The fix2col package*

David Carlisle†

1998/08/17

1 Introduction

This package makes two independent changes to LaTeX's two column output routine to fix the following two longstanding 'features'.

- If the TeX mark system is used (for example using the 'headings' page style in the standard LaTeX classes) then any marks that originate on the first column are 'lost' as LaTeX constructs the second column. An example document showing how this can result in incorrect page headings may be found in the latex bug database:
  `http://www.uni-mainz.de/cgi-bin/ltxbugs2html?pr=latex/2613`
- The second feature is documented in the LaTeX book. By default LaTeX does not attempt to keep double and single column floats in sequence, so if 'Figure 1' is a double column float produced with `figure*`, then it may float after 'Figure 2' if that is a single column, `figure`, float. Further correspondence about this may also be found in the bug database:
  `http://www.uni-mainz.de/cgi-bin/ltxbugs2html?pr=latex/2346`

2 Notes on the Implementation Strategies

2.1 Preserving Marks

The standard LaTeX twocolumn system works internally by making each column a separate 'page' that is passed independently to TeX's pagebreaker. (Unlike say the multicol package, where all columns are gathered together and then split into columns later, using `\vsplit`.) This means that the primitive TeX marks that are normally used for header information, are globally reset after the first column. By default LaTeX does nothing about this. A good solution is provided by Piet van Oostrum (building on earlier work of Joe Pallas) in his fixmarks package.

*This file has version number v0.03, last revised 1998/08/17.

†Part one is essentially a copy of the fixmarks package by Piet van Oostrum, itself based on earlier work by Joe Pallas. Part two is loosely based on the fixfloats package, originally by Ed Sznyter, with some modifications by Bil Kleb.

After the first column box has been collected the mark information for that box is saved, so that any `\firstmark` can be 'artificially' used to set the page-level marks after the second column has been collected. (The second column `\firstmark` is not normally required.) Unfortunately TEX does not provide a direct way of knowing if any marks are in the page, `\firstmark` always has a value from previous pages, even if there is no mark in this page. The solution is to make a copy of the box and then `\vsplit` it so that any marks show up as `\splitfirstmark`.

The use of `\vsplit` does mean that the output routine will globally change the value of `\splitfirstmark` and `\splitbotmark`. The fixmarks package goes to some trouble to save and restore these values so that the output routine does *not* change the values. This part of fixmarks is not copied here as it is quite costly (having to be run on every page) and there is no reason why anyone writing code using `\vsplit` should allow the output routine to be triggered before the split marks have been accessed.

2.2 Preserving Float Order

The standard output routine maintains two lists of floats that have been 'deferred' for later consideration. One list for single column floats, and one for double column floats (which are always immediately put onto their deferred list). This mechanism means that LATEX 'knows' which type of float is contained in each box by the list that it is processing, but having two lists means that there is no mechanism for preserving the order between the floats in each list.

The solution to this problem consists of two small changes to the output routine.

Firstly, abandon the 'double column float list' `\@dbldeferlist` and change every command where it is used so that instead the same `\@deferlist` is used as for single column floats. That one change ensures that double and single column floats stay in the same sequence, but as LATEX no longer 'knows' whether a float is double or single column, it will happily insert a double float into a single column, overprinting the other column, or the margin.

The second change is to provide an alternative mechanism for recording the two column floats. LATEX already has a compact mechanism for recording float information, an integer count register assigned to each float records information about the 'type' of float 'figure', 'table' and the position information 'htp' etc.

The type information is stored in the 'high' bits, one bit position (above '32') allocated to each float type. The 'low' bits store information about the allowed positions, one bit each allocated for `h t b p`. In the LATEX2.09 system, the bit corresponding to '16' formed a 'boundary' between these two sets of information, and it was never actually used by the system. Ed Sznyter's fixfloats package not unreasonably used this position to store the double column information, setting the bit for double column floats. Then at each point in the output routine at which a float is committed to a certain region, an additional check must be made to check that the float is (or is not) double column. If it spans the wrong number of columns it is deferred rather than being added.

Unfortunately the bit '16' is not available in LaTeX 2_ε. It is used to encode the extra float position possibility '!' that was added in that system. It would be possible to use position '32' and to move the flags for 'table', 'figure',... up one position, to start at 64, but this would mean that in principle one less float type would be supported, and more importantly is likely to break any other packages that assume anything about the output routine internals. So here I instead use another mechanism for flagging double column floats: By default all floats have depth 0pt. This package arranges that double column ones have depth 1sp. This information may then be used in the same manner as in the fixfloats package, to defer any floats that are not of the correct column spanning type.

Use of the package showed that one also has to change the way LaTeX handles star-form floats: if they are immediately deferred (as done normally) certain situations can still result in the float sequence getting out of order. This happens when a floats are placed in the middle of a paragraph. In that case the wide float is deferred immediately while a column wide float early on in the same paragraph might not be handled until the end of the paragraph when it is finally seen by the output routine. Since by that time the wide float is already on the `\@deferlist` the column float will also end up there (which is not only incorrect because it may have fitted onto the page but also because it is then placed at the end of this list). Version v0.03 now fixes this problem.

3 Implementation

1 ⟨*package⟩

3.1 Preserving Marks

This is just a change to the single command `\@outputdblcol` so that it saves mark information for the first column and restores it in the second column.

```
2 \def\@outputdblcol{%
3   \if@firstcolumn
4     \global\@firstcolumnfalse
```

Save the left column

```
5     \global\setbox\@leftcolumn\copy\@outputbox
```

Remember the marks from the first column

```
6     \splitmaxdepth\maxdimen
7     \vbadness\maxdimen
8     \setbox\@outputbox\vsplit\@outputbox to\maxdimen
```

One minor difference from the current fixmarks, pass the marks through a token register to stop any # tokens causing an error in a `\def`.

```
9     \toks@\expandafter{\topmark}%
10    \xdef\@firstcoltopmark{\the\toks@}%
11    \toks@\expandafter{\splitfirstmark}%
12    \xdef\@firstcolfirstmark{\the\toks@}%
```

This test does not work if truly empty marks have been inserted, but LaTeX marks should always have (at least) two brace groups. (Except before the first mark is used, when the marks are empty, but that is OK here.)

```
    \ifx\@firstcolfirstmark\@empty
      \global\let\@setmarks\relax
    \else
      \gdef\@setmarks{%
        \let\firstmark\@firstcolfirstmark
        \let\topmark\@firstcoltopmark}%
    \fi
```

End of change

```
  \else
    \global\@firstcolumntrue
    \setbox\@outputbox\vbox{%
     \hb@xt@\textwidth{%
        \hb@xt@\columnwidth{\box\@leftcolumn \hss}%
        \hfil
        \vrule \@width\columnseprule
        \hfil
       \hb@xt@\columnwidth{\box\@outputbox \hss}}}%
  \@combinedblfloats
```

Override current first and top with those of first column if necessary

```
    \@setmarks
```

End of change

```
    \@outputpage
    \begingroup
      \@dblfloatplacement
      \@startdblcolumn
      \@whilesw\if@fcolmade \fi{\@outputpage\@startdblcolumn}%
    \endgroup
  \fi}
```

3.2 Preserving Float Order

Changes `\@dbldeferlist` to `\@deferlist` are not explicitly noted but are flagged by blank comment lines around the changed line.

```
\def\end@dblfloat{%
\if@twocolumn
  \@endfloatbox
  \ifnum\@floatpenalty <\z@
    \@largefloatcheck
```

Force the depth of two column float boxes.

```
    \global\dp\@currbox1sp %
```

Next line assumes that first token of `\end@float` is `\@endfloatbox` so we gobble that.

```
%    \@cons\@deferlist\@currbox
     \expandafter\@gobble\end@float
```

`\@Esphack` is then added by `\@endfloat` above.

```
  \fi
%    \ifnum \@floatpenalty =-\@Mii \@Esphack\fi
\else
  \end@float
\fi
}
```

Test if the float box has the wrong width. (Actually as noted above the test is for a conventional depth setting rather than for the width of the float).

```
\def\@testwrongwidth #1{%
  \ifdim\dp#1=\f@depth
  \else
    \global\@testtrue
  \fi}
```

Normally looking for single column floats, which have zero depth.

```
\let\f@depth\z@
```

but when making two column float area, look for floats with 1sp depth.

```
\def\@dblfloatplacement{\global\@dbltopnum\c@dbltopnumber
   \global\@dbltoproom \dbltopfraction\@colht
   \@textmin \@colht
   \advance \@textmin -\@dbltoproom
   \@fpmin \dblfloatpagefraction\textheight
   \@fptop \@dblfptop
   \@fpsep \@dblfpsep
   \@fpbot \@dblfpbot
   \def\f@depth{1sp}}
```

All the remaining changes are replacing the double column defer list or insering the extra test `\@testwrongwidth{`$\langle box \rangle$`}` at suitable places. That is at plces where a box is taken off the deferlist.

```
\def \@doclearpage {%
     \ifvoid\footins
       \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
       \setbox\@tempboxa\box\@cclv
       \xdef\@deferlist{\@toplist\@botlist\@deferlist}%
       \global \let \@toplist \@empty
       \global \let \@botlist \@empty
       \global \@colroom \@colht
       \ifx \@currlist\@empty
       \else
          \@latexerr{Float(s) lost}\@ehb
          \global \let \@currlist \@empty
       \fi
       \@makefcolumn\@deferlist
       \@whilesw\if@fcolmade \fi{\@opcol\@makefcolumn\@deferlist}%
```

```
      \if@twocolumn
        \if@firstcolumn
          \xdef\@deferlist{\@dbltoplist\@deferlist}%
          \global \let \@dbltoplist \@empty
          \global \@colht \textheight
          \begingroup
             \@dblfloatplacement
             \@makefcolumn\@deferlist
             \@whilesw\if@fcolmade \fi{\@outputpage
                                       \@makefcolumn\@deferlist}%
          \endgroup
        \else
          \vbox{}\clearpage
        \fi
      \fi
```

the next line is needed to avoid losing floats in certain circumstances a single call to the original `\doclearpage` will now no longer output all floats.

```
      \ifx\@deferlist\@empty \else\clearpage \fi
    \else
      \setbox\@cclv\vbox{\box\@cclv\vfil}%
      \@makecol\@opcol
      \clearpage
    \fi
}
\def \@startdblcolumn {%
  \@tryfcolumn \@deferlist
  \if@fcolmade
  \else
    \begingroup
      \let \reserved@b \@deferlist
      \global \let \@deferlist \@empty
      \let \@elt \@sdblcolelt
      \reserved@b
    \endgroup
  \fi
}
\def\@addtonextcol{%
  \begingroup
   \@insertfalse
   \@setfloattypecounts
   \ifnum \@fpstype=8
   \else
     \ifnum \@fpstype=24
     \else
       \@flsettextmin
       \@reqcolroom \ht\@currbox
```

```
      \advance \@reqcolroom \@textmin
      \ifdim \@colroom>\@reqcolroom
        \@flsetnum \@colnum
        \ifnum\@colnum>\z@
           \@bitor\@currtype\@deferlist
           \@testwrongwidth\@currbox
           \if@test
           \else
             \@addtotoporbot
           \fi
        \fi
      \fi
    \fi
  \fi
  \if@insert
  \else
    \@cons\@deferlist\@currbox
  \fi
 \endgroup
}
\def\@addtodblcol{%
 \begingroup
  \@insertfalse
  \@setfloattypecounts
  \@getfpsbit \tw@
  \ifodd\@tempcnta
    \@flsetnum \@dbltopnum
    \ifnum \@dbltopnum>\z@
      \@tempswafalse
      \ifdim \@dbltoproom>\ht\@currbox
        \@tempswatrue
      \else
        \ifnum \@fpstype<\sixt@@n
          \advance \@dbltoproom \@textmin
          \ifdim \@dbltoproom>\ht\@currbox
            \@tempswatrue
          \fi
          \advance \@dbltoproom -\@textmin
        \fi
      \fi
      \if@tempswa
          \@bitor \@currtype \@deferlist
```

not in fixfloats?

```
         \@testwrongwidth\@currbox
          \if@test
          \else
             \@tempdima -\ht\@currbox
             \advance\@tempdima
```

```
                -\ifx \@dbltoplist\@empty \dbltextfloatsep \else
                                          \dblfloatsep \fi
              \global \advance \@dbltoproom \@tempdima
              \global \advance \@colht \@tempdima
              \global \advance \@dbltopnum \m@ne
              \@cons \@dbltoplist \@currbox
              \@inserttrue
            \fi
        \fi
      \fi
    \fi
    \if@insert
    \else
      \@cons\@deferlist\@currbox
    \fi
  \endgroup
}
\def \@addtocurcol {%
    \@insertfalse
    \@setfloattypecounts
    \ifnum \@fpstype=8
    \else
      \ifnum \@fpstype=24
      \else
        \@flsettextmin
        \advance \@textmin \@textfloatsheight
        \@reqcolroom \@pageht
        \ifdim \@textmin>\@reqcolroom
          \@reqcolroom \@textmin
        \fi
        \advance \@reqcolroom \ht\@currbox
        \ifdim \@colroom>\@reqcolroom
          \@flsetnum \@colnum
          \ifnum \@colnum>\z@
            \@bitor\@currtype\@deferlist
```

We need to defer the float also if its width doesn't fit.

```
           \@testwrongwidth\@currbox
           \if@test
           \else
             \@bitor\@currtype\@botlist
             \if@test
               \@addtobot
             \else
               \ifodd \count\@currbox
                 \advance \@reqcolroom \intextsep
                 \ifdim \@colroom>\@reqcolroom
                   \global \advance \@colnum \m@ne
                   \global \advance \@textfloatsheight \ht\@currbox
```

```
                  \global \advance \@textfloatsheight 2\intextsep
                  \@cons \@midlist \@currbox
                  \if@nobreak
                    \nobreak
                    \@nobreakfalse
                    \everypar{}%
                  \else
                    \addpenalty \interlinepenalty
                  \fi
                  \vskip \intextsep
                  \box\@currbox
                  \penalty\interlinepenalty
                  \vskip\intextsep
                  \ifnum\outputpenalty <-\@Mii \vskip -\parskip\fi
                  \outputpenalty \z@
                  \@inserttrue
                \fi
              \fi
              \if@insert
              \else
                \@addtotoporbot
              \fi
            \fi
          \fi
        \fi
      \fi
    \fi
  \fi
  \if@insert
  \else
    \@resethfps
    \@cons\@deferlist\@currbox
  \fi
}
\def\@xtryfc #1{%
  \@next\reserved@a\@trylist{}{}%
  \@currtype \count #1%
  \divide\@currtype\@xxxii
  \multiply\@currtype\@xxxii
  \@bitor \@currtype \@failedlist
  \@testfp #1%
  \@testwrongwidth #1%
  \ifdim \ht #1>\@colht
     \@testtrue
  \fi
  \if@test
    \@cons\@failedlist #1%
  \else
    \@ytryfc #1%
```

```
  \fi}
\def\@ztryfc #1{%
  \@tempcnta\count #1%
  \divide\@tempcnta\@xxxii
  \multiply\@tempcnta\@xxxii
  \@bitor \@tempcnta {\@failedlist \@flfail}%
  \@testfp #1%
```

not in fixfloats?

```
  \@testwrongwidth #1%
  \@tempdimb\@tempdima
  \advance\@tempdimb\ht #1%
  \advance\@tempdimb\@fpsep
  \ifdim \@tempdimb >\@colht
    \@testtrue
  \fi
  \if@test
    \@cons\@flfail #1%
  \else
    \@cons\@flsucceed #1%
    \@tempdima\@tempdimb
  \fi}
```

⟨/package⟩

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.