The afterpag package*

David Carlisle
carlisle@cs.man.ac.uk

1994/05/23

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

1. Sometimes LaTeX's float positioning mechanism gets overloaded, and all floating `figure`s and `table`s 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, `longtable`s 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 apear on the same page as the end of the longtable, the second ensures that the surrounding text starts again on a new page.

```
1 ⟨*package⟩
```

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

*This file has version number v1.07, last revised 1994/05/23.

[1] This 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 LaTeX's floating environments.

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

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

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

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`.

```
\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.

```
  \ifvmode
```

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

```
    \let\AP@noindent\empty
  \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.

```
    \setbox\z@\lastbox
    \edef\AP@noindent
      {{\everypar{}\ifvoid\z@\noindent\else\indent\fi}}%
    \endgraf
  \else
```

The remaining (even worse) possibility that the ouput 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.

```
    \abovedisplayshortskip\z@\abovedisplayskip\z@
    \belowdisplayshortskip\z@\belowdisplayskip\z@
    \xdef\AP@disp{%
      \predisplaysize\the\predisplaysize
      \prevgraf\the\prevgraf\relax}%
    $$\vskip-\baselineskip\vskip-\parskip
    \edef\AP@noindent{%
```

Do not insert `\everydisplay` tokens again.

```
      \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.

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

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

```
  \begingroup
  \nointerlineskip\null
    \output{%
      \global\setbox\AP@partial\vbox{%
        \unvbox\@cclv
         \global\setbox\@ne\lastbox}}%
  \eject
  \endgroup}
```

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

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

Restore the \everydisplay register. \ignorespaces prevents a space or newline after $$ creating rogue a indentation or paragraph.

```
\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.

```
\def\AP@@{%
  \AP@savetop
  \global\expandafter\let\expandafter\AP@\expandafter\relax\AP@
  \endgraf
```

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).

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

Now repair things if we started off in horizontal mode.

```
  \AP@noindent}
```

If \AP@ is not \relax then the current page already has some 'afterpage' commands, 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.

```
\long\def\afterpage#1{%
  \ifx\AP@\relax
    \gdef\AP@{{#1\endgraf}}%
    \global\output{%
      \the\AP@output
      \ifnum\outputpenalty>-\@Mi
        \global\output\expandafter{\the\AP@output}%
        \aftergroup\AP@@
      \fi}%
  \else
    \expandafter\gdef\expandafter\AP@\expandafter{\AP@{#1\endgraf}}%
  \fi}
```

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

```
\let\AP@clearpage\clearpage
\def\clearpage{%
  \ifx\AP@\relax
    \AP@clearpage
  \else
    \global\output\expandafter{\the\AP@output}%
    \AP@clearpage
    \expandafter\AP@\fi\global\let\AP@\relax}
\let\AP@enddocument\enddocument
\def\enddocument{%
  \ifx\AP@\relax\else
    \global\output\expandafter{\the\AP@output}%
    \AP@clearpage
    \expandafter\AP@\fi\global\let\AP@\relax
    \AP@enddocument}
```

`\addboxcontents` Given a vbox `#1`, add to the current vertical list such that the end result is equivalent to the list that TeX 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 TeX would normally have placed before the first box in `#1`. Also `\prevdepth` must be set at the end.

```
\def\addboxcontents#1{{%
```

Perhaps I shouldnt 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`.

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

Put the breakpoint back.

```
  \setbox#1\vbox{\break\unvbox#1}%
```

Set `\skip@` to be height of `#1` (without top glue)

```
  \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`.)

```
  \splittopskip\vsize
  \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@`.

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

Now fake TeX's `\baselineskip` calculation.

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

Finally add the glue.

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

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

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

⟨/package⟩