of package hyperref [24] – in the relevant sectioning macros to recover string substitution on a case-by-case basis.

Example

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

■

5 Other Packages for Fine LaTeX Typography

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

enumitem — Flexible and consistent definition of all basic LaTeX-list types plus inline lists [5].

geometry — Powerful and sophisticated setup of the page layout [30]. Best accompanied by layout [19] to visualize the page geometries.

hyphenat — Hyphens that do not inhibit further auto-hyphenation of a compound word [38].

microtype — Fine control of spacing, tracking, sidebearings, character protrusion into the margins, font expansion, and much more [25].

See Section 3.12 for a front-end to microtype offered by this package.

See also Khirevich's discussion [15].

ragged2e — Improved versions of environments `raggedleft`, `raggedright`, and `center` [26].

setspace — Consistently set the line-spacing of a document, i. e., control `\baselineskip` [29].

See Section 3.16 for a front-end to setspace offered by this package.

A typog-grep

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

A.1 Name

typog-grep - specialized grep for typog-inspect elements in LaTeX log files

A.2 Synopsis

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

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

The first form, "discovery mode", shows all *ID*s 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 *ID*s match *REGEXP* in *LOG-FILE*.

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

A.3 Description

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

In the LaTeX 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 *ID*s 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 LaTeX source file line number of the beginning of the `typoginspect` environment, and *PAGE-NUMBER* is the page where the output of `TEXT-OR-CODE-TO-INVESTIGATE` occurs.

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

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

A.4 Options

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

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

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

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 *ID*s. The argument *WHEN* determines when to apply color: `always`, `never`, or `auto`. The setting `auto` checks whether standard output has been redirected. This is the default.

-C, **--config** *KEY=VALUE*[:*KEY=VALUE*[:...]]

Set one or more configuration *KEY* to *VALUE* pairs. See section A.5 for a description of all available configuration items. Use option --**show-config** to display the default configuration.

--debug

Turn on debug output on *stderr*.

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

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

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

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

-h, **--help**

Display brief help then exit.

-i, **--[no-]id**

Print the actual ID-name that matched *REGEXP*. Control the appearance of the matching *ID* with configuration item `id-heading`.

-y, **--[no-]ignore-case**

Match *ID*s while ignoring case distinctions in patterns and data.

-j, **--[no-]job-name**

Print the `\jobname` that **latex** associated with the input file.

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

Print the line number where the `typoginspect` environment was encountered in the LaTeX source file.

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

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

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

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

-P, **--[no-]pager**

Redirect output from *stdout* to the configured pager.

--show-config

Show the default configuration and exit.

--show-encodings

Show all known encodings and exit.

-V, --version

Show version information and exit.

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

Match only whole words.

A.5 Configuration

`id-format=`*FORMAT*

Control the *FORMAT* for printing matching ids in inline-mode, where *FORMAT* is passed to Perl's `printf`. Default: `%s:`.

`id-heading=0|1`

Choose between printing the matching *ID*s with option **--id**: Inline (`0`) or heading before the matching data (`1`). Default: `0`.

`id-heading-format=`*FORMAT*

Control the *FORMAT* for printing matching *ID*s 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. It 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 *ID*s when printed inline.

`id-heading-color`
: Color of matching *ID*s when printed in heading form.

`line-break-pass-color`
: Color of the lines showing which pass (e.g., `@firstpass`) of the line-breaking algorithm is active.

`line-number-color`
: Color of TeX-source-file line numbers.

`log-line-number-color`
: Color of log-file line numbers.

`math-color`
: Color used for math expressions including their font specs.

`page-number-color`
: Color of page numbers of the final output.

`tightness-color`
: Color of lines with Tight/Loose hbox reports.

`vertical-breakpoint-color`
: Color of possible vertical breakpoints.

A.5.1 Brief summary of colors and attributes

Foreground Color

`black`, `red`, `green`, `yellow`, `blue`, `magenta`, `cyan`, `white`,

Prefix with `bright_` for high-intensity or bold foreground.

Foreground Grey

`grey0`, ..., `grey23`

Background Color

`on_black`, `on_red`, `on_green`, `on_yellow`, `on_blue`, `on_magenta`, `on_cyan`, `on_white`

Replace `on_` with `on_bright_` for high-intensity or bold background.

Background Grey

`on_grey0`, ..., `on_grey23`

Text Attribute

`bold`, `dark`, `italic`, `underline`, `reverse`

A.5.2 Some common encodings

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

Latin-1, Western European

`iso-8859-1`, `cp850`, `cp860`, `cp1252`

Latin-2, Central European

`iso-8859-2`, `cp852`, `cp1250`

Latin-3, South European (Esperanto, Maltese)

`iso-8859-3`

Latin-4, North European (Baltics)

`iso-8859-4`

Cyrillics

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

Arabic

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

Greek

`iso-8859-7`, `cp737`, `cp1253`

Hebrew

`iso-8859-8`, `cp862`, `cp1255`

Turkish
: `iso-8859-9`, `cp857`, `cp1254`

Nordic
: `iso-8859-10`, `cp865`, `cp861` (Icelandic)

Thai
: `iso-8859-11`, `cp874`

Baltic
: `iso-8859-13`, `cp775`, `cp1257`

Celtic
: `iso-8859-14`

Latin-9 (sometimes called Latin0)
: `iso-8859-15`

Latin-10
: `iso-8859-16`

A.6 Exit status

The exit status is 0 if at least one *ID* matched *REGEXP*, 1 if no *ID* matched *REGEXP*, and 2 if an error occurred.

A.7 Caveats

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

A.8 See also

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

B e-TEX: Breaking Paragraphs into Lines

This is an excerpt from the e-TEX manual [7], Sec. 3.8, »Breaking Paragraphs into Lines« that describes the `\lastlinefit` algorithm. Package typog warps `\lastlinefit` in environment `lastlinefitpar`, which was introduced in Sec. 3.10.5 on p. 40.

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

> Traditional typesetting with lead type used to adjust (stretch or shrink) the interword spaces in the last line of a paragraph by the same amount as those in the preceding line. With TEX the last line is, however, usually typeset at its natural width due to infinitely stretchable `\parfillskip` glue. e-TEX allows interpolation between these two extremes by specifying a suitable value for `\lastlinefit`. For a value of 0 or less, e-TEX behaves as TEX, values from 1 to 1000 indicate a glue adjustment fraction f times 1000, values above 1000 are interpreted as $f = 1$.
>
> The new algorithm is used only if
>
> 1. `\lastlinefit` is positive;
> 2. `\parfillskip` has infinite stretchability; and
> 3. the stretchability of `\leftskip` plus `\rightskip` is finite.[43]
>
> Thus the last line of a paragraph would normally be typeset at its natural width and the stretchability of `\parfillskip` glue would be used to achieve the desired line width. The algorithm proceeds as usual, considering all possible sequences of feasible break points and accumulating demerits for the stretching or shrinking of lines as well as for visually incompatible lines. When a candidate for the last line has been reached, the following conditions are tested:
>
> 5. the previous line was not »infinitely bad« and was stretched with positive finite stretchability or was shrunk with positive shrinkability;
> 6. the last line has infinite stretchability entirely due to `\parfillskip` glue;
> 7. if the previous line was stretched or shrunk the last line has positive finite stretchability or shrinkability.
>
> If all three conditions are satisfied, a glue adjustment factor of f times that of the preceding line will be applied to the relevant stretch or shrink components of all glue nodes in the last line, and the corresponding demerits are computed. (The last line will, however, not be stretched beyond the desired line width.)
>
> When all possible candidates for the last line of the paragraph have been examined, the one having fewest accumulated demerits is chosen. If e-TEX's modified algorithm was applied to that last line, the actual stretching or shrinking is achieved by suitably modifying the `\parfillskip` glue node.

43 As usual for parameters influencing TEX's line-breaking algorithm, the values current at the end of the (partial) paragraph are used.

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

Section 3.8 of the e-TEX manual closes with a description of the generalizations of `\clubpenalties`, `\displaywidowpenalties`, `\interlinepenalties`, and `\widowpenalties`. See also Sec. 3.14, »Vertically Partially-Tied Paragraphs«, p. 48 in this manual.

C Package Code

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

```
%<*package>
\NeedsTeXFormat{LaTeX2e}[2005/12/01]
\ProvidesPackage{typog}
                [2025/05/18  v0.5  TypoGraphic extensions]

\RequirePackage{etoolbox}
\RequirePackage{everyhook}
\RequirePackage{xkeyval}

```

Declarations of Lengths, Skips, etc.

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

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

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

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

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

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

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

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

```

`\typog@typeout` Our debug information printer.

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

```

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

```
\newcounter{typog@@iteration}

```

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

```
\ExplSyntaxOn
\let\typog@trim@spaces=\tl_trim_spaces:o
\ExplSyntaxOff

```

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

```
\newcommand{\typog@register@pdfsubstitute}[1]{%
  \AtBeginDocument{%
    \ifdefined\pdfstringdefDisableCommands
      \@ifpackageloaded{pdfcomment}
                       {\PackageWarningNoLine{typog}
                                             {package pdfcomment loaded -
 -\space
                                              typog will not touch PDF in-
  terface}}
                       {\pdfstringdefDisableCommands{#1}}%
    \fi}}
```

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

ftypog@microtype@preloaded

```
\newif\iftypog@microtype@preloaded
```

equire@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.

```
\ifdefined\MT@MT
  \typog@typeout{package microtype preloaded}%
  \typog@microtype@preloadedtrue
  \def\typog@require@preloaded@microtype{\relax}
\else
  \typog@microtype@preloadedfalse
  \def\typog@require@preloaded@microtype
    {\PackageError{typog}%
                  {package microtype not (pre-)loaded}%
                  {package microtype must be loaded before pack-
  age typog}}
\fi
```

\iftypog@microtype@loaded

```
\newif\iftypog@microtype@loaded
```

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

```
\AtBeginDocument{
  \ifdefined\MT@MT
    \typog@typeout{package microtype loaded}%
    \typog@microtype@loadedtrue
    \def\typog@require@microtype{\relax}
```

```
  \else
    \typog@microtype@loadedfalse
    \def\typog@require@microtype
      {\PackageError{typog}%
                    {package microtype not loaded}%
                    {require package microtype before package typog}}%
  \fi
}
```

Our own state ...

nfig@mathitalicscorrection

```
\newmuskip\typog@config@mathitalicscorrection
```

Space around em-dash.

g@config@emdashspace (*dimen*)

```
\newlength{\typog@config@emdashspace}
```

Space around en-dash.

g@config@endashspace (*dimen*)

```
\newlength{\typog@config@endashspace}
```

Actual `\labelitem`⟨*N*⟩ corrections.

og@adjust@labelitemi (*dimen*)

```
\newdimen{\typog@adjust@labelitemi}
```

g@adjust@labelitemii (*dimen*)

```
\newdimen{\typog@adjust@labelitemii}
```

@adjust@labelitemiii (*dimen*)

```
\newdimen{\typog@adjust@labelitemiii}
```

g@adjust@labelitemiv (*dimen*)

```
\newdimen{\typog@adjust@labelitemiv}
```

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

lowercase@labelitemi (*dimen*)

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

owercase@labelitemii (*dimen*)

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

wercase@labelitemiii (*dimen*)

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

owercase@labelitemiv (*dimen*)

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

uppercase@labelitemi (*dimen*)

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

ppercase@labelitemii (*dimen*)

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

percase@labelitemiii (*dimen*)

```
\newdimen{\typog@adjust@uppercase@labelitemiii}
```

ppercase@labelitemiv (*dimen*)

```
\newdimen{\typog@adjust@uppercase@labelitemiv}
```

Other lengths ...

nfig@textitalicscorrection

```
\newlength{\typog@config@textitalicscorrection}
```

\typog@config@ligaturekern

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

\typog@config@lowerslash

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

og@config@raisecapitaldash

```
\newlength{\typog@config@raisecapitaldash}
```

fig@raisecapitalguillemets

```
\newlength{\typog@config@raisecapitalguillemets}
```

@config@raisecapitalhyphen

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

g@config@raisecapitaltimes

```
\newlength{\typog@config@raisecapitaltimes}
```

pog@config@raiseguillemets

```
\newlength{\typog@config@raiseguillemets}
```

pog@config@raisefiguredash

```
\newlength{\typog@config@raisefiguredash}
```

\typog@config@slashkern
```
\newlength{\typog@config@slashkern}
```
\typog@config@breakpenalty
```
\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.
```
\newlength{\typog@dim@unit}
\setlength{\typog@dim@unit}{.001em}
```
g@config@trackingttspacing
```
\newcommand*{\typog@config@trackingttspacing}{300, 90, 60}
```
\typog@default@shrink@i The default configuration for shrink values.
```
\newcommand*{\typog@default@shrink@i}{5}
```
\typog@default@shrink@ii
```
\newcommand*{\typog@default@shrink@ii}{10}
```
\typog@default@shrink@iii
```
\newcommand*{\typog@default@shrink@iii}{20}
```
\typog@shrink@i Configurable shrink values. Initialized from the typog@default@shrink@ set.
```
\newcommand*{\typog@shrink@i}{}
```
\typog@shrink@ii
```
\newcommand*{\typog@shrink@ii}{}
```
\typog@shrink@iii
```
\newcommand*{\typog@shrink@iii}{}
```
\typog@default@stretch@i The default configuration for stretch values.
```
\newcommand*{\typog@default@stretch@i}{5}
```
\typog@default@stretch@ii
```
\newcommand*{\typog@default@stretch@ii}{10}
```
\typog@default@stretch@iii
```
\newcommand*{\typog@default@stretch@iii}{20}
```
\typog@stretch@i Configurable stretch values. Initialized from the typog@default@stretch set.
```
\newcommand*{\typog@stretch@i}{}
```
\typog@stretch@ii
```
\newcommand*{\typog@stretch@ii}{}
```
\typog@stretch@iii
```
\newcommand*{\typog@stretch@iii}{}
```

C.1 Setup and Reconfiguration

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

```
\NewDocumentEnvironment{typogsetup}{m}
 {\def\typog@@arg{#1}%
  \ifx\typog@@arg\empty
    \typog@initialize@options
  \else
    \setkeys{typog}{#1}%
  \fi
  \ignorespaces}
 {\ignorespacesafterend}
```

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

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

```

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

```
\ExplSyntaxOn
\cs_generate_variant:Nn \seq_set_split:Nnn {Nne}
\cs_new:Npn \typog_get_nth_csname:cnn #1#2#3
  {
    \seq_set_split:Nne \l_tmpa_seq {,} {\cs:w typog@config@#2 \cs_end:}
    \cs_gset:cpn {#1} {\seq_item:Nn \l_tmpa_seq {#3}}
  }
\cs_new:Npn \typog_get_nth_dimen:nnn #1#2#3
  {
    \seq_set_split:Nne \l_tmpa_seq {,} {\cs:w typog@config@#2 \cs_end:}
    \dim_set:Nn {#1} {\seq_item:Nn \l_tmpa_seq {#3}}
  }
\NewDocumentCommand{\typoggetnth}{m m m}{
  \token_if_dim_register:NTF {#1}
    {
      \typog_get_nth_dimen:nnn {#1} {#2} {#3}
    }
    {
      \typog_get_nth_csname:cnn {#1} {#2} {#3}
    }
}
\ExplSyntaxOff

```

C.2 Information

\typog@round@dim@to@tenths

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

```

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

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

```

\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.

```
\NewDocumentCommand{\fontsizeinfo}{s m}
  {\global\expandafter\edef\csname typog@fontsize@#2\endcsname
     {\typog@round@dim@to@tenths{\fontdimen6\font}}%
   \global\expandafter\edef\csname typog@linespacing@#2\endcsname
     {\typog@round@dim@to@tenths{\baselineskip}}%
   \protected\expandafter\gdef\csname #2\endcsname
     {\@ifnextchar*{\typog@formatsizeinfo
                      {\csname typog@fontsize@#2\endcsname}%
                      {\csname typog@linespacing@#2\endcsname}%
                      {}% no unit
                      \ignorespaces % eat spaces after star
                      \@gobble}     % consume the star itself
                   {\typog@formatsizeinfo
                      {\csname typog@fontsize@#2\endcsname}%
                      {\csname typog@linespacing@#2\endcsname}%
                      {\,pt}% decorative unit 'pt'
  }}}

```

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

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

typog@inspect@count Counter to supply unique number and in turn ⟨*id*⟩ for those typoinspect environments that were not identified by the user.

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

```

`typoginspect` (*env.*)

```
\define@key[typog]{typoginspect}{tracingboxes}[\maxdimen]%
          {\def\typog@@typoginspect@tracingboxes{#1}}
\NewDocumentEnvironment{typoginspect}{O{} m}
 {\def\typog@@typoginspect@tracingboxes{\m@ne}%
  \setkeys[typog]{typoginspect}{#1}%
```

If the user does not supply an ⟨*id*⟩, we fall back to our own counter and construct a hopefully unique ⟨*id*⟩ from that.

```
  \edef\typog@@arg{#2}%
  \ifx\typog@@arg\empty
    \stepcounter{typog@inspect@count}%
    \edef\typog@@id{\typog@default@inspect@id@prefix
                    \arabic{typog@inspect@count}}%
  \else
    \edef\typog@@id{\typog@trim@spaces{\typog@@arg}}%
  \fi
  \typeout{<typog-inspect\space
           id="\typog@@id"\space
           job="\jobname"\space
           line="\the\inputlineno"\space
           page="\the\value{page}">}%
```

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

```
  \hbadness=\m@ne
  \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.

```
  \tracingnone
  \tracingpages=\@ne
  \tracingparagraphs=\@ne
  \showboxbreadth=\typog@@typoginspect@tracingboxes
  \showboxdepth=\typog@@typoginspect@tracingboxes}
 {\typeout{</typog-inspect>}%
  \ignorespacesafterend}
```

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

```
\NewDocumentEnvironment{typoginspectpar}{m}
 {\typoginspect{#1}}
 {\par\endtypoginspect}
```

C.3 Hyphenation

\typog@allowhyphenation Re-enable automatic hyphenation.

The same or almost the same implementation can be found in babel as macro \bbl@allowhyphens and hyphenat as macro \prw@zbreak.

```
\newcommand*{\typog@allowhyphenation}
  {\ifvmode
     \relax
   \else
     \nobreak
     \hskip\z@skip
   \fi}
```

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

```
\unless\ifdefined\allowhyphenation
  \let\allowhyphenation=\typog@allowhyphenation
\fi
```

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

```
\NewDocumentCommand{\breakpoint}{s}
  {\discretionary{}{}{}%
   \IfBooleanTF{#1}%
     {\ignorespaces}%
     {\typog@allowhyphenation}}
```

PDF-substitute definition

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

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

```
\NewDocumentEnvironment{hyphenmin}{o m}
  {\lefthyphenmin=\IfNoValueTF{#1}{#2}{#1}%
   \righthyphenmin=#2}
  {}
```

C.4 Disable/Break Ligatures

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

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

\nolig

```
\NewDocumentCommand{\nolig}{s o}
```

```
  {\dimen0=\IfNoValueTF{#2}
    {\typog@config@ligaturekern}
    {#2\typog@dim@unit}%
   \IfBooleanTF{#1}%
     {\kern\dimen0\ignorespaces}%
     {\discretionary{\typog@hyphen}{}{\kern\dimen0}%
      \typog@allowhyphenation
      \IfNoValueF{#2}{\ignorespaces}}}

```

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

```
\typog@register@pdfsubstitute{
  \RenewExpandableDocumentCommand{\nolig}{s o m}{%
    \ifx\typog@TYPOG#3\typog@TYPOG
      \relax
    \else
      \ifx\relax#3\relax
        \relax
      \else
        \typog@missing@argument
      \fi
    \fi}
}

```

C.5 Manual Italic Correction

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

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

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

```
\newcommand*{\typog@itcorr@text}[1]
  {\def\typog@@strength{#1}%
   \dimen0=\fontdimen1\font
   \ifdim\dimen0=\z@
     \typog@itcorr@text@unconditional{\typog@@strength}%
   \else
     \kern\typog@@strength\dimen0
   \fi}
```

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

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

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

```
\NewDocumentCommand{\itcorr}{s m}
  {\ifmmode
```

```
    \typog@itcorr@math{#2}%
  \else
    \IfBooleanTF{#1}%
      {\typog@itcorr@text{#2}}%
      {\typog@itcorr@text@unconditional{#2}}%
  \fi}
```

PDF-substitute definition

```
\typog@register@pdfsubstitute{
  \RenewExpandableDocumentCommand{\itcorr}{s m}{}
}
```

C.6 Apply Extra Kerning

Slash

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

```
\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'.

```
\NewDocumentCommand{\kernedslash}{s}
  {\hspace*{\typog@config@slashkern}%
   \raisebox{-\typog@config@lowerslash}{\typog@forwardslash}%
   \IfBooleanTF{#1}%
     {\hspace*{\typog@config@slashkern}\ignorespaces}%
     {\typog@breakpoint
      \typog@allowhyphenation
      \hspace*{\typog@config@slashkern}}}
```

PDF-substitute definition

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

Hyphen

`\kernedhyphen`

```
\NewDocumentCommand{\kernedhyphen}{s O{0} m m}
  {\ifmmode
     \mspace{\muexpr(#3 mu) * 18 / 1000}%
     \raisebox{#2\typog@dim@unit}{$\m@th\mathord{-}$}%
     \mspace{\muexpr(#4 mu) * 18 / 1000}%
   \else
     \def\typog@@auto{*}%
     \def\typog@@optarg{#2}%
     \hspace*{#3\typog@dim@unit}%
     \raisebox{\ifx\typog@@optarg\typog@@auto
```

```
                \typog@config@raisecapitalhyphen
              \else
                \typog@@optarg\typog@dim@unit
              \fi}{\typog@hyphen}%
    \hspace{#4\typog@dim@unit}%
    \IfBooleanT{#1}{\nobreak}%
  \fi}
```

PDF-substitute definition

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

One-argument shorthands.

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

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

PDF-substitute definition

```
\typog@register@pdfsubstitute{
  \RenewExpandableDocumentCommand{\leftkernedhyphen}{s o m}{-}
}

```

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

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

PDF-substitute definition

```
\typog@register@pdfsubstitute{
  \RenewExpandableDocumentCommand{\rightkernedhyphen}{s o m}{-}
}

```

En-Dash and Em-Dash

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

```
\newcommand*{\typog@wrap@endash}[3]
  {#1\hspace{\typog@config@endashspace}%
   #3%
   #2\hspace{\typog@config@endashspace}}
```

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

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

```
  {#1\hspace{\typog@config@emdashspace}%
   #3%
   #2\hspace{\typog@config@emdashspace}}

```

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

```
\NewDocumentCommand{\leftspacedendash}{s o}
  {\IfBooleanTF{#1}
    {\IfNoValueTF{#2}
      {\typog@wrap@endash{\nobreak}{\nobreak}
                         {\textendash}}
      {\typog@wrap@endash{\nobreak}{\nobreak}
                         {\raisebox{#2}{\textendash}}}}
    {\IfNoValueTF{#2}
      {\typog@wrap@endash{\relax}{\nobreak}
                         {\textendash}}
      {\typog@wrap@endash{\nobreak}{\nobreak}
                         {\raisebox{#2}{\textendash}}}}%
   \ignorespaces}
\let\leftspaceddash=\leftspacedendash
```

PDF-substitute definition

```
\typog@register@pdfsubstitute{
  \RenewExpandableDocumentCommand{\leftspacedendash}{s o m}{%
    \ifx\typog@TYPOG#3\typog@TYPOG
      \textendash
    \else
      \ifx\relax#3\relax
        \textendash
      \else
        \typog@missing@argument
      \fi
    \fi}
  \let\leftspaceddash=\leftspacedendash
}

```

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

```
\NewDocumentCommand{\rightspacedendash}{s o}
  {\IfBooleanTF{#1}
    {\IfNoValueTF{#2}
      {\typog@wrap@endash{\nobreak}{\nobreak}
                         {\textendash}}
      {\typog@wrap@endash{\nobreak}{\nobreak}
                         {\raisebox{#2}{\textendash}}}}
    {\IfNoValueTF{#2}
      {\typog@wrap@endash{\nobreak}{\relax}
                         {\textendash}}
      {\typog@wrap@endash{\nobreak}{\nobreak}
```

```
                          {\raisebox{#2}{\textendash}}}}%
  \ignorespaces}
\let\rightspaceddash=\rightspacedendash
\let\spacedendash=\rightspacedendash
\let\spaceddash=\rightspacedendash
```

PDF-substitute definition

```
\typog@register@pdfsubstitute{
  \RenewExpandableDocumentCommand{\rightspacedendash}{s o m}{%
    \ifx\typog@TYPOG#3\typog@TYPOG
      \space\textendash\space
    \else
      \ifx\relax#3\relax
        \space\textendash\space
      \else
        \typog@missing@argument
      \fi
    \fi
    \ignorespaces}
  \let\rightspaceddash=\rightspacedendash
  \let\spacedendash=\rightspacedendash
  \let\spaceddash=\rightspacedendash
}

```

`\swapendashskip`

```
\NewDocumentCommand{\swapendashskip}{m}
  {\skip0=\lastskip
   \unskip
   #1%
   \hskip\skip0}

```

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

The two `\mbox`es turn off automatic hyphenation of the adjacent words. This ensures ›compatibility‹ with the en-dash ␣--␣.

```
\NewDocumentCommand{\leftspacedemdash}{s o}
  {\mbox{}%
   \IfBooleanTF{#1}
    {\IfNoValueTF{#2}
      {\typog@wrap@emdash{\nobreak}{\nobreak}
                         {\textemdash}}
      {\typog@wrap@emdash{\nobreak}{\nobreak}
                         {\raisebox{#2}{\textemdash}}}}
    {\IfNoValueTF{#2}
      {\typog@wrap@emdash{\relax}{\nobreak}
                         {\textemdash}}
      {\typog@wrap@emdash{\nobreak}{\nobreak}
                         {\raisebox{#2}{\textemdash}}}}%
   \mbox{}%
   \ignorespaces}
```

PDF-substitute definition

```
\typog@register@pdfsubstitute{
 \RenewExpandableDocumentCommand{\leftspacedemdash}{s o m}{%
    \ifx\typog@TYPOG#3\typog@TYPOG
      \textemdash
    \else
      \ifx\relax#3\relax
        \textemdash
      \else
        \typog@missing@argument
      \fi
    \fi}
}

```

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

The two `\mbox`es turn off automatic hyphenation of the adjacent words. This ensures ›compatibility‹ with the em-dash ‚---‘.

```
\NewDocumentCommand{\rightspacedemdash}{s o}
  {\mbox{}%
   \IfBooleanTF{#1}
    {\IfNoValueTF{#2}
      {\typog@wrap@emdash{\nobreak}{\nobreak}
                         {\textemdash}}
      {\typog@wrap@emdash{\nobreak}{\nobreak}
                         {\raisebox{#2}{\textemdash}}}}
    {\IfNoValueTF{#2}
      {\typog@wrap@emdash{\nobreak}{\relax}
                         {\textemdash}}
      {\typog@wrap@emdash{\nobreak}{\nobreak}
                         {\raisebox{#2}{\textemdash}}}}%
   \mbox{}%
   \ignorespaces}
\let\spacedemdash=\rightspacedemdash
```

PDF-substitute definition

```
\typog@register@pdfsubstitute{
  \RenewExpandableDocumentCommand{\rightspacedemdash}{s o m}{%
    \ifx\typog@TYPOG#3\typog@TYPOG
      \textemdash
    \else
      \ifx\relax#3\relax
        \textemdash
      \else
        \typog@missing@argument
      \fi
    \fi
    \ignorespaces}
  \let\spacedemdash=\rightspacedemdash
}

```

C.7 Raise Selected Characters

\typog@breakpoint We want our own penalty for a line-break at a particular point. The predefined \allowbreak is too eager. A package-private, user-configurable penalty fits best.

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

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

```
\NewDocumentCommand{\capitalhyphen}{s}
  {\raisebox{\typog@config@raisecapitalhyphen}{\typog@hyphen}%
   \IfBooleanTF{#1}%
     {\ignorespaces}%
     {\typog@breakpoint\typog@allowhyphenation}}
```

The non-hyperref version's code is straightforward. The \pdfstringdefDisableCommands 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.

```
\typog@register@pdfsubstitute{
  \def\capitalhyphen#1{%
    \if*\detokenize{#1}%
      -\ignorespaces
    \else
      -#1%
    \fi}
}

```

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

```
\NewDocumentCommand{\capitalendash}{s}
  {\raisebox{\typog@config@raisecapitaldash}{\textendash}%
   \IfBooleanTF{#1}%
     {\ignorespaces}%
     {\typog@breakpoint\typog@allowhyphenation}}
\let\capitaldash=\capitalendash
```

PDF-substitute definition

```
\typog@register@pdfsubstitute{
  \def\capitalendash#1{%
    \if*\detokenize{#1}%
      \textendash\ignorespaces
    \else
      \textendash#1%
    \fi}
  \let\capitaldash=\capitalendash
}

```

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

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

PDF-substitute definition

```
\typog@register@pdfsubstitute{
  \def\capitalemdash#1{%
    \if*\detokenize{#1}%
      \textemdash\ignorespaces
    \else
      \textemdash#1%
    \fi}
}
```

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

```
\NewDocumentCommand{\leftspacedcapitalendash}{s}
  {\IfBooleanTF{#1}%
    {\typog@wrap@endash{\nobreak}{\nobreak}{\capitalendash*}}
    {\typog@wrap@endash{\relax}{\nobreak}{\capitalendash}}%
   \ignorespaces}
\let\leftspacedcapitaldash=\leftspacedcapitalendash
```

PDF-substitute definition

```
\typog@register@pdfsubstitute{
  \def\leftspacedcapitalendash#1{%
    \if*\detokenize{#1}%
      \textendash\ignorespaces
    \else
      \textendash#1%
    \fi}
  \let\leftspacedcapitaldash=\leftspacedcapitalendash
}
```

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

```
\NewDocumentCommand{\rightspacedcapitalendash}{s}
  {\IfBooleanTF{#1}%
    {\typog@wrap@endash{\nobreak}{\nobreak}{\capitalendash*}}
    {\typog@wrap@endash{\nobreak}{\relax}{\capitalendash}}%
   \ignorespaces}
\let\rightspacedcapitaldash=\rightspacedcapitalendash
\let\spacedcapitalendash=\rightspacedcapitalendash
\let\spacedcapitaldash=\rightspacedcapitalendash
```

PDF-substitute definition

```
\typog@register@pdfsubstitute{
  \def\rightspacedcapitalendash#1{%
    \if*\detokenize{#1}%
      \textendash\ignorespaces
    \else
      \textendash#1%
    \fi}
  \let\rightspacedcapitaldash=\rightspacedcapitalendash
  \let\spacedcapitalendash=\rightspacedcapitalendash
  \let\spacedcapitaldash=\rightspacedcapitalendash
}

```

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

```
\NewDocumentCommand{\leftspacedcapitalemdash}{s}
  {\IfBooleanTF{#1}%
    {\typog@wrap@emdash{\nobreak}{\nobreak}{\capitalemdash*}}
    {\typog@wrap@emdash{\relax}{\nobreak}{\capitalemdash}}%
   \ignorespaces}
```

PDF-substitute definition

```
\typog@register@pdfsubstitute{
  \def\leftspacedcapitalemdash#1{%
    \if*\detokenize{#1}%
      \textemdash\ignorespaces
    \else
      \textemdash#1%
    \fi}
}

```

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

```
\NewDocumentCommand{\rightspacedcapitalemdash}{s}
  {\IfBooleanTF{#1}%
    {\typog@wrap@emdash{\nobreak}{\nobreak}{\capitalemdash*}}
    {\typog@wrap@emdash{\nobreak}{\relax}{\capitalemdash}}%
   \ignorespaces}
\let\spacedcapitalemdash=\rightspacedcapitalemdash
```

PDF-substitute definition

```
\typog@register@pdfsubstitute{
  \def\rightspacedcapitalemdash#1{%
    \if*\detokenize{#1}%
      \textemdash\ignorespaces
    \else
      \textemdash#1%
    \fi}
  \let\spacedcapitalemdash=\rightspacedcapitalemdash
}
```

```
573
```

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

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

PDF-substitute definition

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

`\capitaltimes`

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

PDF-substitute definition

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

`\singleguillemetleft`

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

PDF-substitute definition

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

`\singleguillemetright`

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

PDF-substitute definition

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

`\doubleguillemetleft`

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

PDF-substitute definition

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

\doubleguillemetright

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

PDF-substitute definition

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

\Singleguillemetleft

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

PDF-substitute definition

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

\Singleguillemetright

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

PDF-substitute definition

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

\Doubleguillemetleft

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

PDF-substitute definition

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

\Doubleguillemetright

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

PDF-substitute definition

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

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

onfig@raiseinvertedmarks@i

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

nfig@raiseinvertedmarks@ii

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

fig@raiseinvertedmarks@iii

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

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

inverted@exclamationmark@i

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

og@inverted@questionmark@i

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

nverted@exclamationmark@ii

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

g@inverted@questionmark@ii

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

verted@exclamationmark@iii

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

@inverted@questionmark@iii

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

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

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

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

```
  {\letcs{\@tempa}{typog@inverted@#1@\romannumeral#2}%
   \letcs{\@tempb}{typog@config@raiseinvertedmarks@\romannumeral#2}%
   \ifdim\@tempb=\z@
     \setbox0=\hbox{\@tempa}%
     \dimen0=\dp0
   \else
     \dimen0=\@tempb
   \fi
   \raisebox{\dimen0}{\@tempa}}
```

The user-side macros are almost trivial now.

talinvertedexclamationmark

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

PDF-substitute definition

```
\typog@register@pdfsubstitute{
   \def\capitalinvertedexclamationmark#1{%
     \csname typog@inverted@exclamationmark@\romannumeral#1\endcsname
   }
}
```

apitalinvertedquestionmark

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

PDF-substitute definition

```
\typog@register@pdfsubstitute{
  \def\capitalinvertedquestionmark#1{%
    \csname typog@inverted@questionmark@\romannumeral#1\endcsname
  }
}
```

C.8 Vertically Adjusted Label Items

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

```
\newcommand*{\@typog@uppercase@adjust@labelitem}[1]
  {\@typog@maybe@patch@itemize
   \ifstrequal{#1}{*}
              {\setlength{\typog@adjust@labelitemi}
                         {\typog@adjust@uppercase@labelitemi}%
               \setlength{\typog@adjust@labelitemii}
                         {\typog@adjust@uppercase@labelitemii}%
               \setlength{\typog@adjust@labelitemiii}
                         {\typog@adjust@uppercase@labelitemiii}%
               \setlength{\typog@adjust@labelitemiv}
                         {\typog@adjust@uppercase@labelitemiv}}
              {\ifcase #1% 0
```

```
                  \relax  % outside of any itemize environment
                \or % 1
                  \setlength{\typog@adjust@labelitemi}
                            {\typog@adjust@uppercase@labelitemi}%
                \or % 2
                  \setlength{\typog@adjust@labelitemii}
                            {\typog@adjust@uppercase@labelitemii}%
                \or % 3
                  \setlength{\typog@adjust@labelitemiii}
                            {\typog@adjust@uppercase@labelitemiii}%
                \or % 4
                  \setlength{\typog@adjust@labelitemiv}
                            {\typog@adjust@uppercase@labelitemiv}%
                \else
                  \PackageError{typog}
                               {Itemize level out of range}
                               {Valid levels are 1, 2, 3, 4, and *}
                \fi}}

```

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

```
\newcommand*{\@typog@lowercase@adjust@labelitem}[1]
  {\@typog@maybe@patch@itemize
   \ifstrequal{#1}{*}
              {\setlength{\typog@adjust@labelitemi}
                         {\typog@adjust@lowercase@labelitemi}%
               \setlength{\typog@adjust@labelitemii}
                         {\typog@adjust@lowercase@labelitemii}%
               \setlength{\typog@adjust@labelitemiii}
                         {\typog@adjust@lowercase@labelitemiii}%
               \setlength{\typog@adjust@labelitemiv}
                         {\typog@adjust@lowercase@labelitemiv}}
              {\ifcase #1% 0
                  \relax  % outside of any itemize environment
               \or % 1
                  \setlength{\typog@adjust@labelitemi}
                            {\typog@adjust@lowercase@labelitemi}%
               \or % 2
                  \setlength{\typog@adjust@labelitemii}
                            {\typog@adjust@lowercase@labelitemii}%
               \or % 3
                  \setlength{\typog@adjust@labelitemiii}
                            {\typog@adjust@lowercase@labelitemiii}%
               \or % 4
                  \setlength{\typog@adjust@labelitemiv}
                            {\typog@adjust@lowercase@labelitemiv}%
               \else
                  \PackageError{typog}
                               {Itemize level out of range}
                               {Valid levels are 1, 2, 3, 4, and *}
               \fi}}
```

```
747
```

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

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

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

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

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

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

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

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

```
          {\@typog@noadjust@labelitem{\@itemdepth}}
          {\forcsvlist{\@typog@noadjust@labelitem}{#1}}%
  \ignorespaces}

```

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

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

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

`\@typog@itemize@patch` This is the additional code we inject into plain LaTeX's or package enumitem's `\itemize`.

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

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

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

Sneak in our own macro's name.

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

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

```
   \expandafter\def\csname\@itemitem\endcsname
       {\raisebox{\csname typog@adjust@labelitem\romannumeral\the\@itemdepth\endc
                 {\@typog@old@itemitem}}}

```

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 LaTeX's macro.

```
\newcommand*{\@typog@patch@itemize}
  {\ifdefined\enit@itemize@i
     \patchcmd{\enit@itemize@i}
              {\expandafter}
              {\@typog@itemize@patch\expandafter}
              {\typog@debug@typeout{patching enumitem \string\enit@itemize@i\spac
ceeded}}
              {\PackageError{typog}
                            {Patching enumitem macro \string\enit@itemize@i\space
                            {}}%
   \else
     \patchcmd{\itemize}
              {\expandafter}
              {\@typog@itemize@patch\expandafter}
              {\typog@debug@typeout{patching \string\itemize\space suc-
ceeded}}
              {\PackageError{typog}
                            {Patching plain LaTeX macro \string\itemize\space fai
                            {}}%
   \fi}

```

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

```
\newbool{@typog@itemize@has@been@patched}
\newcommand*{\@typog@maybe@patch@itemize}
  {\ifbool{@typog@itemize@has@been@patched}
          {\relax}
          {\@typog@patch@itemize
           \booltrue{@typog@itemize@has@been@patched}}}

```

Freestanding height-adjusted label items.

`\Adjustedlabelitemi`

```
\NewDocumentCommand{\Adjustedlabelitemi}{}
```

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

\adjustedlabelitemi

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

\Adjustedlabelitemii

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

\adjustedlabelitemii

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

\Adjustedlabelitemiii

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

\adjustedlabelitemiii

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

\Adjustedlabelitemiv

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

\adjustedlabelitemiv

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

And now for their PDFsubstitutes.

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

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

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

```
\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.

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

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

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

Load the ⟨*number-of-steps*⟩ and ensure that it is odd.

```
   \count0=#3\relax
   \unless\ifodd\count0
     \advance\count0 by 1%
   \fi
```

Set the iteration counter.

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

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

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

Box 1 is the accumulator for the raised samples.

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

Build the stairs.

```
   \loop
     \ifnum\thetypog@@iteration=\numexpr\count0 / 2\relax
       \dimen1=3\dimen0
     \else
       \dimen1=\dimen0
     \fi
     \dimen2=\dimexpr#2 * (\thetypog@@iteration - \count0 / 2)\re-
lax
     \setbox1=\hbox{\unhbox1\raisebox{\dimen2}{\kern\dimen1 #5\kern\dimen1}}%
     \addtocounter{typog@@iteration}{1}%
     \unless\ifnum\thetypog@@iteration>\count0
   \repeat
```

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

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

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

```
\NewDocumentCommand{\typogadjuststairs}{O{.5} m m m}
```

```
  {\begingroup
   \unless\ifdim #2>\z@
     \PackageError{typog}
                  {\string\typogadjuststairs\space non-positive step-
  size}
                  {step-size must be a positive dimension}%
   \fi
   \ifnum #3<1
     \PackageError{typog}
                  {\string\typogadjuststairs\space too few number-
  of-steps}
                  {number-of-steps must at least be 1}%
   \fi
   \ifblank{#4}
           {\PackageError{typog}
                         {sample must not be empty}
                         {supply either some uppercase or some low-
  ercase letters}}
           {}%
   \def\arraystretch{1}%
   \begin{tabular}{@{}c@{}}
     \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemi}  \\
     \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemii}  \\
     \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemiii}  \\
     \typogadjuststairsfor{#1}{#2}{#3}{#4}{\labelitemiv}
   \end{tabular}
   \endgroup}

```

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

```
\newcommand*{\typog@uppercase@adjusted@labelitems}
  {\hbox{\raisebox{\typog@adjust@uppercase@labelitemi}{\labelitemi}%
         \raisebox{\typog@adjust@uppercase@labelitemii}{\labelitemii}%
         \raisebox{\typog@adjust@uppercase@labelitemiii}{\labelitemiii}%
         \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.

```
\NewDocumentCommand{\typoguppercaseadjustcheck}{O{.5} m}
  {\setbox0=\hbox{#2}%
   \setbox1=\typog@uppercase@adjusted@labelitems
   \mbox{\rlap{\raisebox{\fpeval{#1}\ht0}
                        {\rule{\wd1}{\typog@hairline@width}}}%
         \box1}}

```

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

```
\newcommand*{\typog@lowercase@adjusted@labelitems}
```

```
  {\hbox{\raisebox{\typog@adjust@lowercase@labelitemi}{\labelitemi}%
         \raisebox{\typog@adjust@lowercase@labelitemii}{\labelitemii}%
         \raisebox{\typog@adjust@lowercase@labelitemiii}{\labelitemiii}%
         \raisebox{\typog@adjust@lowercase@labelitemiv}{\labelitemiv}}}
```

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

```
\NewDocumentCommand{\typoglowercaseadjustcheck}{O{.5} m}
  {\setbox0=\hbox{#2}%
   \setbox1=\typog@lowercase@adjusted@labelitems
   \mbox{\rlap{\raisebox{\fpeval{#1}\ht0}
                        {\rule{\wd1}{\typog@hairline@width}}}%
               \box1}}

```

C.9 Align Last Line of a Paragraph

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

`lastlineraggedleftpar` (*env.*)

```
\NewDocumentEnvironment{lastlineraggedleftpar}{}
  {\lastlinefit=0%
   \setlength{\leftskip}{\z@ \@plus 1fil}%
   \setlength{\rightskip}{-\leftskip}%
   \setlength{\parfillskip}{\leftskip}}
  {\par}
```

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

```
\let\lastlineflushrightpar=\lastlineraggedleftpar
\let\endlastlineflushrightpar=\endlastlineraggedleftpar

```

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

```
\NewDocumentEnvironment{lastlinecenteredpar}{}
  {\lastlinefit=0%
   \setlength{\leftskip}{\z@ \@plus .5fil}%
   \setlength{\rightskip}{-\leftskip}%
   \setlength{\parfillskip}{\z@ \@plus 1fil}}
  {\par}

```

C.10 Fill Last Line of a Paragraph

`shortenpar` (*env.*)

```
\NewDocumentEnvironment{shortenpar}{}
  {\advance\looseness by -1
   \ifnum\tracingparagraphs>0
     \typeout{@ looseness \the\looseness}%
   \fi}
```

```
  {\par}
```

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.

```
\NewDocumentEnvironment{prolongpar}{}
  {\finalhyphendemerits=100000001
   \advance\looseness by 1
   \ifnum\tracingparagraphs>0
     \typeout{@ looseness \the\looseness}%
   \fi}
  {\par}
```

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`.

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

ndentpar@nonzero@parindent

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

covernextindentpar (*env.*)

```
\NewDocumentEnvironment{covernextindentpar}{o}
  {\IfNoValueTF{#1}
     {\ifdim\parindent=\z@
        \dimen0=\dimexpr\linewidth - \typog@covernextindentpar@zero@parindent
      \else
        \dimen0=\dimexpr\linewidth - \typog@covernextindentpar@nonzero@parindent
      \fi}
     {\dimen0=\dimexpr\linewidth - (#1)}%
   \parfillskip=\dimen0 \@minus \dimen0
   \relax}
  {\par}
```

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

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

tlinepar@nonzero@parindent

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

openlastlinepar (*env.*) Compare with the suggestion in Ref. 34.

```
\NewDocumentEnvironment{openlastlinepar}{o}
  {\IfNoValueTF{#1}
     {\ifdim\parindent=\z@
        \skip0=\typog@openlastlinepar@zero@parindent
               \@plus 1fil
               \@minus \typog@openlastlinepar@zero@parindent
```

```
      \else
        \skip0=\typog@openlastlinepar@nonzero@parindent
               \@plus 1fil
               \@minus \typog@openlastlinepar@nonzero@parindent
      \fi}
     {\dimen0=\dimexpr#1\relax
      \skip0=\dimen0 \@plus 1fil \@minus \dimen0}
   \parfillskip=\skip0}
  {\par}

```

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

```
\NewDocumentEnvironment{lastlinefitpar}{O{1000}}
  {\lastlinefit=#1\relax}
  {\par}

```

C.11 Spacing

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

```
\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` $= 0$.

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

`\widespace`

```
\NewDocumentCommand{\widespace}{s}
  {\IfBooleanTF{#1}%
    {\dimen0=\widespacescale\fontdimen2\font}%
    {\ifdim\fontdimen7\font=\z@
       \dimen0=\widespacescale\fontdimen2\font
     \else
       \dimen0=\dimexpr\fontdimen2\font +
               \widespacestrength\fontdimen7\font
     \fi}%
   \hskip \glueexpr\dimen0
          \@plus \widespacescale\fontdimen3\font
          \@minus \widespacescale\fontdimen4\font
   \ignorespaces}

```

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

```
\newcommand*{\narrowspacestrength}{.5}
```

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

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

\narrowspace

```
\NewDocumentCommand{\narrowspace}{s}
  {\IfBooleanTF{#1}%
     {\dimen0=\narrowspacescale\fontdimen2\font}%
     {\ifdim\fontdimen7\font=\z@
        \dimen0=\narrowspacescale\fontdimen2\font
      \else
        \dimen0=\dimexpr\fontdimen2\font -
                \narrowspacestrength\fontdimen7\font
      \fi}%
   \hskip \glueexpr\dimen0
          \@plus \narrowspacescale\fontdimen3\font
          \@minus \narrowspacescale\fontdimen4\font
   \ignorespaces}

```

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

loosespacing (*env.*)

```
\NewDocumentEnvironment{loosespacing}{O{1}}
  {\dimen2=\fontdimen2\font
   \ifcase #1
     \spaceskip=\z@
   \or % 1         +5%
     \spaceskip=1.05\dimen2 \@plus .5\dimen2 \@minus .1\dimen2
   \or % 2         +10%
     \spaceskip=1.1\dimen2 \@plus .5\dimen2 \@minus .1\dimen2
   \or % 3         +20%
     \spaceskip=1.2\dimen2 \@plus .6\dimen2 \@minus .2\dimen2
   \else % >= 4    +30%
     \spaceskip=1.3\dimen2 \@plus .8\dimen2 \@minus .3\dimen2
   \fi
   \ignorespaces}
  {\ignorespacesafterend}

```

tightspacing (*env.*)

```
\NewDocumentEnvironment{tightspacing}{O{1}}
  {\dimen2=\fontdimen2\font
   \ifcase #1
     \spaceskip=\z@
   \or % 1          -1.25%
     \spaceskip=.9875\dimen2 \@plus .0125\dimen2 \@minus .5\dimen2
   \or % 2          -2.5%
     \spaceskip=.975\dimen2 \@plus .025\dimen2 \@minus .5\dimen2
   \or % 3          -5%
```

```
    \spaceskip=.95\dimen2 \@plus .05\dimen2 \@minus .5\dimen2
  \else % >= 4    -10%
    \spaceskip=.9\dimen2 \@plus .1\dimen2 \@minus .5\dimen2
  \fi
  \ignorespaces}
 {\ignorespacesafterend}

```

C.12 Microtype Front-End

Tracking

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

```
\NewDocumentEnvironment{setfonttracking}{m}
 {\edef\MT@letterspace@{#1}%
  \lsstyle
  \ignorespaces}
 {\ignorespacesafterend}

```

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.

```
\newcommand*{\typog@setup@font@expansion}
 {\SetExpansion
    [context = typog@shrink1,
     shrink = \typog@shrink@i,
     stretch = 0]%
    {encoding = {*}}%
    {}
  \SetExpansion
    [context = typog@shrink2,
     shrink = \typog@shrink@ii,
     stretch = 0]%
    {encoding = {*}}%
    {}
  \SetExpansion
    [context = typog@shrink3,
     shrink = \typog@shrink@iii,
     stretch = 0]%
    {encoding = {*}}%
    {}

  \SetExpansion
    [context = typog@stretch1,
     shrink = 0,
     stretch = \typog@stretch@i]%
    {encoding = {*}}%
    {}
```

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

icrotype@expansion@feature We cannot even parse the `\iftypog@microtype@preloaded` part further down unless the `\ifMT@expansion` conditional exists. So, we hoist this test in a macro of its own. It only gets called if package microtype already has been sourced.

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

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

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

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

```
\iftypog@microtype@preloaded
  \typog@test@microtype@expansion@feature
\else
  \def\typog@require@microtype@expansion
    {\PackageError{typog}%
                  {package microtype not (pre-)loaded, %
                   which is required for typog's font expansion}%
                  {require package microtype before package typog}}
\fi
```

`setfontshrink` (*env.*)

```
\NewDocumentEnvironment{setfontshrink}{O{1}}
  {\typog@require@microtype@expansion
   \ifcase#1% 0
     \relax
   \or % 1
     \microtypecontext{expansion=typog@shrink1}%
   \or % 2
     \microtypecontext{expansion=typog@shrink2}%
   \else % >= 3
     \microtypecontext{expansion=typog@shrink3}%
   \fi
   \ignorespaces}
  {\ignorespacesafterend}
```

`setfontstretch` (*env.*)

```
\NewDocumentEnvironment{setfontstretch}{O{1}}
  {\typog@require@microtype@expansion
   \ifcase#1% 0
     \relax
   \or % 1
     \microtypecontext{expansion=typog@stretch1}%
   \or % 2
     \microtypecontext{expansion=typog@stretch2}%
   \else % >= 3
     \microtypecontext{expansion=typog@stretch3}%
   \fi
   \ignorespaces}
  {\ignorespacesafterend}
```

`setfontexpand` (*env.*)

```
\NewDocumentEnvironment{setfontexpand}{O{1}}
  {\typog@require@microtype@expansion
   \ifcase#1% 0
     \relax
```

```
    \or % 1
      \microtypecontext{expansion=typog@expand1}%
    \or % 2
      \microtypecontext{expansion=typog@expand2}%
    \else % >= 3
      \microtypecontext{expansion=typog@expand3}%
    \fi
    \ignorespaces}
  {\ignorespacesafterend}

```

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.

```
\NewDocumentEnvironment{nofontexpansion}{}
  {\ifdefined\microtypesetup
     \microtypesetup{expansion=false}%
   \fi
   \ignorespaces}
  {\ignorespacesafterend}
```

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

```
\let\nofontexpand=\nofontexpansion
\let\endnofontexpand=\endnofontexpansion

```

Character Protrusion

nocharprotrusion (*env.*) See ›Implementation‹ comment of nofontexpansion.

```
\NewDocumentEnvironment{nocharprotrusion}{}
  {\ifdefined\microtypesetup
     \microtypesetup{protrusion=false}%
   \fi
   \ignorespaces}
  {\ignorespacesafterend}

```

C.13 Sloppy Paragraphs

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

```
\newcommand*{\typog@scaled@emergencystretch}[1]
  {\emergencystretch=\ifdim\linewidth=\z@
                       #1%
                     \else
                       \dimexpr (#1) * \linewidth / \textwidth
                     \fi}

```

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

```
\NewDocumentCommand{\slightlysloppy}{O{1}}
 {\ifcase #1% 0
    % \tolerance=200
    % \emergencystretch=\z@
    % \hfuzz=.1\p@
    % \vfuzz=\hfuzz
    \fussy
  \or % 1
    \pretolerance=165%
    \tolerance=330%
    \typog@scaled@emergencystretch{.375em}%
    \hfuzz=.15\p@
    \vfuzz=\hfuzz
  \or % 2
    \pretolerance=265%
    \tolerance=530%
    \typog@scaled@emergencystretch{.75em}%
    \hfuzz=.15\p@
    \vfuzz=\hfuzz
  \or % 3
    \pretolerance=435%
    \tolerance=870%
    \typog@scaled@emergencystretch{1.125em}%
    \hfuzz=.2\p@
    \vfuzz=\hfuzz
  \or % 4
    \pretolerance=705%
    \tolerance=1410%
    \typog@scaled@emergencystretch{1.5em}%
    \hfuzz=.3\p@
    \vfuzz=\hfuzz
  \or % 5
    \pretolerance=1155%
    \tolerance=2310%
    \typog@scaled@emergencystretch{1.875em}%
    \hfuzz=.35\p@
    \vfuzz=\hfuzz
  \or % 6
    \pretolerance=1880%
    \tolerance=3760%
    \typog@scaled@emergencystretch{2.25em}%
    \hfuzz=.4\p@
    \vfuzz=\hfuzz
  \or % 7
    \pretolerance=3065%
    \tolerance=6130%
    \typog@scaled@emergencystretch{2.625em}%
    \hfuzz=.45\p@
```

```
    \vfuzz=\hfuzz
  \else % >= 8
    % \tolerance=9999
    % \emergencystretch=3em
    % \hfuzz=.5\p@
    % \vfuzz=\hfuzz
    \sloppy
  \fi
  \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 ⟨*sloppiness*⟩ *and* the ratio of the actual `\linewidth` to the (maximum) `\textwidth`.
- The `\hfuzz` values are interpolated linearly with ⟨*sloppiness*⟩ 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);`

`slightlysloppypar` (*env.*)

```
\NewDocumentEnvironment{slightlysloppypar}{O{1}}
  {\par\slightlysloppy[#1]\ignorespaces}
  {\par}

```

C.14 Vertically Partially-Tied Paragraphs

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

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

```

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

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

```

\vtietop

```
\NewDocumentCommand{\vtietop}{O{3}}
  {\setcounter{typog@mean@penalty}
              {\typog@geometric@mean{\@M}{\clubpenalty}}%
   \typog@debug@typeout{vtietop: penalties \the\@M--\the\value{typog@mean@penalty
  -\the\clubpenalty}%
   \unless\ifnum\clubpenalty<\@M
     \PackageWarning{typog}{vtietop: clubpenalty=\the\clubpenalty\space>= 10000}%
   \fi
   \ifcase#1% 0
     \relax
   \or % 1
     \relax
   \or % 2
     \clubpenalties 3
         \@M
         \value{typog@mean@penalty}
         \clubpenalty
   \or % 3
     \clubpenalties 4
         \@M \@M
         \value{typog@mean@penalty}
         \clubpenalty
   \or % 4
     \clubpenalties 5
         \@M \@M \@M
         \value{typog@mean@penalty}
         \clubpenalty
   \or % 5
     \clubpenalties 6
         \@M \@M \@M \@M
         \value{typog@mean@penalty}
         \clubpenalty
   \or % 6
     \clubpenalties 7
         \@M \@M \@M \@M \@M
         \value{typog@mean@penalty}
         \clubpenalty
   \or % 7
     \clubpenalties 8
         \@M \@M \@M \@M \@M
         \value{typog@mean@penalty}
         \clubpenalty
   \or % 8
     \clubpenalties 9
         \@M \@M \@M \@M \@M
         \value{typog@mean@penalty}
         \clubpenalty
   \else % >= 9
     \clubpenalties 10
         \@M \@M \@M \@M \@M \@M \@M \@M
         \value{typog@mean@penalty}
```

```
          \clubpenalty
    \fi}
```

vtietoppar (*env.*)

```
\NewDocumentEnvironment{vtietoppar}{O{3}}
  {\vtietop[#1]}
  {\par
   \ignorespacesafterend}
```

\splicevtietop

```
\NewDocumentCommand{\splicevtietop}{O{3}}
  {\let\typog@old@item=\@item
   \def\@item[##1]{\typog@old@item[##1]\vtietop[#1]}%
   \ignorespaces}
```

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).

```
\ifdefined\SetEnumitemKey
  \SetEnumitemKey{vtietop}{first=\splicevtietop}
\fi
```

\vtiebot

```
\NewDocumentCommand{\vtiebot}{O{3}}
  {\setcounter{typog@mean@penalty}
              {\typog@geometric@mean{\@M}{\widowpenalty}}%
   \typog@debug@typeout{vtiebot: penalties \the\@M--\the\value{typog@mean@penalty
  -\the\widowpenalty}%
   \unless\ifnum\widowpenalty<\@M
     \PackageWarning{typog}{vtiebot: widowpenalty=\the\widowpenalty\space>= 10000
   \fi
   \ifcase#1% 0
     \relax
   \or % 1
     \relax
   \or % 2
     \widowpenalties 3
         \@M
         \value{typog@mean@penalty}
         \widowpenalty
   \or % 3
     \widowpenalties 4
         \@M \@M
         \value{typog@mean@penalty}
         \widowpenalty
   \or % 4
     \widowpenalties 5
         \@M \@M \@M
         \value{typog@mean@penalty}
```

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

vtiebotpar (*env.*)

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

\typog@vtiebotdisp

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

```
        \value{typog@mean@penalty}
        \displaywidowpenalty
  \or % 3
    \displaywidowpenalties 4
        \@M \@M
        \value{typog@mean@penalty}
        \displaywidowpenalty
  \or % 4
    \displaywidowpenalties 5
        \@M \@M \@M
        \value{typog@mean@penalty}
        \displaywidowpenalty
  \or % 5
    \displaywidowpenalties 6
        \@M \@M \@M \@M
        \value{typog@mean@penalty}
        \displaywidowpenalty
  \or % 6
    \displaywidowpenalties 7
        \@M \@M \@M \@M \@M
        \value{typog@mean@penalty}
        \displaywidowpenalty
  \or % 7
    \displaywidowpenalties 8
        \@M \@M \@M \@M \@M \@M
        \value{typog@mean@penalty}
        \displaywidowpenalty
  \or % 8
    \displaywidowpenalties 9
        \@M \@M \@M \@M \@M \@M \@M
        \value{typog@mean@penalty}
        \displaywidowpenalty
  \else % >= 9
    \displaywidowpenalties 10
        \@M \@M \@M \@M \@M \@M \@M \@M
        \value{typog@mean@penalty}
        \displaywidowpenalty
  \fi}

```

vtiebotdisp (*env.*)

```
\NewDocumentEnvironment{vtiebotdisp}{O{3}}
  {\typog@vtiebotdisp{#1}}
  {\ignorespacesafterend}

```

vtiebotdisptoppar (*env.*)

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

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

C.15 Breakable Displayed Equations

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

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

C.16 Setspace Front-End

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

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

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

```
1480 \newcommand*{\typog@setbaselineskip@relative@error}{.001}
```

\typog@setbaselineskip Given the ⟨*target-baselineskip*⟩ as argument iterate setting `\setstretch` until the error drops below our threshold.

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

Initialize our “emergency-stop” loop counter.

```
1484   \int_set:Nn \l_tmpa_int {1}
1485   \int_set:Nn \l_tmpb_int {\typog@setbaselineskip@iter@limit}
```

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

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

```
                {\string\setbaselineskip:\space
                  baselineskip\space not\space positive}
                {}
  }

  \dim_compare:nNnTF {\l_tmpa_dim} > {\c_zero_dim}
  {}
  {
    \PackageError{typog}
                {\string\setbaselineskip:\space target\space
                  baselineskip\space must\space be\space
                  positive}
                {}
  }

  \skip_if_eq:nnTF {\l_tmpa_dim} {\glueexpr #1}
  {}
  {
    \PackageWarning{typog}
                  {\string\setbaselineskip:\space argument\space
                    is\space a\space skip;\space
                    will\space ignore\space glue}
                  {}
  }

  \fp_set:Nn \l_tmpa_fp {\l_tmpa_dim / \baselineskip}
  \fp_until_do:nNnn {abs(\l_tmpa_dim / \baselineskip - 1)} <
                    {\typog@setbaselineskip@relative@error}
  {
    \setstretch{\fp_use:N \l_tmpa_fp}
    \fp_set:Nn \l_tmpa_fp
               {\l_tmpa_fp * \l_tmpa_dim / \baselineskip}

    \int_incr:N \l_tmpa_int
    \int_compare:nNnTF {\l_tmpa_int} > {\l_tmpb_int}
    {
      \PackageError{typog}
                  {\string\setbaselineskip:\space excessive\space
                    number\space of\space iterations:\space
                    \int_use:N \l_tmpa_int\space >\space
                    \int_use:N \l_tmpb_int}
                  {}
    }
    {}
  }

  \typog@debug@typeout{\string\setbaselineskip:\space
    final\space \string\setstretch\space argument:\space
    \fp_use:N \l_tmpa_fp}
  \typog@debug@typeout{\string\setbaselineskip:\space
    final\space baselineskip:\space \the\baselineskip}
}
```

```
1549
```

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

> ***Implementation Note***
>
> Viewed as a standalone macro `\setbaselineskip` does not need the decoration `\AfterPreamble`. However, all of its siblings, `\setbaselineskippercentage`, `\setleading`, and `\setleadingpercentage` then would behave differently as they are delayed to the end of the preamble, but `\setbaselineskip` immediately becomes effective. For example, the successive calls
>
> ```
> \setbaselineskippercentage{140}
> \setbaselineskip{12.5pt}
> ```
>
> in the preamble would set the baselineskip to 140% in the document. Therefore, `\setbaselineskip` is delayed too and the order of the calls thus preserved.

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

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

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

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

```
1561 \dim_new:N \typogfontsize
```

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

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

`\setbaselineskippercentage`

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

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

\setleading

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

\setleadingpercentage

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

```
}
\ExplSyntaxOff
```

C.17 Smooth Ragged

`\typog@repeat` As we shall have to repeat the line specifications for our paragraphs so often we introduce the two argument macro `\typog@repeat` that takes a ⟨*repeat-count*⟩ and a ⟨*body*⟩ that is repeated.

```
\ExplSyntaxOn
\cs_new_eq:NN \typog@repeat \prg_replicate:nn
```

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

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

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

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

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

```
\define@key[typog]{smoothraggedrightshapetriplet}{leftskip}%
           {\def\typog@@triplet@leftskip{#1}}
\define@key[typog]{smoothraggedrightshapetriplet}{parindent}%
           {\def\typog@@triplet@parindent{#1}}
\NewDocumentEnvironment{smoothraggedrightshapetriplet}{O{} m m m}
  {\def\typog@@triplet@leftskip{\z@}%
   \def\typog@@triplet@parindent{\z@}%
   \setkeys*[typog]{smoothraggedrightshapetriplet}{#1}%
   \skip0=\typog@@triplet@leftskip\relax
   \skip1=#2\relax
   \skip2=#3\relax
   \skip3=#4\relax
   \typog@debug@typeout{smoothraggedrightshapetriplet: skip0=\the\skip0}%
   \typog@debug@typeout{smoothraggedrightshapetriplet: skip1=\the\skip1}%
   \typog@debug@typeout{smoothraggedrightshapetriplet: skip2=\the\skip2}%
   \typog@debug@typeout{smoothraggedrightshapetriplet: skip3=\the\skip3}%
   \unless\ifnum\typog@mod{\typog@triplet@max@lines}{3}=0
     \PackageError{typog}
                  {Line number of triplet generator\space
                    (\typog@triplet@max@lines) not divisible by 3}
                  {}
   \fi
   \edef\typog@@triplet@linespecs{%
     \glueexpr \skip0 + \typog@@triplet@parindent\relax
           \glueexpr \skip1 - \typog@@triplet@parindent\relax
```

```
                    \skip0 \skip2  \skip0 \skip3
    \typog@repeat{\numexpr\typog@triplet@max@lines / 3 - 1}
                  {\skip0 \skip1  \skip0 \skip2  \skip0 \skip3}}
   \parshape=\typog@triplet@max@lines\typog@@triplet@linespecs\relax}
  {\par}

```

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

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

```

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

```
\define@key[typog]{smoothraggedrightshapequintuplet}{leftskip}
           {\def\typog@@quintuplet@leftskip{#1}}
\define@key[typog]{smoothraggedrightshapequintuplet}{parindent}
           {\def\typog@@quintuplet@parindent{#1}}
\NewDocumentEnvironment{smoothraggedrightshapequintuplet}{O{} m m m m m}
  {\def\typog@@quintuplet@leftskip{\z@}%
   \def\typog@@quintuplet@parindent{\z@}%
   \setkeys*[typog]{smoothraggedrightshapequintuplet}{#1}%
   \skip0=\typog@@quintuplet@leftskip\relax
   \skip1=#2\relax
   \skip2=#3\relax
   \skip3=#4\relax
   \skip4=#5\relax
   \skip5=#6\relax
   \typog@debug@typeout{smoothraggedrightshapequintuplet: skip0=\the\skip0}%
   \typog@debug@typeout{smoothraggedrightshapequintuplet: skip1=\the\skip1}%
   \typog@debug@typeout{smoothraggedrightshapequintuplet: skip2=\the\skip2}%
   \typog@debug@typeout{smoothraggedrightshapequintuplet: skip3=\the\skip3}%
   \typog@debug@typeout{smoothraggedrightshapequintuplet: skip4=\the\skip4}%
   \typog@debug@typeout{smoothraggedrightshapequintuplet: skip5=\the\skip5}%
   \unless\ifnum\typog@mod{\typog@quintuplet@max@lines}{5}=0
     \PackageError{typog}
                  {Line number of quintuplet generator\space
                    (\typog@quintuplet@max@lines) not divisible by 5}
                  {}
   \fi
   \edef\typog@@quintuplet@linespecs{%
     \glueexpr \skip0 + \typog@@quintuplet@parindent\relax
            \glueexpr \skip1 - \typog@@quintuplet@parindent\relax
                      \skip0 \skip2  \skip0 \skip3
                      \skip0 \skip4  \skip0 \skip5
     \typog@repeat{\numexpr\typog@quintuplet@max@lines / 5 - 1}
                  {\skip0 \skip1  \skip0 \skip2  \skip0 \skip3
                   \skip0 \skip4  \skip0 \skip5}}
   \parshape=\typog@quintuplet@max@lines\typog@@quintuplet@linespecs\relax}
  {\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.

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

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

```
\define@key[typog]{smoothraggedrightshapeseptuplet}{leftskip}%
           {\def\typog@@septuplet@leftskip{#1}}
\define@key[typog]{smoothraggedrightshapeseptuplet}{parindent}%
           {\def\typog@@septuplet@parindent{#1}}
\NewDocumentEnvironment{smoothraggedrightshapeseptuplet}{O{} m m m m m m m}
  {\def\typog@@septuplet@leftskip{\z@}%
   \def\typog@@septuplet@parindent{\z@}%
   \setkeys*[typog]{smoothraggedrightshapeseptuplet}{#1}%
   \skip0=\typog@@septuplet@leftskip\relax
   \skip1=#2\relax
   \skip2=#3\relax
   \skip3=#4\relax
   \skip4=#5\relax
   \skip5=#6\relax
   \skip6=#7\relax
   \skip7=#8\relax
   \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip0=\the\skip0}%
   \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip1=\the\skip1}%
   \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip2=\the\skip2}%
   \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip3=\the\skip3}%
   \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip4=\the\skip4}%
   \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip5=\the\skip5}%
   \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip6=\the\skip6}%
   \typog@debug@typeout{smoothraggedrightshapeseptuplet: skip7=\the\skip7}%
   \unless\ifnum\typog@mod{\typog@septuplet@max@lines}{7}=0
     \PackageError{typog}
                  {Line number of septuplet generator\space
                    (\typog@septuplet@max@lines) not divisible by 7}
                  {}
   \fi
   \edef\typog@@septuplet@linespecs{%
     \glueexpr \skip0 + \typog@@septuplet@parindent\relax
           \glueexpr \skip1 - \typog@@septuplet@parindent\relax
                    \skip0 \skip2  \skip0 \skip3  \skip0 \skip4
                    \skip0 \skip5  \skip0 \skip6  \skip0 \skip7
     \typog@repeat{\numexpr\typog@septuplet@max@lines / 7 - 1}
                  {\skip0 \skip1  \skip0 \skip2  \skip0 \skip3  \skip0 \skip4
                   \skip0 \skip5  \skip0 \skip6  \skip0 \skip7}}
   \parshape=\typog@septuplet@max@lines\typog@@septuplet@linespecs\relax}
  {\par}
```

moothraggedrightfuzzfactor

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

smoothraggedrightgenerator

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

\smoothraggedrightleftskip

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

smoothraggedrightparindent

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

\smoothraggedrightragwidth

```
\newlength{\smoothraggedrightragwidth}
\setlength{\smoothraggedrightragwidth}{2em}
```

\typog@fuzzwidth (*dimen*)

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

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

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

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

```
\ExplSyntaxOn
\cs_new:Npn \typog@generator@index:n #1 {
  \str_case:nnTF {#1}
      {
        {triplet} {0}
        {quintuplet} {1}
        {septuplet} {2}
      }
      {}
      {-1}
}
\cs_generate_variant:Nn \typog@generator@index:n {e}
\let\typog@generator@index=\typog@generator@index:e
\ExplSyntaxOff

\NewDocumentEnvironment{smoothraggedrightpar}{O{}}
  {\edef\typog@@linewidth{\linewidth}%
   \setkeys[typog]{smoothraggedrightpar}{#1}%
   \edef\typog@@generatorchoice{\typog@generator@index{\smoothraggedrightgenerato
```

Obey to the indentation prescribed by any list environment.

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

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

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

Now for the generator-specific code ...

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

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

```
    \typog@fuzzwidth=.25\smoothraggedrightragwidth
    \typog@debug@typeout{smoothraggedright: generator=triplet, typog@fuzzwidth=\the\typog@fuzzwidth}%
    \smoothraggedrightshapetriplet[leftskip=\typog@@smoothraggedrightleftskip,
                                   parindent=\glueexpr\smoothraggedrightparindent,
                                   #1]%
        {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth
                   + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
        {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth}% (3)
        {\glueexpr (\typog@@linewidth * 2 - \smoothraggedrightragwidth) / 2
                   + \glueexpr \z@ \@plus \typog@fuzzwidth \@minus \typog@fuzzwidth\relax}% (2)
  \or
```

`generator=quintuplet`.

```
    \typog@fuzzwidth=.125\smoothraggedrightragwidth
    \typog@debug@typeout{smoothraggedright: generator=quintuplet, typog@fuzzwidth=\the\typog@fuzzwidth}%
    \smoothraggedrightshapequintuplet[leftskip=\typog@@smoothraggedrightleftskip
                                      parindent=\glueexpr\smoothraggedrightparindent,
                                      #1]%
        {\glueexpr (\typog@@linewidth * 4 - \smoothraggedrightragwidth * 3) / 4
                   + \glueexpr \z@ \@plus \typog@fuzzwidth \@minus \typog@fuzzwidth\relax}% (2)
        {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth\relax}% (5)
        {\glueexpr (\typog@@linewidth * 2 - \smoothraggedrightragwidth) / 2
                   + \glueexpr \z@ \@plus \typog@fuzzwidth \@minus \typog@fuzzwidth\relax}% (3)
        {\glueexpr (\typog@@linewidth * 4 - \smoothraggedrightragwidth) / 4
                   + \glueexpr \z@ \@plus \typog@fuzzwidth \@minus \typog@fuzzwidth\relax}% (4)
        {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth
                   + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
  \or
```

`generator=septuplet`.

Permutation 3 – 6 – 1 – 5 – 2 – 7 – 4 looks ›random‹ enough for our purposes.

```
    \typog@fuzzwidth=.08333\smoothraggedrightragwidth
    \typog@debug@typeout{smoothraggedright: generator=septuplet, typog@fuzzwidth=\the\typog@fuzzwidth}%
    \smoothraggedrightshapeseptuplet[leftskip=\typog@@smoothraggedrightleftskip,
```

```
                                        parindent=\glueexpr\smoothraggedrightparind
   indent,
                                        #1]%
        {\glueexpr (\typog@@linewidth * 3 - \smoothraggedrightrag-
   width * 2) / 3
                    + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
   nus \typog@fuzzwidth\relax}% (3)
        {\glueexpr (\typog@@linewidth * 6 - \smoothraggedrightrag-
   width) / 6
                    + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
   nus \typog@fuzzwidth\relax}% (6)
        {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth +
                    + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
        {\glueexpr (\typog@@linewidth * 3 - \smoothraggedrightrag-
   width) / 3
                    + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
   nus \typog@fuzzwidth\relax}% (5)
        {\glueexpr (\typog@@linewidth * 6 - \smoothraggedrightrag-
   width * 5) / 6
                    + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
   nus \typog@fuzzwidth\relax}% (2)
        {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth\relax}% (7)
        {\glueexpr (\typog@@linewidth * 2 - \smoothraggedrightrag-
   width) / 2
                    + \glueexpr \z@ \@plus \typog@fuzzwidth \@mi-
   nus \typog@fuzzwidth\relax}% (4)
   \else
     \PackageError{typog}
                  {smoothraggedright: unknown generator name (\smoothraggedright-
   generator)}
                  {valid generator names are triplet, quintuplet, and sep-
   tuplet}
   \fi}
  {\ifcase\typog@@generatorchoice
     \endsmoothraggedrightshapetriplet
   \or
     \endsmoothraggedrightshapequintuplet
   \or
     \endsmoothraggedrightshapeseptuplet
   \fi}

```

smoothraggedright (*env.*)

```
\NewDocumentEnvironment{smoothraggedright}{O{}}
  {\PushPostHook{par}{\hskip-\parindent\smoothraggedrightpar[#1]\relax}}
  {\par\PopPostHook{par}}

```

Change History

v0.1
General: Initial version. i
v0.2
\narrowspace: New macro. 102
\widespace: Add fallback if \fontdimen7 is zero. Extend with a starred version. 101
v0.3
hyphenmin: New environment. 78
\resetbaselineskip: New macro. 115
\setbaselineskip: New macro. 115
\setbaselineskippercentage: New macro. 115
\setleading: New macro. 116
\setleadingpercentage: New macro. 116
\typogfontsize: New dimen. 115
v0.4
\lowercaseadjustlabelitems: New macro. 93
\noadjustlabelitems: New macro. 93
\typogadjuststairs: New macro. 97
\typoggetnth: New macro. 75
\typoglowercaseadjustcheck: New macro. 99
\typoguppercaseadjustcheck: New macro. 98
\uppercaseadjustlabelitems: New macro. 93
v0.4a
smoothraggedrightshapeseptuplet: Add missing backslash to \typog@@septuplet@parindent in local macro \typog@@septuplet@linespecs which rendered the septuplet generator unusable. 119
\typog@register@pdfsubstitute: Using \pdfstringdefDisableCommands breaks package pdfcomment. So we just refrain from using the macro if pdfcomment was loaded. 71
v0.5
General: Rewrite parser for package option lowercaselabelitemadjustments to accept a star as a placeholder. 74
Rewrite parser for package option uppercaselabelitemadjustments to accept a star as a placeholder. 74
\Adjustedlabelitemi: New macro. 95
\adjustedlabelitemi: New macro. 96
\Adjustedlabelitemii: New macro. 96
\adjustedlabelitemii: New macro. 96
\Adjustedlabelitemiii: New macro. 96
\adjustedlabelitemiii: New macro. 96
\Adjustedlabelitemiv: New macro. 96
\adjustedlabelitemiv: New macro. 96
\capitalinvertedexclamationmark: New macro. 91
\capitalinvertedquestionmark: New macro. 91

References

[1] Abrahams, Paul W., Kathryn A. Hargreaves, and Karl Berry. *TEX for the Impatient*. 2020, https://tug.ctan.org/info/impatient/book.pdf.

[2] American Mathematical Society and the LaTeX3 Project Team. *Package amsmath*. 2020, https://ctan.org/pkg/amsmath.

[3] Arseneau, Donald. *Package cite*. 2015, https://ctan.org/pkg/cite.

[4] *The Chicago Manual of Style*, 17th ed. The University of Chicago Press, Chicago/IL, 2017.

[5] Bezos, Javier. *Package enumitem*. 2019, https://ctan.org/pkg/enumitem.

[6] Bezos, Javier. *Package babel*. 2021, https://ctan.org/pkg/babel. The original author of package babel was J. L. Braams.

[7] Breitenlohner, Peter and the NTS Team. *e-TEX*. 1998, https://mirrors.ctan.org/systems/doc/etex/etex_man.pdf.

[8] Burnol, Jean-François. *Package mathastext*. 2023, https://ctan.org/pkg/mathastext.

[9] Carlisle, David. *Russian Paragraph Shapes*. Baskerville, 6(1), 13–15, 1996, http://uk-tug-archive.tug.org/wp-installed-content/uploads/2008/12/61.pdf.

[10] Carlisle, David. *What do different* \fontdimen<num> *mean*. 2013–1–2, https://tex.stackexchange.com/questions/88991/what-do-different-fontdimennum-mean.

[11] Cubitt, Toby. *Package cleveref*. 2018, https://ctan.org/pkg/cleveref.

[12] Eijkhout, Victor. *TEX By Topic, A Texnician's Reference*. 2007, https://www.eijkhout.net/tex/tex-by-topic.html.

[13] Forssman, Friedrich, and Ralf de Jong. *Detailtypographie*. 4. Aufl. Verlag Hermann Schmitz, Mainz, 2008.

[14] Høgholm, Morten, Lars Madsen, and the LaTeX3 Project Team. *Package mathtools*. 2020, https://ctan.org/pkg/mathtools.

[15] Khirevich, Siarhei. *Tips on Writing a Thesis in LaTeX*. 2013, https://www.khirevich.com/latex/microtype.

[16] Kleber, Josef. *Package pdfcomment*. 2018, https://ctan.org/pkg/pdfcomment.

[17] Knuth, Donald Ervin. *The TEXbook*. Addison Wesley, Reading/MA, 1986.

[18] The LaTeX Project. *Package l3experimental*. 2024, https://ctan.org/pkg/l3experimental.

[19] McPherson, Kent. *Package layout*. 2014, https://ctan.org/pkg/layout. The package was converted to LaTeX 2ε by J. L. Braams and modified by H. Umeki.

[20] Middendorp, Jan. *Shaping Text*. bis publishers, Amsterdam, 2014.

[21] Mittelbach, Frank. *Managing forlorn paragraph lines (a. k. a. widows and orphans) in LaTeX*. TUGboat, 39(3), 246–251, 2018, https://tug.org/TUGboat/tb39-3/tb123mitt-widows.pdf.

[22] Mittelbach, Frank. *Package widows-and-orphans*. 2020, https://ctan.org/pkg/widows-and-orphans.

[23] Niederberger, Clemens. *Package tasks*. 2022, https://github.com/cgnieder/tasks.

[24] Rahtz, Sebastian, and Frank Mittelbach. *Package hyperref*. 2020, https://ctan.org/pkg/hyperref. The package is maintained by the LaTeX3 Project Team.

[25] Schlicht, Robert. *Package microtype*. 2020, https://ctan.org/pkg/microtype.

[26] Schröder, Martin. *Package ragged2e*. 2019, https://ctan.org/pkg/ragged2e.

[27] Solomon, David. *Output Routines: Examples and Techniques. Part I: Introduction and Examples*. TUGboat, 11(1), 69–85, 1990, https://www.tug.org/TUGboat/Articles/tb11-1/tb27salomon.pdf.

[28] Strizver, Ilene. *Type rules!: the designer's guide to professional typography*, 4th ed. John Wiley & Sons, Hoboken/NJ, 2014.

[29] Tobin, Geoffrey, and Robin Fairbairns. *Package setspace*. 2011, https://ctan.org/pkg/setspace.

[30] Umeki, Hideo. *Package geometry*. 2020, https://ctan.org/pkg/geometry.

[31] Wermuth, Udo. *Tracing paragraphs*. TUGboat, 37(3), 358–373, 2016, https://tug.org/TUGboat/tb37-3/tb117wermuth.pdf.

[32] Wermuth, Udo. *The optimal value for* `\emergencystretch`. TUGboat, 38(1), 65–86, 2017, https://tug.org/TUGboat/tb38-1/tb118wermuth.pdf.

[33] Wermuth, Udo. *A note on* `\linepenalty`. TUGboat, 38(3), 400–414, 2017, https://tug.org/TUGboat/tb38-3/tb120wermuth.pdf.

[34] Wermuth, Udo. *Experiments with* `\parfillskip`. TUGboat, 39(3), 276–303, 2018, https://tug.org/TUGboat/tb39-3/tb123wermuth-parfillskip.pdf.

[35] Wermuth, Udo. *An attempt at ragged-right typesetting*. TUGboat, 41(1), 73–94, 2020, https://tug.org/TUGboat/tb41-1/tb127wermuth-ragged.pdf.

[36] Wermuth, Udo. Personal communication. August 2, 2022.

[37] Wermuth, Udo. *Vertical alignments in plain TEX*. TUGboat, 44(3), 427–440, 2023, https://tug.org/TUGboat/tb44-3/tb138wermuth-valign.pdf.

[38] Wilson, Peter. *Package hyphenat*. 2004, https://ctan.org/pkg/hyphenat. The package is maintained by W. Robertson.

[39] Wilson, Peter. *Glisterings*. TUGboat, 28(2), 229–232, 2007, https://tug.org/TUGboat/tb28-2/tb89glister.pdf.

[40] Wilson, Peter. *Package needspace*. 2010, https://ctan.org/pkg/needspace. The package is maintained by W. Robertson.

Index

Numbers written in italics refer to definitions; numbers in regular style refer to pages where an entry is used. References to code lines are prefixed with ›ℓ‹. Again italics refer to definitions and regular style to lines where the code is used.

K

L

M

N

O

P

Q

R

S