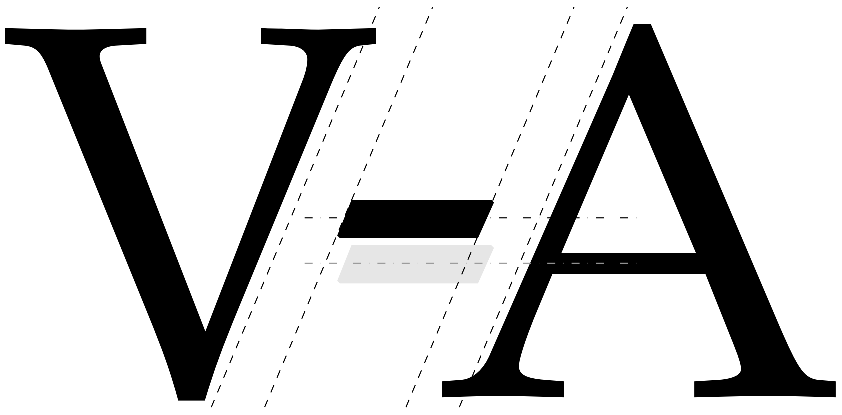
TypoG – Typographic Fine-Tuning

Ch. L. Spiel[*]

v0.3 2024/05/07

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

Table of Contents continued on next page.

List of Tables

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

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

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

Section 3.2 expands the hyphenation facilities of LaTeX.

Sections 3.3 to 3.6 deal with vertically positioning glyphs in a more pleasant way.

Sections 3.7 and 3.8 discuss dearly missed macros for better control of the last line of a paragraph.

Section 3.9 covers the manipulation of the length of a paragraph.

Section 3.10 expounds on the microtype front-end: font tracking (3.10.1), font expansion (3.10.2), and character protrusion (3.10.3).

In Sec. 3.11 we address some shortcomings of spacing control with a replacement for the macro `\sloppy` and the related environment `sloppypar`.

Section 3.12 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. 3.13.

Section 3.14 introduces the setspace front-end.

In the last part, Sec. 3.15, we introduce a novel way of generating ragged paragraphs, which still is experimental.

1.2 Prerequisites

Package typog requires ε-TeX; it relies on the LaTeX3 interface. Parts of it are based on package microtype. However, if the respective functionality is not used, typog can be used without microtype. The same holds true for the setspace front-end.

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

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. 3.10.

\usepackage[...]{setspace} % Only required for macros in Sec. 3.14.

\usepackage[⟨OPTION⟩...]{typog}
```

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

The package ⟨*OPTIONs*⟩ serve as configuration ⟨*key*⟩s, too. This means they can be set with `typogsetup` and their values can be retrieved with `\typogget`. Options that rely on package microtype are indicated with »microtype req.«.

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

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

`ligaturekern=`⟨*dim*⟩
: Set ⟨*dim*⟩ of the kern that is inserted to split a ligature in macro `\nolig`. See Sec. 3.3. Default value: 33⁄1000 em.

We access the configuration values with `\typogget`.

`mathitalicscorrection=`⟨*dim*⟩
: Italics correction in math mode. See Sec. 3.4 and also the complementary configuration option `textitalicscorrection`. Default value: 0.4mu.[1]

`raise*=`⟨*dim*⟩
: Set the length by which selected characters (dash, hyphen, times, and number dash) are raised. Default value: 0pt.

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

`raisecapitaldash=`⟨*dim*⟩
: Set the length that the `\textendash` is raised in `\capitaldash`. See Sec. 3.6.2. Default value: 0.0pt.

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

`raisecapitalhyphen=`⟨*dim*⟩
: Set the length that the hyphen character ⌞-⌝ is raised in `\capitalhyphen`. See Sec. 3.6.1. Default value: 0.0pt.

`raisecapitaltimes=`⟨*dim*⟩
: Set the length that the multiplication symbol ⌞×⌝ is raised in `\capitaltimes`. See Sec. 3.6.4. Default value: 0.0pt.

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

`raisecapitalguillemets=`⟨*dim*⟩
: Set the length that single and double guillemets are raised in the uppercase versions of the guillemet macros. See Sec. 3.6.5. Default value: 0.0pt.

`raiseguillemets=`⟨*dim*⟩
: Set the length that single and double guillemets are raised in the lowercase versions of the guillemet macros. See Sec. 3.6.5. Default value: 0.0pt.

`raisefiguredash=`⟨*dim*⟩
: Set the length that the `\textendash` is raised in `\figuredash`. See Sec. 3.6.3. Default value: 0.0pt.

`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 respective levels. They are used in `setfontshrink` (`shrinklimits` triple only), `setfontstretch` (`stretchlimits` triple only), and `setfontexpand` (both triples of limits). See Sec. 3.10.2.

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. 3.5.1. Default value: 50⁄1000 em.

`textitalicscorrection=`⟨*dim*⟩
: Italics correction fallback-value; used if `\fontdimen1` is zero. See Sec. 3.4 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 `settracking` as described in Sec. 3.10.1.

The argument ⟨*outer-spacing*⟩ gets passed to microtype's `\SetTracking` option `outer spacing` [19, 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.

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`.

`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.

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 internal configuration values of package typog▌ This can be done in a safe way without resorting to code that is bracketed by `\makeatletter`/`\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. ■

3.1 Information

Never forget: The visual output counts;
it must always be checked, [. . .].
— Udo Wermuth [25]

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

We define some functions for introspection of the typesetting process.

3.1.1 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 units (i. e., pt), the starred version `\cs-name*` omits the units. The separating slash is `\kernedslash`, which is introduced in Sec. 3.5.1.

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

Use Cases
Colophon. ¶ Font test pages. ■

3.1.2 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.

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

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 the trace information, but the write operations for trace data and `\immediate\write` which is used to print the end-tag are not synchronized. ■

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

```
<typog-inspect␣id="⟨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.

— Any text tool can be used to ferret out the tags. EMACS users will find `(occur ⟨regexp⟩)` to be useful.

— As long as the tags are not nested **sed** or **perl** extract the information gathered by `typoginspect`, for example:

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

or

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

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

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

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

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 respective command-line.

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

Investigating the badness of a paragraph. It is generally unnecessary to determine the *exact* classification of a paragraph's badness [13, p. 97n], though the curious user can switch on logging of TeX's line-break information with `\tracingparagraphs=1`[5] or simply use the typoginspect environment and check the suffixes

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

of each line in the paragraph, where for ⟨*suffix*⟩ the following mapping holds [13, p. 99]:

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

Example

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

1. The feasible breakpoint ⌞`@@`⌝ number 17 in the paragraph leads to
2. ⌞`line`⌝ 15, which is the loose ⌞`.1`⌝ last ⌞`-`⌝ line of the paragraph.
3. Up to this breakpoint the paragraph has picked up total demerits ⌞`t`⌝ of 142289.
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. ■

5 Reference 24 provides an exceptionally detailed discussion of the output of `\tracingparagraphs`.

6 The author is unaware of any descriptions of `s`, `a`, or `g`. 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.

All of our guillemets were raised by 33/1000 em

Note

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

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

or

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

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

Investigating page-breaks. Use `\tracingpages=1` or the typoginspect environment to switch on tracing of TeX's page-break information [13, 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. ■

8 See also the discussion of the TeX output routines by Solomon [21].

3.2 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, for example, in the innocuous case of 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‹ 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. 10. ¶

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

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

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

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

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

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

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

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

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

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

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

TABLE 1: TEX offers plenty of possibilities to hyphenate a compound. ¶ We use the sample ›hyphenated-compound‹ to show various code 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 [31]

`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

```
\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.

***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 `\hyphenchar`s. If these are reassigned, the usual hyphenation occurs again.

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

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

`\breakpoint`
`\breakpoint*`

The empty discretionary construct [13, p. 95], `\discretionary{}{}{}`, is so helpful that it deserves its own macro – with a descriptive name.

```
\breakpoint
\breakpoint*
```

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

The difference between `\breakpoint` and the LaTeX macro `\allowbreak` is not only that the former has a starred form, but the penalty associated with `\breakpoint` is the current[10] `\exhyphenpenalty`, whereas `\allowbreak` statically assigns a zero penalty.

Use Case

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

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

`hyphenmin` (*env.*)
Since v0.3

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

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

Without optional argument `hyphenmin` sets both `\lefthyphenmin` and `\righthyphenmin` to ⟨*hyphen-minimum*⟩. When called with an optional argument it sets `\lefthyphenmin` to ⟨*left-hyphen-minimum*⟩ and `\righthyphenmin` to ⟨*hyphen-minimum*⟩.[11]

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

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. ■

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

Alternatively use `\texorpdfstring` [18, Sec. 4.1.2, p. 22]. ■

spectively.

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). ■

3.4 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`[12] in both the starred and the unstarred form.

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

Use Cases

Stronger or weaker correction than `\/`. ¶ Correct a non-slanted or non-italicized font. ¶ Negative correction at the left-hand side[13] of italics, i. e., compensate »shift-to-the-right effect« of italics. ¶ Positive correction at the left-hand side of italics, e. g., an opening parenthesis or square bracket followed by an italic *f* (before: 8, after: 7) or *y* (before: 4, after: 1) reaching far to the left below the baseline. ■

The ⟨*strength*⟩ parameter explained. TEX records the slant angle α of a font in `\fontdim1` as 1 pt × sin α. 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

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

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.

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

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

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

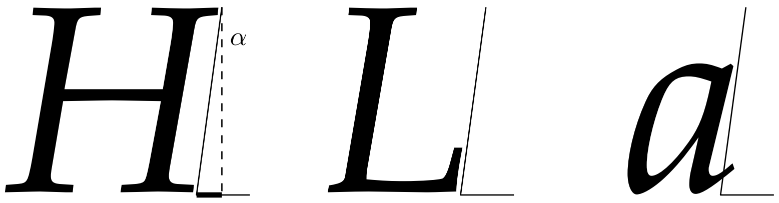


FIGURE 1: Some letters of an italics font. We use the capital ⌞H⌝ to measure the angle α between the plumb-line (drawn dashed) and a tangent to the rightmost parts of the glyph. The length of the plumb-line is proportional to ⟨*strength*⟩ and the short, thick part of the baseline symbolizes the resulting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

3.5 Apply Extra Kerning

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

3.5.1 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`.

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

Use Cases

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

3.5.2 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 [31]) 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. 3.6. 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 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 allow typsetting these characters at a user definable, adjusted height above the baseline. Users can base their own definitions of raised characters on their associated dimensions.[14]

[14] Also compare with Ex. 12 in Ref. 30 for an attempt to automate vertical alignment.

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

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

3.6.1 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. ■

3.6.2 Capital Dash

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

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

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

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

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

`\capitalemdash`
`\capitalemdash*`

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

```
\capitalemdash
\capitalemdash*
```

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

3.6.3 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. 3.6.1 and 3.6.2.

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

TABLE 2: Suggested values for raising the en-dash between lining numerals of some selected fonts.

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

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 [9]).

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*. ■

3.6.4 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. ■

3.6.5 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[15] 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.

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

\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 .025em to .075em are typical for fonts that need this kind of correction. Table 3 summarizes some findings.

Table 3: Suggested values for raising guillemets of some selected fonts.

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

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 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 [20]) are in effect. For an instructive discussion consult Ch. 17, »Paragraph End«, of Ref. 10. The following environments allow to 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 `lastlineragged-leftpar`.

`lastlinecenteredpar` (*env.*)

Center the last lines of the paragraphs enclosed by this environment.[16]

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

16 Also compare the approach taken in Ref. 27.

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

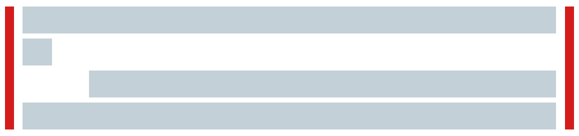
3.8 Fill Last Line of a Paragraph

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.

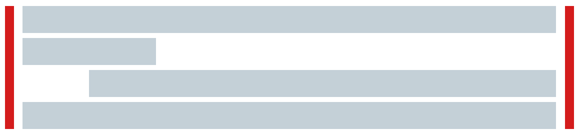
3.8.1 Problem Definition

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

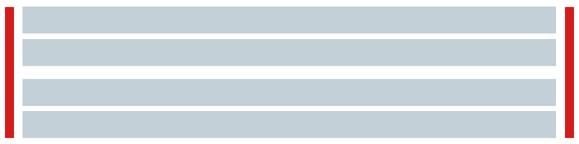
1. \parindent > 0 [27, O1]
   If the last line of a paragraph is shorter than the \parindent of the following paragraph a visual gap tears open.

   The same problem arises with displayed math in a flush-left[17] setting, e. g., amsmath [2] and option fleqn.[18]
   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.

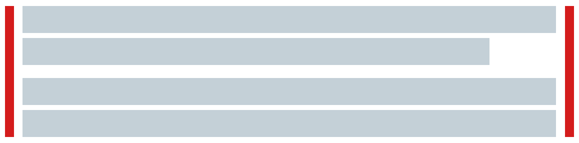
   

2. \parindent $= 0$ [27, O2]
   If the last line of a paragraph is completely filled with text, i. e., flush with the right margin, it may become hard to spot the start of the following paragraph unless \parskip is large.[19]

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

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`[20] for the gap [7].

Tip

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

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

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

Important

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

- Technically, the paragraphs need to contain enough glue (see e. g. Sec. 3.11) 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. ■

3.8.2 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`.

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

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

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

2. Uniform paragraph change.
   (a) Vary spacing.
   Modify the inter-word spacing, for example, with the macros introduced in Sec. 3.9.1.
   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. 3.10.1. 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. 3.10.2.

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. 3.10.2.

3.8.3 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[21] or a club line[22] [13, p. 104 and 16]. (See also Sec. 3.12

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

22 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.[23] 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.

3.8.4 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.

23 Command `\looseness` is a TeX primitive [13, p. 103n]. A thorough discussion of the interaction of `\linepenalty` and `\looseness` can be found in Ref. 26.

3.9 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. 3.10.

3.9.1 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 [15, 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.

`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, respectively.

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

Environment `loosespacing` increases the width of a space by the percentages given in the Tab. 4.

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

TABLE 4: 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. 5.

The default level of `tightspacing` is 1.

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

TABLE 5: 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.

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. ■

3.9.2 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. 6 that associates the `\fontdimen`⟨*number*⟩s with their meanings and also reports on their current values (for this document).[24]

#	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 6: The first column ⌞#⌝ states the index of the `\fontdimen` parameter: ⟨*number*⟩. Column 2 presents short descriptions of the `\fontdimen`⟨*number*⟩ parameters. As examples, the values for the current font are shown in column 3; they are normalized to the quad-size.

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

Typeset a wide, sentence-ending space as if in `\nonfrenchspacing` mode. Consult Table 7 for a comparison of the various sizes.

```
\widespace
\widespace*
```

24 The association is given in Appendix F (p. 433) of Ref. 13. For a concise and understandable explanation of the TeX `\fontdimen` parameters consult Ref. 8.

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

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

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

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

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.

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.[25] ■

3.9.3 Narrow Space

\narrowspace
\narrowspace*
SINCE V0.2

Typeset a narrow space. Consult Table 7 for a comparison of the various sizes.

```
\narrowspace
\narrowspace*
```

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

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

if \fontdimen7 is different from zero or otherwise

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

The starred version, \narrowspace*, unconditionally uses the \fontdimen7 = 0 code-path. Refer to Table 6 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, respectively.

25 See also »Investigating the badness of a paragraph« on Page 7.

Use Case

Tighten loose lines with a series of `\narrowspaces`.[26] ■

TABLE 7: Exemplary comparison of standard `\space` versus `\narrowspace` and `\widespace`. All values are relative to the size of the current font's quad size. `\narrowspace` and `\widespace` use the package's defaults. ¶ The upper values in the Width-column for `\narrowspace`, and `\widespace` refer to the `\fontdimen7` $\neq 0$ case and the lower ones to the `\fontdimen7` $= 0$ code-path.

Name	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

3.10 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 [19] has been loaded, preferably *before* package typog

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

in the document preamble. ■

3.10.1 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.

26 Footnote 25 again applies.

See Sec. 5.3, ›Tracking‹, and 7, »Letterspacing revisited«, in the documentation of microtype [19] 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. ■

3.10.2 Font Expansion

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

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

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

A ⟨*level*⟩ of zero is a no-op. Tables 8 and 9 summarize the values for `stretch` and `shrink` in these environments.

⟨*level*⟩	`stretch`	`shrink`	Comment
	1⁄1000 em	1⁄1000 em	
0	n/a	n/a	no operation
1	0	5	default
2	0	10	
3	0	20	

TABLE 8: Preconfigured values for `shrink` inside of environment `setfontshrink`. Note that all `stretch` values are zero, so the fonts only can shrink.

⟨*level*⟩	`stretch`	`shrink`	
	1⁄1000 em	1⁄1000 em	
0	n/a	n/a	no operation
1	5	0	default
2	10	0	
3	20	0	

TABLE 9: Preconfigured values for `stretch` inside of environment `setfontstretch`. 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 10 gives an overview of the values associated with ⟨*level*⟩.

⟨*level*⟩	`stretch` ¹⁄₁₀₀₀ em	`shrink` ¹⁄₁₀₀₀ em	Comment
0	n/a	n/a	no operation
1	5	5	default
2	10	10	
3	20	20	

TABLE 10: Preconfigured values for `shrink` and `stretch` inside of environment `setfontexpand`. 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 [19].

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

Use Cases

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

`nofontexpansion` (*env.*) Disable the microtype feature ›expansion‹ inside of the environment.

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

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

Use Cases

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

3.10.3 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. ■

3.11 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.[27] The default ⟨*sloppiness*⟩ is 1.

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

Table 11 summarizes the adjustments that `\slightlysloppy` makes depending on the ⟨*sloppiness*⟩ level.

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`. ■

27 Also compare the findings for `\emergencystretch` in Ref. 25.

TABLE 11: Adjustments made by `\slightlysloppy` to various TEX parameters at different levels of ⟨*sloppiness*⟩.

⟨*sloppiness*⟩	`\toler-`	`\hfuzz`	`\emergency-`	Comment
	`ance`	`\vfuzz`	`stretch` G	
		pt	em	
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

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

to prevent excessive stretchability in narrow lines.

3.12 Vertically Partially-Tied Paragraphs

LaTeX provides several macros and environments to tie material vertically – most prominently `samepage` and `minipage`.[28] 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 ε-TeX penalty arrays [6, Sec. 3.8]. In addition the environments `vtietoppar`, `vtiebotpar`, and `vtiebotdisptoppar` explicitly issue a `\par` at the end of the group.

`\vtietop`
`vtietoppar` (*env.*)

Avoid a club line in each partial paragraph.

```
\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 [4]:

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

or shorter and with the default ⟨*number-of-lines*⟩, 3, using the enumitem style[29] `vtie-top`:

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

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

`vtietop` (*enumitem key*)

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

`\vtiebot`
`vtiebotpar` (*env.*)

Avoid a widow line in each partial paragraph.

```
\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.[30]

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

30 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. 3.8.3 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 `vtiebotdisp-par` count backward from the end of each partial paragraph. Again, each displayed math in the paragraph resets the count. According to TEX's rules, a displayed math formula always is counted as *three* lines no matter its contents. Table 12 summarizes these rules with the help of an example.

TABLE 12: 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	} Display math		
5			
6			
7	Text line$_4$	1	2
8	Text line$_5$	2	1

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

‡ The same counting scheme also holds for `vtiebotpar`, `\vtiebot-disp`, and `vtiebotdisppar`. It is implied by ε-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`[31] or `\shortintertext`[32] it is desirable to make the display breakable. This is achieved with `\allowdisplaybreaks` or the environment `breakabledisplay` which will be described in Sec. 3.13. ■

Use Cases

Fix widows and orphans, e. g., those turned up by package widows-and-orphans [17]. ¶ 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. ■

3.13 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 13 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}}
```
■

31 Introduced in package amsmath [2].
32 Defined in package mathtools [11].

Table 13: 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`	Comment
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. ■

3.14 Setspace Front-End

In the copy of this document gets typeset with 10/12.5.

Package setspace [22] 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. 12.5pt) or the lines' leading to a length or percentage of the font's size.[33] This is where the following macros go to bat.

Important

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

```
\usepackage{setspace}
```

in the document preamble. ■

`\setbaselineskip`
SINCE V0.3

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

```
\setbaselineskip{⟨baseline-skip⟩}
```

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

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

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

Example

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

```
\setbaselineskip{12.5pt}
```
■

Tip

To set the `\baselineskip` relative to the current value use

```
\setbaselineskip{⟨factor⟩\baselineskip}
```

where ⟨*factor*⟩ is a floating-point number. ■

`\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.

33 To find out about the current font's size and the `\baselineskip` in printable form check out Sec. 3.1.1 on p. 5.

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

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

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

`\setbaselineskippercentage{125}`

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

`\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 10pt 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 10pt, but now write

`\setleadingpercentage{25}`

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

`\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:[34]

`\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

34 For an overview of the various `\fontdimen`⟨*number*⟩ parameters consult Tab. 6 on p. 26.

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

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

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

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

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

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

or

```
\setleadingpercentage{0}
```

Tip

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

Here are some forms that do work:

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

3.15 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 [28] we prescribe the line-widths with the `\parshape` primitive and leave alone the stretchability or shrinkability of the glue.

`smoothraggedrightshapetriplet` (*env.*)
`smoothraggedrightshapequintuplet` (*env.*)
`smoothraggedrightshapeseptuplet` (*env.*)

We introduce three environments that allow for setting three, five, or seven different line-lengths: `smoothraggedrightshapetriplet`, `smoothraggedrightshapequintuplet`, and `smoothraggedrightshapeseptuplet`; they work for paragraphs up to 99, 95, or 98 lines, respectively.

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

The environments take $N = 3, 5,$ or 7 mandatory line-width parameters, where each ⟨*widthI*⟩, $I = 1, \ldots, N$ 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`.

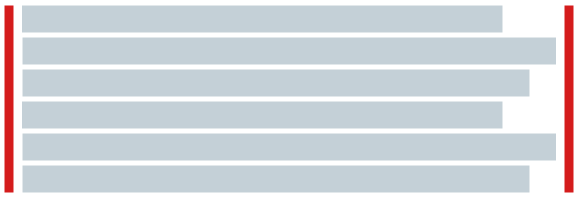
`smoothraggedrightpar` (*env.*)

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.

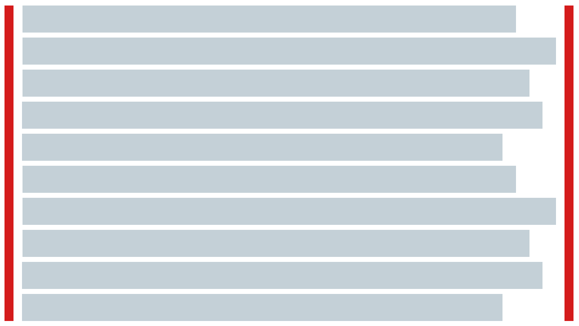
```
\begin{smoothraggedrightpar}[⟨option...⟩]
  ...
\end{smoothraggedrightpar}
```

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

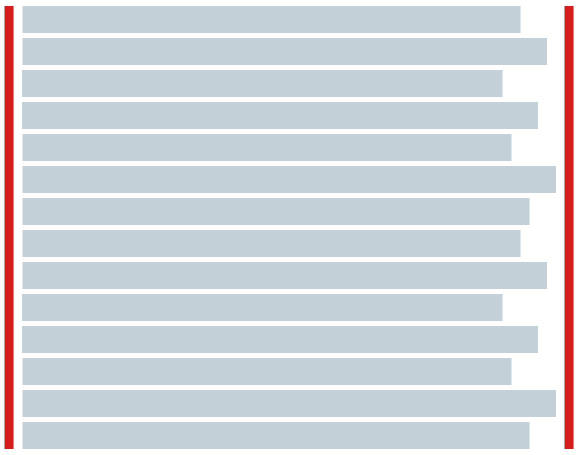
— The triplet generator repeats a *short line – long line – middle-length line* sequence. Shown below are two complete cycles.



— 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[35] 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: 0pt.

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

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

Use Cases
Replacement for `\RaggedRight` [20]. ¶ Design alternative for fully justified paragraphs if used with a small rag-width. ■

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

4 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 in-line lists [4].
geometry	Powerful and sophisticated setup of the page layout [23]. Best accompanied by layout [14] to visualize the page geometries.
hyphenat	Hyphens that do not inhibit further auto-hyphenation of a compound word [31].
microtype	Fine control of spacing, tracking, sidebearings, character protrusion into the margins, font expansion, and much more [19]. See also Khirevich’s discussion [12].
ragged2e	Improved versions of environments `raggedleft`, `raggedright`, and `center` [20].
setspace	Consistently set the document’s line-spacing, i. e., `\baselineskip` [22]

A 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}
                [2024/05/07  v0.3  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 debug information printer.

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

```

`\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
      \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 ty-
pog}}%
  \fi
}

```

Our own state…

`ypog@mathitalicscorrection`

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

ypog@textitalicscorrection

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

\typog@ligaturekern

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

\typog@raisecapitaldash

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

pog@raisecapitalguillemets

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

\typog@raisecapitalhyphen

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

\typog@raisecapitaltimes

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

\typog@raiseguillemets

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

\typog@raisefiguredash

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

\typog@slashkern

```
\newlength{\typog@slashkern}
```

\typog@breakpenalty

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

\typog@trackingttspacing

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

Setup

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

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

```

A.1 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 out 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 id="\typog@@id" job="\jobname" line="\the\inputlineno"
```

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}

```

A.2 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}
  {}

```

A.3 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@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
        \PackageError{typog}
                     {Missing third argument of \nolig}
                     {Append empty group or \relax after macro invocation}
      \fi
    \fi}
}

```

A.4 Manual Italic Correction

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

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

```

A.5 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@slashkern}%
   \typog@forwardslash
   \IfBooleanTF{#1}%
     {\hspace*{\typog@slashkern}\ignorespaces}%
     {\typog@breakpoint\typog@allowhyphenation\hspace*{\typog@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@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}{-}
}

```

A.6 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@breakpenalty}
```

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

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

The non-hyperref version's code is straightforward. The `\pdfstringdef-DisableCommands` version must be expandable and must match the other version's signature. Yikes! We exploit the fact that conditions are expandable. However, we cannot use `\typog@hyphen` in the expansion as `\char` gets in the way. So, we fall back to the least common denominator and use a bare dash.

```
\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@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@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}
}

```

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

```
\NewDocumentCommand{\figuredash}{s}
  {\raisebox{\typog@raisefiguredash}{\textendash}%
   \IfBooleanTF{#1}%
     {\ignorespaces}%
     {\typog@breakpoint\typog@allowhyphenation}}
```

PDF-substitute definition

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

```

`\capitaltimes`

```
\NewDocumentCommand{\capitaltimes}{}
  {\ifmmode
      \mathbin{\raisebox{\typog@raisecapitaltimes}{$\m@th\times$}}%
   \else
     \raisebox{\typog@raisecapitaltimes}{\texttimes}%
   \fi}
```

PDF-substitute definition

```
\typog@register@pdfsubstitute{
  \RenewExpandableDocumentCommand{\capitaltimes}{}{\texttimes}
}

```

`\singleguillemetleft`

```
\NewDocumentCommand{\singleguillemetleft}{}
  {\typog@allowhyphenation
   \raisebox{\typog@raiseguillemets}{\guilsinglleft}}
```

PDF-substitute definition

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

`\singleguillemetright`

```
\NewDocumentCommand{\singleguillemetright}{}
  {\raisebox{\typog@raiseguillemets}{\guilsinglright}%
   \typog@allowhyphenation}
```

PDF-substitute definition

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

`\doubleguillemetleft`

```
\NewDocumentCommand{\doubleguillemetleft}{}
  {\typog@allowhyphenation
   \raisebox{\typog@raiseguillemets}{\guillemotleft}}
```

PDF-substitute definition

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

`\doubleguillemetright`

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

PDF-substitute definition

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

`\Singleguillemetleft`

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

PDF-substitute definition

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

`\Singleguillemetright`

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

PDF-substitute definition

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

`\Doubleguillemetleft`

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

PDF-substitute definition

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

`\Doubleguillemetright`

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

PDF-substitute definition

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

```

A.7 Align Last Line of a Paragraph

The code of environment `lastlineraggedleftpar` has been inspired by macro `\lastlineraggedleft` [32, 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* [10, Sec. 18.3.1].

```
\NewDocumentEnvironment{lastlinecenteredpar}{}
  {\lastlinefit=0%
   \setlength{\leftskip}{\z@ \@plus .5fil}%
   \setlength{\rightskip}{-\leftskip}%
   \setlength{\parfillskip}{\z@ \@plus 1fil}}
  {\par}

```

A.8 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. 27.

```
\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}

```

A.9 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 [10, 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}

```

A.10 Microtype Front-End

Tracking

`setfonttracking` (*env.*) To archieve 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 = {*}}%
     {}
   \SetExpansion
     [context = typog@stretch2,
      shrink = 0,
      stretch = \typog@stretch@ii]%
     {encoding = {*}}%
     {}
   \SetExpansion
     [context = typog@stretch3,
      shrink = 0,
      stretch = \typog@stretch@iii]%
     {encoding = {*}}%
     {}

   \SetExpansion
     [context = typog@expand1,
      shrink = \typog@shrink@i,
      stretch = \typog@stretch@i]%
     {encoding = {*}}%
     {}
   \SetExpansion
     [context = typog@expand2,
      shrink = \typog@shrink@ii,
      stretch = \typog@stretch@ii]%
     {encoding = {*}}%
     {}
   \SetExpansion
     [context = typog@expand3,
      shrink = \typog@shrink@iii,
      stretch = \typog@stretch@iii]%
     {encoding = {*}}%
     {}}
```

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.

```
\newcommand*{\typog@test@microtype@expansion@feature}
  {\ifMT@expansion
     \typog@typeout{microtype preloaded -- font expansion features available}%
     \def\typog@require@microtype@expansion{\relax}
     \typog@setup@font@expansion
   \else
     \PackageWarning{typog}{microtype preloaded,\space
```

```
                            but font expansion is disabled}%
    \def\typog@require@microtype@expansion
      {\PackageError{typog}
                    {microtype font expansion disabled}
                    {pass option ‘expansion’ to package microtype}}
  \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 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}

```

A.11 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(0.1, 0.5, 9);`

`slightlysloppypar` (*env.*)

```
\NewDocumentEnvironment{slightlysloppypar}{O{1}}
  {\par\slightlysloppy[#1]\ignorespaces}
  {\par}
```

A.12 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@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@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}
        \widowpenalty
  \or % 5
    \widowpenalties 6
        \@M \@M \@M \@M
        \value{typog@mean@penalty}
        \widowpenalty
  \or % 6
    \widowpenalties 7
        \@M \@M \@M \@M \@M
        \value{typog@mean@penalty}
        \widowpenalty
  \or % 7
    \widowpenalties 8
        \@M \@M \@M \@M \@M \@M
        \value{typog@mean@penalty}
        \widowpenalty
  \or % 8
    \widowpenalties 9
        \@M \@M \@M \@M \@M \@M \@M
        \value{typog@mean@penalty}
        \widowpenalty
  \else % >= 9
    \widowpenalties 10
        \@M \@M \@M \@M \@M \@M \@M \@M
        \value{typog@mean@penalty}
        \widowpenalty
  \fi}

```

vtiebotpar (*env.*)

```
\NewDocumentEnvironment{vtiebotpar}{O{3}}
  {\vtiebot[#1]}
  {\par
   \ignorespacesafterend}

```

\typog@vtiebotdisp

```
\NewDocumentCommand{\typog@vtiebotdisp}{m}
  {\setcounter{typog@mean@penalty}
              {\typog@geometric@mean{\@M}{\displaywidowpenalty}}%
   \typog@typeout{vtiebotdisp: penalties \the\@M--\the\value{typog@mean@penalty}--\the\displaywidowpenalty}%
   \unless\ifnum\displaywidowpenalty<\@M
     \PackageWarning{typog}{vtiebotdisp: displaywidowpenalty=\the\displaywidowpen
   \fi
   \ifcase#1% 0
     \relax
   \or % 1
     \relax
   \or % 2
     \displaywidowpenalties 3
         \@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}}%
  \edef\typog@@after@display@math{\vtietop[\typog@@top@lines]}%
  \PushPostHook{display}{\aftergroup\typog@@after@display@math}%
  \vtiebotdisp[#1]}
 {\par
  \PopPostHook{display}%
  \ignorespacesafterend}

```

A.13 Breakable Disp. Eqs.

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

```
\newenvironment*{breakabledisplay}[1][3]
 {\allowdisplaybreaks[#1]}
 {\ignorespacesafterend}

```

A.14 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.

```
\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.

```
\newcommand*{\typog@setbaselineskip@relative@error}{.001}
```

\typog@setbaselineskip Given the ⟨*target-baselineskip*⟩ as argument iterate setting `\setstretch` until the error drops below our threshold.

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

Initialize our "emergency-stop" loop counter.

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

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

\typog@typeout{\string\setbaselineskip:\space
  initial\space baselineskip:\space \the\baselineskip}
\typog@typeout{\string\setbaselineskip:\space
  target\space baselineskip:\space \dim_use:N \l_tmpa_dim}

\dim_compare:nNnTF {\baselineskip} > {\c_zero_dim}
{}
{
  \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@typeout{\string\setbaselineskip:\space
    final\space \string\setstretch\space argument:\space
    \fp_use:N \l_tmpa_fp}
  \typog@typeout{\string\setbaselineskip:\space
    final\space baselineskip:\space \the\baselineskip}
}

```

`\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.

```
\cs_new:Npn \setbaselineskip #1
{
  \AfterPreamble{\typog@setbaselineskip{#1}}
  \ignorespaces
}

```

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

```
\cs_new:Npn \resetbaselineskip
{
  \AfterPreamble{\setstretch{1}}
}

```

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

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

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

```
\AfterEndPreamble{
  \dim_set:Nn \typogfontsize {\fontdimen6\font}
  \typog@typeout{\string\typogfontsize =
    \dim_use:N \typogfontsize\space
    (at\space begin\space of\space document)}
}

```

`\setbaselineskippercentage`

```
\cs_new:Npn \setbaselineskippercentage #1
{
  \AfterPreamble{
    \dim_compare:nNnTF {\typogfontsize} > {\c_zero_dim}
    {
      \typog@setbaselineskip{
        \fp_eval:n {(#1) / 100} \typogfontsize}
    }
    {
      \PackageError{typog}
                   {\string\setbaselineskippercentage:\space
                    \string\typogfontsize <= 0}
                   {Maybe\space \string\typogfontsize\space
                     is\space uninitialized?}
    }
  }
  \ignorespaces
}

```

`\setleading`

```
\cs_new:Npn \setleading #1
{
  \AfterPreamble{
    \dim_compare:nNnTF {\typogfontsize} > {\c_zero_dim}
    {
      \typog@setbaselineskip{\typogfontsize + \dimexpr #1}
    }
    {
      \PackageError{typog}
                   {\string\setleading:\space
                    \string\typogfontsize <= 0}
                   {Maybe\space \string\typogfontsize\space
                     is\space uninitialized?}
    }
  }
  \ignorespaces
}

```

\setleadingpercentage

```
\cs_new:Npn \setleadingpercentage #1
{
  \AfterPreamble{
    \dim_compare:nNnTF {\typogfontsize} > {\c_zero_dim}
    {
      \typog@setbaselineskip{
        \fp_eval:n {1 + (#1) / 100} \typogfontsize}
    }
    {
      \PackageError{typog}
                   {\string\setleadingpercentage:\space
                    \string\typogfontsize <= 0}
                   {Maybe\space \string\typogfontsize\space
                     is\space uninitialized?}
    }
  }
  \ignorespaces
}
\ExplSyntaxOff

```

A.15 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@typeout{smoothraggedrightshapetriplet: skip0=\the\skip0}%
   \typog@typeout{smoothraggedrightshapetriplet: skip1=\the\skip1}%
   \typog@typeout{smoothraggedrightshapetriplet: skip2=\the\skip2}%
   \typog@typeout{smoothraggedrightshapetriplet: skip3=\the\skip3}%
   \unless\ifnum\typog@mod{\typog@triplet@max@lines}{3}=0
     \PackageError{typog}
                  {Line number of triplet generator %
                    (\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
   \skip1=#2\relax
   \skip2=#3\relax
   \skip3=#4\relax
   \skip4=#5\relax
   \skip5=#6\relax
   \typog@typeout{smoothraggedrightshapequintuplet: skip0=\the\skip0}%
   \typog@typeout{smoothraggedrightshapequintuplet: skip1=\the\skip1}%
   \typog@typeout{smoothraggedrightshapequintuplet: skip2=\the\skip2}%
   \typog@typeout{smoothraggedrightshapequintuplet: skip3=\the\skip3}%
   \typog@typeout{smoothraggedrightshapequintuplet: skip4=\the\skip4}%
   \typog@typeout{smoothraggedrightshapequintuplet: skip5=\the\skip5}%
   \unless\ifnum\typog@mod{\typog@quintuplet@max@lines}{5}=0
```

```
    \PackageError{typog}
                 {Line number of quintuplet generator %
                   (\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  \s
  \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
  \skip1=#2\relax
  \skip2=#3\relax
  \skip3=#4\relax
  \skip4=#5\relax
  \skip5=#6\relax
  \skip6=#7\relax
  \skip7=#8\relax
  \typog@typeout{smoothraggedrightshapeseptuplet: skip0=\the\skip0}%
  \typog@typeout{smoothraggedrightshapeseptuplet: skip1=\the\skip1}%
  \typog@typeout{smoothraggedrightshapeseptuplet: skip2=\the\skip2}%
  \typog@typeout{smoothraggedrightshapeseptuplet: skip3=\the\skip3}%
  \typog@typeout{smoothraggedrightshapeseptuplet: skip4=\the\skip4}%
  \typog@typeout{smoothraggedrightshapeseptuplet: skip5=\the\skip5}%
  \typog@typeout{smoothraggedrightshapeseptuplet: skip6=\the\skip6}%
  \typog@typeout{smoothraggedrightshapeseptuplet: skip7=\the\skip7}%
  \unless\ifnum\typog@mod{\typog@septuplet@max@lines}{7}=0
    \PackageError{typog}
                 {Line number of septuplet generator %
                   (\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  \
    \typog@repeat{\numexpr\typog@septuplet@max@lines / 7 - 1}
                   {\skip0 \skip1  \skip0 \skip2  \skip0 \skip3  \skip0 \skip4  \s
   \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}}

\NewDocumentEnvironment{smoothraggedrightpar}{O{}}
  {\edef\typog@@linewidth{\linewidth}%
   \setkeys[typog]{smoothraggedrightpar}{#1}%
```

Convert generator name to an integer suitable for \ifcase.

```
   \edef\typog@@generatorchoice{%
     \ifnum\pdf@strcmp{\smoothraggedrightgenerator}{triplet}=\z@
       0%
     \else
       \ifnum\pdf@strcmp{\smoothraggedrightgenerator}{quintuplet}=\z@
         1%
       \else
         \ifnum\pdf@strcmp{\smoothraggedrightgenerator}{septuplet}=\z@
           2%
         \else
           \PackageError{typog}
                        {smoothraggedright: unknown generator name}
                        {valid generator names are triplet, quintuplet, and septuplet}%
         \fi
       \fi
     \fi}%
```

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@typeout{smoothraggedright: generator=triplet, typog@fuzzwidth=\the\ty
     \smoothraggedrightshapetriplet[leftskip=\typog@@smoothraggedrightleftskip,
                                    parindent=\glueexpr\smoothraggedrightparinden
indent,
                                    #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@typeout{smoothraggedright: generator=quintuplet, typog@fuzzwidth=\the\typog@fuzzwidth}%
     \smoothraggedrightshapequintuplet[leftskip=\typog@@smoothraggedrightleftskip
                                       parindent=\glueexpr\smoothraggedrightparin
indent,
                                       #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@typeout{smoothraggedright: generator=septuplet, typog@fuzzwidth=\the\
     \smoothraggedrightshapeseptuplet[leftskip=\typog@@smoothraggedrightleftskip,
                                      parindent=\glueexpr\smoothraggedrightparind
indent,
                                      #1]%
        {\glueexpr (\typog@@linewidth * 3 - \smoothraggedrightragwidth * 2) / 3
                   + \glueexpr \z@ \@plus \typog@fuzzwidth \@minus \typog@fuzzwidth\relax}% (3)
        {\glueexpr (\typog@@linewidth * 6 - \smoothraggedrightragwidth) / 6
                   + \glueexpr \z@ \@plus \typog@fuzzwidth \@minus \typog@fuzzwidth\relax}% (6)
        {\glueexpr \typog@@linewidth - \smoothraggedrightragwidth +
                   + \glueexpr \z@ \@plus \typog@fuzzwidth\relax}% (1)
        {\glueexpr (\typog@@linewidth * 3 - \smoothraggedrightragwidth) / 3
                   + \glueexpr \z@ \@plus \typog@fuzzwidth \@minus \typog@fuzzwidth\relax}% (5)
        {\glueexpr (\typog@@linewidth * 6 - \smoothraggedrightragwidth * 5) / 6
                   + \glueexpr \z@ \@plus \typog@fuzzwidth \@minus \typog@fuzzwidth\relax}% (2)
        {\glueexpr \typog@@linewidth \@minus \typog@fuzzwidth\relax}% (7)
        {\glueexpr (\typog@@linewidth * 2 - \smoothraggedrightragwidth) / 2
                   + \glueexpr \z@ \@plus \typog@fuzzwidth \@minus \typog@fuzzwidth\relax}% (4)
    \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}}

```

B typog-grep

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

NAME

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

SYNOPSIS

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

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

The first form shows all `<typog-inspect id="ID" ...>` elements in *LOG-FILE*.

The second form shows the contents of `<typog-inspect id="ID" ...>` elements whose *ID*s match *REGEXP* in *LOG-FILE*.

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

DESCRIPTION

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

The LaTeX user brackets her text in

```
\begin{typoginspect}{ID}
  Text and 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 environment shows up, packed with tracing information, as

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

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 and code to investigate` occurs.

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

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

OPTIONS

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

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

ID-discovery mode: Show all `typog-inspect` elements independent of any matching patterns.

--color, **colour** *WHEN*

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

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

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

--debug

Turn on debug output on *stderr*.

-h, **--help**

Display brief help then exit.

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

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

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

Match ids while ignoring case distinctions in patterns and data.

-j, --[no-]job-name
: Print the `\jobname` that **tex** 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.

-V, --version
: Show version information and exit.

-w, --[no-]word-regexp
: Match only whole words.

CONFIGURATION

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

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

`id-heading-format=`*FORMAT*
: Control the *FORMAT* for printing matching ids in heading-mode, where *FORMAT* is passed to Perl's `printf`. Default: `--> %s <--`.

`id-indent=`*INDENT*
: Indentation of nested typog-inspect tags. Only used in "discovery" mode (first form), i.e., if **--all** is active. Default: 8.

`id-max-length=`*MAXIMUM-LENGTH*
: Set the maximum length of a matching id for printing. 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 ids when printed inline.

`id-heading-color`
: Color of matching ids when printed in heading form.

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

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

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

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

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

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

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

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`

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.

SEE ALSO

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

Change History

References

[1] Abrahams, Paul W., Hargreaves, Kathryn A., and Karl Berry. *TEX for the Impatient*. 2020, http://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] Bezos, Javier. *Package enumitem*. 2019, https://ctan.org/pkg/enumitem.

[5] Bezos, Javier. *Package babel*. 2021, https://ctan.org/pkg/babel. The original author of package babel was J. L. Braams.

[6] Breitenlohner, Peter and the NTS Team. *ε-TEX*. 1998, https://mirrors.ctan.org/systems/doc/etex/etex_man.pdf.

[7] 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.

[8] Carlisle, David. *What do different* \fontdimen<num> *mean*. 2013-1-2, https://tex.stackexchange.com/questions/88991/what-do-different-fontdimennum-mean.

[9] Cubitt, Toby. *Package cleveref*. 2018, https://ctan.org/pkg/cleveref.

[10] Eijkhout, Victor. *TEX By Topic, A Texnician's Reference*. 2007, https://www.eijkhout.net/tex/tex-by-topic.html.

[11] Høgholm, Morten, Madsen, Lars and the LATEX3 Project Team. *Package mathtools*. 2020, https://ctan.org/pkg/mathtools.

[12] Khirevich, Siarhei. *Tips on Writing a Thesis in LATEX*. 2013, http://www.khirevich.com/latex/microtype.

[13] Knuth, Donald Ervin. *The TEXbook*. Addison Wesley, Reading/MA, 1986.

[14] 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.

[15] Middendorp, Jan. *Shaping Text*. bis publishers, Amsterdam, 2014.

[16] 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.

[17] Mittelbach, Frank. *Package widows-and-orphans*. 2020, https://ctan.org/pkg/widows-and-orphans.

[18] Rahtz, Sebastian, and Frank Mittelbach. *Package hyperref*. 2020, https://ctan.org/pkg/hyperref. The package is maintained by the LaTeX3 Project Team.

[19] Schlicht, Robert. *Package microtype*. 2020, https://ctan.org/pkg/microtype.

[20] Schröder, Martin. *Package ragged2e*. 2019, https://ctan.org/pkg/ragged2e.

[21] Solomon, David. *Output Routines: Examples and Techniques. Part I: Introduction and Examples*. TUGboat, 11(1), 69–85, 1990, http://www.tug.org/TUGboat/Articles/tb11-1/tb27salomon.pdf.

[22] Tobin, Geoffrey, and Robin Fairbairns. *Package setspace*. 2011, https://ctan.org/pkg/setspace.

[23] Umeki, Hideo. *Package geometry*. 2020, https://ctan.org/pkg/geometry.

[24] Wermuth, Udo. *Tracing paragraphs*. TUGboat, 37(3), 358–373, 2016, https://tug.org/TUGboat/tb37-3/tb117wermuth.pdf.

[25] Wermuth, Udo. *The optimal value for \emergencystretch*. TUGboat, 38(1), 65–86, 2017, https://tug.org/TUGboat/tb38-1/tb118wermuth.pdf.

[26] Wermuth, Udo. *A note on \linepenalty*. TUGboat, 38(3), 400–414, 2017, https://tug.org/TUGboat/tb38-3/tb120wermuth.pdf.

[27] Wermuth, Udo. *Experiments with \parfillskip*. TUGboat, 39(3), 276–303, 2018, https://tug.org/TUGboat/tb39-3/tb123wermuth-parfillskip.pdf.

[28] Wermuth, Udo. *An attempt at ragged-right typesetting*. TUGboat, 41(1), 73–94, 2020, https://tug.org/TUGboat/tb41-1/tb127wermuth-ragged.pdf.

[29] Wermuth, Udo. Personal communication. August 2, 2022.

[30] Wermuth, Udo. *Vertical alignments in plain TeX*. TUGboat, 44(3), 427–440, 2023, https://tug.org/TUGboat/tb44-3/tb138wermuth-valign.pdf.

[31] Wilson, Peter. *Package hyphenat*. 2004, https://ctan.org/pkg/hyphenat. The package is maintained by W. Robertson.

[32] Wilson, Peter. *Glisterings*. TUGboat, 28(2), 229–232, 2007, https://tug.org/TUGboat/tb28-2/tb89glister.pdf.

[33] Wilson, Peter. *Package needspace*. 2010, https://ctan.org/pkg/needspace. The package is maintained by W. Robertson.

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

In the Index page ranges are stuck together with `\figuredash*`.