The postnotes package

User manual

gusbrs

https://github.com/gusbrs/postnotes
https://www.ctan.org/pkg/postnotes

Version v0.2.8 – 2023-12-12

Abstract

postnotes is an endnotes package for LaTeX. Its user interface provides means to print multiple sections of notes along the document, and to subdivide them either automatically – by chapter, by section – or at manually specified places, thus being able to easily handle both numbered and unnumbered headings. The package also provides infrastructure for setting up contextual running headers for printed notes. The default is a simple but useful one, in the form "Notes to pages N–M", but more elaborate ones can be built. When hyperref is loaded, postnotes provides hyperlinked notes, including back links.

Contents

1 Introduction

postnotes is an endnotes package for LaTeX. Its user interface provides means to print multiple sections of notes along the document, and to subdivide them either automatically – by chapter, by section – or at manually specified places, thus being able to easily handle both numbered and unnumbered headings. The package also provides infrastructure for setting up contextual running headers for printed notes. The default is a simple but useful one, in the form "Notes to pages N–M", but more elaborate ones can be built. When hyperref is loaded, postnotes provides hyperlinked notes, including back links.

Though this feature set is mostly (albeit not completely) available in one or another of the existing endnotes packages for LaTeX, subsets of it exist in individual packages, not necessarily compatible with each other. postnotes brings these features together in one place, with no external dependencies except an up-to-date kernel.

On the technical side, postnotes is peculiar among existing LaTeX packages in this area of functionality by the fact that it does not use an external file to store the notes. Both the notes' contents and its metadata are stored in variables which are later retrieved at the time of printing. In particular, the content of the note is stored and retrieved with "no manipulation" (as in expl3's N/n function signatures) and only gets to be expanded at the time it is meant to be typeset. The .aux file is leveraged to set page labels for the notes, since that particular information has to be retrieved asynchronously but, other than that, variables are employed to pass information around.

This has some advantages. First, as is well known, sending arbitrary content to a file to be read later is not a noiseless process in LaTeX. Thus, not doing so makes things smoother. Second, the external file approach is strictly linear: the notes which were written to the file get printed as such, in the order they were written. Having the notes available as a set of variables allows for some more flexibility than that, through the possibility of pre-processing the notes before printing. It also brings some extra degrees of freedom in storing note metadata, and in restoring part of the environment where the note is called to where the note's content is printed.

2 Loading the package

postnotes can be loaded with the usual:

\usepackage{postnotes}

The package does not accept load-time options, package options must be set using \postnotesetup (see Section 4, Package options).

3 User interface

\postnote   \postnote[⟨options⟩]{⟨text⟩}

Sets a postnote with content ⟨text⟩. A note "mark" is typeset at the place \postnote is called, and ⟨text⟩ is stored to be typeset later, on the next call to \printpostnotes.

The mark is usually determined by the printed representation of the main counter, postnote, but can be manually set too. \postnote can receive a number of ⟨*options*⟩, which are presented in Section 4, Note options.

\postnotesection | \postnotesection[⟨options⟩]{⟨text⟩}

Sets a postnote section with content ⟨*text*⟩. This is the basic interface for making notes subdivisions, and it places ⟨*text*⟩ between the notes where it occurs, at the point the notes are printed by \printpostnotes. For more details and some examples, see Section 5. Its ⟨*options*⟩ are presented in Section 4, Section options.

\printpostnotes | \printpostnotes

Prints the \postnotes set since the last call of \printpostnotes, or since the beginning of the document. For two basic usage illustrations, see Examples 2 and 8.

\postnoteref | \postnoteref⟨*⟩{⟨label⟩}

Typesets a postnote reference to ⟨*label*⟩. Of course, ⟨*label*⟩ must have been set to a particular postnote, which can be done by the standard \label command. The starred version of the command inhibits hyperlinking. When the zref-user package is loaded, a corresponding \postnotezref is also provided, and if zref-hyperref is also loaded, it is hyperlinked as its counterpart.

4   Options

Package options

\postnotesetup | \postnotesetup{⟨options⟩}

Package options can be configured by means of \postnotesetup, which receives options and values in key=value fashion.

heading
\pnheading

The heading option sets the heading for the printed notes or, more generally put, that which is printed at the beginning of \printpostnotes. Its default value depends on the document class in use. If \chapter is defined, the default is:

    \chapter*{\pntitle}
    \@mkboth{\pnheaderdefault}{\pnheaderdefault}

but otherwise, it is:

    \section*{\pntitle}
    \@mkboth{\pnheaderdefault}{\pnheaderdefault}

where \pntitle is localizable string, which by default contains "Notes" (see Section 8), and \pnheaderdefault is a function which takes no arguments, but processes a number of variables, to set a contextual running header for the printed notes (see Section 6). \pnheaderdefault produces a header in the form "Notes to pages N–M", according to

4

the notes in each page. If you prefer, you can redefine \pnheading instead of using the heading option, to the same effect.

format   The format option stores formatting instructions for printing the notes. It is called at \printpostnotes, every time a block of notes is about to start. The default value is \small.

listenv   The listenv option sets the list environment inside which the notes are printed in \printpostnotes. This must be the name of an existing list environment, and postnotes provides two suitable ones for convenience: postnoteslist, which is the default, and postnoteslisthang which typesets the notes with a hanging indent. You can also create your own, with enumitem or otherwise, of course. listenv can also receive the special value none, in which case the notes blocks are not wrapped in a list environment, but rather typeset as plain paragraphs. listenv=none already sets postprintnote to \par for that reason but, when using none, you'll probably also want to set maketextmark, pretextmark, or posttextmark to values different than the defaults.

makemark
maketextmark
\pnthepage
    The makemark and maketextmark options control how the mark is to be typeset, at the point \postnote is called and at the point the note's text is printed at \printpostnotes, respectively. They both can receive three arguments: #1 is the mark itself, and arguments #2 and #3 are, respectively, the start and the end of the hyperlink (hence they must be used in this order, and always in pair). Their default values are:

```
makemark = {#2\hbox{\@textsuperscript{\normalfont#1}}#3} ,
maketextmark = {#2#1.#3} ,
```

At the point maketextmark gets typeset, the \pnthepage variable contains the value of \thepage where its corresponding note was set.

pretextmark
posttextmark
postprintnote
    The options pretextmark, posttextmark, and postprintnote allow to insert additional material in \printpostnotes, respectively, right before the mark, right after the mark, and after the note's content. All in all, when listenv is not none, each note in the list is laid out in the form:

    \item[⟨pretextmark⟩⟨mark⟩⟨posttextmark⟩]⟨note content⟩⟨postprintnote⟩

and when listenv is none, each note is laid out in the form:

    ⟨pretextmark⟩⟨mark⟩⟨posttextmark⟩⟨note content⟩⟨postprintnote=\par⟩

style   style is just a convenience "meta" option which sets a number of "base" options – such as listenv, format, maketextmark, etc. – in order to emulate known styles of printing the notes. It accepts the values endnotes or pagenote so that \printpostnotes works as its counterparts in each of these packages.

hyperref
backlink
    The hyperref option controls the use of hyperref by postnotes and takes values auto, true or false. The default value, auto, makes postnotes use hyperref if it is loaded. true does the same thing, but warns if hyperref is not loaded (hyperref is never loaded for you). false means not to use hyperref regardless of its availability. The backlink option controls whether only a link from the note's mark to its respective text at \printpostnotes is created, or if a back link from the text at \printpostnotes back to where the note's mark is placed is also made available. It is a boolean option, defaults to true, and is only operational if hyperref is not false. These are both preamble only options.

sort   The sort option controls whether the notes queue is sorted or not at \printpostnotes. Normally, the order the notes should be printed is the one in which the notes were

placed along the document. However, in cases where some manual intervention was required, sorting the notes can be quite useful, and difficult to handle in its absence. Two typical examples are: a note inside a float which disturbed the sequence of the postnote counter (see Example 9 in Section 9 for an illustration) and a manually set mark, in which case postnotes also allows to manually set a sort value with the sortnum option of \postnote. Sorting does not cross boundaries of notes sections, as set by \postnotesection, in other words, if notes sections exist, sorting is only ever carried out within the boundaries of each section. This may be a restriction for cases in which floats cross sections' boundaries, but it's the only reasonable thing to do. sort is a boolean option, and defaults to true.

Note options

The options accepted by \postnote[⟨*options*⟩]{⟨*text*⟩} are the following:

mark
markstr   By default, the mark generated by \postnote is determined by the printed representation of the postnote counter, \thepostnote, which is stepped when \postnote is called. But the mark and markstr options allow you to manually set it, in case you want, or need, to do so. When either mark or markstr is manually set, the postnote counter is not stepped. The difference between them is that mark must receive a number as value, and uses its value to also set the sortnum option, while markstr, differently from the optional argument of \footnote, can receive a string as value which is directly used as the mark, but it does not set sortnum.

sortnum   Normally, the sort value used for sorting the notes queue (see the sort package option above) is determined by the value of the postnote counter (that is, by \the\c@postnote, and not by its printed representation, \thepostnote). But you may specify this sort value manually with the sortnum option, typically, when you have also manually specified the mark. It receives a floating point number as value. So, for example, if one needed to insert a note between notes 2 and 3, without disturbing the numbering of other notes, one could use \postnote[markstr=2*,sortnum=2.5]{⟨*text*⟩}.

nomark   The nomark option makes \postnote inhibit the typesetting of the mark. Of course, normally, we do want the visual cue of the mark, but the intended use case for this option is for a \postnote with nomark to be paired with a \postnoteref, so as to be able to typeset a note in places where doing so directly may be problematic. For a practical example and an illustration on how to use it, see Example 10 in Section 9.

label   The label option sets a standard \label named with the value given to the option. When the zref-user package is loaded, a corresponding zlabel option is also provided. See Section 7 for details about cross-referencing.

Section options

The options accepted by \postnotesection[⟨*options*⟩]{⟨*text*⟩} are the following:

name   For the purposes of building running headers, each \postnotesection can be identified by its name. This is mainly intended to support unnumbered headings, but its mechanism is general. The name of a note section is made available for the first and last note on a given page through the variables \pnhdnamefirst and \pnhdnamelast at the

moment the header of the page is typeset. For details on how to use these variables, see Section 6.

By default, \postnotesection stores its ⟨*text*⟩ argument with "no manipulation". The exp option allows one to fully expand (e-type expansion) ⟨*text*⟩ in place before storing it. It is a boolean option, and the option given with no value is equivalent to exp=true.

5   Notes sections

As mentioned above, \postnotesection is the basic interface for subdividing the notes when printed. For those familiar with it, this command is postnotes's equivalent to endnotes' \addtoendnotes. It has the same intended use – to add text or commands along the notes' sequence at the point it is called – and the way it works is quite similar to \addtoendnotes. But there are some differences, prominently a \postnotesection is skipped at \printpostnotes if it contains no notes. In other words, if two (or more) calls of \postnotesection occur in immediate sequence, with no \postnote in between, the latter call takes precedence over the former, instead of being accumulated in the queue. This is intended to facilitate the automation of the subdivision of the notes. So, one can, for example, use a hook to \chapter and not have to worry if a chapter with no notes will generate an empty section inside \printpostnotes, e.g., by the call to \chapter* at the table of contents, and so on. Or, one can use the heading number for the automated case, but be able to override it manually for an occasional unnumbered one. For this reason, a more semantic name was chosen for it, instead of the generic "add to".

\pnthechapter   Just like with \postnote, the contents of \postnotesection are not expanded in
\pnthesection   place, but rather stored with "no manipulation" to be typeset later at \printpostnotes.
\pnthechapternextnote   For this reason, some contextual information is stored at the place \postnotesection is
\pnthesectionnextnote   called, and made available at the point it's content gets expanded by means of some
\pnidnextnote   variables (you can use \postnotesection[exp] instead, in which case these variables are of little use). When the content of a notes section gets typeset, \pnthechapter contains the value of \thechapter where \postnotesection was originally called. Similarly, \pnthesection contains the value of \thesection. \pnthechapternextnote and \pnthesectionnextnote are meant to support the automation of the notes subdivision by using hooks to sectioning commands, which is a quite natural way to do this. However, since it may be problematic to hook *after* sectioning commands – \section, for example, figures prominently in the documentation of ltcmdhooks as a case of "look ahead" command for which the after hook is not supported – we will typically want to hook *before* them. But, at this point the values of the chapter or section counters have not yet been stepped, therefore \thechapter and \thesection do not correspond to what we would like to refer to. For this reason, \pnthechapternextnote contains the value of \thechapter at the point the "next note" is placed (a \postnote, that is), the first in the chapter, and already inside it, thus with an expected value of the chapter counter. Similarly, \pnthesectionnextnote contains the value of \thesection for the "next note". \pnidnextnote, in turn, stores the Id number of the "next note". Of course, the "next note" variables are *proxies*, but they are meant to support the automation of the subdivision of the notes through the use of before hooks to the document's sectioning commands, in which case the subdivision of the notes will correspond to

the document's structure and, given empty notes sections are skipped, the values will be the ones we are interested in. But more complex use cases may require something different. Either way, it is up to the user to judge whether the *proxy* is a good one for their use case, the variables just do what they say, and contain the values of interest for the "next note".

This is meant to be simple. Some examples might make things more concrete. At its most basic, \postnotesection can just be set manually:

Example 2: Basic usage

```
\documentclass{book}
\usepackage{postnotes}
\usepackage{hyperref}
\begin{document}
\chapter{First chapter}
\postnotesection{\section*{Notes to chapter \pnthechapter}}
Foo.\postnote{Foo note.}
Bar.\postnote{Bar note.}
\chapter{Second chapter}
\postnotesection{\section*{Notes to chapter \pnthechapter}}
\setcounter{postnote}{0}
Baz.\postnote{Baz note.}
Boo.\postnote{Boo note.}
\printpostnotes
\end{document}
```

The document in Example 2 resets the postnote counter for each chapter, and manually sets notes sections by chapter, which results in \printpostnotes being correspondingly subdivided. But it is easy to make this automatic:

Example 3: Automating notes subdivision with a hook

```
\documentclass{book}
\usepackage{postnotes}
\AddToHook{cmd/chapter/before}{%
  \postnotesection{\section*{Notes to chapter \pnthechapternextnote}}}
\counterwithin*{postnote}{chapter}
\usepackage{hyperref}
\begin{document}
\tableofcontents
\chapter{First chapter}
Foo.\postnote{Foo note.}
Bar.\postnote{Bar note.}
\chapter{Second chapter}
Baz.\postnote{Baz note.}
Boo.\postnote{Boo note.}
\printpostnotes
\end{document}
```

Example 3 uses the cmd/chapter/before hook, and thus \pnthechapternextnote to retrieve the correct chapter number for \postnotesection, as explained above. The counter is reset every chapter by means of \counterwithin*. Note that the call to \chapter* inside \tableofcontents does not generate a spurious notes section at

\printpostnotes (as long as you don't place a note in a sectioning command with no short argument, which you shouldn't do anyway). But what if we have, among mostly numbered chapters, an ocasional unnumbered one?[1] \pnthechapternextnote wouldn't possibly work in this case. Since immediately successive calls to \postnotesection override the previous ones, it is straightforward to just manually adjust the exception:

Example 4: Fine-tuning automation

```
\documentclass{book}
\usepackage{postnotes}
\AddToHook{cmd/chapter/before}{%
  \postnotesection{\section*{Notes to chapter \pnthechapternextnote}}}
\counterwithin*{postnote}{chapter}
\usepackage{hyperref}
\begin{document}
\tableofcontents
\chapter*{Introduction}
\postnotesection{\section*{Notes to the introduction}}
Intro.\postnote{Intro note.}
\chapter{First chapter}
Foo.\postnote{Foo note.}
Bar.\postnote{Bar note.}
\chapter{Second chapter}
Baz.\postnote{Baz note.}
Boo.\postnote{Boo note.}
\printpostnotes
\end{document}
```

If one wants to use section names/titles, the technique above (of using something similar to \pnthechapternextnote) is less useful, since if the first note in the section occurs within a subsection we would lose the information of interest. So we have a little more work to do in this case. Example 5 uses the cmd/chapter/before hook to store the value of \@currentlabelname in a dedicated variable after the next call to \NR@gettitle (presuming the use of nameref, through hyperref). We then store the value of this variable for each note using the postnotes/note/store hook and the note's Id number \l_postnotes_note_id_tl. Finally we can retrieve this stored value using \pnidnextnote inside \postnotesection. Indeed, this example is also meant to illustrate the general technique for storing/restoring variables of interest for this purpose.

Example 5: Section names

```
\documentclass{book}
\usepackage{postnotes}
\ExplSyntaxOn
\tl_new:N \g_my_currentname_tl
\prop_new:N \g_my_names_prop
\AddToHook{cmd/chapter/before}{
  \AddToHookNext{cmd/NR@gettitle/after}{
    \tl_gset:Nv \g_my_currentname_tl { @currentlabelname } } }
```

---

[1]The example here counts on the lucky circumstance of having only a single initial unnumbered section. But, in general, if that's not the case, \counterwithin* is insufficient and the resetting of the postnote counter at unnumbered sections must be handled somehow else.

```
\AddToHook{postnotes/note/store}{
  \prop_gput:NeV \g_my_names_prop
    { \l_postnotes_note_id_tl } \g_my_currentname_tl }
\AddToHook{cmd/chapter/before}{
  \postnotesection{
    \section*{Notes~to~\prop_item:NV \g_my_names_prop \pnidnextnote}}}}
\ExplSyntaxOff
\counterwithin*{postnote}{chapter}
\usepackage{hyperref}
\begin{document}
\chapter*{Introduction}
Intro.\postnote{Intro note.}
\chapter{First chapter}
Foo.\postnote{Foo note.}
Bar.\postnote{Bar note.}
\chapter{Second chapter}
Baz.\postnote{Baz note.}
Boo.\postnote{Boo note.}
\printpostnotes
\end{document}
```

While postnotes goes through great lengths to avoid tampering with sectioning commands, the fact that in general we cannot use cmd hooks after \chapter or \section does complicate things. And Example 5 is indeed a good illustration of how a supposedly simple task can become quite convoluted if we are not allowed to observe the variables of interest *after* the sectioning commands. However, things are quite different from the perspective of a user, considering the problem at the document level. In this case, the definition of the sectioning commands is known, and unique, so that it may make sense to use of the traditional technique of storing the definition of the sectioning command, and then redefining it to do what it used to, plus something else.[2] In which case, we can set \postnotesections with full generality and flexibility.

Example 6: Redefining sections

```
\documentclass{book}
\usepackage{postnotes}
\counterwithin*{postnote}{chapter}
\NewCommandCopy\origlatexchapter\chapter
\RenewDocumentCommand{\chapter}{som}{%
  \IfBooleanTF{#1}{%
    \origlatexchapter*{#3}%
    \setcounter{postnote}{0}%
    \postnotesection{\section*{Notes to ``#3''}}%
  }{%
    \IfNoValueTF{#2}{%
      \origlatexchapter{#3}%
    }{%
      \origlatexchapter[#2]{#3}%
```

---

[2]'egreg' commonly applies the technique for the same purpose in answers using endnotes at TeX.SX. For example: `https://tex.stackexchange.com/a/62425`, `https://tex.stackexchange.com/a/109566`, and `https://tex.stackexchange.com/a/85001`. But things are somewhat simpler with postnotes, since there's no need to handle the case of sections with no notes, given that empty \postnotesections are already skipped.

```
    }%
    \postnotesection{\section*{Notes to chapter \pnthechapter}}%
  }%
}
\usepackage{hyperref}
\begin{document}
\chapter*{Introduction}
Intro.\postnote{Intro note.}
\chapter{First chapter}
Foo.\postnote{Foo note.}
Bar.\postnote{Bar note.}
\chapter{Second chapter}
Baz.\postnote{Baz note.}
Boo.\postnote{Boo note.}
\printpostnotes
\end{document}
```

Things could easily get fancier, of course. But that's the basic structure.

6 Headers

postnotes' support for running headers comprises a basic header, enabled by default, generating headers in the form "Notes to pages N–M", but also some infrastructure for users to build more elaborate ones.

The default headers are generated by the function \pnheaderdefault which, as we saw in Section 4 is used to set the headers in option heading (with \@mkboth). So, the default headers are enabled through that particular setting.

Examining the definition of \pnheaderdefault is possibly the most direct way to explore how the feature is meant to work.

```
\ExplSyntaxOn
\NewDocumentCommand \pnheaderdefault {}
  {
    \tl_if_eq:NNTF \pnhdpagefirst \pnhdpagelast
      { \pnhdnotes{} ~ \pnhdtopage{} ~ \pnhdpagefirst }
      { \pnhdnotes{} ~ \pnhdtopages{} ~ \pnhdpagefirst -- \pnhdpagelast }
  }
\ExplSyntaxOff
```

\pnhdnotes, \pnhdtopage, and \pnhdtopages are localizable strings, which by default respectively contain "Notes", "to page", and "to pages" (see Section 8). Let's replace them to examine the interesting part of the definition:

```
\ExplSyntaxOn
\NewDocumentCommand \pnheaderdefault {}
  {
    \tl_if_eq:NNTF \pnhdpagefirst \pnhdpagelast
      { Notes~to~page~ \pnhdpagefirst }
      { Notes~to~pages~ \pnhdpagefirst -- \pnhdpagelast }
  }
\ExplSyntaxOff
```

\pnhdpagefirst and \pnhdpagelast store the value of \thepage for the first and the last notes (where these notes were originally placed) of the current page (at the point they are being printed). These variables are updated every page along the span of \printpostnotes to the values corresponding to their respective first and last note (which start on that page), so that these values are available at the moment the headers get to be typeset. Hence, what the definition of \pnheaderdefault does is to use these variables to build a rule in the form: "if the page of the first and last notes are equal, write a singular form and just one value but, if they are different, write a plural form and a range of both values". \pnhdchapfirst, \pnhdchaplast, \pnhdsectfirst, and \pnhdsectlast provide the same for \thechapter and \thesection. \pnhdnamefirst and \pnhdnamelast contain the name of the notes section, the one given with the name option of \postnotesection (and are empty in case no name was provided).

With that in hand, fancier headers can be built. For example, if we'd like headers in the form "Notes to chapters A–C, pages N–M", we could define:

```
\ExplSyntaxOn
\NewDocumentCommand \mypnheader {}
  {
    \tl_if_eq:NNTF \pnhdchapfirst \pnhdchaplast
      { Notes~to~chapter~\pnhdchapfirst,~ }
      { Notes~to~chapters~\pnhdchapfirst -- \pnhdchaplast,~ }
    \tl_if_eq:NNTF \pnhdpagefirst \pnhdpagelast
      { page~\pnhdpagefirst }
      { pages~\pnhdpagefirst -- \pnhdpagelast }
  }
\ExplSyntaxOff
```

and then set heading to use \mypnheader instead of \pnheaderdefault. This definition should work well as long as all the chapters (containing notes) are numbered, but if unnumbered ones come into play, again we can fine-tune the automation, adjusting for the exception. That's the purpose of the name option for \postnotesection, and of \pnhdnamefirst and \pnhdnamelast. Example 7 illustrates their use (of course, the use of lipsum is just for demonstration):

Example 7: Fancy headers

```
\documentclass{book}
\usepackage{postnotes}
\postnotesetup{
  heading = {%
    \chapter*{\pntitle}
    \markboth{\mypnheader}{\mypnheader}%
  } ,
}
\counterwithin*{postnote}{chapter}
\AddToHook{cmd/chapter/before}{%
  \postnotesection{\section*{Notes to chapter \pnthechapternextnote}}%
}
\ExplSyntaxOn
\NewDocumentCommand \mypnheader {}
  {
    \bool_case:nF
```

```
        {
          {
            \str_if_eq_p:ee { \pnhdnamefirst } { intro } &&
            \str_if_eq_p:ee { \pnhdnamelast } { intro }
          }
          { Notes~to~the~introduction,~ }
          {
            \str_if_eq_p:ee { \pnhdnamefirst } { intro } &&
            ! \str_if_eq_p:ee { \pnhdnamelast } { intro }
          }
          { Notes~to~the~introduction~and~chapter~\pnhdchaplast{},~ }
        }
        {
          \tl_if_eq:NNTF \pnhdchapfirst \pnhdchaplast
            { Notes~to~chapter~\pnhdchapfirst{},~ }
            { Notes~to~chapters~\pnhdchapfirst{} -- \pnhdchaplast{},~ }
        }
      \tl_if_eq:NNTF \pnhdpagefirst \pnhdpagelast
        { page~\pnhdpagefirst{} }
        { pages~\pnhdpagefirst{} -- \pnhdpagelast{} }
    }
  \ExplSyntaxOff
  \usepackage{hyperref}
  \usepackage{lipsum}
  \ExplSyntaxOn
  \NewDocumentCommand \demochapter { m }
    { \prg_replicate:nn { #1 } { \lipsum[1-2]\postnote{\lipsum[3]}\par } }
  \ExplSyntaxOff
  \begin{document}
  \tableofcontents
  \chapter*{Introduction}
  \postnotesection[name=intro]{\section*{Notes to the introduction}}
  \demochapter{12}
  \chapter{Chapter 1}
  \demochapter{15}
  \chapter{Chapter 2}
  \demochapter{15}
  \printpostnotes
  \end{document}
```

7 Cross-references

Cross-referencing with postnotes works in a pretty standard way: set a label, make references to it. However, there are two ways to set a label to a note. One can either set a label with the label option of \postnote, or one can directly set it with the standard \label as part of the note's content. They are both valid, but they are not equivalent, they have different meanings and, accordingly, behave differently.

The label set with the label option is set at the place where \postnote is. The label set with \label in the note's content, is just stored, and only gets expanded when this

content gets to be typeset, at `\printpostnotes`. In short, the label option belongs to the "mark", while the `\label` set in the content belongs to the "text".

Of course, the data stored in each label will correspond to this difference. Even if the plain `\ref` to both will get the same value (the mark), a `\pageref` will be different, the links to either will point to different places, etc.

8 Localization

<div style="text-align: right"><code>\pntitle</code>
<code>\pnhdnotes</code>
<code>\pnhdtopage</code>
<code>\pnhdtopages</code></div>

postnotes uses a few localized strings, stored in the variables `\pntitle`, `\pnhdnotes`, `\pnhdtopage`, and `\pnhdtopages`. The first one is used in the default value of the heading option and defaults to "Notes". The remaining three are used in `\pnheaderdefault` (and thus, indirectly also in the heading option) and their respective defaults are: "Notes", "to page", and "to pages". So, if you changed the default value of heading and is not using `\pnheaderdefault`, you don't have to worry about them.

These strings will automatically adjust to the document language, set either by babel or by polyglossia, *if* the language is supported by postnotes. Currently supported are English, German, French, and Portuguese. Either way, you can always change these variables to the values of your preference. If you are not using either babel or polyglossia, you can do so directly, for example, with:

> `\renewcommand*{\pntitle}{My title}`

However, with babel or polyglossia, and specially in a multi language document, you must use the appropriate hooks of your language package for this, otherwise, the next call to `\selectlanguage` will override your settings. For babel you should use:

> `\addto\extras`⟨language⟩`{\renewcommand*{\pntitle}{My title}}`

and for polyglossia:

> `\gappto\captions`⟨language⟩`{\renewcommand*{\pntitle}{My title}}`

9 Further examples

This section collects some further usage examples which did not fit into the previous sections, but might still be of interest.

Example 8 illustrates a basic procedure of how to obtain a note section for each chapter of a book, by calling `\printpostnotes` at the end of each chapter:

Example 8: Notes for each chapter

```
\documentclass{book}
\usepackage{postnotes}
\postnotesetup{heading={\section*{\pntitle}}}
\usepackage{hyperref}
\begin{document}
\chapter{First chapter}
Foo.\postnote{Foo note.}
Bar.\postnote{Bar note.}
```

```
\printpostnotes
\chapter{Second chapter}
\setcounter{postnote}{0}
Baz.\postnote{Baz note.}
Boo.\postnote{Boo note.}
\printpostnotes
\end{document}
```

Example 9 shows a case of a float which disturbs the order of the notes. It demonstrates a (traditional) technique to deal with the situation, by setting a manual mark and adjusting the counter where appropriate. It also illustrates the role the sorting of notes can have, by producing not only correctly ordered note marks (as a result of the manual adjustments), but also correctly ordered printed notes (as a result of sorting):

Example 9: Sorting and floats

```
\documentclass{book}
\usepackage{postnotes}
\usepackage{hyperref}
\begin{document}
\chapter{First chapter}
\postnote{1}
\postnote{2}
\begin{table}[p]
  \caption{Table}
  Table.\postnote[mark=5]{3}
\end{table}
\postnote{4}
\postnote{5}
\stepcounter{postnote}
\clearpage
\postnote{6}
\printpostnotes
\end{document}
```

Example 10 illustrates a couple of techniques to handle long captions. Captions pose a problem to \postnote because, if you set a \postnote inside a standard caption, whose text is long enough to require two lines, the content of the caption ends up being typeset twice: once to check if it would have fitted in a single line, the second to typeset the two lines since it didn't fit in one. This triggers the postnote counter to be stepped twice (and any other counter that happens to be there too). One way to handle the situation is, if you know(!) that the caption takes two lines, you can decrement the counter just before it, and place the note directly in the caption, this will get you a correctly numbered note. Another way is to use the pairing between a nomark \postnote and \postnoteref: place a note outside the caption (but close to it, since its position will determine the anchor for the backlink) with the nomark option, set a label inside it and, inside the caption make a \postnoteref to the label. In practice:

Example 10: Long caption

```
\documentclass{article}
\usepackage[textwidth=8cm]{geometry}
\usepackage{postnotes}
```

```
\usepackage{hyperref}
\begin{document}
\begin{figure}
  Figure.
  \addtocounter{postnote}{-1}
  \caption[Caption for LOF]{A long caption, long enough to require two
    lines\postnote{A note.}}
\end{figure}
\begin{figure}
  Figure.
  \postnote[nomark]{\label{en:1}A note.}%
  \caption[Caption for LOF]{A long caption, long enough to require two
    lines\postnoteref{en:1}}
\end{figure}
\printpostnotes
\end{document}
```

10   Acknowledgments

Some people have kindly contributed to postnotes. Suggestions, ideas, insightful comments, solutions to problems, bug reports were generously provided by (in chronological order): Ulrike Fischer, David Carlisle, Moritz Wemheuer, Joseph Wright, 'SwitWu', and Jonathan P. Spratte ('Skillmon').

The package's language support have been provided or improved thanks to: 'Pika78' (French) and Herbert Voß (German).

If I have inadvertently left anyone off the list I apologize, and please let me know, so that I can correct the oversight.

Thank you all very much!

11   Change history

A change log with relevant changes for each version, eventual upgrade instructions, and upcoming changes, is maintained in the package's repository, at `https://github.com/gusbrs/postnotes/blob/main/CHANGELOG.md`. The change log is also distributed with the package's documentation through CTAN upon release so, most likely, `texdoc postnotes/changelog` should provide easy local access to it. An archive of historical versions of the package is also kept at `https://github.com/gusbrs/postnotes/releases`.