The perpage package
Version 2.0

David Kastrup*

2014/10/25

1 Description

The perpage package adds the ability to reset counters per page and/or keep their occurences sorted in order of appearance on the page.

It works by attaching itself to the code for \stepcounter and will then modify the given counter according to information written to the .aux file, which means that multiple passes may be needed. Since it uses the internals of the \label mechanism, the need for additional passes will get announced by LaTeX as "labels may have changed".

\MakePerPage

```
\MakePerPage[2]{footnote}
```

will start footnote numbers with 2 on each page (the optional argument defaults to 1). 2 might be a strange number, unless you have used something like

```
\renewcommand\thefootnote{\fnsymbol{footnote}}
```

and want to start off with a dagger. The starting value must not be less than 1 so that the counter logic can detect the reset of a counter reliably.[1] It could be a good idea to redefine \@cnterr if you use a format with limited range: at the first pass, footnotes are not reset across pages and things like \fnsymbol will quickly run out of characters to use.

\theperpage

If you want to label things also on a per page base, for example with

```
\renewcommand{\thefigure}{\thepage-\arabic{figure}}
```

you'll have the problem that \thepage is updated asynchronously with the real page, since TeX does not know which page the figure will end up. If you have used the perpage package for modifying the figure counter, however, at the point where

*dak@gnu.org

[1] This unfortunately means that you can't just use \alph in order to get figures on page 10 numbered as "10", "10a", "10b".

the counter is incremented, the macro `\theperpage` will be set to the correct value corresponding to the actual page location. Note that this macro is shared between all counters, so advancing a different counter under control of `perpage` will render `\thefigure` incorrect.

`\MakeSorted`

```
\MakeSorted{figure}
```

will make the `figure` counter get 'sorted': this means that counter values will be assigned in order of appearance in the output, not in order of appearance in the source code. For example, the order of interspersed one- and two-column figures might get mixed up by LaTeX in the output. Making the counter sorted will fix the order to match the order of appearance. A similar problem is when ordinary footnotes are present in floating material (this does not work in standard LaTeX, but might do so when using `manyfoot.sty` or `bigfoot.sty`): this might jumble their order in the output, and making their counter sorted will make things appear fine again.

While this would not fix the order in the table of figures, fortunately the respective entries actually get written out in order of appearance in the output anyway, so this indeed fixes the problem.

Manually setting the counter does not lead to reliable results in general; as a special case, however, resetting it to zero is recognized (this can also happen automatically when the counter is dependent on some other counter). The point where it is reset in the source code separates 'count groups': everything in the source before that point is assigned sorted numbers separately from everything appearing behind it, and the sequence numbers start again with 1 with the first item appearing in the output (not the source) from the new count group.

`\MakeSortedPerPage`

```
\MakeSortedPerPage[2]{table}
```

will make the table numbers restart at 2 on each page *and* will keep them sorted, to boot. Introducing new count groups by resetting the counter to 0 manually will not work, as it is not clear how to handle count groups scattered between pages. You will usually want to use something like

```
\renewcommand{\thefigure}{\theperpage-\arabic{figure}}
```

to go along with a page-wise figure number.[1] Note that it would be quite silly to start the ranges with 2: this is just an example for the optional argument in case that you ever need it.

`\AddAbsoluteCounter`

```
\AddAbsoluteCounter{equation}
```

will create a counter `absequation` that will advance together with the counter `equation` but will not get reset along with it. This is not sorted into output

[1] Note the use of `\theperpage` here, see above.

order, but just runs along with the sequence in the source file. As a special case, the counter `abspage` is created in this manner and `\theabspage` is defined as an arabic number that works in the same contexts as `\page` (namely, gets properly deferred by `\protected@write`).

2 The documentation driver

This is the default driver for typesetting the docs. Running it through as a separate file will include the code section. Running the original `.dtx` file through will omit the code.

```
⟨*driver⟩
\documentclass{ltxdoc}
\usepackage{perpage}
\MakePerPage{footnote}
\begin{document}
\OnlyDescription
⟨driver⟩ \AlsoImplementation
\DocInput{perpage.dtx}
\end{document}
⟨/driver⟩
```

3 The package interfaces

First identification.

```
⟨*style⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{perpage}[2014/10/25 2.0 Reset/sort counters per page]
```

`\pp@cl@begin` `\pp@cl@end` These macros are considerable tricky. They are called as artificial 'dependent' counters when the counter they are hooked into is advanced. The way in which those counters are called are one of the following:

```
\def\@stpelt#1{\global\csname c@#1\endcsname \z@}
```

which is the default way of resetting a subordinate counter used in LaTeX, or

```
\def\@stpelt#1{\global\csname c@#1\endcsname \m@ne\stepcounter{#1}}
```

which is a little present from `fixltx2e.sty` as of 2014/05/01, quite complicating this feat.

The startup code swallows either `\global \advance` or `\global`.

```
\def\pp@cl@begin{\z@\z@ \begingroup}
```

The command used for ending our fake counters checks for the `\m@ne` condition. We don't want to bump our auxiliary counters twice, so we remove the following `\stepcounter` command. Things will go haywire if there is none, of course.

```
\def\pp@cl@end{\afterassignment\pp@cl@end@ii \count@}
\def\pp@cl@end@ii{%
  \relax
  \expandafter\endgroup
  \ifnum\count@<\z@
    \expandafter\pp@cl@end@iii
  \fi}
\def\pp@cl@end@iii\stepcounter#1{}
```

`\AddAbsoluteCounter` adds a counter with prefix `abs` to a given counter. It typesets as an arabic number and never gets reset. And it is advanced whenever the unprefixed counter gets advanced.

```
\newcommand\AddAbsoluteCounter[1]
{\@ifundefined{c@abs#1}{%
    \expandafter\newcount\csname c@abs#1\endcsname
    \global\value{abs#1}\@ne
    \global\expandafter\let\csname cl@abs#1\endcsname\@empty
    \expandafter\xdef\csname theabs#1\endcsname{%
      \noexpand\number \csname c@abs#1\endcsname}%
    \global\@namedef{c@pabs@#1}{\pp@cl@begin
      \stepcounter{abs#1}%
      \pp@cl@end}%
    \@addtoreset{pabs@#1}{#1}}{}}
```

`\c@perpage` We now create the absolute counter `perpage`:

```
\AddAbsoluteCounter{page}
```

`\theabspage` This has to be specially defined so that it will expand as late as `\thepage` does. Several commands set the latter temporarily to `\relax` in order to inhibit expansion, and we will more or less imitate its behavior when found set in that manner.

```
\def\theabspage{\ifx\thepage\relax
    \noexpand\theabspage
  \else
    \number\c@abspage
  \fi}
```

Here follow the three commands for defining counters per page:

`\MakePerPage` This creates a counter reset per page. An optional second argument specifies the starting point of the sequence.

```
\newcommand*\MakePerPage[2][\@ne]{%
  \pp@makeperpage{#2}\c@pchk@{#1}}
```

`\MakeSorted` This will create a counter sorted in appearance on the page. No optional argument is given: set the counter to a desired starting value manually if you need to. Resetting it to zero will start a new count group, setting it to other values is probably not reliable.

```
\newcommand*\MakeSorted[1]{%
  \setcounter{#1}{\z@}%
  \pp@makeperpage{#1}\c@schk@{\@ne}}
```

\MakeSortedPerPage This will create output in sorted order, reset on each page. Use an optional argument to specify the starting value per page. This must not be 0, unfortunately.

```
\newcommand*\MakeSortedPerPage[2][\@ne]{%
  \pp@makeperpage{#2}\c@spchk@{#1}}
```

All of those must only occur in the preamble since we can't do the initialization of the counter values otherwise.

```
\@onlypreamble\MakePerPage
\@onlypreamble\MakeSorted
\@onlypreamble\MakeSortedPerPage
```

4 Internals

It works in the following manner: The basic work is done through attaching help code to the counter's reset list. Each counter has an associated absolute id that is counted through continuously and is never reset, thus providing a unique frame of reference. Sorted and perpage counters work by writing out information to the `.aux` file.

The information we maintain for each counter while processing the source file are:

- The absolute counter id.
- The last counter value so that we can check whether the sequence has been interrupted.
- The current scope id.
- Its starting value.

The information written to the file consists of:

- The absolute counter id.
- The current scope id.
- The scope's starting value.
- The absolute counter id of a superior counter.

Sorted counters work by writing out the current absolute id and range id into the `.aux` file each time the counter gets incremented. Whenever the counter is changed in a manner different from being incremented, a new counter scope gets started. Each counter scope has its own independently assigned counter numbers and is associated with its absolute id starting value. So as each counter

is incremented, we write out the triple of current absolute id, counter scope and initial value for the scope. Scope changes when a value assigned from the file differs from the 'natural' value. When the file is read in, counter movements are tracked. Each counter that does not have its 'natural' value, is having a counter setting recorded.

The stuff works by adding a pseudo-reset counter to the counter's dependent counter list.

`\pp@makeperpage` This does the relevant things for modifying a counter. It defines its reset value, it defines the correspoding absolute counter. The absolute counter serves a double function: it is also used for assigning numbers while reading the `.aux` file. For this purpose it is assigned the initialized values here and in the enddocument hook (which is called before rereading the `.aux` file and checking for changed labels), while the counter is reset to zero at the start of the document.

```
\def\pp@makeperpage#1#2#3{%
  \global\expandafter\mathchardef\csname c@pp@r@#1\endcsname=#3\relax
  \global\@namedef{c@pchk@#1}{#2{#1}}%
  \newcounter{pp@a@#1}%
  \setcounter{pp@a@#1}{#3}%
  \addtocounter{pp@a@#1}\m@ne
  \@addtoreset{pchk@#1}{#1}%
  \AtBeginDocument{\setcounter{pp@a@#1}\z@}%
  \edef\next{\noexpand\AtEndDocument
    {\noexpand\setcounter{pp@a@#1}{%
        \number\value{pp@a@#1}}}}\next}
\@onlypreamble\pp@makeperpage
```

`\pp@chkvlist` Check for an empty vertical list. If we have one, that is worth warning about.

```
\def\pp@chkvlist{%
  \ifcase
    \ifvmode
      \ifx\lastnodetype\@undefined
        \ifdim-\@m\p@=\prevdepth\ifdim\lastskip=\z@\ifnum\lastpenalty=\z@
          \@ne
        \fi\fi\fi
      \else
        \ifnum\lastnodetype=\m@ne \@ne \fi
      \fi
    \fi \z@
  \or
    \PackageWarning{perpage}{\string\stepcounter\space probably at start of
      vertical list:^^JYou might need to use \string\leavevmode\space
      before it to avoid vertical shifts}%
  \fi}
```

`\pp@fetchctr` `\theperpage` This fetches the counter information and puts it into `\pp@label`, `\pp@page` and (globally) into `\theperpage`.

```
\def\pp@fetchctr#1{\expandafter\expandafter\expandafter\pp@fetchctrii
```

```
  \csname pp@r@#1@\number\value{pp@a@#1}\endcsname
  {}{}}

\global\let\theperpage\@empty

\def\pp@fetchctrii#1#2#3{\def\pp@label{#1}%
  \def\pp@page{#2}%
  \gdef\theperpage{#3}}
```

Ok, let's put together all the stuff for the simplest case, counters numbered per page without sorting:

`\c@pchk@` This is the code buried into to the reset list. When the reset list is executed in the context of advancing a counter, we call something like

```
\global\c@pchk@{countername}\z@
```

since the reset list expected a counter here instead of some generic command. That is the reason we start off this command by giving `\global` something to chew on.

```
\def\c@pchk@#1{\pp@cl@begin
```

Now we fetch the page value corresponding to the not yet adjusted value of the absolute counter to see whether the previous counter advance happened on the same page.

```
  \pp@fetchctr{#1}\let\next\pp@page
  \addtocounter{pp@a@#1}\@ne
  \pp@fetchctr{#1}%
```

We compare the pages for current and last advance of the counter. If they differ, we reset the counter to its starting value. We do the same if the counter has been reset to zero manually, likely by being in the reset list of some other counter.

```
  \ifcase\ifx\next\pp@page\else\@ne\fi
    \ifnum\value{#1}=\z@\@ne\fi\z@
  \else
    \setcounter{#1}{\value{pp@r@#1}}%
  \fi
  \pp@writectr\pp@pagectr{#1}{\noexpand\theabspage}}
```

`\pp@writectr` This is the common ending of all pseudo reset counters. It writes out an appropriate command to the `.aux` file with all required information. We try to replicate any sentinel kerns or penalties.

```
\def\pp@writectr#1#2#3{\edef\next{%
    \string#1{#2}{\number\value{pp@a@#2}}{#3}{\noexpand\thepage}}%
  \pp@chkvlist
  \dimen@=\lastkern
  \ifdim\dimen@=\z@ \else \unkern\fi
  \count@=\lastpenalty
  \protected@write\@auxout{}{\next}%
  \ifdim\dimen@=\z@
```

```
    \penalty \ifnum\count@<\@M \@M \else \count@ \fi
  \else \kern\dimen@\fi
  \pp@cl@end}
```

`\pp@labeldef` This is a helper macro.

```
\def\pp@labeldef#1#2#3#4#5{\@newl@bel{pp@r@#2}{#3}{{#1}{#4}{#5}}}
```

`\pp@pagectr` This is the workhorse for normal per page counters. It is called whenever the `.aux` file is read in and establishes the appropriate information for each counter advancement in a pseudolabel.

```
\def\pp@pagectr#1#2#3#4{\@ifundefined{c@pp@a@#1}{}{%
    \addtocounter{pp@a@#1}\@ne
    \expandafter\pp@labeldef\expandafter
      {\number\value{pp@a@#1}}{#1}{#2}{#3}{#4}}}
```

`\c@schk@` This is called for implementing sorted counters. Sorted counters maintain a "count group", and the values in each count group are numbered independently from that of other count groups. Whenever a counter is found to have been reset, it will start a new count group. At the end of document, the count group counters need to get reset, too, so that the check for changed `.aux` files will still work.

```
\def\c@schk@#1{\pp@cl@begin
  \addtocounter{pp@a@#1}\@ne
  \ifnum\value{#1}=\@ne
    \expandafter\xdef\csname pp@g@#1\endcsname{\number\value{pp@a@#1}}%
    \edef\next{\noexpand\AtEndDocument{\global\let
      \expandafter\noexpand\csname pp@g@#1@\number\value{pp@a@#1}\endcsname
      \relax}}\next
  \fi
  \pp@fetchctr{#1}%
  \ifx\pp@page\@empty
  \else \setcounter{#1}{\pp@label}\fi
  \pp@writectr\pp@spagectr{#1}{\csname pp@g@#1\endcsname}}%
```

`\pp@spagectr` This is the code advancing the respective value of the appropriate count group and assigning the label.

```
\def\pp@spagectr#1#2#3#4{\@ifundefined{c@pp@a@#1}{}{%
    \count@0\csname pp@g@#1@#3\endcsname
    \advance\count@\@ne
    \expandafter\xdef\csname pp@g@#1@#3\endcsname{\number\count@}%
    \expandafter\pp@labeldef\expandafter
      {\number\count@}{#1}{#2}{#3}{#4}}}
```

`\c@spchk@` And this finally is the counter advance code for sorted counters per page. Basically, we just use one count group per page. Resetting a counter manually will not introduce a new count group, and it would be hard to decide what to do in case count groups and page positions overlap.

```
\def\c@spchk@#1{\pp@cl@begin
  \addtocounter{pp@a@#1}\@ne
```

```
  \pp@fetchctr{#1}%
  \ifx\pp@page\@empty
  \else \setcounter{#1}{\pp@label}\fi
  \pp@writectr\pp@ppagectr{#1}{\noexpand\theabspage}}

\pp@ppagectr
\def\pp@ppagectr#1#2#3#4{\@ifundefined{c@pp@a@#1}{}{%
    \def\next{#3}%
    \expandafter\ifx\csname pp@page@#1\endcsname\next
      \addtocounter{pp@a@#1}\@ne
    \else
      \setcounter{pp@a@#1}{\value{pp@r@#1}}%
    \fi
    \global\expandafter\let\csname pp@page@#1\endcsname\next
    \expandafter\pp@labeldef\expandafter
      {\number\value{pp@a@#1}}{#1}{#2}{#3}{#4}}}
```

`\@testdef` LaTeX's current (2007) definition of this macro causes save stack overflow. We fix this by an additional grouping. Delay to the beginning of document to keep Babel happy.

```
\AtBeginDocument{%
  \begingroup
    \@testdef{}{undefined}{}%
    \expandafter
  \endgroup
  \ifx\@undefined\relax
    \let\pp@@testdef\@testdef
    \def\@testdef#1#2#3{{\pp@@testdef{#1}{#2}{#3}%
        \if@tempswa\aftergroup\@tempswatrue\fi}}%
  \fi}
```

⟨/style⟩