

The `afterpage` package*

David Carlisle

2023/07/04

This file is maintained by the L^AT_EX Project team.
Bug reports can be opened (category `tools`) at
<https://latex-project.org/bugs.html>.

This package implements a command, `\afterpage`, that causes the commands specified in its argument to be expanded after the current page is output.¹

1. Sometimes L^AT_EX's float positioning mechanism gets overloaded, and all floating `figures` and `tables` drift to the end of the document. One may flush out all the unprocessed floats by issuing a `\clearpage` command, but this has the effect of making the current page end prematurely. Now you can issue `\afterpage{\clearpage}` and the current page will be filled up with text as usual, but then a `\clearpage` command will flush out all the floats before the next text page begins.
2. An earlier mechanism to help with float placement was the optional argument `[H]` (meaning **HERE!**) which was originally added to the standard floating environments by `here.sty`, and is now provided by `float.sty`. However some `[H]` users have commented that they did not really mean 'Here!' They actually wanted 'Somewhere close'. This can now be achieved by `\afterpage{\clearpage\begin{figure}[H] ... \end{figure}}`
This ensures that the figure is at the top of the next page. (The `\clearpage` stops any other figures drifting past the `[H]` figure.)
3. Floating longtables. `longtable.sty` provides the `longtable` environment, a multi-page version of `tabular`. Many `longtable` users have told me that it is difficult to set the text surrounding the long table, and that they wanted a 'floating' version. As, presumably, `longtables` are long, they are probably too large to hold in memory, and float in the way that the `table` environment is floated, however if the table is in a separate file, say `ltfile.tex`, you can now use one of:
`\afterpage{\clearpage\input{ltfile}}`
`\afterpage{\clearpage\input{ltfile}\clearpage}`
The first form lets text appear on the same page as the end of the longtable, the second ensures that the surrounding text starts again on a new page.

*This file has version number v1.08, last revised 2023/07/04.

¹This is really a pre-release, to see whether people like the idea of a command like this. This implementation is *not* particularly robust. This implementation does not work in two column mode, and can get 'confused' by L^AT_EX's floating environments.

```
1 (*package)
```

`\afterpage` The token register used to save the old output routine.

```
2 \newtoks\AP@output
3 \global\AP@output\expandafter{\the\output}
```

A box register used to save any part of the next page which has already been processed.

```
4 \newbox\AP@partial
```

A box register used to save any footnote texts that are ‘tied’ to the text that gets saved in `\AP@partial`.

```
5 \newbox\AP@footins
```

The following macro attempts to get safely into vertical mode, and then invokes a special output routine to grab the current page into `\AP@partial`.

```
6 \def\AP@savetop{%
```

Now begins a test to see what state we are in. `\AP@noindent` will be defined so as to return to this state (well, almost!) after `afterpage` has finished.

```
7 \ifvmode
```

Vertical mode. This is the simplest case, do nothing.

```
8 \let\AP@noindent\empty
```

```
9 \else\ifhmode
```

Horizontal mode. ‘Back out’ into vertical mode, removing the indentation box as we go. If in fact there was no indentation box, the output routine was invoked by `\noindent` (what bad luck!) so we have to remember to re-insert the `\noindent` before the paragraph is seen again. `\everypar` tokens have already been inserted, so don’t insert them again.

```
10 \setbox\z@\lastbox
```

```
11 \edef\AP@noindent
```

```
12  {\everypar{ }\ifvoid\z@\noindent\else\indent\fi}}%
```

```
13 \par
```

```
14 \else
```

The remaining (even worse) possibility that the output routine was triggered by the start of `displaymath` within a paragraph.

Come out of `displaymath` with `$$`, then adjust the spacing (getting into `vmode` at the same time). `\AP@noindent` will restart display math later. `\everydisplay` tokens have already been inserted (they apply to the math list that will be started by `\AP@noindent`, even though they were triggered by the display math that was closed by the lines below!). Save the values `\prevgraf` and `\predisplaysize` for use in the re-started math list.

```
15 \abovedisplayshortskip\z@\abovedisplayskip\z@
```

```
16 \belowdisplayshortskip\z@\belowdisplayskip\z@
```

```
17 \xdef\AP@disp{%
```

```
18 \predisplaysize\the\predisplaysize
```

```
19 \prevgraf\the\prevgraf\relax}%
```

```
20 $$\vskip-\baselineskip\vskip-\parskip
```

```
21 \edef\AP@noindent{%
```

Do not insert `\everydisplay` tokens again.

```
22 \toks@{\the\everydisplay}\everydisplay{}}%
```

Start `displaymath` mode with no spurious paragraph line above it. Restore `\prevgraf` and `\predisplaysize`. Use `\aftergroup` to restore the correct setting for `\everydisplay` after this display has finished.

```
23     {\everypar{} \noindent} $$ \AP@disp \aftergroup \noexpand \AP@ed}%
24 \fi \fi
```

Now switch the output routine and remove everything from the current page into the box `\AP@partial`.

```
25 \begingroup
26 \nointerlineskip \null
27 \output{%
28   \global \setbox \AP@partial \vbox{%
29     \unvbox \@cclv
30     \global \setbox \@ne \lastbox}%
```

If the text that is saved in `\AP@partial` had footnotes, we'd better grab them as well, otherwise they may come out on a page with the 'afterpage' text, before the page that has the footnote mark! (Added at v1.08.)

```
31   \global \setbox \AP@footins \box \footins}%
```

Having defined the output routine, trigger it...

```
32 \eject
33 \endgroup}
```

`\AP@` stores all the commands that must be executed after the page break.

```
34 \let \AP@ \relax
```

Restore the `\everydisplay` register. `\ignorespaces` prevents a space or new-line after `$$` creating a rogue indentation or paragraph.

```
35 \def \AP@ed {\everydisplay \expandafter {\the \toks@} \ignorespaces}
```

Remove the current vertical list, insert the commands `\AP@` at the top of the page, and then re-insert the saved text.

```
36 \def \AP@@ {%
37   \AP@savetop
38   \global \expandafter \let \expandafter \AP@ \expandafter \relax \AP@
39   \par
```

The text originally at the top of this page is now stored in the box `\AP@partial`, including `\topskip` glue. Now we want to unbox `\AP@partial`, placing the baseline of the first row `\baselineskip` below the baseline of the last line coming from the afterpage text. If we assumed nothing has too much height or depth (and `\topskip` is rigid), it would be fairly trivial to position the contents of `\AP@partial` so that the baseline of the first row was `\baselineskip` below the last row just added.

In this version, I thought it might be fun to try to exactly achieve the `\baselineskip` or `\lineskip` calculation that `TeX` normally does internally. The call to `\addboxcontents` does the right thing (I hope).

```
40 \addboxcontents \AP@partial
```

Now re-insert any footnote text. This may not be quite the right place, as the text that has just been unboxed may break over a page in its new position. Also it may not be the right number if the text from `\afterpage` itself contains footnotes. Too bad!

```
41 \ifvoid \AP@footins \else
42   \insert \footins {\unvbox \AP@footins} \fi
```

Now repair things if we started off in horizontal mode.

```

43 \AP@noindent}
    If \AP@ is not \relax then the current page already has some ‘afterpage’ com-
    mands, so just add the new commands to the end of the list. Otherwise save the
    commands in \AP@. (within a local group), and switch the output routine. (The
    new output routine just calls the old one if it is invoked by a LATEX float.)
44 \long\def\afterpage#1{%
45 \ifx\AP@\relax
46 \gdef\AP@{#{1\par}}%
47 \global\output{%
48 \the\AP@output
49 \ifnum\outputpenalty>-\@Mi
50 \global\output\expandafter{\the\AP@output}%
51 \aftergroup\AP@@
52 \fi}%
53 \else
54 \expandafter\gdef\expandafter\AP@\expandafter{\AP@{#{1\par}}}%
55 \fi}

```

If we have got to the end of the document or clearpage just put the stuff out without any trickery.

```

56 \let\AP@clearpage\clearpage
57 \def\clearpage{%
58 \ifx\AP@\relax
59 \AP@clearpage
60 \else
61 \global\output\expandafter{\the\AP@output}%
62 \AP@clearpage

```

At this point (since v1.08) need to clear \AP@ *before* using its expansion, as otherwise hit an infinite loop. Sigh.

```

63 \global\expandafter\let\expandafter\AP@\expandafter\relax
64 \expandafter\expandafter\AP@
65 \fi}
66 \let\AP@enddocument\enddocument
67 \def\enddocument{%
68 \ifx\AP@\relax\else
69 \global\output\expandafter{\the\AP@output}%
70 \AP@clearpage
71 \global\expandafter\let\expandafter\AP@\expandafter\relax
72 \expandafter\expandafter\AP@
73 \fi
74 \AP@enddocument}

```

\addboxcontents Given a vbox #1, add to the current vertical list such that the end result is equivalent to the list that T_EX would have built had the contents of #1 (apart from any initial glue) been added individually to the current list.

So essentially, the problem is that of unboxing #1, but replacing the glue at the top of #1 with (something equivalent to) the \baselineskip or \lineskip glue that T_EX would normally have placed before the first box in #1. Also \prevdepth must be set at the end.

```

75 \def\addboxcontents#1{%

```

Perhaps I shouldn't use grouping here, as I probably don't really want to save #1. If it is removed, `\splittopskip` and `\splitmaxdepth` would need to be restored by hand.

First replace any glue at the top by `\vskip 0pt`.

```
76 \splittopskip\z@
77 \splitmaxdepth\maxdimen
78 \setbox#1\vbox{\break\unvbox#1}%
79 \setbox\z@\vsplit#1to\z@
```

Put the breakpoint back.

```
80 \setbox#1\vbox{\break\unvbox#1}%
Set \skip@ to be height of #1 (without top glue)
81 \skip@\ht#1%
```

Now make the first baseline of the first row be `\vsize` from the top. (This assumes that the first row has height less than `\vsize`.)

```
82 \splittopskip\vsize
83 \setbox\z@\vsplit#1to\z@
```

Subtract the new height of #1 from `\skip@`, and add back on `\splittopskip`, so `\skip@` is now the height of the first row of #1. This may still be 0pt if (eg) a mark or whatsit is between the top glue and the first box. Save (this height - `\splittopskip`) in `\skip\tw@`.

```
84 \advance\skip@-\ht#1%
85 \skip\tw@\skip@
86 \advance\skip@\splittopskip
```

Now fake TeX's `\baselineskip` calculation.

```
87 \advance\skip@\prevdepth
88 \advance\skip@-\baselineskip
89 \advance\skip\tw@\ifdim-\skip@<\lineskiplimit\lineskip\else-\skip@\fi
```

Finally add the glue.

```
90 \vskip\skip\tw@
```

Now unbox the box, setting `\prevdepth` by hand, as `\unvbox` (unlike `\box`) does not automatically set it.

```
91 \global\dimen@i\dp#1%
92 \unvbox#1}%
93 \prevdepth\dimen@i}
```

```
94 </package>
```