

The `postnotes` package^{*}

Code documentation

Gustavo Barros[†]

2023-02-15

Contents

1	Initial setup	2
2	Data	2
3	Options	5
4	<code>\postnote</code>	10
5	<code>\postnoteref</code>	15
6	<code>\postnotesection</code>	16
7	<code>\printpostnotes</code>	17
8	Headers	23
9	Compatibility	30
10	Languages	37
	Index	40

^{*}This file describes v0.2.2, released 2023-02-15.

[†]<https://github.com/gusbrs/postnotes>

1 Initial setup

Start the DocStrip guards.

```

1 <*package>
   Identify the internal prefix (LATEX3 DocStrip convention).
2 <@@=postnotes>

```

The new syntax for file/package hooks, which the package assumes, requires kernel 2021-11-15 (ltnnews34, ltfilehook). Furthermore, the kernel of 2022-06-01 introduced a couple of very nice features which simplifies the relation with hyperref (ltnnews35, hyperref-linktarget): the provision of \MakeLinkTarget and the definition by the kernel of the starred version of \ref, which we can use regardless of hyperref being loaded. So we require the 2022-06-01 kernel or newer.

```

3 \providecommand\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
4 \IfFormatAtLeastTF{2022-06-01}
5 {}
6 {%
7   \PackageError{postnotes}{LaTeX kernel too old}
8   {%
9     'postnotes' requires a LaTeX kernel 2022-06-01 or newer.%
10    \MessageBreak Loading will abort!%
11   }%
12 \endinput
13 }%

14 \ProvidesExplPackage {postnotes} {2023-02-15} {0.2.2}
15 {Endnotes for LaTeX}

```

2 Data

_postnotes_data_name:n Returns the name of the property list variable which stores the data of the \postnote with <note id> number.

```

    \_postnotes_data_name:n {\<note id>}

16 \cs_new:Npn \_postnotes_data_name:n #1
17   { g\_postnotes_ #1 _data_prop }
18 \cs_generate_variant:Nn \_postnotes_data_name:n { e }

```

(End definition for _postnotes_data_name:n.)

postnotes provides a number of hooks from the new hook system to grant some points of access in key places of the package. Note that hooks created with \NewHook are meant to be public interfaces (see <https://chat.stackexchange.com/transcript/message/62955941#62955941>, and following discussion).

_postnotes_store:nn Stores the metadata and <note content> of \postnote with ID <note id>, from where it is called. The postnotes/note/store hook is intended to add further data to the note, when required to support packages with specific needs.

```

    \_postnotes_store:nn {\<note id>} {\<note content>}

```

```

19 \NewHook { postnotes/note/store }
20 \cs_new_protected:Npn \__postnotes_store:nn #1#2
21 {
22   \prop_new:c { \__postnotes_data_name:e {#1} }
23   \prop_gput:cnn { \__postnotes_data_name:e {#1} } { type } { note }
24   \prop_gput:cnx { \__postnotes_data_name:e {#1} } { mark }
25     { \l__postnotes_mark_tl }
26   \prop_gput:cnx { \__postnotes_data_name:e {#1} } { counter }
27     { \int_use:N \c@postnote }
28   \prop_gput:cnx { \__postnotes_data_name:e {#1} } { sortnum }
29     {
30       \bool_if:NTF \l__postnotes_manual_sortnum_bool
31         { \fp_use:N \l__postnotes_sort_num_fp }
32         { \int_use:N \c@postnote }
33     }
34   \cs_if_exist:cT { chapter }
35     {
36       \prop_gput:cnx { \__postnotes_data_name:e {#1} }
37         { thechapter } { \thechapter }
38     }
39   \prop_gput:cnx { \__postnotes_data_name:e {#1} } { thesection }
40     { \thesection }
41   \prop_gput:cnx { \__postnotes_data_name:e {#1} } { pnsectname }
42     { \g__postnotes_section_name_tl }
43   \prop_gput:cnx { \__postnotes_data_name:e {#1} } { pnsectid }
44     { \int_use:N \g__postnotes_sectid_int }
45   \prop_gput:cnx { \__postnotes_data_name:e {#1} } { multibool }
46     { \bool_to_str:N \l__postnotes_maybe_multi_bool }
47   \prop_gput:cnn { \__postnotes_data_name:e {#1} } { content } {#2}
48   \UseHook { postnotes/note/store }
49 }

```

(End definition for __postnotes_store:nn.)

__postnotes_store_section:nn Stores the metadata and $\langle \text{note content} \rangle$ of \postnotessection with ID $\langle \text{note id} \rangle$, from where it is called.

```

\__postnotes_store_section:nn {\langle \text{note id} \rangle} {\langle \text{note content} \rangle}
50 \cs_new_protected:Npn \__postnotes_store_section:nn #1#2
51 {
52   \prop_new:c { \__postnotes_data_name:e {#1} }
53   \prop_gput:cnn { \__postnotes_data_name:e {#1} } { type } { section }
54   \cs_if_exist:cT { chapter }
55     {
56       \prop_gput:cnx { \__postnotes_data_name:e {#1} }
57         { thechapter } { \thechapter }
58     }
59   \prop_gput:cnx { \__postnotes_data_name:e {#1} } { thesection }
60     { \thesection }
61   \prop_gput:cnn { \__postnotes_data_name:e {#1} } { content } {#2}
62 }
63 \cs_generate_variant:Nn \__postnotes_store_section:nn { nx }

```

(End definition for __postnotes_store_section:nn.)

`__postnotes_prop_get:nnN` Convenience functions to retrieve and clear data from a note based on the ID number.
`__postnotes_prop_item:nn`
`__postnotes_prop_gclear:n`

```

\__postnotes_prop_get:nnN {<note id>} {<property>} {<tl var to set>}
\__postnotes_prop_item:nn {<note id>} {<property>}
\__postnotes_prop_gclear:n {<note id>}

64 \cs_new_protected:Npn \__postnotes_prop_get:nnN #1#2#3
65 {
66   \prop_get:cnNF { \__postnotes_data_name:e {#1} } {#2} #3
67   { \tl_clear:N #3 }
68 }
69 \cs_new:Npn \__postnotes_prop_item:nn #1#2
70 { \prop_item:cn { \__postnotes_data_name:e {#1} } {#2} }
71 \cs_new_protected:Npn \__postnotes_prop_gclear:n #1
72 { \prop_gclear:c { \__postnotes_data_name:e {#1} } }

(End definition for \__postnotes_prop_get:nnN, \__postnotes_prop_item:nn, and \__postnotes_prop_gclear:n.)

```

`\post@note` The `\newlabel` equivalent for postnotes. Based on the kernel's `\@newl@bel` so that we get L^AT_EX checks for multiple and changed references for free (procedure learnt from `zref`). `\post@note`, when the `.aux` file is read, defines macros named `\postnote@r@<label name>`, according to the prefix set by `\c__postnotes_ref_prefix_tl`.

```

\post@note {<label name>} {<label content>}

73 \tl_const:Nn \c__postnotes_ref_prefix_tl { postnote@r }
74 \cs_new_protected:Npx \post@note #1#2
75 { \exp_not:N \@newl@bel { \c__postnotes_ref_prefix_tl } {#1} {#2} }

(End definition for \post@note.)

```

`__postnotes_set_mark_page_label:n` Label setting functions for each pertinent context. They must use `\iow_shipout_x:Nn`, since the main information we are interested in is the `page`.
`__postnotes_set_text_page_label:n`
`__postnotes_set_print_page_label:n`

```

\__postnotes_set_mark_page_label:n {<label name>}
\__postnotes_set_text_page_label:n {<label name>}
\__postnotes_set_print_page_label:n {<label name>}

76 \cs_new_protected:Npn \__postnotes_set_mark_page_label:n #1
77 {
78   \iow_shipout_x:Nn \@auxout
79   { \token_to_str:N \post@note { mark@ #1 } { \thepage } }
80 }
81 \cs_generate_variant:Nn \__postnotes_set_mark_page_label:n { x }
82 \cs_new_protected:Npn \__postnotes_set_text_page_label:n #1
83 {
84   \iow_shipout_x:Nn \@auxout
85   { \token_to_str:N \post@note { text@ #1 } { \int_use:N \c@page } }
86 }
87 \cs_generate_variant:Nn \__postnotes_set_text_page_label:n { x }
88 \cs_new_protected:Npn \__postnotes_set_print_page_label:n #1
89 {
90   \iow_shipout_x:Nn \@auxout
91   { \token_to_str:N \post@note { print@ #1 } { \int_use:N \c@page } }
92 }
93 \cs_generate_variant:Nn \__postnotes_set_print_page_label:n { x }

```

(End definition for `_postnotes_set_mark_page_label:n`, `_postnotes_set_text_page_label:n`, and `_postnotes_set_print_page_label:n`.)

`_postnotes_get_pageref:Nn`
`_postnotes_extract_pageref:n`

Reference data extraction functions.

```

\__postnotes_get_pageref:Nn {\tl var to set}} {\label name}}
\__postnotes_extract_pageref:n {\label name}}

94 \cs_new_protected:Npn \__postnotes_get_pageref:Nn #1#2
95 {
96   \cs_if_exist:cTF { \c__postnotes_ref_prefix_tl @ #2 }
97   { \tl_set:Nv #1 { \c__postnotes_ref_prefix_tl @ #2 } }
98   { \tl_clear:N #1 }
99 }
100 \cs_generate_variant:Nn \__postnotes_get_pageref:Nn { Nx }
101 \cs_new:Npn \__postnotes_extract_pageref:n #1
102 {
103   \cs_if_exist:cTF { \c__postnotes_ref_prefix_tl @ #1 }
104   { \exp_not:v { \c__postnotes_ref_prefix_tl @ #1 } }
105   { \c_empty_tl }
106 }
107 \cs_generate_variant:Nn \__postnotes_extract_pageref:n { e }

```

(End definition for `_postnotes_get_pageref:Nn` and `_postnotes_extract_pageref:n`.)

3 Options

heading option

```

108 \keys_define:nn { postnotes/setup }
109 {
110   heading .cs_set_protected:Np = \pnheading ,
111   heading .value_required:n = true ,
112 }

```

`\pnheading` Provide default value for `\pnheading`.

```

113 \cs_if_exist:cTF { chapter }
114 {
115   \cs_new_protected:Npn \pnheading
116   {
117     \chapter*{\pntitle}
118     \@mkboth{\pnheaderdefault}{\pnheaderdefault}
119   }
120 }
121 {
122   \cs_new_protected:Npn \pnheading
123   {
124     \section*{\pntitle}
125     \@mkboth{\pnheaderdefault}{\pnheaderdefault}
126   }
127 }

```

(End definition for `\pnheading`.)

format option

```
128 \tl_new:N \l__postnotes_print_format_tl
129 \keys_define:nn { postnotes/setup }
130 {
131     format .tl_set:N = \l__postnotes_print_format_tl ,
132     format .initial:n = { \small } ,
133     format .value_required:n = true ,
134 }
```

listenv option

```
135 \tl_new:N \l__postnotes_print_env_tl
136 \bool_new:N \l__postnotes_print_as_list_bool
137 \keys_define:nn { postnotes/setup }
138 {
139     listenv .code:n =
140     {
141         \tl_if_eq:nnTF {#1} { none }
142         {
143             \bool_set_false:N \l__postnotes_print_as_list_bool
144             \tl_set:Nn \l__postnotes_post_printnote_tl { \par }

```

A sensible default just in case. It should not get to be used though.

```
145         \tl_set:Nn \l__postnotes_print_env_tl { itemize }
146     }
147     {
148         \bool_set_true:N \l__postnotes_print_as_list_bool
149         \tl_set:Nn \l__postnotes_print_env_tl {#1}
150     }
151 },
152 listenv .initial:n = { postnoteslist } ,
153 listenv .value_required:n = true ,
154 }
```

A couple of built-in list environments provided for convenience, and `postnoteslist` as default. The horizontal setup of the label in these lists is based on the description environment of the standard classes (see the *The L^AT_EX Companion*).

```
155 \NewDocumentEnvironment { postnoteslist } { }
156 {
157     \list { }
158     {
159         \setlength { \leftmargin } { Opt }
160         \setlength { \labelwidth } { Opt }
161         \setlength { \itemindent } { .5\parindent }
162         \cs_set_eq:NN \makelabel \l__postnotes_list_makelabel:n
163         \setlength { \rightmargin } { Opt }
164         \setlength { \listparindent } { \parindent }
165         \setlength { \parsep } { \parskip }
166         \setlength { \itemsep } { Opt }
167         \setlength { \topsep } { .5\topsep }
168         \setlength { \partopsep } { .5\partopsep }
169     }
170 }
171 { \endlist }
172 \NewDocumentEnvironment { postnoteslisthang } { }
```

```

173 {
174   \list { }
175   {
176     \setlength { \leftmargin } { 1em }
177     \setlength { \labelwidth } { -\leftmargin }
178     \setlength { \itemindent } { -2\leftmargin }
179     \cs_set_eq:NN \makelabel \_postnotes_list_makelabel:n
180     \setlength { \rightmargin } { 0pt }
181     \setlength { \listparindent } { \parindent }
182     \setlength { \parsep } { \parskip }
183     \setlength { \itemsep } { 0pt }
184     \setlength { \topsep } { .5\topsep }
185     \setlength { \partopsep } { .5\partopsep }
186   }
187 }
188 { \endlist }
189 \cs_new:Npn \_postnotes_list_makelabel:n #1
190 { \hspace { \labelsep } \normalfont ~ #1 }

```

makemark and maketextmark options

The arguments are: #1 is the mark, #2 and #3 are, respectively, the start and the end of the backlink.

```

191 \keys_define:nn { postnotes/setup }
192 {
193   makemark .cs_set:Np = \_postnotes_make_mark:nnn #1#2#3 ,
194   makemark .value_required:n = true ,

```

From the default kernel definition of \@makefnmark.

```

195   makemark .initial:n =
196     { #2 \hbox { \@textsuperscript { \normalfont #1 } } #3 } ,
197   maketextmark .cs_set:Np = \_postnotes_make_text_mark:nnn #1#2#3 ,
198   maketextmark .value_required:n = true ,
199   maketextmark .initial:n = { #2 #1 . #3 } ,
200 }

```

pretextmark, posttextmark, postprintnote options

```

201 \tl_new:N \l__postnotes_pre_textmark_tl
202 \tl_new:N \l__postnotes_post_textmark_tl
203 \tl_new:N \l__postnotes_post_printnote_tl
204 \keys_define:nn { postnotes/setup }
205 {
206   pretextmark .tl_set:N = \l__postnotes_pre_textmark_tl ,
207   pretextmark .value_required:n = true ,
208   posttextmark .tl_set:N = \l__postnotes_post_textmark_tl ,
209   posttextmark .value_required:n = true ,
210   postprintnote .tl_set:N = \l__postnotes_post_printnote_tl ,
211   postprintnote .value_required:n = true ,
212 }

```

hyperref and backlink options

```

213 \bool_new:N \l__postnotes_hyperlink_bool
214 \bool_new:N \l__postnotes_hyperref_warn_bool

```

```

215 \bool_new:N \l__postnotes_backlink_bool
216 \keys_define:nn { postnotes/setup }
217 {
218   hyperref .choice: ,
219   hyperref / auto .code:n =
220   {
221     \bool_set_true:N \l__postnotes_hyperlink_bool
222     \bool_set_false:N \l__postnotes_hyperref_warn_bool
223   } ,
224   hyperref / true .code:n =
225   {
226     \bool_set_true:N \l__postnotes_hyperlink_bool
227     \bool_set_true:N \l__postnotes_hyperref_warn_bool
228   } ,
229   hyperref / false .code:n =
230   {
231     \bool_set_false:N \l__postnotes_hyperlink_bool
232     \bool_set_false:N \l__postnotes_hyperref_warn_bool
233   } ,
234   hyperref .initial:n = auto ,
235   hyperref .default:n = true ,
236   backlink .bool_set:N = \l__postnotes_backlink_bool ,
237   backlink .initial:n = true ,
238   backlink .default:n = true ,
239 }
240 \AddToHook { begindocument }
241 {
242   \IfPackageLoadedTF { hyperref }
243   { }
244   {
245     \bool_if:NT \l__postnotes_hyperref_warn_bool
246     { \msg_warning:nn { postnotes } { missing-hyperref } }
247     \bool_set_false:N \l__postnotes_hyperlink_bool
248   }
249   \keys_define:nn { postnotes/setup }
250   {
251     hyperref .code:n =
252     {
253       \msg_warning:nnn { postnotes }
254       { option-preamble-only } { hyperref }
255     } ,
256     backlink .code:n =
257     {
258       \msg_warning:nnn { postnotes }
259       { option-preamble-only } { backlink }
260     } ,
261   }
262 }
263 \msg_new:nnn { postnotes } { option-preamble-only }
264 { Option~'#1'~only~available~in~the~preamble~\msg_line_context:. }
265 \msg_new:nnn { postnotes } { missing-hyperref }
266 { Missing~'hyperref'~package.~Setting~'hyperref=false'. }

```


sort option

```
267 \bool_new:N \l__postnotes_sort_bool
268 \keys_define:nn { postnotes/setup }
269 {
270   sort .bool_set:N = \l__postnotes_sort_bool ,
271   sort .initial:n = true ,
272   sort .default:n = true ,
273 }
```

style option

```
274 \keys_define:nn { postnotes/setup }
275 {
276   style .choice: ,
277   style / endnotes .meta:n =
278   {
279     listenv = none ,
280     format =
281     {
282       \footnotesize
283       \setlength { \rightskip } { 0pt }
284       \setlength { \leftskip } { 0pt }
285       \setlength { \parindent } { 1.8em }
286     } ,
287     pretextmark = { \par } ,
```

endnotes uses a zero width box to get the desired alignment to the right, but that does not play well with the backlinks, so we have a little more work to do to get this right.

```
288     maketextmark =
289     {
290       \hbox_set:Nn \l_tmpa_box { \@textsuperscript { \normalfont ##1 } }
291       \skip_horizontal:n { - \box_wd:N \l_tmpa_box }
292       ##2 \box_use:N \l_tmpa_box ##3
293     } ,
294   } ,
295   style / pagenote .meta:n =
296   {
297     listenv = none ,
298     format = { } ,
299     pretextmark = { \par\noindent } ,
300     maketextmark = { { \normalfont ##2 ##1 . ##3 } } ,
301     posttextmark = { ~ } ,
302   } ,
303 }
```

\postnotesetup

\postnotesetup Provide \postnotesetup.

\postnotesetup{<options>}

```
304 \NewDocumentCommand \postnotesetup { m }
305 { \keys_set:nn { postnotes/setup } {#1} }
```

(End definition for \postnotesetup.)

4 `\postnote`

Different from the traditional `\footnotemark` / `\footnotetext` system, in the context of end notes, the functionality which corresponds to `\footnotetext` is simply to store the data to be typeset later. Hence, some of the problems that afflict footnotes do not apply to end notes. Namely, and as far as I can tell, they can be used in “inner horizontal mode” (`\mbox` etc.), and in math mode, and if the “text” will be typeset in the same page as the “mark” is of little concern.

However, the separation between “mark” and “text” is still useful in other contexts: floats and contexts where multiple typesetting passes are performed. David Carlisle and Ulrike Fischer shared some thoughts on the matter at the TeX.SX chat: <https://chat.stackexchange.com/transcript/message/60754383#60754383>.

The interesting questions here are: if they are replaceable in their roles in these contexts and how much would we lose by providing them. In analyzing this, we have to distinguish two situations: when `\footnotemark` is called with no argument (and thus steps the counter), and when it is called with the optional argument (and thus refrains from stepping the counter).

For floats, the problem they pose is that they may disturb the *ordering* of the notes. This particular issue can be solved by using `\footnotemark` without argument, and manually adjusting the counter on subsequent calls to `\footnotetext`. A good example of the technique is <https://tex.stackexchange.com/a/43694>. True, a user may wish to specify the mark explicitly, but doesn’t necessarily need to do it to solve the ordering issue.

Multiple typesetting passes of content are much harder. And they abound: the standard classes’ `\caption` typesets the caption once, if it is short, but twice if it is longer than a line; `amsmath`’s math environments perform a measuring pass before actually typesetting the equations; `amsmath`’s `\text` macro runs the contents through `\mathchoice` (which typesets the contents four times) when in math mode; `tabularx` and `tabularray` also perform measuring passes of their tables; so does `csquotes`’ blockquotes; and certainly more that I’m unaware. A number of these places offer some one or another way to mitigate the issue: `amsmath`, `tabularx`, `csquotes` and (optionally) `tabularray` restore counter values after measuring steps; `amsmath` offers a boolean to indicate when it is a measuring pass; `csquotes` offers further handles. But the standard `\caption` offers none, and neither does `amsmath`’s `\text` macro. Well, the `pkgcaption` package has can disable the multiple passes for `\caption` with the option `singlelinecheck`, but it is not reasonable to require it for our purposes, so we must assume the worst case.

Enrico Gregorio is categorical in stating that `\endnotemark` and `\endnotetext` are required for `enotez` to handle `\caption`, which apparently it didn’t offer originally: “The package should implement `\endnotemark` and `\endnotetext` for this case. According to the documentation, the author deems them to not be needed: he’s wrong.” (<https://tex.stackexchange.com/a/314937>). See also <https://tex.stackexchange.com/a/43794> and <https://tex.stackexchange.com/a/358207>.

In this scenario, when there’s no way around the multiple passes, `\footnotemark` can only handle the general case if used with an argument, precisely because it inhibits the stepping of the counter. Otherwise the counter is stepped multiple times, and we’d get the wrong number (and mark). In some circumstances, if we know the number of passes is deterministic, we might get away by adjusting the counter manually (`\caption` may be dealt with this way: if we know it to be two lines, we can decrement the counter before it and get correct results, even hyperlinked). But in cases which adjusting the counter is sufficient, end notes can be dealt with in the same way, and doesn’t need the

separation between “mark” and “text”. So, what is distinctive of the kernel’s footnote apparatus, which allows it as much flexibility as one would like, is receiving an arbitrary number as argument and not stepping the counter. And as far as `\footnotemark` and `\footnotetext` are concerned, the main point of the optional argument is really to “manually establish the relation” between the two of them. So, if *not stepping the counter* is what is needed to handle the general case, is it viable to do so? What would we lose in so doing?

When receiving an arbitrary number as argument, as the kernel functionality for footnotes and other endnotes packages do, this value is expected to be printed as such, hence it must correspond to the `postnote` counter (in our case). But this counter is in the hands of the user, and can be reset along the document, thus its uniqueness cannot be ensured. But not stepping `postnote` is perfectly viable, as it just aims at storing how the mark is to be typeset. However, not stepping the ID counter would complicate things considerably. Not doing so implies we’d lose the connection we have between the “mark” and the corresponding “text”. We might add the “text” to the queue, but all the metadata would be lost, including the `hyperref` anchor, but really the set of data without which the kind of functionality offered would be nonviable, or severely hampered. Not stepping `postnote` but stepping the ID counter also is not sufficient, because we’d get a note in duplicity. We could naively think that a gap in the ID is not a problem, and just not add the duplicate to the queue. But how could we tell the difference between a legitimate and an illegitimate step of the ID counter?

I have not been able to devise a way to “reconnect” “text” and “mark” in the absence of the unique ID counter. The most promising idea was to have mandatory arguments to `\postnotemark` and `\postnotetext` receiving a *label* which we could use to identify their counterparts, but I was not able to go through with this, and the attempts all increased complexity considerably. It is not just a label/ref system, there’s got to be a one-to-one correspondence between the sets, uniqueness has to be ensured on both sides, and there cannot be “lone” marks or texts (a bijection). Besides, this label based system of identification would have to live side-by-side with the one based on the counter. So, even if we’d have unique IDs, we wouldn’t know beforehand in what form it comes. Considering the ID is used to build the variable name in which we store the note’s information, this would also complicate things.

Besides, there are ways to get things working with multiple passes without the “mark”/“text” partition. As mentioned, there are a number of cases which offer some kind of “handle” or way to identify the multiple passes. `csquotes` has a dedicated hook that can be used. `amsmath` sets the `measuring@` boolean (which `hyperref` also defines). So, not all cases are as tricky as `\caption` or `\text`, and even that can be decently dealt with without a separation between “mark” and “text”. Besides, in difficult cases, the package offers a `nomark` option to `\postnote` to place a note, but typeset no mark. Then we can typeset a mark with `\postnoteref` referring to a `\label` in the note of interest. This would result in a correct mark without duplicity, and in a correct link from there to the note’s text at `\printpostnotes`. The drawback is that the placement of `\postnote` would be important, and results sensitive to it. All the metadata is collected at the point of `\postnote`, anchor included, not at the point of `\postnoteref`. So the consequences are a slightly off backlink, possibly imprecise metadata, etc. Considering `hyperref` itself shies away completely from linking `\footnotemark` with an argument, I’d say there’s some gain.

The truth is there are some trade-offs, there’s no “ideal” solution. Still, all in all, my judgment is that the unique ID counter is worth more than the inconveniences of an occasional `\postnote[nomark]` referenced with `\postnoteref`, and even that should not

be needed much. So, for the time being, until something else shakes this balance, I won't be offering `\postnotemark` and `\postnotetext`.

For the `hyperref` support for cross-references in `\postnote`, I've moved back and forth quite a lot. One of the ideas I fancied was using `\refstepcounter` and let `hyperref` do its job. But, since I want to have control of the anchor/destination name on both "sides", I'd have to set `\theHpostnote` locally before calling `\refstepcounter`, otherwise results might be sensitive to user calls to `\counterwithin` (see <https://github.com/latex3/hyperref/issues/230>, thanks Ulrike Fischer). However, even if that worked well for the default case, we still had to setup things manually, in case of a manually supplied mark. All in all, I'm just calling `\stepcounter`, setting the relevant cross-reference variables once and setting the anchor manually.

`postnote` is the public, user facing, counter for `\postnote`. It determines how the note's mark gets to be typeset. It can be reset, set, and have its printed representation changed. Of course, whether those are meaningful is up to the user.

```
306 \newcounter { postnote }
```

`\g__postnotes_note_id_int` `\g__postnotes_note_id_int` is the internal, unique counter which provides the ID number of each note. It ties "mark" and "text" together, is also the connection between each note and its data, including the content, which is stored in a property list named according to `__postnotes_data_name:n` and the ID number. `\l_postnotes_note_id_tl` is a convenience variable storing the counter's value. `\g__postnotes_queue_seq` stores the sequence of notes' IDs to be processed by the next call of `\printpostnotes`.

```
307 \int_new:N \g__postnotes_note_id_int
308 \tl_new:N \l_postnotes_note_id_tl
309 \tl_set:Nn \l_postnotes_note_id_tl { \int_use:N \g__postnotes_note_id_int }
310 \seq_new:N \g__postnotes_queue_seq
```

(End definition for `\g__postnotes_note_id_int`, `\l_postnotes_note_id_tl`, and `\g__postnotes_queue_seq`. This function is documented on page ??.)

`\postnote` Provide `\postnote`.

```
\postnote [<options>] {\<note text>}
```

```
311 \NewDocumentCommand \postnote { 0 { } +m }
312 { \__postnotes_note:nn {#1} {#2} }
```

(End definition for `\postnote`.)

`__postnotes_note:nn` The internal version of `\postnote`. The `postnotes/note/begin` hook is meant to provide a place from where some additional setup for the note can be performed. This is being used for adding support for some features/packages, but can also be used, for example, to add an extra local property for `zref`.

```
\__postnotes_note:nn [<options>] {\<note content>}
```

```

313 \NewHook { postnotes/note/begin }
314 \cs_new_protected:Npn \__postnotes_note:nn #1#2
315 {
316   \group_begin:
317   \keys_set:nn { postnotes/note } {#1}
318   \__postnotes_inhibit_note:F
319   {
320     \int_gincr:N \g__postnotes_note_id_int
321     \tl_if_empty:NT \l__postnotes_mark_tl
322     {
323       \stepcounter { postnote }
324       \tl_set:Nx \l__postnotes_mark_tl { \thepostnote }
325     }
326     \seq_gput_right:Nx \g__postnotes_queue_seq
327     { \l_postnotes_note_id_tl }
328     \UseHook { postnotes/note/begin }
329     \cs_set:Npn \@currentcounter { postnote }
330     \cs_set:Npx \@currentlabel { \p@postnote \l__postnotes_mark_tl }
331     \MakeLinkTarget* { postnote. \l_postnotes_note_id_tl .mark }
332     \__postnotes_set_mark_page_label:x { \l_postnotes_note_id_tl }
333     \__postnotes_set_user_labels:
334     \bool_if:NF \l__postnotes_nomark_bool
335     {
336       \__postnotes_typeset_mark:xV
337       { \l_postnotes_note_id_tl } \l__postnotes_mark_tl
338     }
339     \__postnotes_store:nn { \l_postnotes_note_id_tl } {#2}
340   }
341   \group_end:
342 }

```

(End definition for __postnotes_note:nn.)

Options for \postnote.

```

343 \tl_new:N \l__postnotes_mark_tl
344 \bool_new:N \l__postnotes_nomark_bool
345 \fp_new:N \l__postnotes_sort_num_fp
346 \tl_new:N \l__postnotes_note_label_tl
347 \bool_new:N \l__postnotes_manual_sortnum_bool
348 \bool_new:N \l__postnotes_maybe_multi_bool
349 \keys_define:nn { postnotes/note }
350 {
351   markstr .tl_set:N = \l__postnotes_mark_tl ,
352   markstr .value_required:n = true ,
353   sortnum .code:n =
354   {
355     \fp_set:Nn \l__postnotes_sort_num_fp {#1}
356     \bool_set_true:N \l__postnotes_manual_sortnum_bool
357   } ,
358   sortnum .value_required:n = true ,
359   mark .meta:n =
360   {
361     markstr = {#1} ,
362     sortnum = {#1} ,

```

```

363     } ,
364     mark .value_required:n = true ,
365     nomark .bool_set:N = \l__postnotes_nomark_bool ,
366     nomark .default:n = true ,
367     label .tl_set:N = \l__postnotes_note_label_tl ,
368     label .value_required:n = true ,
369 }

```

`__postnotes_inhibit_note:TF` In contexts of multiple passes of content, it may be needed, or preferred, to inhibit the note altogether to avoid side effects and duplicity. This conditional, obviously, will always return the true branch unless something is done in the `postnotes/note/inhibit` hook. This hook is meant to handle support for packages or features which may justify note inhibition, and the code there should set `\l__postnotes_inhibit_note_bool` and `\l__postnotes_print_plain_mark_bool` as appropriate to the case.

```

370 \bool_new:N \l__postnotes_inhibit_note_bool
371 \bool_new:N \l__postnotes_print_plain_mark_bool
372 \NewHook { postnotes/note/inhibit }
373 \prg_new_protected_conditional:Npnn \__postnotes_inhibit_note: { F }
374 {
375     \bool_set_false:N \l__postnotes_inhibit_note_bool
376     \bool_set_false:N \l__postnotes_print_plain_mark_bool
377     \UseHook { postnotes/note/inhibit }

```

Printing a plain mark here may be needed because, if we are inhibiting the note in a “measuring context” and omit it completely, the measuring being performed will be off by the size of the mark. So, to ensure the measuring can be done correctly, we place the mark. Since we’d only print this mark in case of inhibition, when we don’t actually step the counter, to typeset correctly the mark that would be printed if the counter had been stepped, we increment `\c@postnote` locally and grouped, and smuggle `\thepostnote` out of the group.

```

378     \bool_if:NT \l__postnotes_print_plain_mark_bool
379     {
380         \tl_if_empty:NT \l__postnotes_mark_tl
381         {
382             \group_begin:
383             \int_incr:N \c@postnote
384             \exp_args:NNNx
385             \group_end:
386             \tl_set:Nn \l__postnotes_mark_tl { \thepostnote }
387         }
388         \__postnotes_typeset_mark_wrapper:n
389         { \__postnotes_make_mark:nnn { \l__postnotes_mark_tl } { } { } }
390     }
391     \bool_if:NTF \l__postnotes_inhibit_note_bool
392     { \prg_return_true: }
393     { \prg_return_false: }
394 }

```

(End definition for `__postnotes_inhibit_note:TF`.)

`__postnotes_typeset_mark:nn` Auxiliary functions for mark typesetting in `__postnotes_note:nn`. `__postnotes_typeset_mark_wrapper:n` is based on the definition of `\@footnotemark` in the kernel.

```

    \_postnotes_typeset_mark:nn {\<note id>} {\<mark>}}
    \_postnotes_typeset_mark_wrapper:n {\<mark>}}

395 \cs_new_protected:Npn \_postnotes_typeset_mark:nn #1#2
396 {
397   \_postnotes_typeset_mark_wrapper:n
398   {
399     \bool_if:NTF \l__postnotes_hyperlink_bool
400     {
401       \_postnotes_make_mark:nnn {#2}
402       { \hyper@linkstart { link } { postnote. #1 .text } }
403       { \hyper@linkend }
404     }
405     { \_postnotes_make_mark:nnn {#2} { } { } }
406   }
407 }
408 \cs_generate_variant:Nn \_postnotes_typeset_mark:nn { xV }
409 \tl_new:N \l__postnotes_saved_spacefactor_tl
410 \cs_new_protected:Npn \_postnotes_typeset_mark_wrapper:n #1
411 {
412   \mode_leave_vertical:
413   \mode_if_horizontal:T
414   {
415     \tl_set:Nx \l__postnotes_saved_spacefactor_tl { \the\spacefactor }
416     \nobreak
417   }
418   #1
419   \mode_if_horizontal:T
420   { \spacefactor \l__postnotes_saved_spacefactor_tl }
421   \scan_stop:
422 }

```

(End definition for _postnotes_typeset_mark:nn and _postnotes_typeset_mark_wrapper:n.)

_postnotes_set_user_labels: Auxiliary function for user label setting in _postnotes_note:nn. Supports the label and zlabel options of \postnote.

```

423 \cs_new_protected:Npn \_postnotes_set_user_labels:
424 {
425   \tl_if_empty:NF \l__postnotes_note_label_tl
426   { \exp_args:NV \label \l__postnotes_note_label_tl }
427   \tl_if_empty:NF \l__postnotes_note_zlabel_tl
428   { \exp_args:NV \zlabel \l__postnotes_note_zlabel_tl }
429 }

```

(End definition for _postnotes_set_user_labels:.)

5 \postnoteref

\postnoteref Provide \postnoteref.

```

    \postnoteref{*}{\<label>}

430 \NewDocumentCommand \postnoteref { s m }
431 { \_postnotes_note_ref:nn {#1} {#2} }

```

(End definition for \postnoteref.)

_postnotes_note_ref:nn The internal version of \postnoteref.

```

    \_postnotes_note_ref:nn {\star bool} {\label}

432 \cs_new_protected:Npn \_postnotes_note_ref:nn #1#2
433 {
434   \group_begin:
435   \_postnotes_typeset_mark_wrapper:n
436   {
437     \bool_lazy_and:nnTF
438     { ! #1 }
439     { \l__postnotes_hyperlink_bool }
440     {
441       \hyperref [#2]
442       { \_postnotes_make_mark:nnn { \ref*{#2} } { } { } }
443     }
444     { \_postnotes_make_mark:nnn { \ref*{#2} } { } { } }
445   }
446   \group_end:
447 }

```

(End definition for _postnotes_note_ref:nn.)

6 \postnotesection

\postnotesection Provide \postnotesection and \postnotesectionx.
\postnotesectionx

```

    \postnotesection[<options>]{<section content>}
    \postnotesectionx[<options>]{<section content>}

448 \NewDocumentCommand \postnotesection { 0 { } +m }
449 { \_postnotes_section:nn {#1} {#2} }
450 \NewDocumentCommand \postnotesectionx { 0 { } +m }
451 {
452   % NOTE Command deprecated in 2022-12-27 for v0.2.0.
453   \msg_warning:nn { postnotes } { postnotesectionx-deprecated }
454   \postnotesection [ #1 , exp ] {#2}
455 }
456 \msg_new:nnn { postnotes } { postnotesectionx-deprecated }
457 {
458   '\iow_char:N\postnotesectionx'~is~deprecated~\msg_line_context:~
459   Use~the~'exp'~option~of~'\iow_char:N\postnotesection'~instead.
460 }

```

(End definition for \postnotesection and \postnotesectionx.)

_postnotes_section:nn The internal version of \postnotesection.

```

    \_postnotes_section:nn {\options} {\content}

```



```

461 \int_new:N \g__postnotes_sectid_int
462 \cs_new_protected:Npn \__postnotes_section:nn #1#2
463 {
464   \group_begin:
465   \int_gincr:N \g__postnotes_sectid_int
466   \int_gincr:N \g__postnotes_note_id_int
467   \seq_gput_right:Nx \g__postnotes_queue_seq { \l_postnotes_note_id_tl }
468   \tl_gclear:N \g__postnotes_section_name_tl
469   \keys_set:nn { postnotes/section } {#1}
470   \bool_if:NTF \l__postnotes_section_exp_bool
471     { \__postnotes_store_section:nx { \l_postnotes_note_id_tl } {#2} }
472     { \__postnotes_store_section:nn { \l_postnotes_note_id_tl } {#2} }
473   \group_end:
474 }

```

(End definition for `__postnotes_section:nn`.)

Options for `\postnotessection`. Actually, I would have preferred to use “label” for the `name` option, but I feared I might need it further down the road for the traditional meaning.

```

475 \tl_new:N \g__postnotes_section_name_tl
476 \bool_new:N \l__postnotes_section_exp_bool
477 \keys_define:nn { postnotes/section }
478 {
479   name .tl_gset:N = \g__postnotes_section_name_tl ,
480   name .value_required:n = true ,
481   exp .bool_set:N = \l__postnotes_section_exp_bool ,
482   exp .initial:n = false ,
483   exp .default:n = true ,
484 }

```

7 `\printpostnotes`

`\printpostnotes` Provide `\printpostnotes`.

`\printpostnotes`

```

485 \NewDocumentCommand \printpostnotes { }
486 { \__postnotes_print_notes: }

```

(End definition for `\printpostnotes`.)

`\pnthechapter` User facing variables, aimed at making available some of the notes’ and sections’ metadata for the user at specific contexts.

```

\pnthechapternextnote 487 \tl_new:N \pnthechapter
\pnthesectionnextnote 488 \tl_new:N \pnthesection
\pnthepage             489 \tl_new:N \pnthechapternextnote
                      490 \tl_new:N \pnthesectionnextnote
                      491 \tl_new:N \pnthepage

```

(End definition for `\pnthechapter` and others.)

```

\g__postnotes_print_postnotes_int
\l__postnotes_print_note_id_tl
\l__postnotes_print_note_id_next_tl
\l__postnotes_print_counter_tl
\l__postnotes_print_mark_tl
\l__postnotes_print_type_curr_tl
\l__postnotes_print_type_next_tl
\l__postnotes_print_type_prev_tl
\l__postnotes_print_content_tl
\l__postnotes_clear_queue_seq

Auxiliary variables for \__postnotes_print_notes:.
492 \int_new:N \g__postnotes_print_postnotes_int
493 \tl_new:N \l__postnotes_print_note_id_tl
494 \tl_new:N \l__postnotes_print_note_id_next_tl
495 \tl_new:N \l__postnotes_print_counter_tl
496 \tl_new:N \l__postnotes_print_mark_tl
497 \tl_new:N \l__postnotes_print_type_curr_tl
498 \tl_new:N \l__postnotes_print_type_next_tl
499 \tl_new:N \l__postnotes_print_type_prev_tl
500 \tl_new:N \l__postnotes_print_content_tl
501 \seq_new:N \l__postnotes_clear_queue_seq

```

(End definition for `\g__postnotes_print_postnotes_int` and others. This function is documented on page ??.)

`__postnotes_print_notes:` hooks. Both meant at providing points of entry for additional setup, specially to add support to packages and features which require it. The `postnotes/print/begin` hook is run early in `__postnotes_print_notes:` and only once per call, after the user options have been processed. The `postnotes/print/note/begin` hook is run once for each note, at the point where environment variables are being set or restored, before the typesetting of either the mark or the text, but within a group of its own of each note.

```

502 \NewHook { postnotes/print/begin }
503 \NewHook { postnotes/print/note/begin }

```

The `postnotetext` is a counter used to restore the original value of `postnote` at the time of printing, for the purposes of cross-referencing, it should be different from `postnote` if a note may occur inside `\printpostnotes`. The `postnotessection` is a counter which is stepped for every postnote section which gets to be actually typeset. It's aim is to provide a valid “enclosing counter” to `postnote` in the context of `\printpostnotes`. Since we don't know where `postnote` may have been reset along the document, in the general case, we can't rely on any other preexisting counter. This means that the particular value of `postnotessection` is of little practical meaning, it really is just meant to provide recognizable “bounds” for `postnote` along the printing of the notes. Indeed, it is initialized to a very high value (larger than the conceivable number of postnote sections in a document), so that “marks” and “texts” don't mix in the same reference list, which would occur if the enclosing counters of both belonged to the same range, and with somewhat arbitrary results, since we cannot ensure the step of the enclosing counter along the document matches `postnotessection`. This is actually a tricky problem from the cross-referencing standpoint: two different things, which should be of the same type, are reset along the document, but shouldn't really be mixed together. They are both $\text{\LaTeX} 2_{\epsilon}$ counters, since they may be required externally. Their main intended use case is to support `zref-clever`, but in principle they can be of general use.

```

504 \newcounter { postnotetext }
505 \newcounter { postnotessection }
506 \setcounter { postnotessection } { 10000 }

```

`__postnotes_print_notes:` The internal version of `\printpostnotes`.

```

\__postnotes_print_notes:

```

```

507 \cs_new_protected:Npn \__postnotes_print_notes:
508 {
509   \group_begin:
510   \int_gincr:N \g__postnotes_print_postnotes_int
511   \seq_if_empty:NTF \g__postnotes_queue_seq
512     { \msg_warning:nn { postnotes } { empty-printpostnotes } }
513   {
514     \pnheading
515     \UseHook { postnotes/print/begin }
516     \tl_set:Nn \l__postnotes_print_type_prev_tl { open }
517     \seq_set_eq:NN \l__postnotes_clear_queue_seq \g__postnotes_queue_seq
518     \__postnotes_verify_multipass:N \g__postnotes_queue_seq
519     \bool_if:NT \l__postnotes_sort_bool
520       { \__postnotes_sort_queue:N \g__postnotes_queue_seq }
521     \bool_gset_true:N \g__postnotes_header_vars_next_bool
522     \__postnotes_get_headers_data:N \g__postnotes_queue_seq
523     \__postnotes_set_headers_vars_first:

```

Ensure the first note after a heading has paragraph indentation when `listenv` is `none`. `endnotes` uses a workaroundsish solution in `\enoteheading`, setting a box and then skipping back a line. Enrico Gregorio is correct, though, in criticizing it at https://tex.stackexchange.com/q/575905#comment1450213_575915, and suggests the use of `\@afterindenttrue`, which is what `indentfirst` does (we do the same, just locally).

```

524     \bool_if:NF \l__postnotes_print_as_list_bool
525     {
526       \cs_set_eq:NN \@afterindentfalse \@afterindenttrue
527       \@afterindenttrue
528     }
529     \bool_until_do:nn { \seq_if_empty_p:N \g__postnotes_queue_seq }
530     {
531       \seq_gpop_left:NN \g__postnotes_queue_seq
532       \l_postnotes_print_note_id_tl
533       \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
534       { type } \l__postnotes_print_type_curr_tl
535       \tl_if_eq:NnTF \l__postnotes_print_type_curr_tl { section }
536       { % type_curr = 'section'
537         \seq_if_empty:NTF \g__postnotes_queue_seq
538         {
539           \tl_set:Nn \l__postnotes_print_note_id_next_tl { noid }
540           \tl_set:Nn \l__postnotes_print_type_next_tl { close }
541         }
542         {
543           \seq_get_left:NN \g__postnotes_queue_seq
544           \l__postnotes_print_note_id_next_tl
545           \__postnotes_prop_get:nnN
546           { \l__postnotes_print_note_id_next_tl }
547           { type } \l__postnotes_print_type_next_tl
548         }

```

We only process the entry if `type_next` is `note`: here are skipped empty sections.

```

549     \tl_if_eq:NnT \l__postnotes_print_type_next_tl { note }
550     {
551       \stepcounter { postnotessection }
552       \group_begin:

```

```

553         \_postnotes_prop_get:nnN
554         { \l_postnotes_print_note_id_tl }
555         { thechapter } \pnthechapter
556     \_postnotes_prop_get:nnN
557     { \l_postnotes_print_note_id_tl }
558     { thesection } \pnthesection
559     \_postnotes_prop_get:nnN
560     { \l__postnotes_print_note_id_next_tl }
561     { thechapter } \pnthechapternextnote
562     \_postnotes_prop_get:nnN
563     { \l__postnotes_print_note_id_next_tl }
564     { thesection } \pnthesectionnextnote
565     \_postnotes_prop_get:nnN
566     { \l_postnotes_print_note_id_tl }
567     { content } \l__postnotes_print_content_tl
568     \l__postnotes_print_content_tl
569     \group_end:

```

Set `type_prev` for the next iteration.

```

570         \tl_set:NV \l__postnotes_print_type_prev_tl
571         \l__postnotes_print_type_curr_tl
572     }
573 }
574 { % type_curr = 'note'
575   \tl_if_eq:NnF \l__postnotes_print_type_prev_tl { note }
576   {
577     \bool_if:NTF \l__postnotes_print_as_list_bool
578     { \exp_args:Nx \begin { \l__postnotes_print_env_tl } }
579     { \group_begin: }
580     \l__postnotes_print_format_tl
581   }
582   \group_begin:
583   \UseHook { postnotes/print/note/begin }
584   \_postnotes_get_pageref:Nx \pnthepage
585   { mark@ \l_postnotes_print_note_id_tl }
586   \_postnotes_prop_get:nnN
587   { \l_postnotes_print_note_id_tl }
588   { mark } \l__postnotes_print_mark_tl
589   \_postnotes_prop_get:nnN
590   { \l_postnotes_print_note_id_tl }
591   { counter } \l__postnotes_print_counter_tl
592   \_postnotes_prop_get:nnN
593   { \l_postnotes_print_note_id_tl }
594   { content } \l__postnotes_print_content_tl
595   \cs_set:Npn \@currentcounter { postnotetext }
596   \int_set:Nn \c@postnotetext
597   { \l__postnotes_print_counter_tl }
598   \cs_set:Npx \@currentlabel
599   { \p@postnote \l__postnotes_print_mark_tl }
600   \_postnotes_text_mark_wrapper:n
601   {
602     \MakeLinkTarget*
603     { postnote. \l_postnotes_print_note_id_tl .text }
604     \_postnotes_set_text_page_label:x
605     { \l_postnotes_print_note_id_tl }

```

```

606         \__postnotes_typeset_text_mark:eV
607         { \l__postnotes_print_note_id_tl }
608         \l__postnotes_print_mark_tl
609     }

```

Leave vertical mode to avoid “perhaps a missing \item” error.

```

610         \mode_leave_vertical:
611         \l__postnotes_print_content_tl
612         \l__postnotes_post_printnote_tl
613     \group_end:

```

For notes, query for next note’s type after the current note was typeset, to handle possible nesting. Even if nesting is not a feature, this should avoid hard crashes related to “lonely \item” or “extra \endgroup” errors, in case it occurs.

```

614         \seq_if_empty:NTF \g__postnotes_queue_seq
615         {
616             \tl_set:Nn \l__postnotes_print_note_id_next_tl { noid }
617             \tl_set:Nn \l__postnotes_print_type_next_tl { close }
618         }
619         {
620             \seq_get_left:NN \g__postnotes_queue_seq
621             \l__postnotes_print_note_id_next_tl
622             \__postnotes_prop_get:nnN
623             { \l__postnotes_print_note_id_next_tl }
624             { type } \l__postnotes_print_type_next_tl
625         }
626         \tl_if_eq:NnF \l__postnotes_print_type_next_tl { note }
627         {
628             \bool_if:NTF \l__postnotes_print_as_list_bool
629             { \exp_args:Nx \end { \l__postnotes_print_env_tl } }
630             { \group_end: }
631         }

```

Set `type_prev` for the next iteration.

```

632         \tl_set:NV \l__postnotes_print_type_prev_tl
633         \l__postnotes_print_type_curr_tl
634     }
635 }
636 \AddToHookNext { shipout/after }
637 { \bool_gset_false:N \g__postnotes_header_vars_next_bool }

```

We won’t use the variables anymore, clear them to reduce memory usage. Given how we populated `\l__postnotes_clear_queue_seq`, this won’t catch nested notes. But it’s not worth to conditionally add new items along the way (testing it every iteration) for this. Again, not a feature.

```

638         \seq_map_inline:Nn \l__postnotes_clear_queue_seq
639         { \__postnotes_prop_gclear:n { ##1 } }
640     }
641     \group_end:
642 }

```

(End definition for `__postnotes_print_notes:.`)

```

643 \msg_new:nnn { postnotes } { empty-printpostnotes }
644 { Empty~'\iow_char:N\printpostnotes'~\msg_line_context:. }

```

```

\__postnotes_typeset_text_mark:nn
\__postnotes_text_mark_wrapper:n

\__postnotes_typeset_text_mark:nn {<note id>} {<mark>}
\__postnotes_text_mark_wrapper:n {<mark>}

645 \cs_new_protected:Npn \__postnotes_typeset_text_mark:nn #1#2
646 {
647   \bool_lazy_and:nnTF
648     { \l__postnotes_hyperlink_bool }
649     { \l__postnotes_backlink_bool }
650   {
651     \__postnotes_make_text_mark:nnn {#2}
652     { \hyper@linkstart { link } { postnote. #1 .mark } }
653     { \hyper@linkend }
654   }
655   { \__postnotes_make_text_mark:nnn {#2} { } { } }
656 }
657 \cs_generate_variant:Nn \__postnotes_typeset_text_mark:nn { eV }
658 \cs_new_protected:Npn \__postnotes_text_mark_wrapper:n #1
659 {
660   \bool_if:NTF \l__postnotes_print_as_list_bool
661     { \item [ \l__postnotes_pre_textmark_tl #1 \l__postnotes_post_textmark_tl ] }
662     { \l__postnotes_pre_textmark_tl #1 \l__postnotes_post_textmark_tl }
663 }

```

(End definition for `__postnotes_typeset_text_mark:nn` and `__postnotes_text_mark_wrapper:n`.)

Print auxiliary

`__postnotes_verify_multipass:N` provides a general procedure for handling cases of multiple passes of content. Ideally, the job should be done at `__postnotes_inhibit_note:F`, if at all possible. But, failing that, we can rely on the fact that `\postnotes` of measuring/trial passes don't end up being output and hence don't generate labels in the `.aux` file. This is the equivalent for `postnotes` to the effect of write restrictions for the packages based on external files, which is how they actually handle these cases. However, despite this being a general test, and a reasonable one, I'd like to restrain its use to the minimum possible. First, using this criterion across the board would result in large swings on the content of `\printpostnotes` and spurious warnings in an initial compilation since the labels are not available on the first run. Second, I'd prefer not to interfere with the queue, unless we really need to. Hence, we only apply this check for "eligible" items. For signaling this eligibility, the note must have been stored with the `\l__postnotes_maybe_multi_bool` set to `true`, which is then saved in the `multibool` property. One implication of this procedure is that, if there are any new notes marked as `multibool`, three rounds of compilation will be needed, since the labels of the printed notes will be written only on the second run and the document will thus require a third one to stabilize.

```

\__postnotes_verify_multipass:N

\__postnotes_verify_multipass:N <\g__postnotes_queue_seq>

664 \cs_new_protected:Npn \__postnotes_verify_multipass:N #1
665 {
666   \group_begin:
667   \seq_clear:N \l_tmpa_seq

```

```

668 \seq_map_inline:Nn #1
669 {
670   \__postnotes_prop_get:nnN {##1} { multibool } \l_tmpa_tl
671   \tl_if_eq:NnTF \l_tmpa_tl { true }
672   {
673     \cs_if_exist:cT
674     { \c__postnotes_ref_prefix_tl @ mark@ ##1 }
675     { \seq_put_right:Nn \l_tmpa_seq {##1} }
676   }
677   { \seq_put_right:Nn \l_tmpa_seq {##1} }
678 }
679 \seq_gset_eq:NN #1 \l_tmpa_seq
680 \group_end:
681 }

```

(End definition for __postnotes_verify_multipass:N.)

__postnotes_sort_queue:N Sorting function for __postnotes_print_notes:.

```

\__postnotes_sort_queue:N (\g__postnotes_queue_seq)
682 \cs_new_protected:Npn \__postnotes_sort_queue:N #1
683 {
684   \group_begin:
685   \seq_gsort:Nn #1
686   {
687     \__postnotes_prop_get:nnN {##1} { pnsectid } \l_tmpa_tl
688     \__postnotes_prop_get:nnN {##2} { pnsectid } \l_tmpb_tl
689     \tl_if_eq:NNTF \l_tmpa_tl \l_tmpb_tl
690     {
691       \__postnotes_prop_get:nnN {##1} { type } \l_tmpa_tl
692       \__postnotes_prop_get:nnN {##2} { type } \l_tmpb_tl
693       \bool_lazy_and:nnTF
694       { \str_if_eq_p:Vn \l_tmpa_tl { note } }
695       { \str_if_eq_p:Vn \l_tmpb_tl { note } }
696       {
697         \__postnotes_prop_get:nnN {##1} { sortnum } \l_tmpa_tl
698         \__postnotes_prop_get:nnN {##2} { sortnum } \l_tmpb_tl
699         \fp_compare:nNnTF { \l_tmpa_tl } > { \l_tmpb_tl }
700         { \sort_return_swapped: }
701         { \sort_return_same: }
702       }
703       { \sort_return_same: }
704     }
705     { \sort_return_same: }
706   }
707   \group_end:
708 }

```

(End definition for __postnotes_sort_queue:N.)

8 Headers

The headers infrastructure of postnotes is comprised of three basic parts:

1. For each `\postnote`, labels are set storing the `page` where the note occurs. Each note actually generates a pair of such labels, once when `\postnote` is called (with the mark), and another where the note is printed (in `\printpostnotes`). The former ones store `\thepage`, since we want the printed representation of it for typesetting purposes, the latter ones store the value of the `page` counter, since we don't need to typeset it, but do need to perform algebraic operations with it. These labels are set by `__postnotes_set_mark_page_label:n`, `__postnotes_set_text_page_label:n`, and `__postnotes_set_print_page_label:n` at the appropriate places. The set of these labels provides a mapping from each note's "mark" and "text" to the page where it occurs.
2. This information set is processed by `__postnotes_get_headers_data:N` at the beginning of `__postnotes_print_notes:` to identify the first and last note of each page in `\printpostnotes`, and to generate a mapping from these first and last notes on each page to the pages where their corresponding marks occur. We also take the opportunity to enrich this mapping with other metadata of each note. So we get also mappings from the first and last note on each page to `\thechapter`, `\thesection`, and the `name` of the section in which they occur. These mappings are stored in property lists `\g__postnotes_header_{info}_first_prop` and `\g__postnotes_header_{info}_last_prop` where the `key` is the page in `\printpostnotes` where their note's content is typeset (or rather where it starts to be typeset, it is the page where the text's mark is printed).
3. Based on these mappings, along the span of notes section we run `__postnotes_set_headers_vars_next:` at each `shipout/before` hook to set user facing variables for the *next* page, which will be available when their heading gets typeset. Given that at `shipout` we can rely on a correct value of the `page` counter, we use it as `key` to query the property lists generated in the previous step. These user facing variables are called `\pnhd{info}first` and `\pnhd{info}last`. Since we cannot rely on the shipout hook for the first page of `\printpostnotes`, `__postnotes_set_headers_vars_first:` is run at its beginning to ensure correct values are in place on the first page of the notes section.

These `\pnhd{info}first` and `\pnhd{info}last` variables can then be used to build simple functions which can be passed to mark commands to achieve rich contextual running headers.

<code>\pnhdpagefirst</code> <code>\pnhdpagelast</code> <code>\pnhdchapfirst</code> <code>\pnhdchaplast</code> <code>\pnhdsectfirst</code> <code>\pnhdsectlast</code> <code>\pnhdnamefirst</code> <code>\pnhdnamelast</code>	User facing variables, aimed at making available header data for the user. Setting these variables with correct values at the moment the header gets typeset is <i>the</i> objective of the whole headers infrastructure of the package. <div style="font-family: monospace; font-size: 0.8em;"> 709 \tl_new:N \pnhdpagefirst 710 \tl_new:N \pnhdpagelast 711 \tl_new:N \pnhdchapfirst 712 \tl_new:N \pnhdchaplast 713 \tl_new:N \pnhdsectfirst 714 \tl_new:N \pnhdsectlast 715 \tl_new:N \pnhdnamefirst 716 \tl_new:N \pnhdnamelast </div>
--	---

(End definition for `\pnhdpagefirst` and others.)

\g__postnotes_header_page_first_prop	
\g__postnotes_header_page_last_prop	717 \prop_new:N \g__postnotes_header_page_first_prop
\g__postnotes_header_chap_first_prop	718 \prop_new:N \g__postnotes_header_page_last_prop
\g__postnotes_header_chap_last_prop	719 \prop_new:N \g__postnotes_header_chap_first_prop
\g__postnotes_header_sect_first_prop	720 \prop_new:N \g__postnotes_header_chap_last_prop
\g__postnotes_header_sect_last_prop	721 \prop_new:N \g__postnotes_header_sect_first_prop
\g__postnotes_header_name_first_prop	722 \prop_new:N \g__postnotes_header_sect_last_prop
\g__postnotes_header_name_last_prop	723 \prop_new:N \g__postnotes_header_name_first_prop
\g__postnotes_header_prev_last_page_tl	724 \prop_new:N \g__postnotes_header_name_last_prop
\g__postnotes_header_prev_last_chap_tl	725 \tl_new:N \g__postnotes_header_prev_last_page_tl
\g__postnotes_header_prev_last_sect_tl	726 \tl_new:N \g__postnotes_header_prev_last_chap_tl
\g__postnotes_header_prev_last_name_tl	727 \tl_new:N \g__postnotes_header_prev_last_sect_tl
\l__postnotes_prev_text_page_tl	728 \tl_new:N \g__postnotes_header_prev_last_name_tl
\l__postnotes_curr_text_page_tl	729 \tl_new:N \l__postnotes_prev_text_page_tl
\l__postnotes_prev_mark_page_tl	730 \tl_new:N \l__postnotes_curr_text_page_tl
\l__postnotes_prev_mark_chap_tl	731 \tl_new:N \l__postnotes_prev_mark_page_tl
\l__postnotes_prev_mark_sect_tl	732 \tl_new:N \l__postnotes_prev_mark_chap_tl
\l__postnotes_prev_mark_name_tl	733 \tl_new:N \l__postnotes_prev_mark_sect_tl
	734 \tl_new:N \l__postnotes_prev_mark_name_tl

Auxiliary variables for the headers infrastructure.

(End definition for \g__postnotes_header_page_first_prop and others.)

__postnotes_get_headers_data:N	Process header data for __postnotes_set_headers_vars:n.
---------------------------------	--

```

\__postnotes_get_headers_data:N <\g__postnotes_queue_seq>

735 \cs_new_protected:Npn \__postnotes_get_headers_data:N #1
736 {
737   \group_begin:
738   \tl_gclear:N \pnhdpagefirst
739   \tl_gclear:N \pnhdpagelast
740   \tl_gclear:N \pnhdchapfirst
741   \tl_gclear:N \pnhdchaplast
742   \tl_gclear:N \pnhdsectfirst
743   \tl_gclear:N \pnhdsectlast
744   \tl_gclear:N \pnhdnamefirst
745   \tl_gclear:N \pnhdnamelast
746   \prop_gclear:N \g__postnotes_header_page_first_prop
747   \prop_gclear:N \g__postnotes_header_page_last_prop
748   \prop_gclear:N \g__postnotes_header_chap_first_prop
749   \prop_gclear:N \g__postnotes_header_chap_last_prop
750   \prop_gclear:N \g__postnotes_header_sect_first_prop
751   \prop_gclear:N \g__postnotes_header_sect_last_prop
752   \prop_gclear:N \g__postnotes_header_name_first_prop
753   \prop_gclear:N \g__postnotes_header_name_last_prop
754   \tl_gclear:N \g__postnotes_header_prev_last_page_tl
755   \tl_gclear:N \g__postnotes_header_prev_last_chap_tl
756   \tl_gclear:N \g__postnotes_header_prev_last_sect_tl
757   \tl_gclear:N \g__postnotes_header_prev_last_name_tl
758   \tl_clear:N \l__postnotes_prev_text_page_tl
759   \tl_clear:N \l__postnotes_curr_text_page_tl
760   \tl_clear:N \l__postnotes_prev_mark_page_tl
761   \tl_clear:N \l__postnotes_prev_mark_chap_tl
762   \tl_clear:N \l__postnotes_prev_mark_sect_tl

```

```

763 \tl_clear:N \l__postnotes_prev_mark_name_tl
764 \seq_map_inline:Nn #1
765 {
766   \exp_args:Nx \tl_if_eq:nnT
767   { \__postnotes_prop_item:nn {##1} { type } }
768   { note }
769   {
770     \__postnotes_get_pageref:Nn
771     \l__postnotes_curr_text_page_tl { text@ ##1 }
772     \tl_if_empty:NF \l__postnotes_curr_text_page_tl
773     {
774       \tl_if_eq:NNTF
775       \l__postnotes_prev_text_page_tl
776       \l__postnotes_curr_text_page_tl
777       {

```

We are on the same page as the previous note, just update the `prev_mark` data.

```

778       \__postnotes_get_pageref:Nn
779       \l__postnotes_prev_mark_page_tl { mark@ ##1 }
780       \__postnotes_prop_get:nnN {##1} { thechapter }
781       \l__postnotes_prev_mark_chap_tl
782       \__postnotes_prop_get:nnN {##1} { thesection }
783       \l__postnotes_prev_mark_sect_tl
784       \__postnotes_prop_get:nnN {##1} { pnsectname }
785       \l__postnotes_prev_mark_name_tl
786     }
787   {

```

We are on the transition between two pages, current ID is the first note of the new page (or on the very first note of `\printpostnotes`, given `\l__postnotes_prev_text_page_tl` is initialized to empty).

Set ‘last’ values for previous page, based on the last valid `prev_mark` stored ones. There is no previous page to the first one of `\printpostnotes`, so we don’t set ‘last’ values for it (conditioning on `\l__postnotes_prev_text_page_tl` being empty, which only occurs on the first note).

```

788       \tl_if_empty:NF \l__postnotes_prev_text_page_tl
789       {
790         \prop_gput:Nxx \g__postnotes_header_page_last_prop
791         { \l__postnotes_prev_text_page_tl }
792         { \l__postnotes_prev_mark_page_tl }
793         \prop_gput:Nxx \g__postnotes_header_chap_last_prop
794         { \l__postnotes_prev_text_page_tl }
795         { \l__postnotes_prev_mark_chap_tl }
796         \prop_gput:Nxx \g__postnotes_header_sect_last_prop
797         { \l__postnotes_prev_text_page_tl }
798         { \l__postnotes_prev_mark_sect_tl }
799         \prop_gput:Nxx \g__postnotes_header_name_last_prop
800         { \l__postnotes_prev_text_page_tl }
801         { \l__postnotes_prev_mark_name_tl }
802       }

```

Set ‘first’ values for current page, based on the current note ID.

```

803       \prop_gput:Nxx \g__postnotes_header_page_first_prop
804       { \l__postnotes_curr_text_page_tl }
805       { \__postnotes_extract_pageref:n { mark@ ##1 } }

```

```

806         \prop_gput:Nxx \g__postnotes_header_chap_first_prop
807         { \l__postnotes_curr_text_page_tl }
808         { \__postnotes_prop_item:nn {##1} { thechapter } } }
809     \prop_gput:Nxx \g__postnotes_header_sect_first_prop
810     { \l__postnotes_curr_text_page_tl }
811     { \__postnotes_prop_item:nn {##1} { thesection } } }
812     \prop_gput:Nxx \g__postnotes_header_name_first_prop
813     { \l__postnotes_curr_text_page_tl }
814     { \__postnotes_prop_item:nn {##1} { pnsectname } } }

```

Store `prev_mark` data for the first note on the page.

```

815     \__postnotes_get_pageref:Nn
816     \l__postnotes_prev_mark_page_tl { mark@ ##1 }
817     \__postnotes_prop_get:nnN {##1} { thechapter }
818     \l__postnotes_prev_mark_chap_tl
819     \__postnotes_prop_get:nnN {##1} { thesection }
820     \l__postnotes_prev_mark_sect_tl
821     \__postnotes_prop_get:nnN {##1} { pnsectname }
822     \l__postnotes_prev_mark_name_tl

```

Set `\l__postnotes_prev_text_page_tl` for the next page (`\l__postnotes_curr_text_page_tl` is never empty at this point, since we conditioned to it).

```

823     \tl_set:NV \l__postnotes_prev_text_page_tl
824     \l__postnotes_curr_text_page_tl
825   }
826 }
827 }
828 }

```

We can't catch the transition from the last page of `\printpostnotes` to the following one through the mapping above, but the `prev_mark` values of the last note in the loop are the ones we want, so we set 'last' values for the last page based on them.

```

829     \tl_if_empty:NF \l__postnotes_prev_text_page_tl
830     {
831       \prop_gput:Nxx \g__postnotes_header_page_last_prop
832       { \l__postnotes_prev_text_page_tl }
833       { \l__postnotes_prev_mark_page_tl }
834       \prop_gput:Nxx \g__postnotes_header_chap_last_prop
835       { \l__postnotes_prev_text_page_tl }
836       { \l__postnotes_prev_mark_chap_tl }
837       \prop_gput:Nxx \g__postnotes_header_sect_last_prop
838       { \l__postnotes_prev_text_page_tl }
839       { \l__postnotes_prev_mark_sect_tl }
840       \prop_gput:Nxx \g__postnotes_header_name_last_prop
841       { \l__postnotes_prev_text_page_tl }
842       { \l__postnotes_prev_mark_name_tl }
843     }
844   \group_end:
845 }

```

(End definition for `__postnotes_get_headers_data:N`.)

The sequence of pages processed in `__postnotes_get_headers_data:N` is not ensured to be continuous, since not every page of `\printpostnotes` starts a note. There may be notes that fill whole pages, or the last page of the notes may end with a note

that started on the penultimate page. We must handle this case at `__postnotes_set_headers_vars:n`. For every page for which there is information provided by `__postnotes_get_headers_data:N` we store a `header_prev_last` (the last value of the previous header) for each of the variables of interest. If the next page is skipped in the sequence (no notes starting on it), we can use these stored values to set both ‘first’ and ‘last’ variables based on them for that page.

`__postnotes_set_headers_vars:n` Set user facing variables based on data generated by `__postnotes_get_headers_data:N`.

```

\__postnotes_set_headers_vars:n {(page number)}

846 \cs_new_protected:Npn \__postnotes_set_headers_vars:n #1
847 {
848   \group_begin:
849   \prop_get:NnNTF \g__postnotes_header_page_first_prop
850     {#1} \l_tmpa_tl
851     { \tl_gset:NV \pnhdpagefirst \l_tmpa_tl }
852     { \tl_gset:NV \pnhdpagefirst \g__postnotes_header_prev_last_page_tl }
853   \prop_get:NnNTF \g__postnotes_header_page_last_prop
854     {#1} \l_tmpa_tl
855     {
856       \tl_gset:NV \pnhdpagelast \l_tmpa_tl
857       \tl_gset:NV \g__postnotes_header_prev_last_page_tl \l_tmpa_tl
858     }
859     { \tl_gset:NV \pnhdpagelast \g__postnotes_header_prev_last_page_tl }
860   \prop_get:NnNTF \g__postnotes_header_chap_first_prop
861     {#1} \l_tmpa_tl
862     { \tl_gset:NV \pnhdchapfirst \l_tmpa_tl }
863     { \tl_gset:NV \pnhdchapfirst \g__postnotes_header_prev_last_chap_tl }
864   \prop_get:NnNTF \g__postnotes_header_chap_last_prop
865     {#1} \l_tmpa_tl
866     {
867       \tl_gset:NV \pnhdchaplast \l_tmpa_tl
868       \tl_gset:NV \g__postnotes_header_prev_last_chap_tl \l_tmpa_tl
869     }
870     { \tl_gset:NV \pnhdchaplast \g__postnotes_header_prev_last_chap_tl }
871   \prop_get:NnNTF \g__postnotes_header_sect_first_prop
872     {#1} \l_tmpa_tl
873     { \tl_gset:NV \pnhdsectfirst \l_tmpa_tl }
874     { \tl_gset:NV \pnhdsectfirst \g__postnotes_header_prev_last_sect_tl }
875   \prop_get:NnNTF \g__postnotes_header_sect_last_prop
876     {#1} \l_tmpa_tl
877     {
878       \tl_gset:NV \pnhdsectlast \l_tmpa_tl
879       \tl_gset:NV \g__postnotes_header_prev_last_sect_tl \l_tmpa_tl
880     }
881     { \tl_gset:NV \pnhdsectlast \g__postnotes_header_prev_last_sect_tl }
882   \prop_get:NnNTF \g__postnotes_header_name_first_prop
883     {#1} \l_tmpa_tl
884     { \tl_gset:NV \pnhdnamefirst \l_tmpa_tl }
885     { \tl_gset:NV \pnhdnamefirst \g__postnotes_header_prev_last_name_tl }
886   \prop_get:NnNTF \g__postnotes_header_name_last_prop
887     {#1} \l_tmpa_tl

```

```

888     {
889       \tl_gset:NV \pnhdnamelast \l_tmpa_tl
890       \tl_gset:NV \g__postnotes_header_prev_last_name_tl \l_tmpa_tl
891     }
892     { \tl_gset:NV \pnhdnamelast \g__postnotes_header_prev_last_name_tl }
893   \group_end:
894 }
895 \cs_generate_variant:Nn \__postnotes_set_headers_vars:n { x }

```

(End definition for `__postnotes_set_headers_vars:n`.)

`__postnotes_set_headers_vars_next:` The functions that actually call `__postnotes_set_headers_vars:n` at the appropriate contexts with appropriate page values. Though we set `__postnotes_set_headers_vars_next:` to run at every shipout/before hook of the document, it is made no-op by `\g__postnotes_header_vars_next_bool` which only has a true value inside `\printpostnotes`. `__postnotes_set_headers_vars_first:` must set a label and retrieve its value to be able to have a reliable value of its own page.

```

896 \AddToHook { shipout/before } [ postnotes/header ]
897 { \__postnotes_set_headers_vars_next: }
898 \bool_new:N \g__postnotes_header_vars_next_bool
899 \cs_new_protected:Npn \__postnotes_set_headers_vars_next:
900 {
901   \bool_if:NT \g__postnotes_header_vars_next_bool
902   { \__postnotes_set_headers_vars:x { \int_eval:n { \c@page + 1 } } }
903 }
904 \cs_new_protected:Npn \__postnotes_set_headers_vars_first:
905 {
906   \__postnotes_set_print_page_label:x
907   { \int_use:N \g__postnotes_print_postnotes_int }
908   \__postnotes_set_headers_vars:x
909   {
910     \__postnotes_extract_pageref:e
911     { print@ \int_use:N \g__postnotes_print_postnotes_int }
912   }
913 }

```

(End definition for `__postnotes_set_headers_vars_next:` and `__postnotes_set_headers_vars_first:`.)

`\pnheaderdefault` A basic header function to be used as default in the `heading` option. It produces a header in the form “Notes to pages N–M”, with a text which can be localized (see Section 10).

```

\pnheaderdefault

914 \NewDocumentCommand \pnheaderdefault {}
915 {
916   \tl_if_eq:NNTF \pnhdpagefirst \pnhdpagelast
917   { \pnhdnotes{} ~ \pnhdtopage{} ~ \pnhdpagefirst }
918   { \pnhdnotes{} ~ \pnhdtopages{} ~ \pnhdpagefirst -- \pnhdpagelast }
919 }

```

(End definition for `\pnheaderdefault`.)

9 Compatibility

A dedicated temp variable for restoring data.

```
920 \tl_new:N \l__postnotes_restore_tmp_tl
```

`\caption`

For `\caption`'s possible two passes. This catches more than just captions, of course, but is not overkill.

From the user's perspective, one-line captions will just work. For two-line captions, there are two alternatives: i) decrement the counter by 1 `\addtocounter{postnote}{-1}` before the caption, then call `\postnote` inside the caption; or ii) right before the caption, call `\postnote[nomark]{\label{mynote}...}`, then use `\postnoteref{mynote}` inside the caption.

```
921 \AddToHook { postnotes/note/begin } [ postnotes ]
922 {
923   \cs_if_exist:NT \@captype
924   { \bool_set_true:N \l__postnotes_maybe_multi_bool }
925 }
```

biblatex

Thanks Moritz Wemheuer: https://tex.stackexchange.com/q/597359#comment1594585_597389.

```
926 \AddToHook { package/biblatex/after }
927 {
```

Let biblatex know we are in a “notes” context. See <https://tex.stackexchange.com/a/304464>, including comments.

```
928   \AddToHook { postnotes/print/begin } [ postnotes ]
929   { \toggletrue { blx@footnote } }
```

Make biblatex's `\mkbibendnote` use `\postnote`. This is very likely desired in most cases, but may occasionally not be, so we add it to an individually labeled hook, which can be disabled with `\RemoveFromHook{begindocument/before}[postnotes/mkbibendnote]` in the preamble.

```
930   \AddToHook { begindocument/before } [ postnotes/mkbibendnote ]
931   {
932     \cs_set:Npn \blx@theendnote { \postnote }
933     \cs_set:Npn \blx@theendnotetext
934     { \blx@err@endnote \footnotetext }
935   }
936 }
937 <*gobble>
```

I had made an initial experimental attempt to support biblatex's `refsegments`, `refcontexts` and `refsections`. However, this attempt was rash. Even if I could get many example files to work for `refsegments` and `refcontexts`, I could not do so for `refsections`. More importantly, with this partial implementation, I could also generate documents which confused biblatex more than it helped. Things I couldn't understand well, or fix. All in all, I don't think this partial implementation is tenable, and I could

not take it further. Hence, postnotes support for this feature set of biblatex will depend, as it should, on proper upstream support for “saving” and “restoring” citation “context” information.

I have made a feature request at biblatex for this (<https://github.com/plk/biblatex/issues/1226>), which was (understandably) classified as “long term, no promises”.

The attempt was the following (currently “gobbled” from the package):

```
938 \AddToHook { package/biblatex/after }
939 {
```

Store biblatex variables for each note.

```
940   \AddToHook { postnotes/note/store } [ postnotes ]
941   {
942     \prop_gput:cnx { \l__postnotes_data_name:e { \l_postnotes_note_id_tl } }
943     { biblatex@refsection } { \int_use:N \c@refsection }
944     \prop_gput:cnx { \l__postnotes_data_name:e { \l_postnotes_note_id_tl } }
945     { biblatex@refsegment } { \int_use:N \c@refsegment }
946     \prop_gput:cnx { \l__postnotes_data_name:e { \l_postnotes_note_id_tl } }
947     { biblatex@refcontextbool }
948     { \iftoggle { blx@refcontext } { true } { false } }
949     \prop_gput:cnV { \l__postnotes_data_name:e { \l_postnotes_note_id_tl } }
950     { biblatex@refcontext } \blx@refcontext@context
951   }
```

biblatex setup, once for \printpostnotes call.

```
952   \AddToHook { postnotes/print/begin } [ postnotes ]
953   {
954     \l__postnotes_biblatex_endrefcontext_local:
955     \l__postnotes_biblatex_citereset_local:
956   }
```

Restore biblatex variables for each note.

```
957   \AddToHook { postnotes/print/note/begin } [ postnotes ]
958   {
959     \l__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
960     { biblatex@refsection } \l__postnotes_restore_tmp_tl
961     \int_set:Nn \c@refsection { \l__postnotes_restore_tmp_tl }
962     \l__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
963     { biblatex@refsegment } \l__postnotes_restore_tmp_tl
964     \int_set:Nn \c@refsegment { \l__postnotes_restore_tmp_tl }
965     \l__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
966     { biblatex@refcontextbool } \l__postnotes_restore_tmp_tl
967     \use:c { toggle \l__postnotes_restore_tmp_tl } { blx@refcontext }
968     \l__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
969     { biblatex@refcontext } \l__postnotes_restore_tmp_tl
970     \blx@edef@refcontext { \l__postnotes_restore_tmp_tl }
971   }
```

Auxiliary functions.

`\l__postnotes_biblatex_endrefcontext_local:` Replicate the job of \endrefcontext, but with local effects, restrained to the group of \printpostnotes.

```
972   \cs_new_protected:Npn \l__postnotes_biblatex_endrefcontext_local:
973   {
```

```

974 \togglefalse { blx@refcontext }
975 \tl_clear:N \blx@refcontext@labelprefix
976 \tl_clear:N \blx@refcontext@labelprefix@real
977 \tl_set:Nx \blx@refcontext@sortingtemplatename { \blx@sorting }
978 \tl_set:Nn \blx@refcontext@sortingnamekeytemplatename { global }
979 \tl_set:Nn \blx@refcontext@uniquenametemplatename { global }
980 \tl_set:Nn \blx@refcontext@labelalphanametemplatename { global }
981 \blx@edef@refcontext
982 {
983   \blx@refcontext@sortingtemplatename /
984   \blx@refcontext@sortingnamekeytemplatename /
985   /
986   \blx@refcontext@uniquenametemplatename /
987   \blx@refcontext@labelalphanametemplatename
988 }
989 }

```

(End definition for _postnotes_biblatex_endrefcontext_local:.)

_postnotes_biblatex_citereset_local: Replicate the job of \citereset, but with local effects, restrained to the group of \printpostnotes.

```

990 \cs_new_protected:Npn \_postnotes_biblatex_citereset_local:
991 {
992   \global\cslet{blx@bsee@the\c@refsection}\empty
993   \global\cslet{blx@fsee@the\c@refsection}\empty
994   \tl_clear:c { blx@bsee@ \int_use:N \c@refsection }
995   \tl_clear:c { blx@fsee@ \int_use:N \c@refsection }
996   \blx@ibidreset@force
997   \undef \blx@lastkey@text
998   \undef \blx@lastkey@foot
999   \blx@idemreset@force
1000   \undef \blx@lasthash@text
1001   \undef \blx@lasthash@foot
1002   \blx@opcitreset@force
1003   \clist_map_inline:Nn \blx@trackhash@text
1004     { \csundef { blx@lastkey@text@ ##1 } }
1005   \tl_clear:N \blx@trackhash@text
1006   \clist_map_inline:Nn \blx@trackhash@foot
1007     { \csundef { blx@lastkey@foot@ ##1 } }
1008   \tl_clear:N \blx@trackhash@foot
1009   \blx@loccitreset@force
1010   \clist_map_inline:Nn \blx@trackkeys@text
1011     { \csundef { blx@lastnote@text@ ##1 } }
1012   \tl_clear:N \blx@trackkeys@text
1013   \clist_map_inline:Nn \blx@trackkeys@foot
1014     { \csundef { blx@lastnote@foot@ ##1 } }
1015   \tl_clear:N \blx@trackkeys@foot
1016   and all of them do:
1017   \cs_set_eq:NN \blx@lastmpfn \z@
1018 }

```


(End definition for _postnotes_biblatex_citereset_local:.)

1012 }

biblatex’s `refsections`, contrary to `refsegments` and `refcontexts` which are handled in the L^AT_EX side of things (as far as I can tell), need to go through `biber`, and must have correct corresponding citation data written to the `.bcf` file. And the way `\refsection` is implemented presumes each section is only ever begun once (fair...), thus making it difficult to “reopen” it, or append new citations to it later on, when the notes are printed. The start of a `refsection` must be registered on the `.bcf` file, and this is done by `\refsection` (and its auxiliary functions). However, a number of its characteristics make things particularly difficult for the purpose at hand: i) it unconditionally sets a label for the section which, of course, cannot be done twice; and, critically, ii) the optional argument of the environment (which receives the $\langle resources \rangle$) is used to set a local assignment to `\blx@bibfiles`, based on which the relevant information is written to the `.bcf` file, and when the group closes the information is gone. My best attempt is below but it is not good. It feels a wrong approach to “go around” the intended use of `\refsection` so much, and it can’t handle at all its optional argument, for the reasons above. It’s also incomplete, since it does not handle restoring `\l__postnotes_biblatex_orig_refsection_tl`.

```

1013 \AddToHook { package/biblatex/after }
1014 {
1015   \tl_new:N \l__postnotes_biblatex_orig_refsection_tl
1016   \tl_new:N \g__postnotes_biblatex_prev_refsection_tl
1017   \AddToHook { postnotes/print/begin } [ postnotes ]
1018   {
1019     \tl_set:Nx \l__postnotes_biblatex_orig_refsection_tl
1020     { \int_use:N \c@refsection }
1021     \tl_gset:Nx \g__postnotes_biblatex_prev_refsection_tl
1022     \l__postnotes_biblatex_orig_refsection_tl
1023   }
1024   \AddToHook { postnotes/print/note/begin } [ postnotes ]
1025   {
1026     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1027     { biblatex@refsection } \l__postnotes_restore_tmp_tl
1028     \tl_if_eq:NNF
1029     \l__postnotes_restore_tmp_tl
1030     \g__postnotes_biblatex_prev_refsection_tl
1031     {
1032       \int_set:Nn \c@blx@maxsection
1033       { \l__postnotes_restore_tmp_tl - 1 }
1034       \tl_gset_eq:NN \g__postnotes_biblatex_prev_refsection_tl
1035       \l__postnotes_restore_tmp_tl
1036       \group_begin:
1037       \cs_set_eq:NN \label \use_none:n
1038       \cs_set_eq:NN \blx@info \use_none:n
1039       \blx@endrefsection
1040       \refsection
1041       \group_end:
1042     }
1043   }
1044 }
1045 \gobble

```

zref-user

`\l__postnotes_note_zlabel_tl` Even though the `zlabel` option is provided only when `zref-user` is loaded, `\l__postnotes_note_zlabel_tl` must be unconditionally defined, since it is presumed to exist by `__postnotes_set_user_labels:`.

```
1046 \tl_new:N \l__postnotes_note_zlabel_tl
```

(End definition for `\l__postnotes_note_zlabel_tl`.)

```
1047 \AddToHook { package/zref-user/after }
1048 {
```

Provide `zlabel` option.

```
1049 \keys_define:nn { postnotes/note }
1050 {
1051     zlabel .tl_set:N = \l__postnotes_note_zlabel_tl ,
1052     zlabel .value_required:n = true ,
1053 }
```

`\postnotezref` Provide `\postnotezref`.

```
\postnotezref(*){\label{}}
```

```
1054 \NewDocumentCommand \postnotezref { s m }
1055 { \__postnotes_note_zref:nn {#1} {#2} }
```

(End definition for `\postnotezref`.)

`__postnotes_note_zref:nn` The internal version of `\postnotezref`.

```
\__postnotes_note_zref:nn {<star bool>} {\label{}}
```

```
1056 \cs_new_protected:Npn \__postnotes_note_zref:nn #1#2
1057 {
1058     \group_begin:
1059     \__postnotes_typeset_mark_wrapper:n
1060     {
1061         \bool_lazy_all:nTF
1062         {
1063             { ! #1 }
1064             { \l__postnotes_hyperlink_bool }
1065             { \l__postnotes_zrefhyperref_bool }
1066         }
1067         {
1068             \hyperlink
1069             { \zref@extractdefault {#2} { anchor } { } }
1070             { \__postnotes_make_mark:nnn { \zref{#2} } { } { } }
1071         }
1072         { \__postnotes_make_mark:nnn { \zref{#2} } { } { } }
1073     }
1074     \group_end:
1075 }
```

(End definition for `__postnotes_note_zref:nn`.)

```
1076 }
1077 \bool_new:N \l__postnotes_zrefhyperref_bool
1078 \AddToHook { package/zref-hyperref/after }
1079 { \bool_set_true:N \l__postnotes_zrefhyperref_bool }
```

zref-clever

```
1080 \AddToHook { package/zref-clever/after }
1081 {
1082   \zcsetup
1083   {
1084     countertype = { postnote = endnote } ,
1085     countertype = { postnotetext = endnote } ,
1086   }
1087   \AddToHook { postnotes/print/begin } [ postnotes ]
1088   { \zcsetup { counterresetby = { postnotetext = postnotessection } } }
1089 }
```

zref-check

```
1090 \AddToHook { package/zref-check/after }
1091 {
1092   \IfPackageAtLeastTF { zref-check } { 2022-07-05 }
1093   {
1094     \AddToHook { postnotes/note/store } [ postnotes ]
1095     {
1096       \prop_gput:cnx { \l__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1097       { zref-check@abschap } { \int_use:N \c@zc@abschap }
1098       \prop_gput:cnx { \l__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1099       { zref-check@abssec } { \int_use:N \c@zc@abssec }
1100     }
1101     \AddToHook { postnotes/print/note/begin } [ postnotes ]
1102     {
1103       \l__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1104       { zref-check@abschap } \l__postnotes_restore_tmp_tl
1105       \int_set:Nn \c@zc@abschap { \l__postnotes_restore_tmp_tl }
1106       \l__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1107       { zref-check@abssec } \l__postnotes_restore_tmp_tl
1108       \int_set:Nn \c@zc@abssec { \l__postnotes_restore_tmp_tl }
1109     }
1110   }
1111   { }
1112 }
```

amsmath

```
1113 \AddToHook { package/amsmath/after }
1114 {
1115   \AddToHook { postnotes/note/inhibit } [ postnotes ]
1116   {
1117     \legacy_if:nT { measuring@ }
1118     {
1119       \bool_set_true:N \l__postnotes_inhibit_note_bool
1120       \bool_set_true:N \l__postnotes_print_plain_mark_bool
1121     }
1122   }
```

However, the `\text` macro, defined by `amstext` (required by `amsmath`), poses problems if its own. Despite my best efforts, I could not salvage things from the use of `\mathchoice`

and the redefinitions of `\setcounter` and `\addtocounter` performed by `amstext`. Setting `\l__postnotes_maybe_multi_bool` when `firstchoice@` is `false` grants us a working situation for display style. But the use of `\postnote` inside `\text` (and, if `amsmath` is loaded, `\textnormal`, `\textup`, etc.) in inline math environments is not supported. If a note really needs to be there, one can use the `nomark` option and `\postnoteref`. Things should work in text mode and in display style. For some related discussion with regard to footnotes, see <https://tex.stackexchange.com/a/82820> and, in particular, Barbara Beeton’s comment: “This is certainly bravura code. I do hope it doesn’t result in a request to add `\footnote` capabilities to `amsmath`’s multi-line display facilities. (The answer will almost certainly be no. We agree with Kopka & Daly.)”

```

1123 \AddToHook { postnotes/note/begin } [ postnotes ]
1124 {
1125     \legacy_if:nF { firstchoice@ }
1126     { \bool_set_true:N \l__postnotes_maybe_multi_bool }
1127 }
1128 }

```

csquotes

```

1129 \AddToHook { package/csquotes/after }
1130 {
1131     \bool_new:N \l__postnotes_csquotes_measuring_bool
1132     \BlockquoteDisable
1133     { \bool_set_true:N \l__postnotes_csquotes_measuring_bool }
1134     \AddToHook { postnotes/note/inhibit } [ postnotes ]
1135     {
1136         \bool_if:NT \l__postnotes_csquotes_measuring_bool
1137         {
1138             \bool_set_true:N \l__postnotes_inhibit_note_bool
1139             \bool_set_true:N \l__postnotes_print_plain_mark_bool
1140         }
1141     }
1142 }

```

tabularx

For the identification of the trial passes in `tabularx`, see <https://tex.stackexchange.com/a/640035> (including discussion in the comments, thanks David Carlisle), and also <https://tex.stackexchange.com/a/227155> and <https://tex.stackexchange.com/a/352134>.

```

1143 \AddToHook { package/tabularx/after }
1144 {
1145     \bool_new:N \l__postnotes_tabularx_inside_env_bool
1146     \AddToHook { env/tabularx/begin } [ postnotes ]
1147     {
1148         \bool_set_true:N \l__postnotes_tabularx_inside_env_bool
1149         \cs_set_eq:NN \__postnotes_tabularx_saved_write:Nn \write
1150     }
1151     \AddToHook { postnotes/note/inhibit } [ postnotes ]
1152     {
1153         \bool_lazy_and:nnT
1154         { \l__postnotes_tabularx_inside_env_bool }
1155         { ! \cs_if_eq_p:NN \write \__postnotes_tabularx_saved_write:Nn }
1156         {

```

```

1157         \bool_set_true:N \l__postnotes_inhibit_note_bool
1158         \bool_set_true:N \l__postnotes_print_plain_mark_bool
1159     }
1160 }
1161 }

```

tabularray

I’ve tried, but I could not find any “handle” to distinguish in `tabularray` a trial/measure pass from the final one. So we use `__postnotes_verify_multipass:N` for it.

```

1162 \AddToHook { package/tabularray/after }
1163 {
1164     \clist_map_inline:nn
1165     { tblr , longtblr , talltblr , booktabs , longtabs , talltabs , +array }
1166     {
1167         \AddToHook { env/#1/begin } [ postnotes ]
1168         { \bool_set_true:N \l__postnotes_maybe_multi_bool }
1169     }
1170 }

```

10 Languages

<code>\pntitle</code> <code>\pnhdnotes</code> <code>\pnhdtopage</code> <code>\pnhdtopages</code>	<p>Set of language specific user variables. They are used in the default value of the <code>heading</code> option and in <code>\pnheaderdefault</code> which, ultimately, is also used in the same place.</p> <pre> 1171 \tl_new:N \pntitle 1172 \tl_new:N \pnhdnotes 1173 \tl_new:N \pnhdtopage 1174 \tl_new:N \pnhdtopages 1175 \tl_set:Nn \pntitle { Notes } 1176 \tl_set:Nn \pnhdnotes { Notes } 1177 \tl_set:Nn \pnhdtopage { to~page } 1178 \tl_set:Nn \pnhdtopages { to~pages } </pre>
---	---

(End definition for `\pntitle` and others.)

<code>__postnotes_define_language:nn</code>	<p>Defines language specific values for <i>⟨postnote language⟩</i> by storing a set of assignments for the language specific variables in <i>⟨setup⟩</i>. <i>⟨postnote language⟩</i> is an internal name, typically the “main” name of the language, based on which we can set specific <code>babel</code> or <code>polyglossia</code> languages or variants.</p>
--	---

```

        \__postnotes_define_language:nn {⟨postnote language⟩} {⟨setup⟩}

1179 \cs_new_protected:Npn \__postnotes_define_language:nn #1#2
1180 {
1181     \tl_new:c { g__postnotes_language_ #1 _tl }
1182     \tl_gset:cn { g__postnotes_language_ #1 _tl } {#2}
1183 }

```

(End definition for `__postnotes_define_language:nn`.)

For `babel` we use the new hook system, it’s clean, and avoids the `\addto` pitfalls. The appropriate hook to use is `babel/⟨language⟩/beforeextras` so that users can override it with a traditional `\addto\extras⟨language⟩`.

Note that, for `babel`, the captions are currently handled in two different ways – the “old way” and the “new way” – and which of them is used depends on the language. Most still use the “old way”, but the problem is that it is not universal. And the “new way” uses a different naming scheme – `\<language>\<caption>`, which is meant to be set with `\setlocalecaption`, and not suitable for our needs. The `\extras<language>` macros are meant for “arbitrary” code to be run when the language is selected, which is what we want. The captions used to work in the same way, but no longer for languages which use the “new way”.

Note also that there seems to exist some qualms about `babel`’s `\addto`. A number of packages define their own versions of it. Do so at least `varioref` (probably the original), `backref`, and `cleveref`. The latter comments that `\addto` is “flawed”. `babel` itself comments the definition recognizing that there is an “inconsistency”: depending on the case, the operation will be either local or global. This is documented in the manual, which explains this inconsistent behavior is preserved for backward compatibility, and recommends `etoolbox`’s facilities if available. `polyglossia` also recommends `etoolbox`’s `\gappto`. All in all, if there’s need to use the traditional way instead of the new hooks, just rely on `expl3` and use `\tl_gput_right:Nn`.

`__postnotes_set_babel_language:nn` Sets `\<babel language>` to execute the setup defined by `__postnotes_define_language:nn` for `\<postnote language>` at the `babel/\<language>/beforeextras` hook.

```

\__postnotes_set_babel_language:nn {\<babel language>} {\<postnote language>}

1184 \cs_new_protected:Npn \__postnotes_set_babel_language:nn #1#2
1185 {
1186   \ActivateGenericHook { babel/#1/beforeextras }
1187   \exp_args:Nnv \AddToHook { babel/#1/beforeextras }
1188   { g__postnotes_language_ #2 _tl }
1189 }

```

(End definition for `__postnotes_set_babel_language:nn`.)

`polyglossia` uses a similar set of macros for setting up languages as `babel` does. However, the `\blockextras@<language>` macros are unfortunately internal (despite what the manual says, that’s what the code does), thus requiring `\makeatletter/\makeatother` for user configuration, which would be an inconvenience. On the other hand, `polyglossia`’s `\captions<language>` works as in `babel`’s “old way”, meaning it is just a “hook” to which we can append some code. So we use `\captions<language>` for `polyglossia`. Things may complicate here if there’s need to set up different values for different language variants, since the hooks available are all necessarily internal, but I doubt we’ll ever need variants for these simple strings.

`__postnotes_set_polyglossia_language:nn` Sets `\<polyglossia language>` to execute the setup defined by `__postnotes_define_language:nn` for `\<postnote language>` at the `polyglossia \captions<language>` hook.

```

\__postnotes_set_polyglossia_language:nn {\<polyglossia language>}
{\<postnote language>}

1190 \cs_new_protected:Npn \__postnotes_set_polyglossia_language:nn #1#2
1191 {
1192   \AddToHook { package/polyglossia/after }
1193   {
1194     \exp_args:Nnv \csgappto { captions #1 }

```

```

1195         { g__postnotes_language_ #2 _tl }
1196     }
1197 }

```

(End definition for _postnotes_set_polyglossia_language:nn.)

English

```

1198 \_postnotes_define_language:nn { english }
1199 {
1200     \tl_set:Nn \pntitle      { Notes }
1201     \tl_set:Nn \pnhdnotes   { Notes }
1202     \tl_set:Nn \pnhdtopage  { to~page }
1203     \tl_set:Nn \pnhdtopages { to~pages }
1204 }
1205 \_postnotes_set_babel_language:nn { english } { english }
1206 \_postnotes_set_babel_language:nn { british } { english }
1207 \_postnotes_set_babel_language:nn { american } { english }
1208 \_postnotes_set_babel_language:nn { canadian } { english }
1209 \_postnotes_set_babel_language:nn { australian } { english }
1210 \_postnotes_set_babel_language:nn { newzealand } { english }
1211 \_postnotes_set_babel_language:nn { UKenglish } { english }
1212 \_postnotes_set_babel_language:nn { USenglish } { english }
1213 \_postnotes_set_polyglossia_language:nn { english } { english }

```

Portuguese

```

1214 \_postnotes_define_language:nn { portuguese }
1215 {
1216     \tl_set:Nn \pntitle      { Notas }
1217     \tl_set:Nn \pnhdnotes   { Notas }
1218     \tl_set:Nn \pnhdtopage  { da~página }
1219     \tl_set:Nn \pnhdtopages { das~páginas }
1220 }
1221 \_postnotes_set_babel_language:nn { portuguese } { portuguese }
1222 \_postnotes_set_babel_language:nn { brazilian } { portuguese }
1223 \_postnotes_set_babel_language:nn { portuges } { portuguese }
1224 \_postnotes_set_babel_language:nn { brazil } { portuguese }
1225 \_postnotes_set_polyglossia_language:nn { portuguese } { portuguese }

```

French

French localization validated by ‘Pika78’ at issue [#1](#).

babel-french also has .ldfs for `francais`, `frenchb`, and `canadien`, but they are deprecated as options and, if used, they fall back to either `french` or `acadian`.

```

1226 \_postnotes_define_language:nn { french }
1227 {
1228     \tl_set:Nn \pntitle      { Notes }
1229     \tl_set:Nn \pnhdnotes   { Notes }
1230     \tl_set:Nn \pnhdtopage  { de~la~page }
1231     \tl_set:Nn \pnhdtopages { des~pages }
1232 }
1233 \_postnotes_set_babel_language:nn { french } { french }
1234 \_postnotes_set_babel_language:nn { acadian } { french }
1235 \_postnotes_set_polyglossia_language:nn { french } { french }

```

German

German localization provided by Herbert Voß at issue [#2](#).

`babel-german` also has `.ldfs` for `germanb` and `ngermanb`, but they are deprecated as options and, if used, they fall back respectively to `german` and `ngerman`.

```

1236 \_postnotes_define_language:nn { german }
1237 {
1238   \tl_set:Nn \pntitle      { Endnoten }
1239   \tl_set:Nn \pnhdnotes   { Endnoten }
1240   \tl_set:Nn \pnhdtopage  { zu-Seite }
1241   \tl_set:Nn \pnhdtopages { zu-Seiten }
1242 }
1243 \_postnotes_set_babel_language:nn { german }      { german }
1244 \_postnotes_set_babel_language:nn { ngerman }     { german }
1245 \_postnotes_set_babel_language:nn { austrian }   { german }
1246 \_postnotes_set_babel_language:nn { naustrian }  { german }
1247 \_postnotes_set_babel_language:nn { swissgerman } { german }
1248 \_postnotes_set_babel_language:nn { nswissgerman } { german }
1249 \_postnotes_set_polyglossia_language:nn { german } { german }
1250 </package>

```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

A			
<code>\ActivateGenericHook</code>	1186	<code>\bool_set_false:N</code>	143, 222, 231, 232, 247, 375, 376
<code>\addto</code>	37, 38	<code>\bool_set_true:N</code>	148, 221, 226, 227, 356, 924, 1079, 1119, 1120, 1126, 1133, 1138, 1139, 1148, 1157, 1158, 1168
<code>\addtocounter</code>	36	<code>\bool_to_str:N</code>	46
<code>\AddToHook</code>	240, 896, 921, 926, 928, 930, 938, 940, 952, 957, 1013, 1017, 1024, 1047, 1078, 1080, 1087, 1090, 1094, 1101, 1113, 1115, 1123, 1129, 1134, 1143, 1146, 1151, 1162, 1167, 1187, 1192	<code>\bool_until_do:nn</code>	529
<code>\AddToHookNext</code>	636	box commands:	
		<code>\box_use:N</code>	292
		<code>\box_wd:N</code>	291
		<code>\l_tmpa_box</code>	290, 291, 292
B		C	
<code>\begin</code>	578	<code>\caption</code>	10, 11, 30
<code>\BlockquoteDisable</code>	1132	<code>\chapter</code>	117
bool commands:		<code>\citereset</code>	32
<code>\bool_gset_false:N</code>	637	clist commands:	
<code>\bool_gset_true:N</code>	521	<code>\clist_map_inline:Nn</code>	998, 1001, 1004, 1007
<code>\bool_if:NTF</code>	30, 245, 334, 378, 391, 399, 470, 519, 524, 577, 628, 660, 901, 1136	<code>\clist_map_inline:nn</code>	1164
<code>\bool_lazy_all:nTF</code>	1061	<code>\counterwithin</code>	12
<code>\bool_lazy_and:nnTF</code>	437, 647, 693, 1153	cs commands:	
<code>\bool_new:N</code>	136, 213, 214, 215, 267, 344, 347, 348, 370, 371, 476, 898, 1077, 1131, 1145	<code>\cs_generate_variant:Nn</code>	18, 63, 81, 87, 93, 100, 107, 408, 657, 895
		<code>\cs_if_eq_p:NN</code>	1155

<code>\cs_if_exist:NTF</code>		H	
..... 34, 54, 96, 103, 113, 673, 923		<code>\hbox</code>	196
<code>\cs_new:Npn</code>	16, 69, 101, 189	hbox commands:	
<code>\cs_new_protected:Npn</code>	20,	<code>\hbox_set:Nn</code>	290
50, 64, 71, 76, 82, 88, 94, 115, 122,		<code>\hspace</code>	190
314, 395, 410, 423, 432, 462, 507,		<code>\hyperlink</code>	1068
645, 658, 664, 682, 735, 846, 899,		<code>\hyperref</code>	441
904, 972, 990, 1056, 1179, 1184, 1190			
<code>\cs_new_protected:Npx</code>	74	I	
<code>\cs_set:Npn</code>	329, 595, 932, 933	<code>\IfFormatAtLeastTF</code>	3, 4
<code>\cs_set:Npx</code>	330, 598	<code>\IfPackageAtLeastTF</code>	1092
<code>\cs_set_eq:NN</code>		<code>\IfPackageLoadedTF</code>	242
162, 179, 526, 1010, 1037, 1038, 1149		<code>\iftoggle</code>	948
<code>\csgappto</code>	1194	int commands:	
<code>\csundef</code>	999, 1002, 1005, 1008	<code>\int_eval:n</code>	902
		<code>\int_gincr:N</code>	320, 465, 466, 510
E		<code>\int_incr:N</code>	383
<code>\end</code>	629	<code>\int_new:N</code>	307, 461, 492
<code>\endgroup</code>	21	<code>\int_set:Nn</code>	
<code>\endinput</code>	12 596, 961, 964, 1032, 1105, 1108	
<code>\endlist</code>	171, 188	<code>\int_use:N</code>	
<code>\endnotemark</code>	10	. 27, 32, 44, 85, 91, 309, 907, 911,	
<code>\endnotetext</code>	10	943, 945, 992, 993, 1020, 1097, 1099	
<code>\endrefcontext</code>	31	iow commands:	
<code>\enoteheading</code>	19	<code>\iow_char:N</code>	458, 459, 644
exp commands:		<code>\iow_shipout_x:Nn</code>	4, 78, 84, 90
<code>\exp_args:NNNx</code>	384	<code>\item</code>	21, 661
<code>\exp_args:Nnv</code>	1187, 1194	<code>\itemindent</code>	161, 178
<code>\exp_args:Nv</code>	426, 428	<code>\itemsep</code>	166, 183
<code>\exp_args:Nx</code>	578, 629, 766		
<code>\exp_not:N</code>	75	K	
<code>\exp_not:n</code>	104	keys commands:	
		<code>\keys_define:nn</code>	108, 129, 137, 191,
F		204, 216, 249, 268, 274, 349, 477, 1049	
<code>\fmtversion</code>	3	<code>\keys_set:nn</code>	305, 317, 469
<code>\footnote</code>	36		
<code>\footnotemark</code>	10, 11	L	
<code>\footnotesize</code>	282	<code>\label</code>	11, 426, 1037
<code>\footnotetext</code>	10, 11, 934	<code>\labelsep</code>	190
fp commands:		<code>\labelwidth</code>	160, 177
<code>\fp_compare:nNnTF</code>	699	<code>\leftmargin</code>	159, 176, 177, 178
<code>\fp_new:N</code>	345	<code>\leftskip</code>	284
<code>\fp_set:Nn</code>	355	legacy commands:	
<code>\fp_use:N</code>	31	<code>\legacy_if:nTF</code>	1117, 1125
G		<code>\list</code>	157, 174
<code>\gappto</code>	38	<code>\listparindent</code>	164, 181
group commands:			
<code>\group_begin:</code>		M	
..... 316, 382, 434, 464, 509, 552,		<code>\makeatletter</code>	38
579, 582, 666, 684, 737, 848, 1036, 1058		<code>\makeatother</code>	38
<code>\group_end:</code>		<code>\makelabel</code>	162, 179
..... 341, 385, 446, 473, 569, 613,		<code>\MakeLinkTarget</code>	2, 331, 602
630, 641, 680, 707, 844, 893, 1041, 1074		<code>\mathchoice</code>	10, 35
		<code>\mbox</code>	10
		<code>\MessageBreak</code>	10

`\mkbibendnote` 30
mode commands:
`\mode_if_horizontal:TF` 413, 419
`\mode_leave_vertical:` 412, 610
msg commands:
`\msg_line_context:` 264, 458, 644
`\msg_new:nnn` 263, 265, 456, 643
`\msg_warning:nn` 246, 453, 512
`\msg_warning:nnn` 253, 258

N

`\newcounter` 306, 504, 505
`\NewDocumentCommand`
304, 311, 430, 448, 450, 485, 914, 1054
`\NewDocumentEnvironment` 155, 172
`\NewHook` 2, 19, 313, 372, 502, 503
`\newlabel` 4
`\nobreak` 416
`\noindent` 299
`\normalfont` 190, 196, 290, 300

P

`\PackageError` 7
`\par` 144, 287, 299
`\parindent` 161, 164, 181, 285
`\parsep` 165, 182
`\parskip` 165, 182
`\partopsep` 168, 185
`\pnhdchapfirst` 709, 740, 862, 863
`\pnhdchaplast` 709, 741, 867, 870
`\pnhdnamefirst` 709, 744, 884, 885
`\pnhdnamelast` 709, 745, 889, 892
`\pnhdnotes`
917, 918, 1171, 1201, 1217, 1229, 1239
`\pnhdpagefirst`
.... 709, 738, 851, 852, 916, 917, 918
`\pnhdpagelast` . 709, 739, 856, 859, 916, 918
`\pnhdsectfirst` 709, 742, 873, 874
`\pnhdsectlast` 709, 743, 878, 881
`\pnhdtopage` 917, 1171, 1202, 1218, 1230, 1240
`\pnhdtopages`
... 918, 1171, 1203, 1219, 1231, 1241
`\pnheaderdefault` ... 29, 37, 118, 125, 914
`\pnheading` 5, 110, 113, 514
`\pnthechapter` 487, 555
`\pnthechapternextnote` 487, 561
`\pnthepage` 487, 584
`\pnthesection` 487, 558
`\pnthesectionnextnote` 487, 564
`\pntitle`
117, 124, 1171, 1200, 1216, 1228, 1238
`\postnote` 2,
10-13, 15, 22, 24, 30, 36, 311, 932
`\postnotemark` 11, 12
`\postnoteref` 11, 15, 16, 36, 430
postnotes commands:
`\l_postnotes_note_id_tl` 12,
307, 327, 331, 332, 337, 339, 467,
471, 472, 942, 944, 946, 949, 1096, 1098
`\l_postnotes_print_note_id_tl` ...
.... 492, 532, 533, 554, 557, 566,
585, 587, 590, 593, 603, 605, 607,
959, 962, 965, 968, 1026, 1103, 1106
postnotes internal commands:
`\l__postnotes_backlink_bool`
..... 215, 236, 649
`__postnotes_biblatex_citereset_-`
local: 955, 990, 990
`__postnotes_biblatex_endrefcontext_-`
local: 954, 972, 972
`\l__postnotes_biblatex_orig_-`
refsection_tl . 33, 1015, 1019, 1022
`\g__postnotes_biblatex_prev_-`
refsection_tl 1016, 1021, 1030, 1034
`\l__postnotes_clear_queue_seq` ...
..... 21, 492, 517, 638
`\l__postnotes_csquotes_measuring_-`
bool 1131, 1133, 1136
`\l__postnotes_curr_text_page_tl` .
..... 27, 717, 759,
771, 772, 776, 804, 807, 810, 813, 824
`__postnotes_data_name:n` .. 2, 12,
16, 16, 18, 22, 23, 24, 26, 28, 36, 39,
41, 43, 45, 47, 52, 53, 56, 59, 61, 66,
70, 72, 942, 944, 946, 949, 1096, 1098
`__postnotes_define_language:nn` .
..... 37,
38, 1179, 1179, 1198, 1214, 1226, 1236
`__postnotes_extract_pageref:n` ..
..... 5, 94, 101, 107, 805, 910
`__postnotes_get_headers_data:N` .
..... 24, 25, 27, 28, 522, 735, 735
`__postnotes_get_pageref:Nn`
.... 5, 94, 94, 100, 584, 770, 778, 815
`\g__postnotes_header_chap_first_-`
prop 717, 748, 806, 860
`\g__postnotes_header_chap_last_-`
prop 717, 749, 793, 834, 864
`\g__postnotes_header_name_first_-`
prop 717, 752, 812, 882
`\g__postnotes_header_name_last_-`
prop 717, 753, 799, 840, 886
`\g__postnotes_header_page_first_-`
prop 717, 746, 803, 849
`\g__postnotes_header_page_last_-`
prop 717, 747, 790, 831, 853
`\g__postnotes_header_prev_last_-`
chap_tl 717, 755, 863, 868, 870

\g__postnotes_header_prev_last_- name_tl	717, 757, 885, 890, 892	\l__postnotes_prev_mark_chap_tl	717, 761, 781, 795, 818, 836
\g__postnotes_header_prev_last_- page_tl	717, 754, 852, 857, 859	\l__postnotes_prev_mark_name_tl	717, 763, 785, 801, 822, 842
\g__postnotes_header_prev_last_- sect_tl	717, 756, 874, 879, 881	\l__postnotes_prev_mark_page_tl	717, 760, 779, 792, 816, 833
\g__postnotes_header_sect_first_- prop	717, 750, 809, 871	\l__postnotes_prev_mark_sect_tl	717, 762, 783, 798, 820, 839
\g__postnotes_header_sect_last_- prop	717, 751, 796, 837, 875	\l__postnotes_prev_text_page_tl	26, 27, 717, 758, 775, 788, 791, 794, 797, 800, 823, 829, 832, 835, 838, 841
\g__postnotes_header_vars_next_- bool	29, 521, 637, 898, 901	\l__postnotes_print_as_list_bool	136, 143, 148, 524, 577, 628, 660
\l__postnotes_hyperlink_bool	213, 221, 226, 231, 247, 399, 439, 648, 1064	\l__postnotes_print_content_tl	492, 567, 568, 594, 611
\l__postnotes_hyperref_warn_bool	214, 222, 227, 232, 245	\l__postnotes_print_counter_tl	492, 591, 597
__postnotes_inhibit_note:	373	\l__postnotes_print_env_tl	135, 145, 149, 578, 629
__postnotes_inhibit_note:TF	22, 318, 370	\l__postnotes_print_format_tl	128, 131, 580
\l__postnotes_inhibit_note_bool	14, 370, 375, 391, 1119, 1138, 1157	\l__postnotes_print_mark_tl	492, 588, 599, 608
__postnotes_list_makelabel:n	162, 179, 189	\l__postnotes_print_note_id_- next_tl	492, 539, 544, 546, 560, 563, 616, 621, 623
__postnotes_make_mark:nnn	193, 389, 401, 405, 442, 444, 1070, 1072	__postnotes_print_notes:	18, 22-24, 486, 507, 507
__postnotes_make_text_mark:nnn	197, 651, 655	\l__postnotes_print_plain_mark_- bool	14, 371, 376, 378, 1120, 1139, 1158
\l__postnotes_manual_sortnum_- bool	30, 347, 356	\g__postnotes_print_postnotes_- int	492, 510, 907, 911
\l__postnotes_mark_tl	25, 321, 324, 330, 337, 343, 351, 380, 386, 389	\l__postnotes_print_type_curr_tl	492, 534, 535, 571, 633
\l__postnotes_maybe_multi_bool	22, 36, 46, 348, 924, 1126, 1168	\l__postnotes_print_type_next_tl	492, 540, 547, 549, 617, 624, 626
\l__postnotes_nomark_bool	334, 344, 365	\l__postnotes_print_type_prev_tl	492, 516, 570, 575, 632
__postnotes_note:nn	12, 14, 15, 312, 313, 314	__postnotes_prop_gc_clear:n	4, 64, 71, 639
\g__postnotes_note_id_int	12, 307, 320, 466	__postnotes_prop_get:nnN	4, 64, 64, 533, 545, 553, 556, 559, 562, 565, 586, 589, 592, 622, 670, 687, 688, 691, 692, 697, 698, 780, 782, 784, 817, 819, 821, 959, 962, 965, 968, 1026, 1103, 1106
\l__postnotes_note_label_tl	346, 367, 425, 426	__postnotes_prop_item:nn	4, 64, 69, 767, 808, 811, 814
__postnotes_note_ref:nn	16, 431, 432, 432	\g__postnotes_queue_seq	12, 22, 23, 25, 307, 326, 467, 511, 517, 518, 520, 522, 529, 531, 537, 543, 614, 620
\l__postnotes_note_zlabel_tl	34, 427, 428, 1046, 1051	\c__postnotes_ref_prefix_tl	4, 73, 75, 96, 97, 103, 104, 674
__postnotes_note_zref:nn	34, 1055, 1056, 1056		
\l__postnotes_post_printnote_tl	144, 203, 210, 612		
\l__postnotes_post_textmark_tl	202, 208, 661, 662		
\l__postnotes_pre_textmark_tl	201, 206, 661, 662		

475, 487, 488, 489, 490, 491, 493, 494, 495, 496, 497, 498, 499, 500, 709, 710, 711, 712, 713, 714, 715, 716, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 920, 1015, 1016, 1046, 1171, 1172, 1173, 1174, 1181	<code>\togglefalse</code> 974 <code>\toggletrue</code> 929 token commands: <code>\token_to_str:N</code> 79, 85, 91 <code>\topsep</code> 167, 184
<code>\tl_set:Nn</code> 97, 144, 145, 149, 309, 324, 386, 415, 516, 539, 540, 570, 616, 617, 632, 823, 977, 978, 979, 980, 1019, 1175, 1176, 1177, 1178, 1200, 1201, 1202, 1203, 1216, 1217, 1218, 1219, 1228, 1229, 1230, 1231, 1238, 1239, 1240, 1241	U <code>\undef</code> 994, 995, 996, 997 use commands: <code>\use:N</code> 967 <code>\use_none:n</code> 1037, 1038 <code>\UseHook</code> 48, 328, 377, 515, 583
<code>\l_tmpa_tl</code> 670, 671, 687, 689, 691, 694, 697, 699, 850, 851, 854, 856, 857, 861, 862, 865, 867, 868, 872, 873, 876, 878, 879, 883, 884, 887, 889, 890	W <code>\write</code> 1149, 1155
<code>\l_tmpb_tl</code> . 688, 689, 692, 695, 698, 699	Z <code>\zcsetup</code> 1082, 1088 <code>\zlabel</code> 428 <code>\zref</code> 1070, 1072